ARM-based Embedded MPU

SAM9X25

DATASHEET

Description

The SAM9X25 is a high-performance ARM926EJ-S™-based embedded microprocessor unit, running at 400 MHz and featuring multiple networking/connectivity peripherals, optimized for industrial applications such as building automation, gateways and medical.

The SAM9X25 features two 2.0A/B compatible Controller Area Network (CAN) interfaces and two IEEE Std 802.3-compatible 10/100 Mbps Ethernet MACs. Additional communication interfaces include a soft modem supporting exclusively the Conexant SmartDAA line driver, HS USB Device and Host, FS USB Host, two HS SDCard/SDIO/MMC interfaces, USARTs, SPIs, I2S, TWIs and 10-bit ADC.

To ensure uninterrupted data transfer with minimum processor overhead, the SAM9X25 offers a 10-layer bus matrix coupled with 2 x 8 central DMA channels and dedicated DMAs for the high-speed connectivity peripherals.

The External Bus Interface incorporates controllers for 4-bank and 8-bank DDR2/LPDDR, SDRAM/LPSDRAM, static memories, and specific circuitry for MLC/SLC NAND Flash with integrated ECC.

The SAM9X25 is available in a 217-ball BGA package with 0.8 mm ball pitch.

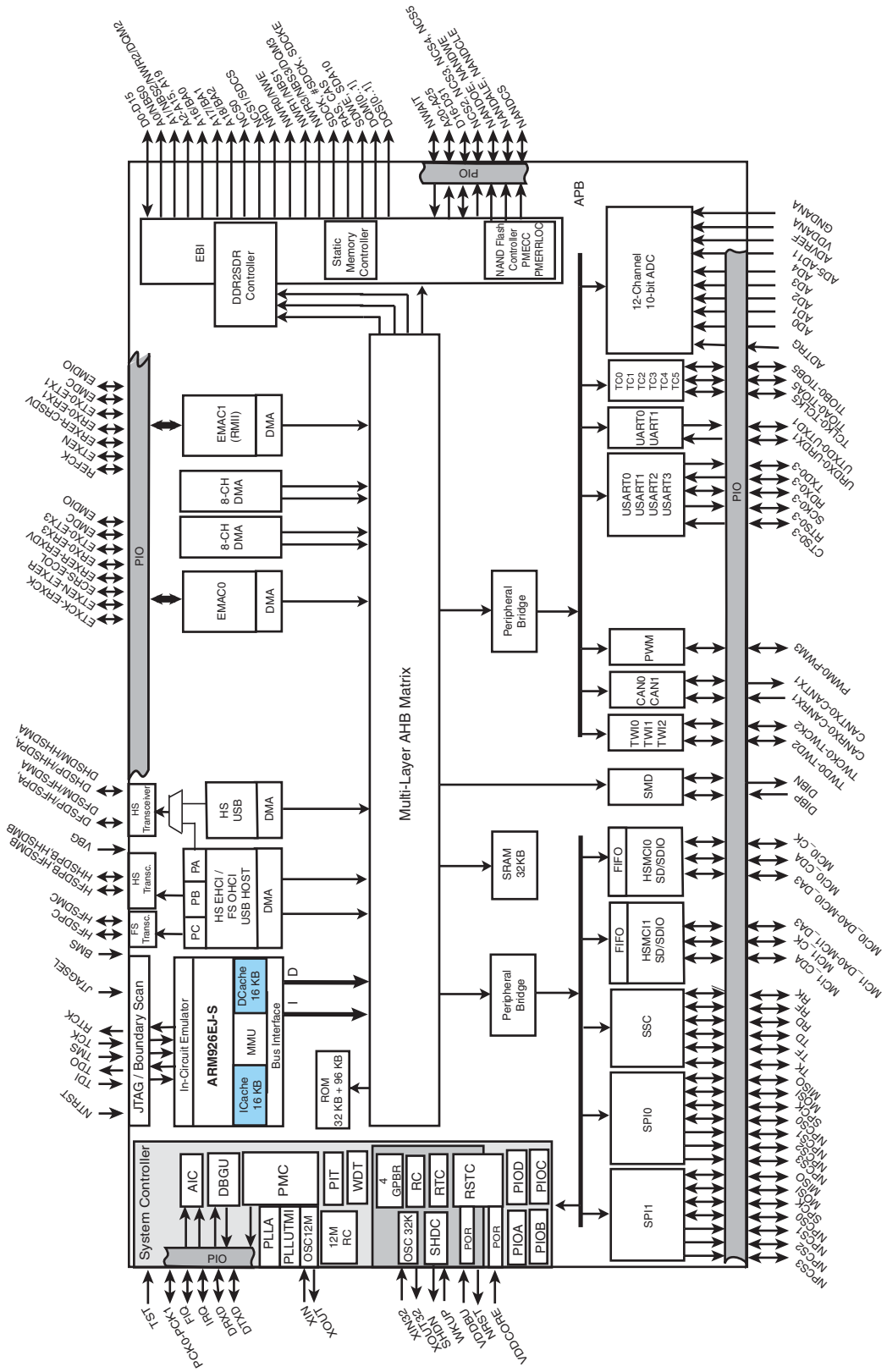
1. Features

- Core
  - ARM926EJ-S™ ARM® Thumb® Processor running at up to 400 MHz @ 1.0V +/- 10%
  - 16 Kbytes Data Cache, 16 Kbytes Instruction Cache, Memory Management Unit
- Memories
  - One 64-Kbyte internal ROM embedding bootstrap routine: Boot on NAND Flash, SDCard, DataFlash® or serial DataFlash. Programmable order.
  - One 32-Kbyte internal SRAM, single-cycle access at system speed
  - High Bandwidth Multi-port DDR2 Controller
  - 32-bit External Bus Interface supporting 4-bank and 8-bank DDR2/LPDDR, SDR/LPSDR, Static Memories
  - MLC/SLC 8-bit NAND Controller, with up to 24-bit Programmable Multi-bit Error Correcting Code (PMECC)
- System running at up to 133 MHz
  - Power-on Reset Cells, Reset Controller, Shut Down Controller, Periodic Interval Timer, Watchdog Timer and Real Time Clock
  - Boot Mode Select Option, Remap Command
  - Internal Low Power 32 kHz RC and Fast 12 MHz RC Oscillators
  - Selectable 32768 Hz Low-power Oscillator and 12 MHz Oscillator
  - One PLL for the system and one PLL at 480 MHz optimized for USB High Speed
  - Twelve 32-bit-layer AHB Bus Matrix for large Bandwidth transfers
  - Dual Peripheral Bridge with dedicated programmable clock for best performance
  - Two dual port 8-channel DMA Controllers
  - Advanced Interrupt Controller and Debug Unit
  - Two Programmable External Clock Signals
- Low Power Mode
  - Shut Down Controller with four 32-bit Battery Backup Registers
  - Clock Generator and Power Management Controller
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
- Peripherals
  - USB Device High Speed, USB Host High Speed and USB Host Full Speed with dedicated On-Chip Transceiver
  - Two 10/100 Mbps Ethernet MAC Controllers
  - Two High Speed Memory Card Hosts
  - Two CAN Controllers
  - Two Master/Slave Serial Peripheral Interface
  - Two 3-channel 32-bit Timer/Counters
  - One Synchronous Serial Controller
  - One 4-channel 16-bit PWM Controller
  - Three Two-wire Interfaces
  - Four USARTs, two UARTs, one DBGU
  - One 12-channel 10-bit Analog-to-Digital Converter
  - Soft Modem
  - Write Protected Registers
- I/O
  - Four 32-bit Parallel Input/Output Controllers
  - 105 Programmable I/O Lines Multiplexed with up to Three Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line, optional Schmitt trigger input

    - Individually Programmable Open-drain, Pull-up and pull-down resistor, Synchronous Output
- Package
    - 217-ball BGA, pitch 0.8 mm

2. Block Diagram

Figure 2-1. SAM9X25 Block Diagram



3. Signal Description

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

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level
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	
VBG	Bias Voltage Reference for USB	Analog	
PCK0–PCK1	Programmable Clock Output	Output	
Shutdown, Wakeup Logic			
SHDN	Shut-Down Control	Output	
WKUP	Wake-Up Input	Input	
ICE and JTAG			
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
JTAGSEL	JTAG Selection	Input	
RTCK	Return Test Clock	Output	
Reset/Test			
NRST	Microcontroller Reset	I/O	Low
TST	Test Mode Select	Input	
NTRST	Test Reset Signal	Input	
BMS	Boot Mode Select	**Input**	
Debug Unit - DBGU			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	
Advanced Interrupt Controller - AIC			
IRQ	External Interrupt Input	Input	
FIQ	Fast Interrupt Input	Input	
PIO Controller - PIOA - PIOB - PIOC - PIOD			
PA0–PA31	Parallel IO Controller A	I/O	
PB0–PB18	Parallel IO Controller B	I/O	
PC0–PC31	Parallel IO Controller C	I/O	
PD0–PD21	Parallel IO Controller D	I/O	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
External Bus Interface - EBI			
D0–D15	Data Bus	I/O	
D16–D31	Data Bus	I/O	
A0–A25	Address Bus	Output	
NWAIT	External Wait Signal	Input	Low
Static Memory Controller - SMC			
NCS0–NCS5	Chip Select Lines	Output	Low
NWR0–NWR3	Write Signal	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0–NBS3	Byte Mask Signal	Output	Low
NAND Flash Support			
NFD0–NFD16	NAND Flash I/O	I/O	
NANDCS	NAND Flash Chip Select	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
DDR2/SDRAM/LPDDR Controller			
SDCK,#SDCK	DDR2/SDRAM Differential Clock	Output	
SDCKE	DDR2/SDRAM Clock Enable	Output	High
SDCS	DDR2/SDRAM Controller Chip Select	Output	Low
BA[0..2]	Bank Select	Output	Low
SDWE	DDR2/SDRAM Write Enable	Output	Low
RAS-CAS	Row and Column Signal	Output	Low
SDA10	SDRAM Address 10 Line	Output	
DQS[0..1]	Data Strobe	I/O	
DQM[0..3]	Write Data Mask	Output	
High Speed MultiMedia Card Interface - HSMCI0–1			
MCI0_CK, MCI1_CK	Multimedia Card Clock	I/O	
MCI0_CDA, MCI1_CDA	Multimedia Card Slot Command	I/O	
MCI0_DA0–MCI0_DA3	Multimedia Card 0 Slot A Data	I/O	
MCI1_DA0–MCI1_DA3	Multimedia Card 1 Slot A Data	I/O	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
Universal Synchronous Asynchronous Receiver Transmitter - USARTx			
SCKx	USARTx Serial Clock	I/O	
TXDx	USARTx Transmit Data	Output	
RXDx	USARTx Receive Data	Input	
RTSx	USARTx Request To Send	Output	
CTSx	USARTx Clear To Send	Input	
Universal Asynchronous Receiver Transmitter - UARTx			
UTXDx	UARTx Transmit Data	Output	
URXDx	UARTx Receive Data	Input	
Synchronous Serial Controller - SSC			
TD	SSC Transmit Data	Output	
RD	SSC Receive Data	Input	
TK	SSC Transmit Clock	I/O	
RK	SSC Receive Clock	I/O	
TF	SSC Transmit Frame Sync	I/O	
RF	SSC Receive Frame Sync	I/O	
Timer/Counter - TCx x=0..5			
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	
Serial Peripheral Interface - SPIx			
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 - TWIx			
TWDx	Two-wire Serial Data	I/O	
TWCKx	Two-wire Serial Clock	I/O	
Pulse Width Modulation Controller - PWMC			
PWM0–PWM3	Pulse Width Modulation Output	Output	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
USB Host High Speed Port - UHPHS			
HFSDPA	USB Host Port A Full Speed Data +	Analog	
HFSDMA	USB Host Port A Full Speed Data -	Analog	
HHSDPA	USB Host Port A High Speed Data +	Analog	
HHSDMA	USB Host Port A High Speed Data -	Analog	
HFSDPB	USB Host Port B Full Speed Data +	Analog	
HFSDMB	USB Host Port B Full Speed Data -	Analog	
HHSDPB	USB Host Port B High Speed Data +	Analog	
HHSDMB	USB Host Port B High Speed Data -	Analog	
HFSDMC	USB Host Port C Full Speed Data -	Analog	
HFSDPC	USB Host Port C Full Speed Data +	Analog	
USB Device High Speed Port - UDPHS			
DFSDM	USB Device Full Speed Data -	Analog	
DFSDP	USB Device Full Speed Data +	Analog	
DHSDM	USB Device High Speed Data -	Analog	
DHSDP	USB Device High Speed Data +	Analog	
Ethernet 10/100 - EMAC0			
ETXCK	Transmit Clock or Reference Clock	Input	
ERXCK	Receive Clock	Input	
ETXEN	Transmit Enable	Output	
ETX0–ETX3	Transmit Data	Output	
ETXER	Transmit Coding Error	Output	
ERXDV	Receive Data Valid	Input	
ERX0–ERX3	Receive Data	Input	
ERXER	Receive Error	Input	
ECRS	Carrier Sense and Data Valid	Input	
ECOL	Collision Detect	Input	
EMDC	Management Data Clock	Output	
EMDIO	Management Data Input/Output	I/O	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
RMII Ethernet 10/100 - EMAC1			
REFCK	Transmit Clock or Reference Clock	Input	
ETXEN	Transmit Enable	Output	
ETX0–ETX1	Transmit Data	Output	
CRSDV	Receive Data Valid	Input	
ERX0–ERX1	Receive Data	Input	
ERXER	Receive Error	Input	
EMDC	Management Data Clock	Output	
EMDIO	Management Data Input/Output	I/O	
Analog-to-Digital Converter - ADC			
AD0–AD11	12 Analog Inputs	Analog	
ADTRG	ADC Trigger	Input	
ADVREF	ADC Reference	Analog	
CAN Controller - CANx			
CANRXx	CAN input	Input	
CANTXx	CAN output	Output	
Soft Modem - SMD			
DIBN	Soft Modem Signal	I/O	
DIBP	Soft Modem Signal	I/O	

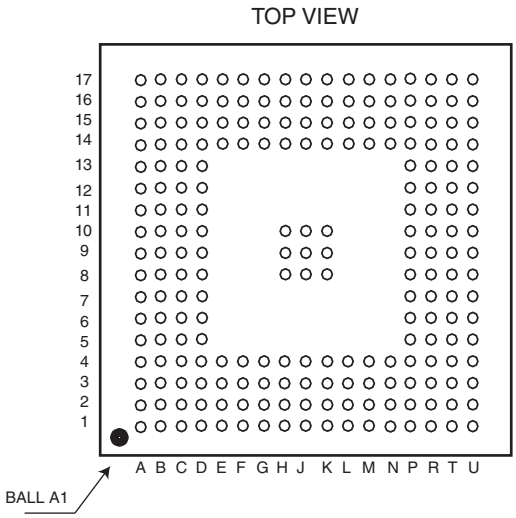
4. Package and Pinout

The SAM9X25 is available in 217-ball BGA package.

4.1 Overview of the 217-ball BGA Package

Figure 4-1 shows the orientation of the 217-ball BGA Package.

Figure 4-1. Orientation of the 217-ball BGA Package



4.2 I/O Description

Table 4-1. SAM9X25 I/O Type Description

I/O Type	Voltage Range	Analog	Pull-up	Pull-down	Schmitt Trigger
GPIO	1.65–3.6V		Switchable	Switchable	Switchable
GPIO_CLK	1.65–3.6V		Switchable	Switchable	Switchable
GPIO_CLK2	1.65–3.6V		Switchable	Switchable	Switchable
GPIO_ANA	3.0–3.6V	I	Switchable		Switchable
EBI	1.65–1.95V, 3.0–3.6V		Switchable	Switchable	
EBI_O	1.65–1.95V, 3.0–3.6V		Reset State	Reset State	
EBI_CLK	1.65–1.95V, 3.0–3.6V				
RSTJTAG	3.0–3.6V		Reset State	Reset State	Reset State
SYSC	1.65–3.6V		Reset State	Reset State	Reset State
VBG	0.9–1.1V	I			
USBFS	3.0–3.6V	I/O			
USBHS	3.0–3.6V	I/O			
CLOCK	1.65–3.6V	I/O			
DIB	3.0–3.6V	I/O			

When “Reset State” is mentioned, the configuration is defined by the “Reset State” column of the Pin Description table.

Table 4-2. SAM9X25 I/O Type Assignment and Frequency

I/O Type	I/O Frequency (MHz)	Charge Load (pF)	Output Current	Signal Name
GPIO	40	10		All PIO lines except GPIO_CLK, GPIO_CLK2, and GPIO_ANA
GPIO_CLK	54	10		MCI0CK, MCI1CK, SPI0SPCK, SPI1SPCK, EMACx_ETXCK, ISI_MCK
GPIO_CLK2	75	10		
GPIO_ANA	25	10	16 mA, 40 mA (peak)	ADx, GPADx
EBI	133	50 (3.3V) 30 (1.8V)		All data lines (Input/output)
EBI_O	66	50 (3.3V) 30 (1.8V)		All address and control lines (output only) except EBI_CLK
EBI_CLK	133	10		CK, #CK
RSTJTAG	10	10		NRST, NTRST, BMS, TCK, TDI, TMS, TDO, RTCK
SYSC	0.25	10		WKUP, SHDN, JTAGSEL, TST, SHDN
VBG	0.25	10		VBG
USBFS	12	10		HFSDPA, HFSDPB/DFSDP, HFSDPC, HFSDMA, HFSDMB/DFSDM, HFSDMC
USBHS	480	10		HHSDPA, HHSDPB/DHSDP, HHSDMA, HHSDMB/DHSDM
CLOCK	50	50		XIN, XOUT, XIN32, XOUT32
DIB	25	25		DIBN, DIBP

4.2.1 Reset State

In the tables that follow, the column “Reset State” indicates the reset state of the line with mnemonics.

- “PIO” “/” signal

Indicates whether the PIO Line resets in I/O mode or in peripheral mode. If “PIO” is mentioned, the PIO Line is maintained in a static state as soon as the reset is released. As a result, the bit corresponding to the PIO Line in the register PIO_PSR (Peripheral Status Register) resets low.

If a signal name is mentioned in the “Reset State” column, the PIO Line is assigned to this function and the corresponding bit in PIO_PSR resets high. This is the case of pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released.

- “I”/“O”

Indicates whether the signal is input or output state.

- “PU”/“PD”

Indicates whether Pull-Up, Pull-Down or nothing is enabled.

- “ST”

Indicates if Schmitt Trigger is enabled.

Note: Example: The PB18 “Reset State” column shows “PIO, I, PU, ST”. That means the line PIO18 is configured as an Input with Pull-Up and Schmitt Trigger enabled. PD14 reset state is “PIO, I, PU”. That means PIO Input with Pull-Up. PD15 reset state is “A20, O, PD” which means output address line 20 with Pull-Down.

4.3 217-ball BGA Package Pinout

Table 4-3. Pin Description BGA217

			Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
Ball	Power Rail	I/O Type	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
L3	VDDIOP0	GPIO	PA0	I/O			TXD0	O	SPI1_NPCS1	O			PIO, I, PU, ST
P1	VDDIOP0	GPIO	PA1	I/O			RXD0	I	SPI0_NPCS2	O			PIO, I, PU, ST
L4	VDDIOP0	GPIO	PA2	I/O			RTS0	O	MCI1_DA1	I/O	E0_TX0	O	PIO, I, PU, ST
N4	VDDIOP0	GPIO	PA3	I/O			CTS0	I	MCI1_DA2	I/O	E0_TX1	O	PIO, I, PU, ST
T3	VDDIOP0	GPIO	PA4	I/O			SCK0	I/O	MCI1_DA3	I/O	E0_TXER	O	PIO, I, PU, ST
R1	VDDIOP0	GPIO	PA5	I/O			TXD1	O	CANTX1	O			PIO, I, PU, ST
R4	VDDIOP0	GPIO	PA6	I/O			RXD1	I	CANRX1	I			PIO, I, PU, ST
R3	VDDIOP0	GPIO	PA7	I/O			TXD2	O	SPI0_NPCS1	O			PIO, I, PU, ST
P4	VDDIOP0	GPIO	PA8	I/O			RXD2	I	SPI1_NPCS0	I/O			PIO, I, PU, ST
U3	VDDIOP0	GPIO	PA9	I/O			DRXD	I	CANRX0	I			PIO, I, PU, ST
T1	VDDIOP0	GPIO	PA10	I/O			DTXD	O	CANTX0	O			PIO, I, PU, ST
U1	VDDIOP0	GPIO	PA11	I/O			SPI0_MISO	I/O	MCI1_DA0	I/O			PIO, I, PU, ST
T2	VDDIOP0	GPIO	PA12	I/O			SPI0_MOSI	I/O	MCI1_CDA	I/O			PIO, I, PU, ST
T4	VDDIOP0	GPIO_CLK	PA13	I/O			SPI0_SPCK	I/O	MCI1_CK	I/O			PIO, I, PU, ST
U2	VDDIOP0	GPIO	PA14	I/O			SPI0_NPCS0	I/O					PIO, I, PU, ST
U4	VDDIOP0	GPIO	PA15	I/O			MCI0_DA0	I/O					PIO, I, PU, ST
P5	VDDIOP0	GPIO	PA16	I/O			MCI0_CDA	I/O					PIO, I, PU, ST
R5	VDDIOP0	GPIO_CLK	PA17	I/O			MCI0_CK	I/O					PIO, I, PU, ST
U5	VDDIOP0	GPIO	PA18	I/O			MCI0_DA1	I/O					PIO, I, PU, ST
T5	VDDIOP0	GPIO	PA19	I/O			MCI0_DA2	I/O					PIO, I, PU, ST
U6	VDDIOP0	GPIO	PA20	I/O			MCI0_DA3	I/O					PIO, I, PU, ST
T6	VDDIOP0	GPIO	PA21	I/O			TIOA0	I/O	SPI1_MISO	I/O			PIO, I, PU, ST
R6	VDDIOP0	GPIO	PA22	I/O			TIOA1	I/O	SPI1_MOSI	I/O			PIO, I, PU, ST
U7	VDDIOP0	GPIO_CLK	PA23	I/O			TIOA2	I/O	SPI1_SPCK	I/O			PIO, I, PU, ST
T7	VDDIOP0	GPIO	PA24	I/O			TCLK0	I	TK	I/O			PIO, I, PU, ST
T8	VDDIOP0	GPIO	PA25	I/O			TCLK1	I	TF	I/O			PIO, I, PU, ST
R7	VDDIOP0	GPIO	PA26	I/O			TCLK2	I	TD	O			PIO, I, PU, ST
P8	VDDIOP0	GPIO	PA27	I/O			TIOB0	I/O	RD	I			PIO, I, PU, ST
U8	VDDIOP0	GPIO	PA28	I/O			TIOB1	I/O	RK	I/O			PIO, I, PU, ST
R9	VDDIOP0	GPIO	PA29	I/O			TIOB2	I/O	RF	I/O			PIO, I, PU, ST
R8	VDDIOP0	GPIO	PA30	I/O			TWD0	I/O	SPI1_NPCS3	O	E0_MDC	O	PIO, I, PU, ST
U9	VDDIOP0	GPIO	PA31	I/O			TWCK0	O	SPI1_NPCS2	O	E0_TXEN	O	PIO, I, PU, ST
D3	VDDANA	GPIO	PB0	I/O			E0_RX0	I	RTS2	O			PIO, I, PU, ST
D4	VDDANA	GPIO	PB1	I/O			E0_RX1	I	CTS2	I			PIO, I, PU, ST
D2	VDDANA	GPIO	PB2	I/O			E0_RXER	I	SCK2	I/O			PIO, I, PU, ST
E4	VDDANA	GPIO	PB3	I/O			E0_RXDV	I	SPI0_NPCS3	O			PIO, I, PU, ST
D1	VDDANA	GPIO_CLK	PB4	I/O			E0_TXCK	I	TWD2	I/O			PIO, I, PU, ST

Table 4-3. Pin Description BGA217 (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
E3	VDDANA	GPIO	PB5	I/O			E0_MDIO	I/O	TWCK2	O			PIO, I, PU, ST
B3	VDDANA	GPIO_ANA	PB6	I/O	AD7	I	E0_MDC	O					PIO, I, PU, ST
C2	VDDANA	GPIO_ANA	PB7	I/O	AD8	I	E0_TXEN	O					PIO, I, PU, ST
C5	VDDANA	GPIO_ANA	PB8	I/O	AD9	I	E0_TXER	O					PIO, I, PU, ST
C1	VDDANA	GPIO_ANA	PB9	I/O	AD10	I	E0_TX0	O	PCK1	O			PIO, I, PU, ST
B2	VDDANA	GPIO_ANA	PB10	I/O	AD11	I	E0_TX1	O	PCK0	O			PIO, I, PU, ST
A3	VDDANA	GPIO_ANA	PB11	I/O	AD0	I	E0_TX2	O	PWM0	O			PIO, I, PU, ST
B4	VDDANA	GPIO_ANA	PB12	I/O	AD1	I	E0_TX3	O	PWM1	O			PIO, I, PU, ST
A2	VDDANA	GPIO_ANA	PB13	I/O	AD2	I	E0_RX2	I	PWM2	O			PIO, I, PU, ST
C4	VDDANA	GPIO_ANA	PB14	I/O	AD3	I	E0_RX3	I	PWM3	O			PIO, I, PU, ST
C3	VDDANA	GPIO_ANA	PB15	I/O	AD4	I	E0_RXCK	I					PIO, I, PU, ST
A1	VDDANA	GPIO_ANA	PB16	I/O	AD5	I	E0_CRS	I		I			PIO, I, PU, ST
B1	VDDANA	GPIO_ANA	PB17	I/O	AD6	I	E0_COL	I		I			PIO, I, PU, ST
D5	VDDANA	GPIO	PB18	I/O			IRQ	I	ADTRG	I			PIO, I, PU, ST
E2	VDDIOP1	GPIO	PC0	I/O							TWD1	I/O	PIO, I, PU, ST
F4	VDDIOP1	GPIO	PC1	I/O							TWCK1	O	PIO, I, PU, ST
F3	VDDIOP1	GPIO	PC2	I/O							TIOA3	I/O	PIO, I, PU, ST
H2	VDDIOP1	GPIO	PC3	I/O							TIOB3	I/O	PIO, I, PU, ST
E1	VDDIOP1	GPIO	PC4	I/O							TCLK3	I	PIO, I, PU, ST
G4	VDDIOP1	GPIO	PC5	I/O							TIOA4	I/O	PIO, I, PU, ST
F2	VDDIOP1	GPIO	PC6	I/O							TIOB4	I/O	PIO, I, PU, ST
F1	VDDIOP1	GPIO	PC7	I/O							TCLK4	I	PIO, I, PU, ST
G1	VDDIOP1	GPIO	PC8	I/O							UTXD0	O	PIO, I, PU, ST
G3	VDDIOP1	GPIO	PC9	I/O							URXD0	I	PIO, I, PU, ST
G2	VDDIOP1	GPIO	PC10	I/O							PWM0	O	PIO, I, PU, ST
H3	VDDIOP1	GPIO	PC11	I/O							PWM1	O	PIO, I, PU, ST
J3	VDDIOP1	GPIO	PC12	I/O							TIOA5	I/O	PIO, I, PU, ST
L2	VDDIOP1	GPIO	PC13	I/O							TIOB5	I/O	PIO, I, PU, ST
H1	VDDIOP1	GPIO	PC14	I/O							TCLK5	I	PIO, I, PU, ST
J2	VDDIOP1	GPIO_CLK	PC15	I/O							PCK0	O	PIO, I, PU, ST
J1	VDDIOP1	GPIO	PC16	I/O					E1_RXER	I	UTXD1	O	PIO, I, PU, ST
L1	VDDIOP1	GPIO	PC17	I/O							URXD1	I	PIO, I, PU, ST
K2	VDDIOP1	GPIO	PC18	I/O					E1_TX0	O	PWM0	O	PIO, I, PU, ST
N3	VDDIOP1	GPIO	PC19	I/O					E1_TX1	O	PWM1	O	PIO, I, PU, ST
K1	VDDIOP1	GPIO	PC20	I/O					E1_RX0	I	PWM2	O	PIO, I, PU, ST
M3	VDDIOP1	GPIO	PC21	I/O					E1_RX1	I	PWM3	O	PIO, I, PU, ST
P3	VDDIOP1	GPIO	PC22	I/O					TXD3	O			PIO, I, PU, ST
J4	VDDIOP1	GPIO	PC23	I/O					RXD3	I			PIO, I, PU, ST

Table 4-3. Pin Description BGA217 (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
K3	VDDIOP1	GPIO	PC24	I/O					RTS3	O			PIO, I, PU, ST
M2	VDDIOP1	GPIO	PC25	I/O					CTS3	I			PIO, I, PU, ST
P2	VDDIOP1	GPIO	PC26	I/O					SCK3	I/O			PIO, I, PU, ST
M1	VDDIOP1	GPIO	PC27	I/O					E1_TXEN	O	RTS1	O	PIO, I, PU, ST
K4	VDDIOP1	GPIO	PC28	I/O					E1_CRSDV	I	CTS1	I	PIO, I, PU, ST
N1	VDDIOP1	GPIO_CLK	PC29	I/O					E1_TXCK	I	SCK1	I/O	PIO, I, PU, ST
R2	VDDIOP1	GPIO_CLK2	PC30	I/O					E1_MDC	O			PIO, I, PU, ST
N2	VDDIOP1	GPIO	PC31	I/O			FIQ	I	E1_MDIO	I/O	PCK1	O	PIO, I, PU, ST
P13	VDDNF	EBI	PD0	I/O			NANDOE	O					PIO, I, PU
R14	VDDNF	EBI	PD1	I/O			NANDWE	O					PIO, I, PU
R13	VDDNF	EBI	PD2	I/O			A21/NANDALE	O					A21,O, PD
P15	VDDNF	EBI	PD3	I/O			A22/NANDCLE	O					A22,O, PD
P12	VDDNF	EBI	PD4	I/O			NCS3	O					PIO, I, PU
P14	VDDNF	EBI	PD5	I/O			NWAIT	I					PIO, I, PU
N14	VDDNF	EBI	PD6	I/O			D16	O					PIO, I, PU
R15	VDDNF	EBI	PD7	I/O			D17	O					PIO, I, PU
M14	VDDNF	EBI	PD8	I/O			D18	O					PIO, I, PU
N16	VDDNF	EBI	PD9	I/O			D19	O					PIO, I, PU
N17	VDDNF	EBI	PD10	I/O			D20	O					PIO, I, PU
N15	VDDNF	EBI	PD11	I/O			D21	O					PIO, I, PU
K15	VDDNF	EBI	PD12	I/O			D22	O					PIO, I, PU
M15	VDDNF	EBI	PD13	I/O			D23	O					PIO, I, PU
L14	VDDNF	EBI	PD14	I/O			D24	O					PIO, I, PU
M16	VDDNF	EBI	PD15	I/O			D25	O	A20	O			A20, O, PD
L16	VDDNF	EBI	PD16	I/O			D26	O	A23	O			A23, O, PD
L15	VDDNF	EBI	PD17	I/O			D27	O	A24	O			A24, O, PD
K17	VDDNF	EBI	PD18	I/O			D28	O	A25	O			A25, O, PD
J17	VDDNF	EBI	PD19	I/O			D29	O	NCS2	O			PIO, I, PU
K16	VDDNF	EBI	PD20	I/O			D30	O	NCS4	O			PIO, I, PU
J16	VDDNF	EBI	PD21	I/O			D31	O	NCS5	O			PIO, I, PU
D10 D13 F14	VDDIOM	POWER	VDDIOM	I									I
J14 K14	VDDNF	POWER	VDDNF	I									I
H9 H10 J9 J10	GNDIOM	GND	GNDIOM	I									I
P7	VDDIOP0	POWER	VDDIOP0	I									I

Table 4-3. Pin Description BGA217 (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
H4	VDDIOP1	POWER	VDDIOP1	I									I
M4 P6	GNDIOP	GND	GNDIOP	I									I
B5	VDDBU	POWER	VDDBU	I									I
B6	GNDBU	GND	GNDBU	I									I
C6	VDDANA	POWER	VDDANA	I									I
D6	GNDANA	GND	GNDANA	I									I
R12	VDDPLLA	POWER	VDDPLLA	I									I
T13	VDDOSC	POWER	VDDOSC	I									I
U13	GNDOSC	GND	GNDOSC	I									I
H14 K8 K9	VDDCORE	POWER	VDDCORE	I									I
H8 J8 K10	GNDCORE	GND	GNDCORE	I									I
U16	VDDUTMII	POWER	VDDUTMII	I									I
T17	VDDUTMIC	POWER	VDDUTMIC	I									I
T16	GNDUTMI	GND	GNDUTMI	I									I
D14	VDDIOM	EBI	D0	I/O									O, PD
D15	VDDIOM	EBI	D1	I/O									O, PD
A16	VDDIOM	EBI	D2	I/O									O, PD
B16	VDDIOM	EBI	D3	I/O									O, PD
A17	VDDIOM	EBI	D4	I/O									O, PD
B15	VDDIOM	EBI	D5	I/O									O, PD
C14	VDDIOM	EBI	D6	I/O									O, PD
B14	VDDIOM	EBI	D7	I/O									O, PD
A15	VDDIOM	EBI	D8	I/O									O, PD
C15	VDDIOM	EBI	D9	I/O									O, PD
D12	VDDIOM	EBI	D10	I/O									O, PD
C13	VDDIOM	EBI	D11	I/O									O, PD
A14	VDDIOM	EBI	D12	I/O									O, PD
B13	VDDIOM	EBI	D13	I/O									O, PD
A13	VDDIOM	EBI	D14	I/O									O, PD
C12	VDDIOM	EBI	D15	I/O									O, PD
J15	VDDIOM	EBI_O	A0	O	NBS0	O							O, PD
H16	VDDIOM	EBI_O	A1	O	NBS2/DQM/ NWR2	O							O, PD
H15	VDDIOM	EBI_O	A2	O									O, PD
H17	VDDIOM	EBI_O	A3	O									O, PD

Table 4-3. Pin Description BGA217 (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
G17	VDDIOM	EBI_O	A4	O									O, PD
G16	VDDIOM	EBI_O	A5	O									O, PD
F17	VDDIOM	EBI_O	A6	O									O, PD
E17	VDDIOM	EBI_O	A7	O									O, PD
F16	VDDIOM	EBI_O	A8	O									O, PD
G15	VDDIOM	EBI_O	A9	O									O, PD
G14	VDDIOM	EBI_O	A10	O									O, PD
F15	VDDIOM	EBI_O	A11	O									O, PD
D17	VDDIOM	EBI_O	A12	O									O, PD
C17	VDDIOM	EBI_O	A13	O									O, PD
E16	VDDIOM	EBI_O	A14	O									O, PD
D16	VDDIOM	EBI_O	A15	O									O, PD
C16	VDDIOM	EBI_O	A16	O	BA0	O							O, PD
B17	VDDIOM	EBI_O	A17	O	BA1	O							O, PD
E15	VDDIOM	EBI_O	A18	O	BA2	O							O, PD
E14	VDDIOM	EBI_O	A19	O									O, PD
B9	VDDIOM	EBI_O	NCS0	O									O, PU
B8	VDDIOM	EBI_O	NCS1	O	SDCS	O							O, PU
D9	VDDIOM	EBI_O	NRD	O									O, PU
C9	VDDIOM	EBI_O	NWR0	O	NWRE	O							O, PU
C7	VDDIOM	EBI_O	NWR1	O	NBS1	O							O, PU
A8	VDDIOM	EBI_O	NWR3	O	NBS3/DQM3	O							O, PU
D11	VDDIOM	EBI_CLK	SDCK	O									O
C11	VDDIOM	EBI_CLK	#SDCK	O									O
B12	VDDIOM	EBI_O	SDCKE	O									O, PU
B11	VDDIOM	EBI_O	RAS	O									O, PU
C10	VDDIOM	EBI_O	CAS	O									O, PU
A12	VDDIOM	EBI_O	SDWE	O									O, PU
C8	VDDIOM	EBI_O	SDA10	O									O, PU
A10	VDDIOM	EBI_O	DQM0	O									O, PU
B10	VDDIOM	EBI_O	DQM1	O									O, PU
A11	VDDIOM	EBI	DQS0	I/O									O, PD
A9	VDDIOM	EBI	DQS1	I/O									O, PD
A4	VDDANA	POWER	ADVREF	I									I
U17	VDDUTMIC	VBG	VBG	I									I
T14	VDDUTMII	USBFS	HFSDPA	I/O	DFSDP	I/O							O, PD
T15	VDDUTMII	USBFS	HFSDMA	I/O	DFSDM	I/O							O, PD
U14	VDDUTMII	USBHS	HHSDPA	I/O	DHSDP	I/O							O, PD

Table 4-3. Pin Description BGA217 (Continued)

Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, ST
U15	VDDUTMII	USBHS	HHSDMA	I/O	DHSDM	I/O							O, PD
R16	VDDUTMII	USBFS	HFSDPB	I/O									O, PD
P16	VDDUTMII	USBFS	HFSDMB	I/O									O, PD
R17	VDDUTMII	USBHS	HHSDPB	I/O									O, PD
P17	VDDUTMII	USBHS	HHSDMB	I/O									O, PD
L17	VDDUTMII	USBFS	HFSDPC	I/O									O, PD
M17	VDDUTMII	USBFS	HFSDMC	I/O									O, PD
R11	VDDIOP0	DIB	DIBN	I/O									O, PU
P11	VDDIOP0	DIB	DIBP	I/O									O, PU
A7	VDDBU	SYSC	WKUP	I									I, ST
D8	VDDBU	SYSC	SHDN	O									O
P9	VDDIOP0	RSTJTAG	BMS	I									I, PD, ST
D7	VDDBU	SYSC	JTAGSEL	I									I, PD
B7	VDDBU	SYSC	TST	I									I, PD, ST
U10	VDDIOP0	RSTJTAG	TCK	I									I, ST
T9	VDDIOP0	RSTJTAG	TDI	I									I, ST
T10	VDDIOP0	RSTJTAG	TDO	O									O
U11	VDDIOP0	RSTJTAG	TMS	I									I, ST
R10	VDDIOP0	RSTJTAG	RTCK	O									O
P10	VDDIOP0	RSTJTAG	NRST	I/O									I, PU, ST
T11	VDDIOP0	RSTJTAG	NTRST	I									I, PU, ST
A6	VDDBU	CLOCK	XIN32	I									I
A5	VDDBU	CLOCK	XOUT32	O									O
T12	VDDOSC	CLOCK	XIN	I									I
U12	VDDOSC	CLOCK	XOUT	O									O

5. Power Considerations

5.1 Power Supplies

The SAM9X25 has several types of power supply pins.

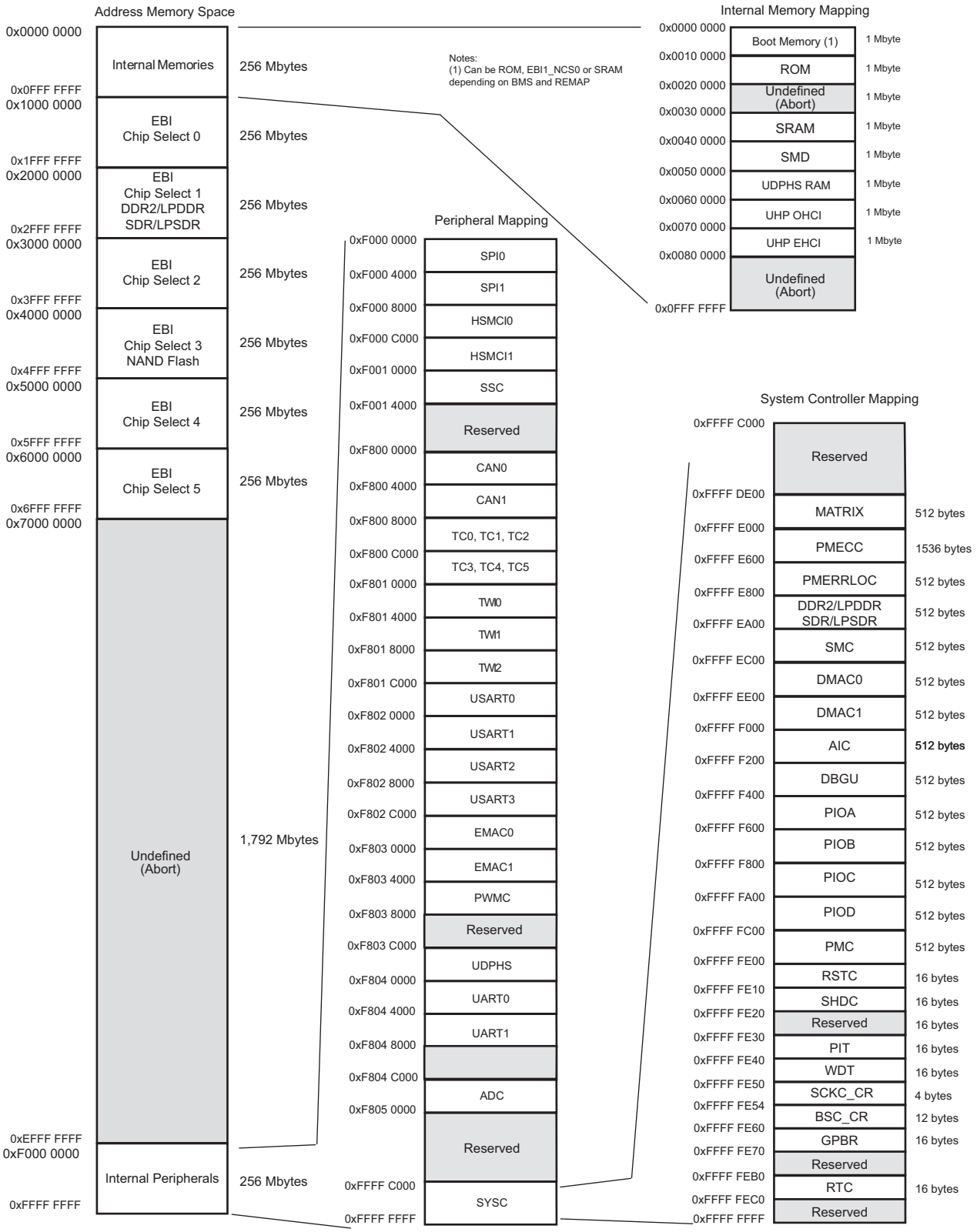
Table 5-1. SAM9X25 Power Supplies

Name	Voltage Range, nominal	Powers	Associated Ground
VDDCORE	0.9–1.1V, 1.0V	ARM core, internal memories, internal peripherals and part of the system controller.	GNDCORE
VDDIOM	1.65–1.95V, 1.8V 3.0–3.6V, 3.3V	External Memory Interface I/O lines	GNDIOM
VDDNF	1.65–1.95V, 1.8V 3.0–3.6V, 3.3V	NAND Flash I/O and control, D16–D31 and multiplexed SMC lines	GNDIOM
VDDIOP0	1.65–3.6V	Part of Peripheral I/O lines(1)	GNDIOP
VDDIOP1	1.65–3.6V	Part of Peripheral I/O lines (1)	GNDIOP
VDDBU	1.65–3.6V	Slow Clock oscillator, the internal 32 kHz RC oscillator and backup part of the System Controller	GNDBU
VDDUTMIC	0.9–1.1V, 1.0V	USB transceiver core logic	GNDUTMI
VDDUTMII	3.0–3.6V, 3.3V	USB transceiver interface	GNDUTMI
VDDPLLA	0.9–1.1V, 1.0V	PLLA and PLLUTMI cells	GNDOSC
VDDOSC	1.65–3.6V	Main Oscillator cells	GNDOSC
VDDANA	3.0–3.6V, 3.3V	Analog-to-Digital Converter	GNDANA

Note: 1. Refer to Table 4-2 for more details.

6. Memories

Figure 6-1. SAM9X25 Memory Mapping



6.1 Memory Mapping

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

Decoding breaks up the 4 Gbytes of address space into 16 banks of 256 Mbytes. Banks 1 to 6 are directed to the EBI that associates these banks to the external chip selects, EBI_NCS0 to EBI_NCS5. Bank 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1 Mbyte 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.

6.2 Embedded Memories

6.2.1 Internal SRAM

The SAM9X25 embeds a total of 32 Kbytes of high-speed SRAM.

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

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

6.2.2 Internal ROM

The SAM9X25 embeds an Internal ROM, which contains the SAM-BA® program.

At any time, the ROM is mapped at address 0x0010 0000. It is also accessible at address 0x0 (BMS = 1) after the reset and before the Remap Command.

6.3 External Memories

6.3.1 External Bus Interface

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - DDR2/SDRAM Controller
  - MLC NAND Flash ECC Controller
- Additional logic for NAND Flash and CompactFlash®
- 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, NCS1, NCS2, NCS3, NCS4, NCS5
  - DDR2/SDRAM Controller (SDCS) or Static Memory Controller on NCS1
  - Optional NAND Flash support on NCS3

6.3.2 Static Memory Controller

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

6.3.3 DDR2SDR Controller

- Supports 4-bank and 8-bank DDR2, LPDDR, SDR and LPSDR
- Numerous Configurations Supported
  - 2K, 4K, 8K, 16K Row Address Memory Parts
  - SDRAM with 8 Internal Banks
  - SDR-SDRAM with 32-bit Data Path
  - DDR2/LPDDR with 16-bit Data Path
  - One Chip Select for SDRAM Device (256 Mbyte Address Space)
- Programming Facilities
  - Multibank Ping-pong Access (Up to 8 Banks Opened at Same Time = Reduces Average Latency of Transactions)
  - Timing Parameters Specified by Software
  - Automatic Refresh Operation, Refresh Rate is Programmable
  - Automatic Update of DS, TCR and PASR Parameters (LPSDR)
- Energy-saving Capabilities
  - Self-refresh, Power-down and Deep Power Modes Supported
- SDRAM Power-up Initialization by Software
- CAS Latency of 2, 3 Supported
- Auto Precharge Command Not Used
- SDR-SDRAM with 16-bit Datapath and Eight Columns Not Supported
  - Clock Frequency Change in Precharge Power-down Mode Not Supported

7. System Controller

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

The System Controller User Interface also embeds the registers that configure the Matrix and a set of registers for the chip configuration. The chip configuration registers configure the EBI chip select assignment and voltage range for external memories.

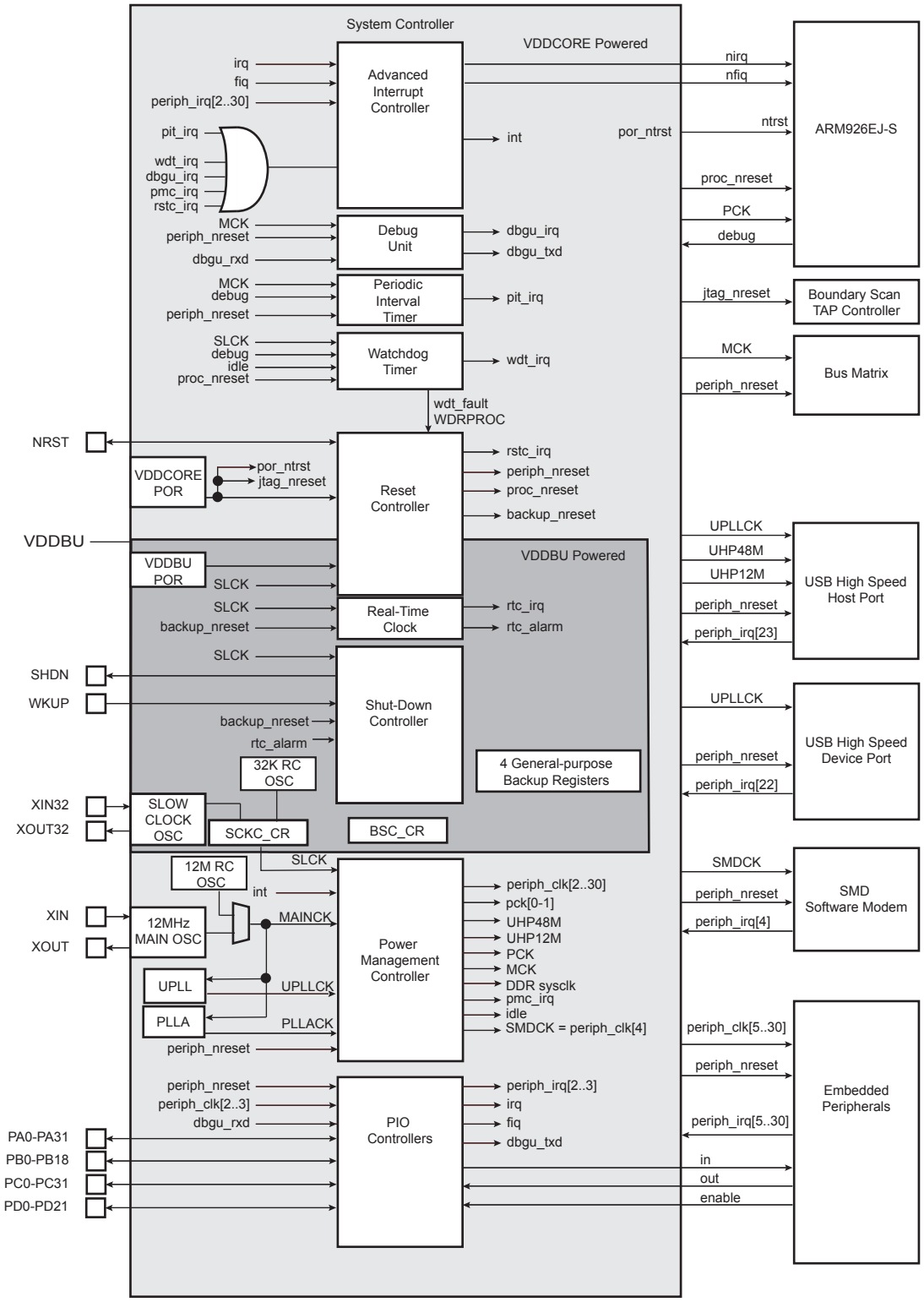
The System Controller's peripherals are all mapped within the highest 16 Kbytes of address space, between addresses 0xFFFF_C000 and 0xFFFF_FFFF.

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

Figure 7-1 on page 23 shows the System Controller block diagram.

Figure 6-1 on page 19 shows the mapping of the User Interface of the System Controller peripherals.

Figure 7-1. SAM9X25 System Controller Block Diagram



7.1 Chip Identification

- Chip ID: 0x819A_05A1
- Chip ID Extension: 4
- JTAG ID: 0x05B2_F03F
- ARM926 TAP ID: 0x0792_603F

7.2 Backup Section

The SAM9X25 features a Backup Section that embeds:

- RC Oscillator
- Slow Clock Oscillator
- Real Time Counter (RTC)
- Shutdown Controller
- 4 Backup Registers
- Slow Clock Controller Configuration Register (SCKC_CR)
- Boot Sequence Configuration Register (BSC_CR)
- A part of the Reset Controller (RSTC)

This section is powered by the VDDBU rail.

8. Peripherals

8.1 Peripheral Mapping

As shown in Figure 6-1, the Peripherals are mapped in the upper 256 Mbytes of the address space between the addresses 0xF000 _000 and 0xFFFF_C000.

Each User Peripheral is allocated 16 Kbytes of address space.

8.2 Peripheral Identifiers

Table 8-1 defines the Peripheral Identifiers of the SAM9X25. 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 8-1. Peripheral Identifiers

Instance ID	Instance Name	Instance Description	External Interrupt	Wired-OR interrupt
0	AIC	Advanced Interrupt Controller	FIQ	
1	SYS	System Controller Interrupt		DBGU, PMC, SYSC, PMECC, PMERRLOC, RTSC, SHDC, PIT WDT, RTC
2	PIOA, PIOB	Parallel I/O Controller A and B		
3	PIOC, PIOD	Parallel I/O Controller C and D		
4	SMD	SMD Soft Modem		
5	USART0	USART 0		
6	USART1	USART 1		
7	USART2	USART 2		
8	USART3	USART 3		
9	TWI0	Two-Wire Interface 0		
10	TWI1	Two-Wire Interface 1		
11	TWI2	Two-Wire Interface 2		
12	HSMCI0	High Speed Multimedia Card Interface 0		
13	SPI0	Serial Peripheral Interface 0		
14	SPI1	Serial Peripheral Interface 1		
15	UART0	UART 0		
16	UART1	UART 1		
17	TC0,TC1	Timer Counter 0,1,2,3,4,5		
18	PWM	Pulse Width Modulation Controller		
19	ADC	ADC Controller		
20	DMAC0	DMA Controller 0		
21	DMAC1	DMA Controller 1		
22	UHPHS	USB Host High Speed		
23	UDPHS	USB Device High Speed		
24	EMAC0	Ethernet MAC0		
25	–	Reserved		
26	HSMCI1	High Speed Multimedia Card Interface 1		
27	EMAC1	Ethernet MAC1		
28	SSC	Synchronous Serial Controller		
29	CAN0	CAN Controller 0		
30	CAN1	CAN Controller 1		
31	AIC	Advanced Interrupt Controller	IRQ	

8.3 Peripheral Signal Multiplexing on I/O Lines

The SAM9X25 features four PIO Controllers (PIOA, PIOB, PIOC and PIOD) which multiplex the I/O lines of the peripheral set.

Each PIO Controller controls 32 lines, 19 lines, 32 lines and 22 lines respectively for PIOA, PIOB, PIOC and PIOD. Each line can be assigned to one of three peripheral functions, A, B or C. Refer to Section 4. “Package and Pinout”, Table 4-3 to see the PIO assignments.

9. ARM926EJ-S™

9.1 Description

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

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

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

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

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

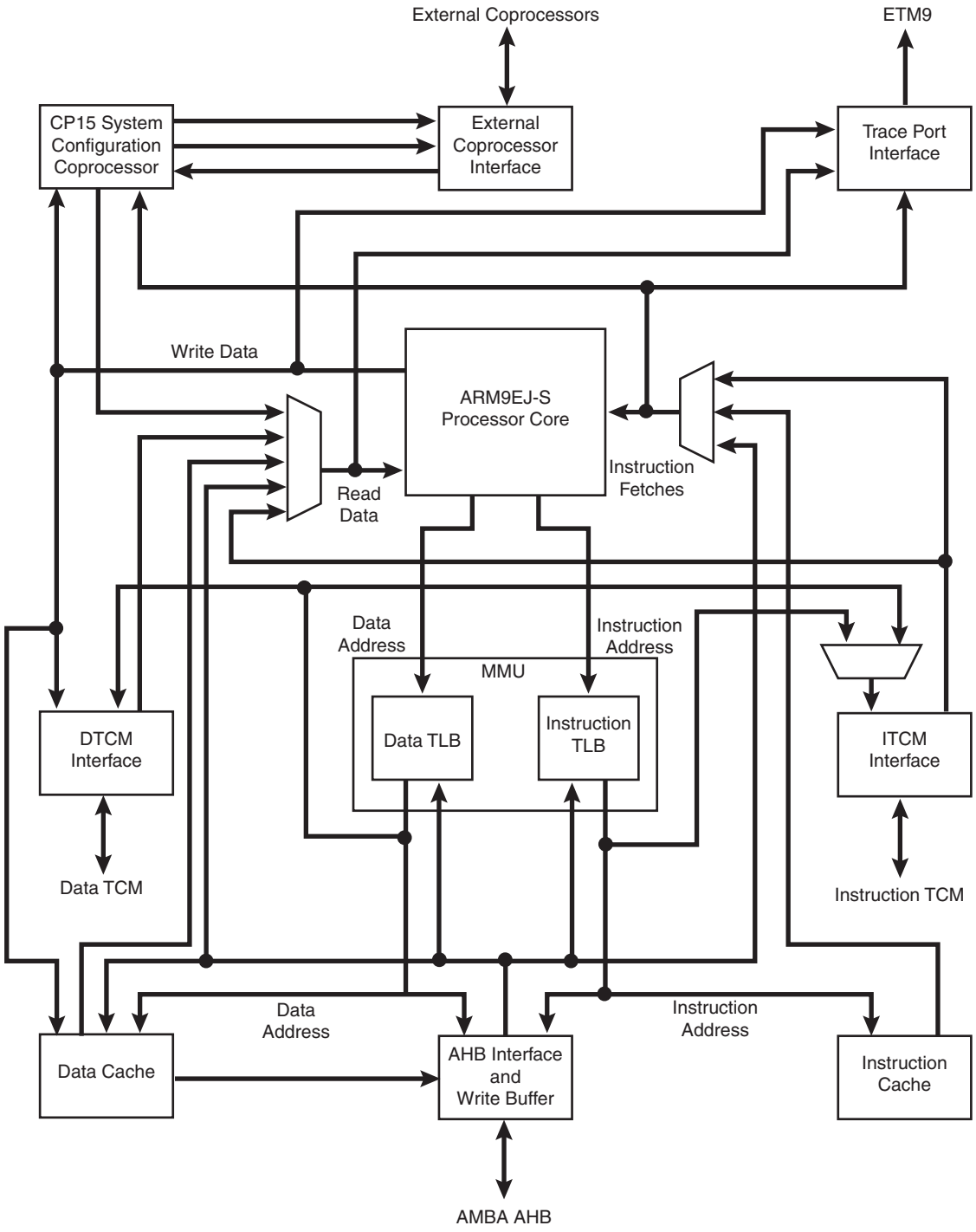
9.2 Embedded Characteristics

- ARM9EJ-S™ Based on ARM® Architecture v5TEJ with Jazelle Technology
- Three Instruction Sets
  - ARM® High-performance 32-bit Instruction Set
  - Thumb® High Code Density 16-bit Instruction Set
  - Jazelle® 8-bit Instruction Set
- 5-Stage Pipeline Architecture when Jazelle is not Used
  - Fetch (F)
  - Decode (D)
  - Execute (E)
  - Memory (M)
  - Writeback (W)
- 6-Stage Pipeline when Jazelle is Used
  - Fetch
  - Jazelle/Decode (Two Cycles)
  - Execute
  - Memory
  - Writeback
- ICache and DCache
  - Virtually-addressed 4-way Set Associative Caches
  - 8 Words per Line
  - Critical-word First Cache Refilling
  - Write-though and Write-back Operation for DCache Only
  - Pseudo-random or Round-robin Replacement
  - Cache Lockdown Registers
  - Cache Maintenance
- Write Buffer
  - 16-word Data Buffer
  - 4-address Address Buffer
  - Software Control Drain
- DCache Write-back Buffer
  - 8 Data Word Entries
  - One Address Entry
  - Software Control Drain
- Memory Management Unit (MMU)
  - Access Permission for Sections
  - Access Permission for Large Pages and Small Pages
  - 16 Embedded Domains
  - 64 Entry Instruction TLB and 64 Entry Data TLB
- Memory Access
  - 8-bit, 16-bit, and 32-bit Data Types
  - Separate AMBA AHB Buses for Both the 32-bit Data Interface and the 32-bit Instructions Interface
- Bus Interface Unit
  - Arbitrates and Schedules AHB Requests
  - Enables Multi-layer AHB to be Implemented

- Increases Overall Bus Bandwidth
- Makes System Architecture Mode Flexible

9.3 Block Diagram

Figure 9-1. ARM926EJ-S Internal Functional Block Diagram



9.4 ARM9EJ-S Processor

9.4.1 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.

9.4.2 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.

9.4.3 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.

9.4.4 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.

9.4.5 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.

9.4.6 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.

9.4.7 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 9-1 shows all the registers in all modes.

Table 9-1. 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

For more details, refer to ARM Software Development Kit.

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

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

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

9.4.7.1 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 9-2. Status Register Format

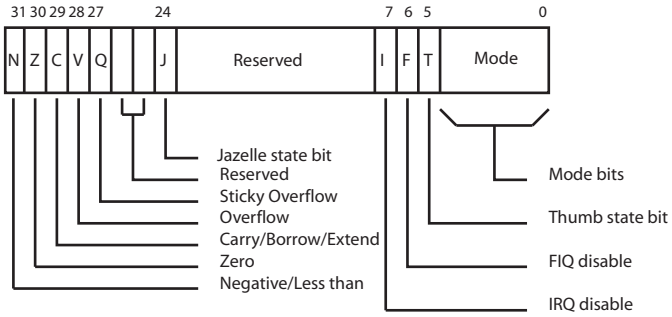


Figure 9-2 shows the status register format, where:

- 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

9.4.7.2 Exceptions

Exception Types and Priorities

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

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

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

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

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

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

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

Exception Modes and Handling

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

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

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

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

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

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

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

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

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

9.4.8 ARM Instruction Set Overview

The ARM instruction set is divided into:

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

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

For further details, see the ARM Technical Reference Manual.

Table 9-2 gives the ARM instruction mnemonic list.

Table 9-2. 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

Table 9-2. ARM Instruction Mnemonic List (Continued)

Mnemonic	Operation
BX	Branch and Exchange
LDR	Load Word
LDRSH	Load Signed Halfword
LDRSB	Load Signed Byte
LDRH	Load Half Word
LDRB	Load Byte
LDRBT	Load Register Byte with Translation
LDRT	Load Register with Translation
LDM	Load Multiple
SWP	Swap Word
MCR	Move To Coprocessor
LDC	Load To Coprocessor
CDP	Coprocessor Data Processing

Mnemonic	Operation
SWI	Software Interrupt
STR	Store Word
STRH	Store Half Word
STRB	Store Byte
STRBT	Store Register Byte with Translation
STRT	Store Register with Translation
STM	Store Multiple
SWPB	Swap Byte
MRC	Move From Coprocessor
STC	Store From Coprocessor

9.4.9 New ARM Instruction Set

Table 9-3. 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.

9.4.10 Thumb Instruction Set Overview

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

The Thumb instruction set is divided into:

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

For further details, see the ARM Technical Reference Manual.

Table 9-4 gives the Thumb instruction mnemonic list.

Table 9-4. Thumb Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply	BLX	Branch, Link, and Exchange
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Register to stack	POP	Pop Register from stack
BCC	Conditional Branch	BKPT	Breakpoint

9.5 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 9-5.

Table 9-5. 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
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.

9.5.1 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.

9.6 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 9-6. 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

9.6.1 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).

9.6.2 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.

9.6.3 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.

9.6.4 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.

9.7 Caches and Write Buffer

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

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

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

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

9.7.1 Instruction Cache (ICache)

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

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

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

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

9.7.2 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.

9.7.2.1 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).

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.

9.7.2.2 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.

9.8 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.

9.8.1 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 9-7 provides an overview of the supported transfers and different kinds of transactions they are used for.

Table 9-7. 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

9.8.2 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.

9.8.3 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.

10. Debug and Test

10.1 Description

The SAM9X25 features a number of complementary debug and test capabilities. A common JTAG/ICE (In-Circuit Emulator) port is used for standard debugging functions, such as downloading code and single-stepping through programs. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

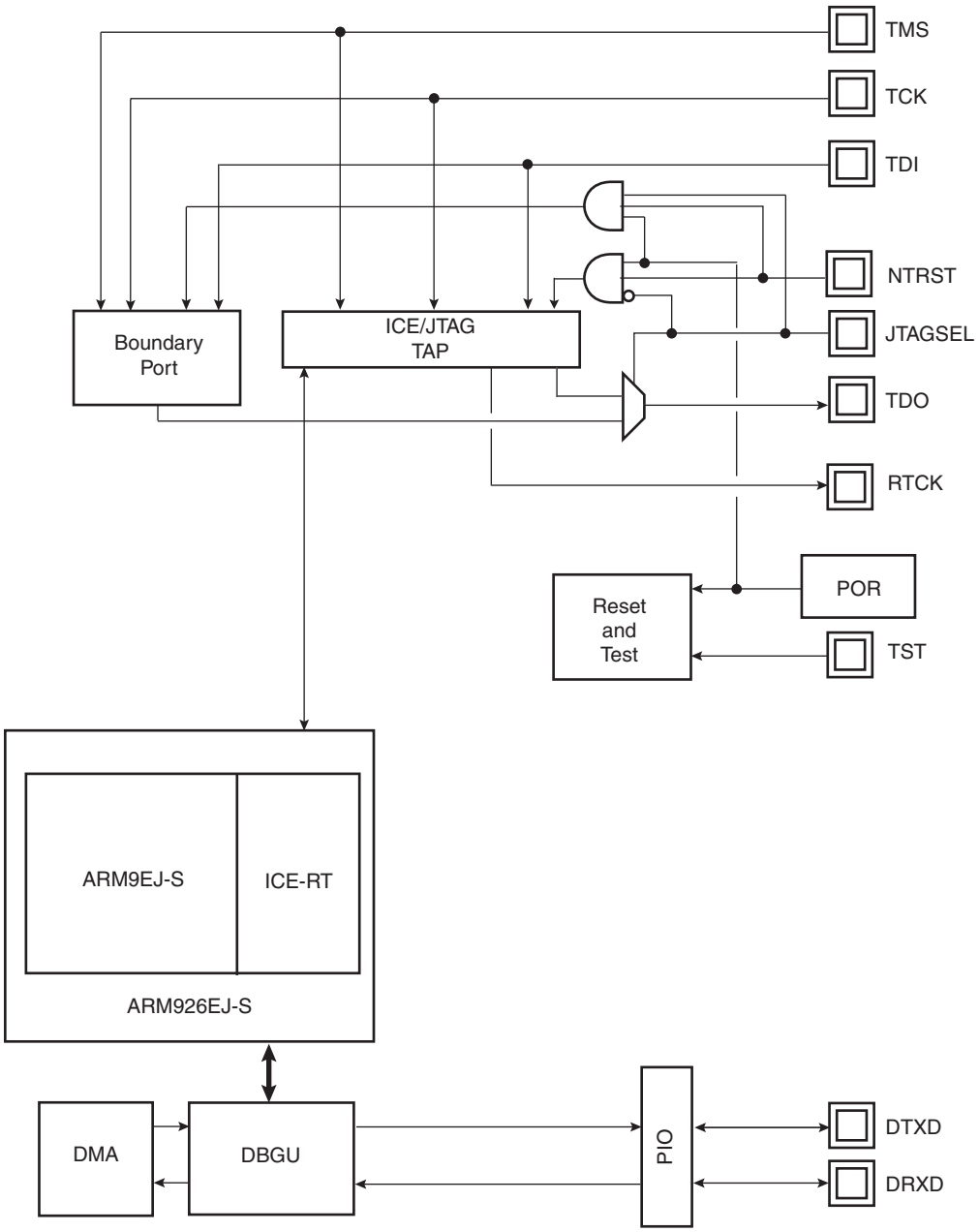
A set of dedicated debug and test input/output pins gives direct access to these capabilities from a PC-based test environment.

10.2 Embedded Characteristics

- ARM926 Real-time In-circuit Emulator
  - Two real-time Watchpoint Units
  - Two Independent Registers: Debug Control Register and Debug Status Register
  - Test Access Port Accessible through JTAG Protocol
  - Debug Communications Channel
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel Interrupt Handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins

10.3 Block Diagram

Figure 10-1. Debug and Test Block Diagram



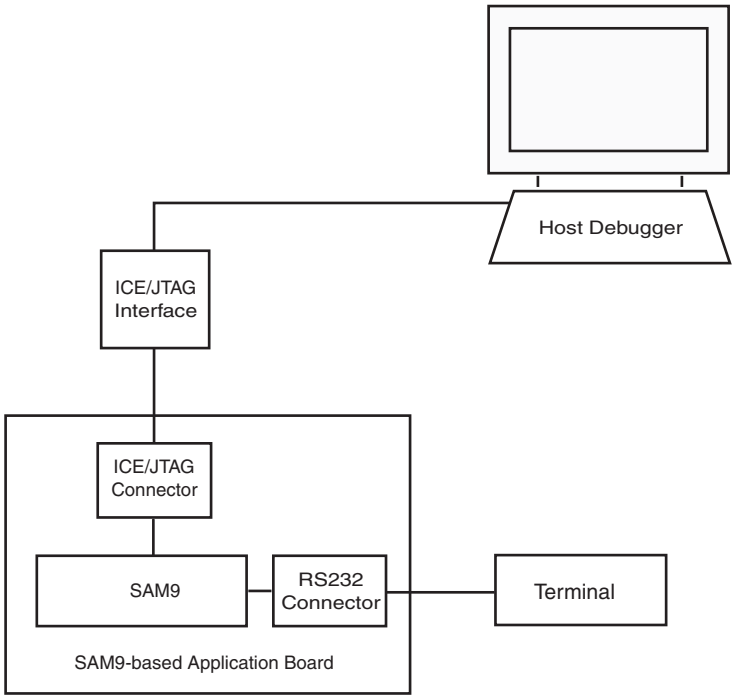
TAP: Test Access Port

10.4 Application Examples

10.4.1 Debug Environment

Figure 10-2 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

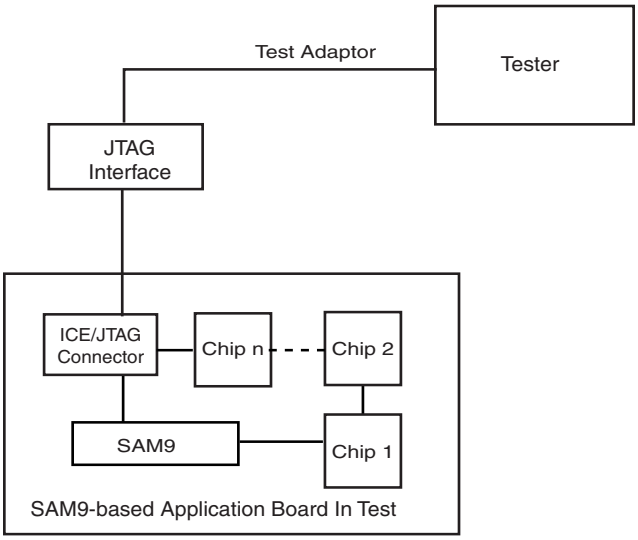
Figure 10-2. Application Debug and Trace Environment Example



10.4.2 Test Environment

Figure 10-3 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 10-3. Application Test Environment Example



10.5 Debug and Test Pin Description

Table 10-1. Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
NTRST	Test Reset Signal	Input	Low
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	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

10.6 Functional Description

10.6.1 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.

10.6.2 EmbeddedICE™

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

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

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

For further details on the EmbeddedICE-RT, see the ARM document:

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

10.6.3 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.

10.6.4 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 device Debug Unit Chip ID value is 0x819A_05A1 on 32-bit width.

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

10.6.5 IEEE 1149.1 JTAG Boundary Scan

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

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

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

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

10.6.6 JTAG ID Code Register

Access: Read-only

31	30	29	28	27	26	25	24
VERSION				PART NUMBER			

23	22	21	20	19	18	17	16
PART NUMBER							

15	14	13	12	11	10	9	8
PART NUMBER				MANUFACTURER IDENTITY			

7	6	5	4	3	2	1	0
MANUFACTURER IDENTITY							1

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

Set to 0x0.

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

Product part Number is 0x5B2F

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

Set to 0x01F.

Bit[0] required by IEEE Std. 1149.1.

Set to 0x1.

JTAG ID Code value is 0x05B2_F03F.

11. Boot Strategies

The system always boots at address 0x0. To ensure maximum boot possibilities, the memory layout can be changed thanks to the BMS pin. This allows the user to layout the ROM or an external memory to 0x0. The sampling of the BMS pin is done at reset.

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

In this boot mode, the chip starts with its default parameters (all registers in their reset state), including as follows:

- The main clock is the on-chip 12 MHz RC oscillator
- The Static Memory Controller is configured with its default parameters

The user software in the external memory performs a complete configuration:

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

If BMS is detected at 1, the boot memory is the embedded ROM and the Boot Program described below is executed. (Section 11.1 “ROM Code”) .

11.1 ROM Code

The ROM Code is a boot program contained in the embedded ROM. It is also called “First level bootloader”.

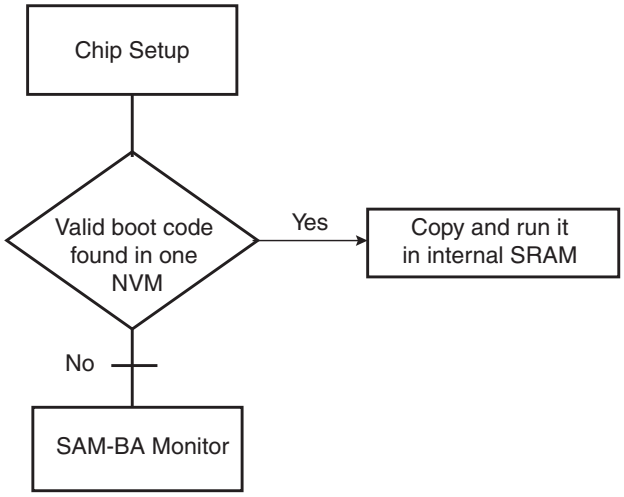
The ROM Code performs several steps:

- Basic chip initialization: XTal or external clock frequency detection
- Attempt to retrieve a valid code from external non-volatile memories (NVM)
- Execution of a monitor called SAM-BA Monitor, in case no valid application has been found on any NVM

11.2 Flow Diagram

The ROM Code implements the algorithm shown below in Figure 11-1.

Figure 11-1. ROM Code Algorithm Flow Diagram



11.3 Chip Setup

At boot start-up, the processor clock (PCK) and the master clock (MCK) source is the 12 MHz Fast RC Oscillator.

Initialization follows the steps described below:

1. **Stack setup** for ARM supervisor mode.
2. **Main Oscillator Detection:** the Main Clock is switched to the 32 kHz RC oscillator to allow external clock frequency to be measured. Then the Main Oscillator is enabled and set in bypass mode. If the MOSCSELS bit rises, an external clock is connected, and the next step is Main Clock Selection (3)**.** If not, the bypass mode is cleared to attempt external quartz detection. This detection is successful when the MOSCXTS and MOSCSELS bits rise, else the 12 MHz Fast RC internal oscillator is used as the Main Clock.
3. **Main Clock Selection**: the Master Clock source is switched from the Slow Clock to the Main Oscillator without prescaler. The PMC Status Register is polled to wait for MCK Ready. PCK and MCK are now the Main Clock.
4. **C variable initialization:** non zero-initialized data is initialized in the RAM (copy from ROM to RAM). Zero-initialized data is set to 0 in the RAM.
5. **PLLA initialization**: PLLA is configured to get a PCK at 96 MHz and an MCK at 48 MHz. If an external clock or crystal frequency running at 12 MHz is found, then the PLLA is configured to allow communication on the USB link for the SAM-BA Monitor; else the Main Clock is switched to the internal 12 MHz Fast RC, but USB will not be activated

.

Table 11-1. External Clock and Crystal Frequencies allowed for Boot Sequence (in MHz)

Boot Sequence	≤ 4	12	≥ 28
Boot on External Memories	Yes	Yes	Yes
SAM-BA Monitor through DBGU	Yes	Yes	Yes
SAM-BA Monitor through USB	No	Yes	No

Note that if the clock frequency is provided not at 12 MHz but between 4 and 28 MHz, it is considered by the ROM Code as the 12 MHz clock frequency, and the PLL settings are configured accordingly.

11.4 NVM Boot

11.4.1 NVM Boot Sequence

The boot sequence on external memory devices can be controlled using the Boot Sequence Configuration Register (BSC_CR). The 3 LSBs of the BSC_CR are available to control the sequence. See the “Boot Sequence Controller (BSC)” section for more details.

The user can then choose to bypass some steps shown in Figure 11-2 “NVM Bootloader Sequence Diagram” according to the BSC_CR Value.

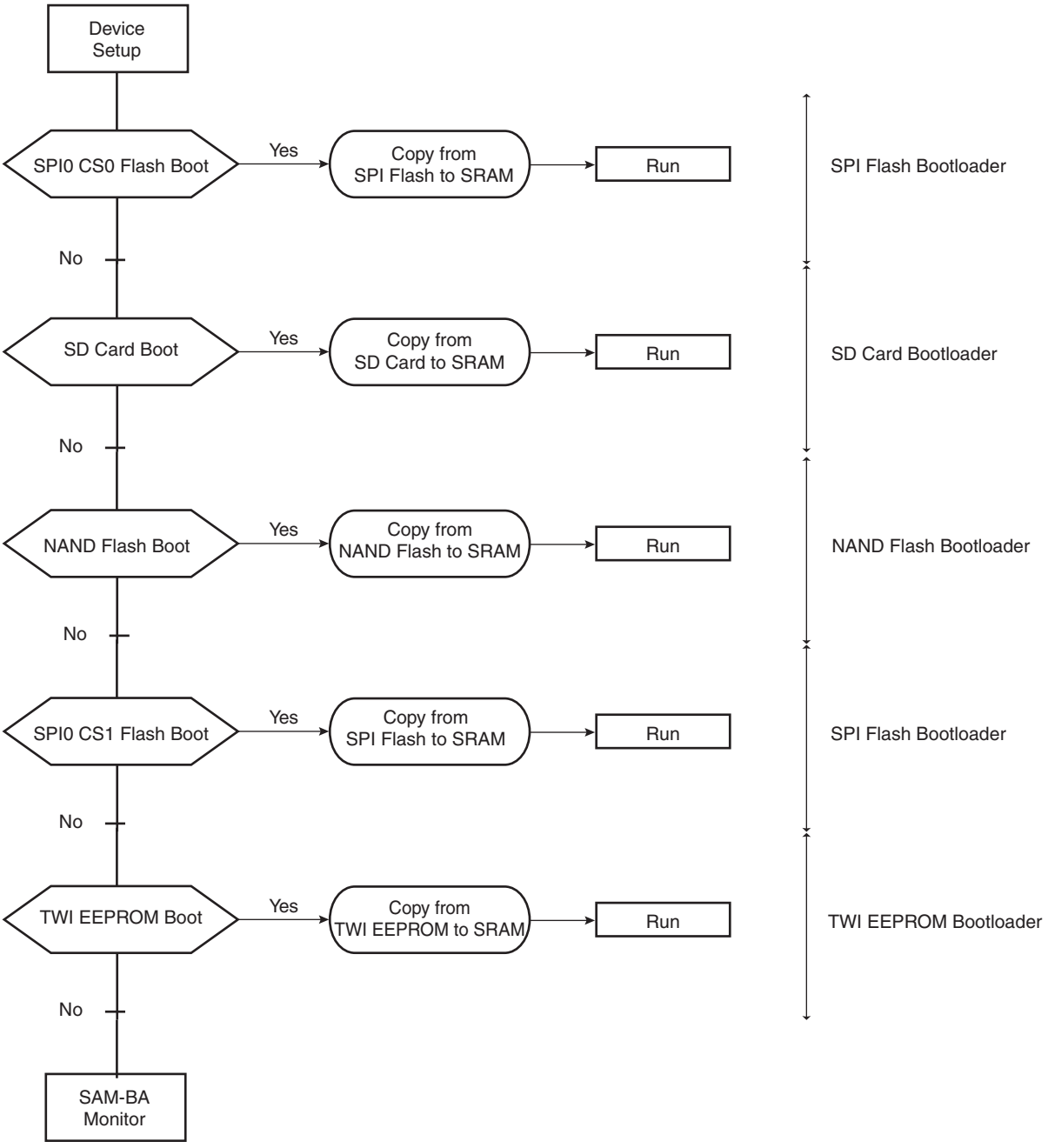
Table 11-2. Boot Sequence Configuration Register Values

BOOT Value	SPI0 NPCS0	SDCard	NAND Flash	SPI0 NPCS1	TWI EEPROM	SAM-BA Monitor
0	Y	Y	Y	Y	Y	Y
1	Y	-	Y	Y	Y	Y
2	Y	-	-	Y	Y	Y
3	Y	-	-	Y	Y	Y
4	Y	-	-	-	Y	Y

Table 11-2. Boot Sequence Configuration Register Values

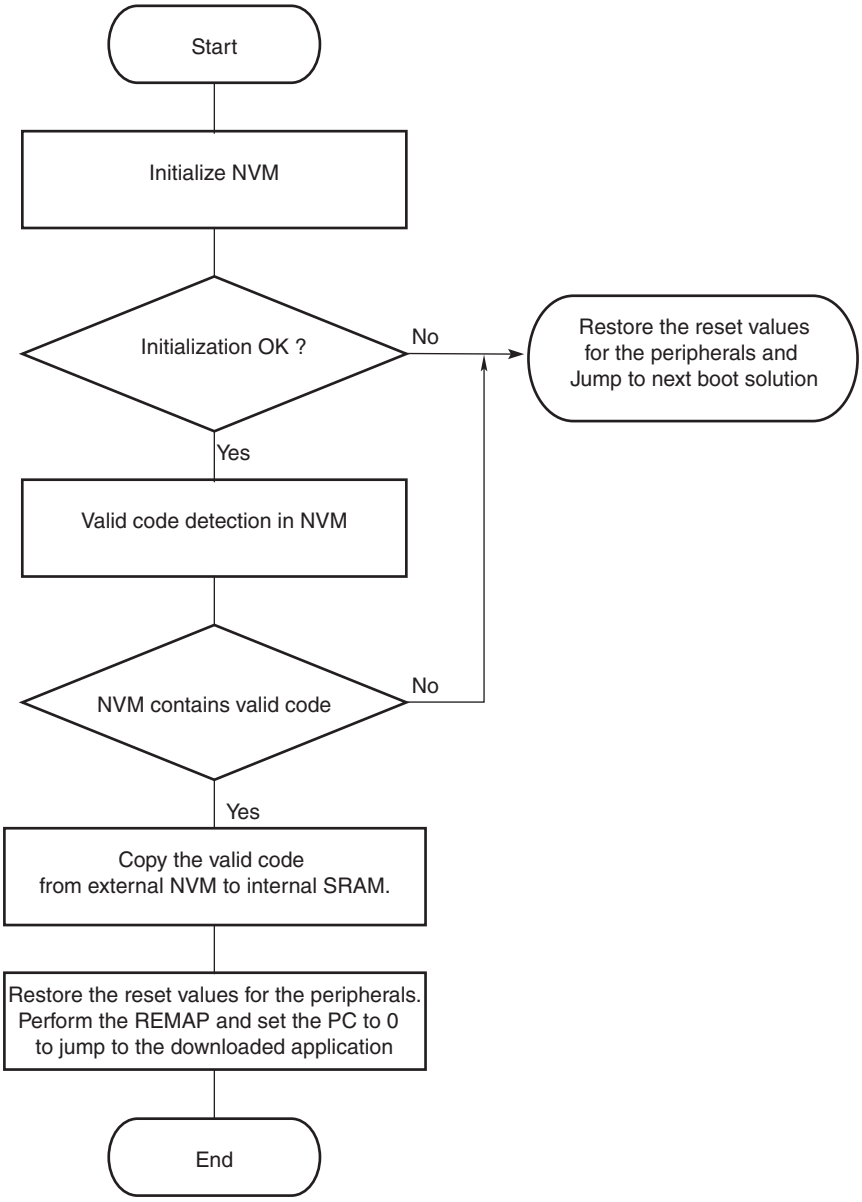
BOOT Value	SPI0 NPCS0	SDCard	NAND Flash	SPI0 NPCS1	TWI EEPROM	SAM-BA Monitor
5	-	-	-	-	-	Y
6	-	-	-	-	-	Y
7	-	-	-	-	-	Y

Figure 11-2. NVM Bootloader Sequence Diagram



11.4.2 NVM Bootloader Program Description

Figure 11-3. NVM Bootloader Program Diagram



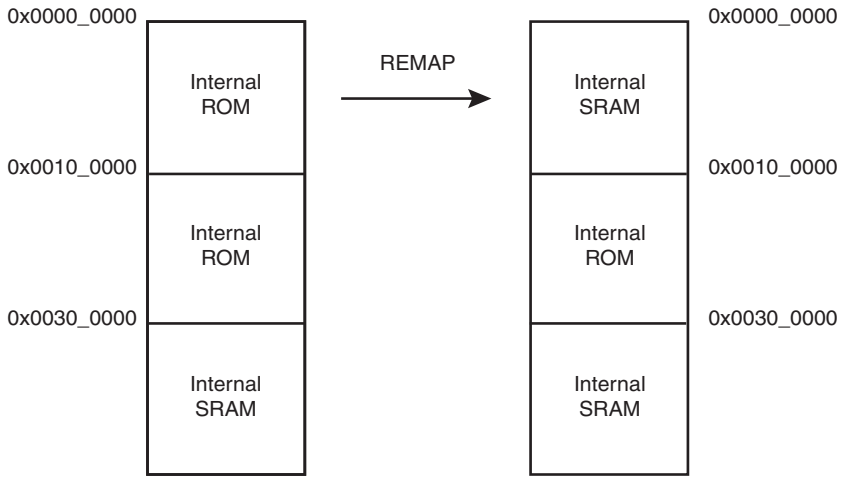
The NVM bootloader program first initializes the PIOs related to the NVM device. Then it configures the right peripheral depending on the NVM and tries to access this memory. If the initialization fails, it restores the reset values for the PIO and the peripheral and then tries the same operations on the next NVM of the sequence.

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

If the NVM does not contain valid code, the NVM bootloader program restores the reset value for the peripherals and then tries the same operations on the next NVM of the sequence.

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

Figure 11-4. Remap Action after Download Completion



11.4.3 Valid Code Detection

There are two kinds of valid code detection.

11.4.3.1 ARM Exception Vectors Check

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

Figure 11-5. LDR Opcode

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

Figure 11-6. B Opcode

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

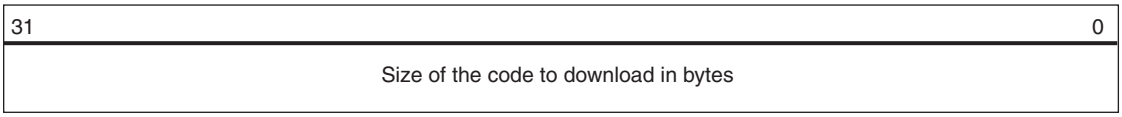
Unconditional instruction: 0xE for bits 31 to 28

Load PC with PC relative addressing instruction:

- Rn = Rd = PC = 0xF
- I==0 (12-bit immediate value)
- P==1 (pre-indexed)
- U offset added (U==1) or subtracted (U==0)
- W==1

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

Figure 11-7. Structure of the ARM Vector 6



The value has to be smaller than 24 kbytes. This size is the internal SRAM size minus the stack size used by the ROM Code at the end of the internal SRAM.

Example

An example of valid vectors follows:

```
00          ea000006          B0x20
04          eafffffe          B0x04
08          ea00002f          B_main
0c          eafffffe          B0x0c
10          eafffffe          B0x10
14          00001234          B0x14<- Code size = 4660 bytes
18          eafffffe          B0x18
```

11.4.3.2 boot.bin File Check

This method is the one used on FAT formatted SDCard. The boot program must be a file named "`boot.bin`" written in the root directory of the filesystem. Its size must not exceed the maximum size allowed: 24 kbytes (0x6000).

11.4.4 Detailed Memory Boot Procedures

11.4.4.1 NAND Flash Boot: NAND Flash Detection

After NAND Flash interface configuration, a reset command is sent to the memory.

The Boot Program first tries to find valid software on a NAND Flash device connected to EBI CS3, with data lines connected to D0-D7, then on NAND Flash connected to D16-D23. Hardware ECC detection and correction are provided by the PMECC peripheral (refer to the PMECC section in the datasheet for more information).

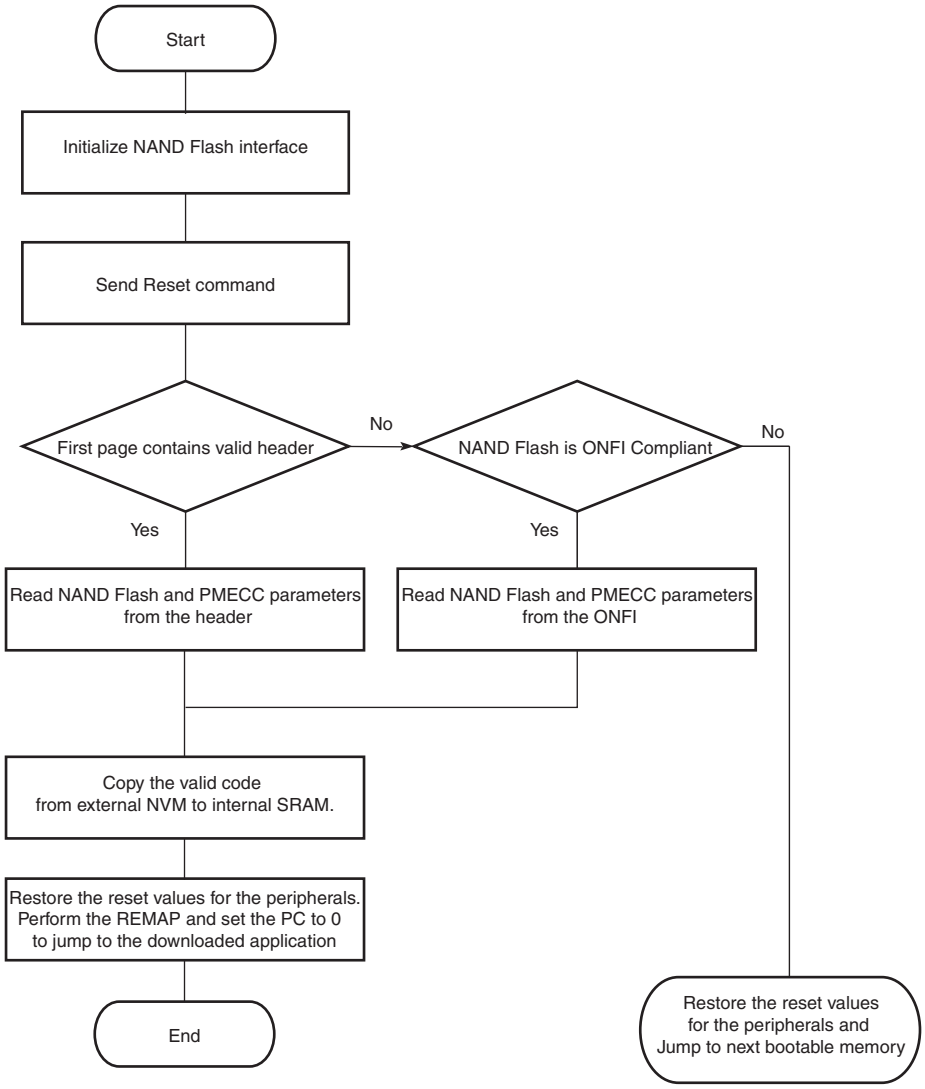
The Boot Program is able to retrieve NAND Flash parameters and ECC requirements using two methods as follows:

- The detection of a specific header written at the beginning of the first page of NAND Flash,

or

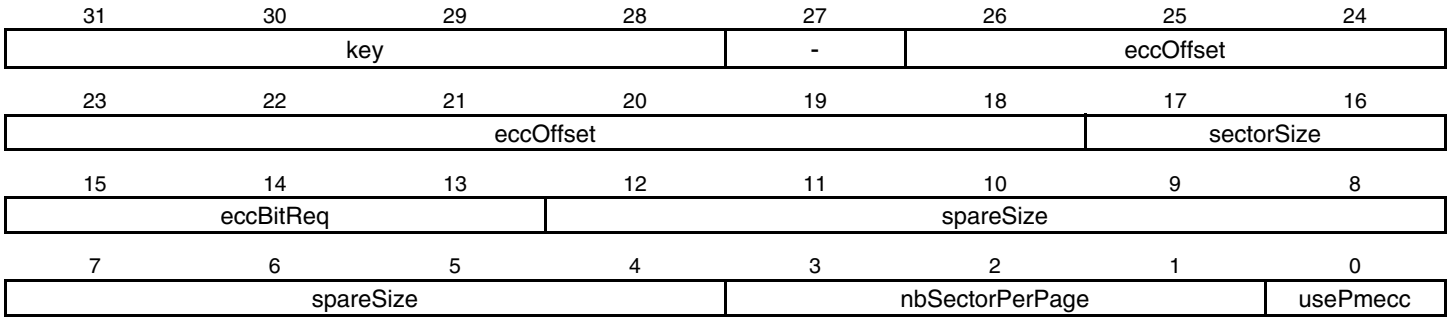
- Through the ONFI parameters for ONFI compliant memories.

Figure 11-8. Boot NAND Flash Download



NAND Flash Specific Header Detection

This is the first method used to determine NAND Flash parameters. After Initialization and Reset command, the Boot Program reads the first page without ECC check, to determine if the NAND parameter header is present. The header is made of 52 times the same 32-bit word (for redundancy reasons) which must contain NAND and PMECC parameters used to correctly perform the read of the rest of the data in the NAND. This 32-bit word is described below:



- **usePmecc: Use PMECC**

0 = Do not use PMECC to detect and correct the data.

1 = Use PMECC to detect and correct the data.

- **nbSectorPerPage: Number of sectors per page**
- **spareSize: Size of the spare zone in bytes**
- **eccBitReq: Number of ECC bits required**
- **sectorSize: Size of the ECC sector**

0 = for 512 bytes.

1 = for 1024 bytes per sector.

Other value for future use.

- **eccOffset: Offset of the first ECC byte in the spare zone**

A value below 2 is not allowed and will be considered as 2.

- **key: value 0xC must be written here to validate the content of the whole word.**

If the header is valid, the Boot Program will continue with the detection of valid code.

ONFI 2.2 Parameters

In case no valid header has been found, the Boot Program will check if the NAND Flash is ONFI compliant, sending a Read Id command (0x90) with 0x20 as parameter for the address. If the NAND Flash is ONFI compliant, the Boot Program retrieves the following parameters with the help of the Get Parameter Page command:

- Number of bytes per page (byte 80)
- Number of bytes in spare zone (byte 84)
- Number of ECC bit correction required (byte 112)
- ECC sector size: by default set to 512 bytes, or 1024 bytes if the ECC bit capability above is 0xFF

By default, ONFI NAND Flash detection will turn ON the usePmecc parameter, and ECC correction algorithm is automatically activated.

Once the Boot Program retrieves the parameter, using one of the two methods described above, it will read the first page again, with or without ECC, depending on the usePmecc parameter. Then it looks for a valid code programmed just after the header offset 0xD0. If the code is valid, the program is copied at the beginning of the internal SRAM.

Note: Booting on 16-bit NAND Flash is not possible, only 8-bit NAND Flash memories are supported.

11.4.4.2 NAND Flash Boot: PMECC Error Detection and Correction

NAND Flash boot procedure uses PMECC to detect and correct errors during NAND Flash read operations in two cases:

- When the usePmecc flag is set in the specific NAND header. If the flag is not set, no ECC correction is performed during NAND Flash page read.
- When the NAND Flash has been detected using ONFI parameters.

The ROM code embeds the software used in the process of ECC detection/correction: the Galois Field tables, and the function PMECC_CorrectionAlgo(). The user does not need to embedd it in other software.

This function can be called by user software when PMECC status returns errors after a read page command.

Its address can be retrieved by reading the third vector of the ROM Code interrupt vector table, at address 0x100008.

The API of this function is:

```
unsigned int PMECC_CorrectionAlgo(AT91PS_PMECC pPMECC,
                                  AT91PS_PMERRLOC pPMERRLOC,
                                  PMECC_paramDesc_struct *PMECC_desc,
                                  unsigned int PMECC_status,
                                  unsigned int pageBuffer)
```

pPMECC : pointer to the PMECC base address,

pPMERRLOC : pointer to the PMERRLOC base address,

PMECC_desc : pointer to the PMECC descriptor,

PMECC_status : the status returned by the read of PMECCISR register;

pageBuffer : address of the buffer containing the page to be corrected.

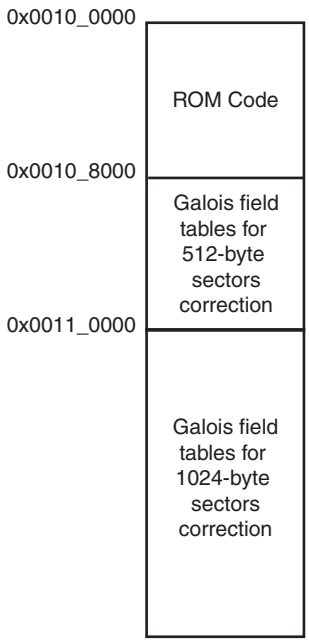
The PMECC descriptor structure is:

```
typedef struct _PMECC_paramDesc_struct {
    unsigned int pageSize;
    unsigned int spareSize;
    unsigned int sectorSize;  // 0 for 512, 1 for 1024 bytes
    unsigned int errBitNbrCapability;
    unsigned int eccSizeByte;
    unsigned int eccStartAddr;
    unsigned int eccEndAddr;

    unsigned int nandWR;
    unsigned int spareEna;
    unsigned int modeAuto;
```

```
    unsigned int clkCtrl;
    unsigned int interrupt;

    int tt;
    int mm;
    int nn;

    short *alpha_to;
    short *index_of;

    short partialSyn[100];
    short si[100];

    /* sigma table */
    short smu[TT_MAX + 2][2 * TT_MAX + 1];
    /* polynom order */
    short lmu[TT_MAX + 1];

} PMECC_paramDesc_struct;
```

The Galois field tables are mapped in the ROM just after the ROM code, as described in Figure 11-9 below:

Figure 11-9. Galois Field Table Mapping



For a full description and an example of how to use the PMECC detection and correction feature, refer to the software package dedicated to this device on Atmel's web site.

11.4.4.3 SD Card Boot

The SD Card bootloader uses MCI0. It looks for a "`boot.bin`" file in the root directory of a FAT12/16/32 formatted SD Card.

Supported SD Card Devices

SD Card Boot supports all SD Card memories compliant with SD Memory Card Specification V2.0. This includes SDHC cards.

11.4.4.4 SPI Flash Boot

Two kinds of SPI Flash are supported: SPI Serial Flash and SPI DataFlash.

The SPI Flash bootloader tries to boot on SPI0 Chip Select 0, first looking for SPI Serial Flash, and then for SPI DataFlash.

It uses only one valid code detection: analysis of ARM exception vectors.

The SPI Flash read is done by means of a Continuous Read command from address 0x0. This command is 0xE8 for DataFlash and 0x0B for Serial Flash devices.

Supported DataFlash Devices

The SPI Flash Boot program supports all Atmel DataFlash devices.

Table 11-3. DataFlash Device

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

Supported Serial Flash Devices

The SPI Flash Boot program supports all SPI Serial Flash devices responding correctly at both Get Status and Continuous Read commands.

11.4.4.5 TWI EEPROM Boot

The TWI EEPROM Bootloader uses the TWI0. It uses only one valid code detection. It analyzes the ARM exception vectors.

Supported TWI EEPROM Devices

TWI EEPROM Boot supports all I^2C-compatible TWI EEPROM memories using 7-bit device address 0x50.

11.4.5 Hardware and Software Constraints

The NVM drivers use several PIOs in peripheral mode to communicate with external memory devices. Care must be taken when these PIOs are used by the application. The devices connected could be unintentionally driven at boot time, and electrical conflicts between output pins used by the NVM drivers and the connected devices may occur.

To assure correct functionality, it is recommended to plug in critical devices to other pins not used by NVM.

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

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

PIO Driven during Boot Program Execution

NVM Bootloader	Peripheral	Pin	PIO Line
NAND	EBI CS3 SMC	NANDOE	PIOD0
	EBI CS3 SMC	NANDWE	PIOD1
	EBI CS3 SMC	NANDCS	PIOD4
	EBI CS3 SMC	NAND ALE	A21
	EBI CS3 SMC	NAND CLE	A22
	EBI CS3 SMC	Cmd/Addr/Data	D[16:0]
SD Card	MCI0	MCI0_CK	PIOA17
	MCI0	MCI0_D0	PIOA15
	MCI0	MCI0_D1	PIOA18
	MCI0	MCI0_D2	PIOA19
	MCI0	MCI0_D3	PIOA20
SPI Flash	SPI0	MOSI	PIOA10
	SPI0	MISO	PIOA11
	SPI0	SPCK	PIOA13
	SPI0	NPCS0	PIOA14
	SPI0	NPCS1	PIOA7
TWI0 EEPROM	TWI0	TWD0	PIOA30
	TWI0	TWCK0	PIOA31
SAM-BA Monitor	DBGU	DRXD	PIOA9
	DBGU	DTXD	PIOA10

11.5 SAM-BA Monitor

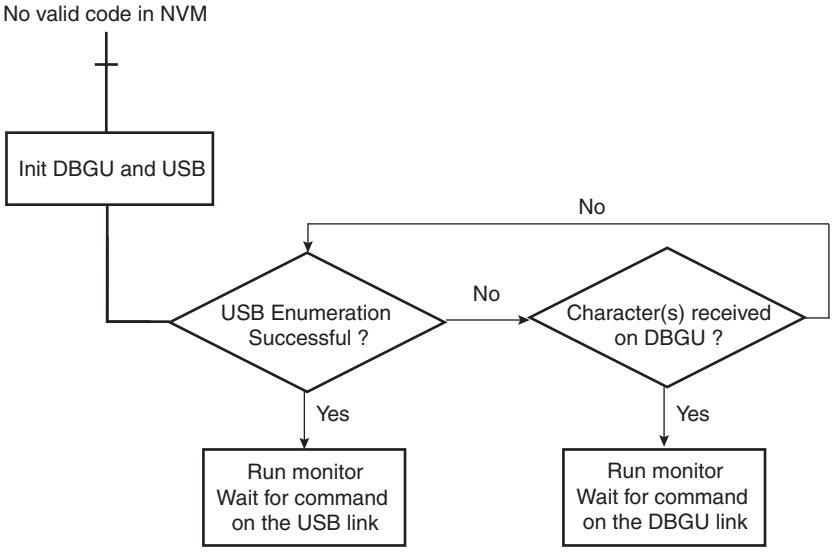
If no valid code has been found in NVM during the NVM bootloader sequence, the SAM-BA Monitor program is launched.

The SAM-BA Monitor principle is to:

- Initialize DBGU and USB
- Check if USB Device enumeration has occurred
- Check if characters have been received on the DBGU

Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as listed in Table 11-4.

Figure 11-10.SAM-BA Monitor Diagram



11.5.1 Command List

Table 11-4. Commands Available through the SAM-BA Monitor

Command	Action	Argument(s)	Example
N	set Normal mode	No argument	**N#**
T	set Terminal mode	No argument	**T#**
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#**

- Mode commands:
  - Normal mode configures SAM-BA Monitor to send / receive data in binary format,
  - Terminal mode configures SAM-BA Monitor to send / receive data in ascii format.
- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target.
  - *Address*: Address in hexadecimal.
  - *Value*: Byte, halfword or word to write in hexadecimal.
  - *Output*: '>'
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target.
  - *Address*: Address in hexadecimal.
  - *Output*: The byte, halfword or word read in hexadecimal followed by '>'
- Send a file (**S**): Send a file to a specified address.
  - *Address*: Address in hexadecimal.
  - *Output*: '>'

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

- Receive a file (**R**): Receive data into a file from a specified address
  - *Address*: Address in hexadecimal.
  - *NbOfBytes*: Number of bytes in hexadecimal to receive.
  - *Output*: '>'
- Go (**G**): Jump to a specified address and execute the code.
  - *Address*: Address to jump in hexadecimal.
  - *Output*: '>'once returned from the program execution. If the executed program does not handle the link register at its entry and does not return, the prompt will not be displayed.
- Get Version (**V**): Return the Boot Program version.
  - *Output*: version, date and time of ROM code followed by '>'.

11.5.2 DBGU Serial Port

Communication is performed through the DBGU serial port initialized to 115,200 Baud, 8 bits of data, no parity, 1 stop bit.

11.5.2.1 Supported External Crystal/External Clocks

The SAM-BA Monitor supports a frequency of 12 MHz to allow DBGU communication for both external crystal and external clock.

11.5.2.2 Xmodem Protocol

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

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC16 to guarantee detection of a maximum bit error.

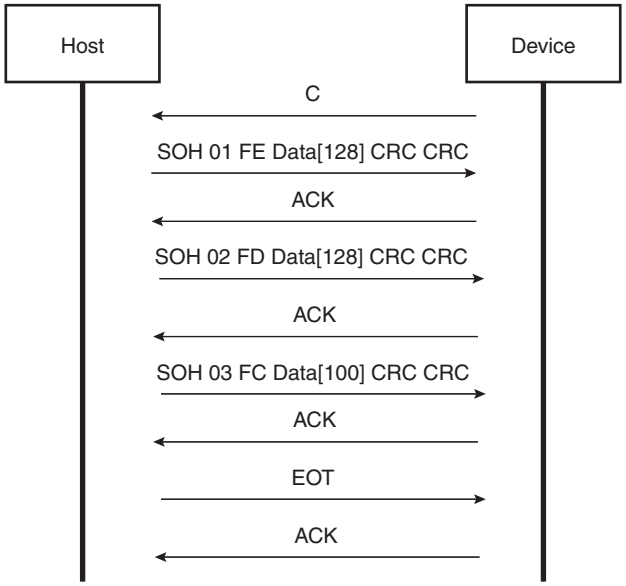
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<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 11-11 shows a transmission using this protocol.

Figure 11-11.Xmodem Transfer Example



11.5.3 USB Device Port

11.5.3.1 Supported External Crystal / External Clocks

The only frequency supported by SAM-BA Monitor to allow USB communication is a 12 MHz crystal or external clock.

11.5.3.2 USB Class

The device uses the USB Communication Device Class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, from Windows 98SE® to Windows XP®. The CDC document, available at www.usb.org, describes how 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.

11.5.3.3 Enumeration Process

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

Table 11-5. Handled Standard Requests

Request	Definition
GET_DESCRIPTOR	Returns the current device configuration value.
SET_ADDRESS	Sets the device address for all future device access.
SET_CONFIGURATION	Sets the device configuration.
GET_CONFIGURATION	Returns the current device configuration value.
GET_STATUS	Returns status for the specified recipient.
SET_FEATURE	Used to set or enable a specific feature.
CLEAR_FEATURE	Used to clear or disable a specific feature.

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

Table 11-6. Handled Class Requests

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

Unhandled requests are STALLed.

11.5.3.4 Communication Endpoints

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint and endpoint 2 is a 64-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through 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.

12. Boot Sequence Controller (BSC)

12.1 Description

The System Controller embeds a Boot Sequence Configuration Register to save timeout delays on boot. The boot sequence is programmable through the Boot Sequence Configuration Register (BSC_CR).

This register is powered by VDDBU, the modification is saved and applied after the next reset. The register is taking Factory Value in case of battery removing.

This register is programmable with user programs or SAM-BA and it is key-protected.

12.2 Embedded Characteristics

- VDDBU powered register

12.3 Product Dependencies

- Product-dependent order

12.4 Boot Sequence Controller (BSC) User Interface

Table 12-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Boot Sequence Configuration Register	BSC_CR	Read-write	–

12.4.1 Boot Sequence Configuration Register

Name: BSC_CR

Address: 0xFFFFFE54

Access: Read-write

Factory Value: 0x0000_0000

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

- **BOOT: Boot Media Sequence**

This value is defined in the product-dependent ROM code. It is only written if BOOTKEY carries the valid value.
Please refer to the “NVM Boot Sequence” section of this datasheet for details on BOOT value.

- **BOOTKEY**

0x6683 (BSC_KEY): Valid key to write the BSC_CR register; it needs to be written at the same time as the BOOT field.

Other values disable the write access. This key field is write-only.

13. Advanced Interrupt Controller (AIC)

13.1 Description

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to thirty-two interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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

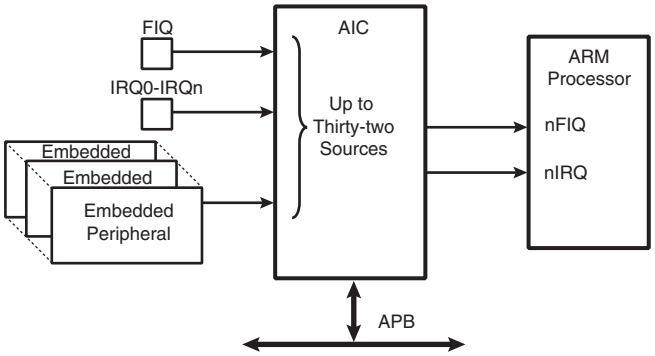
The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

13.2 Embedded Characteristics

- Controls the Interrupt Lines (nIRQ and nFIQ) of an 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
  - Source 2 to Source 31 Control up to Thirty Embedded Peripheral Interrupts or External Interrupts
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive External Sources
- 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 to the Fast Interrupt of the Processor
- General Interrupt Mask
  - Provides Processor Synchronization on Events Without Triggering an Interrupt
- Write Protected Registers

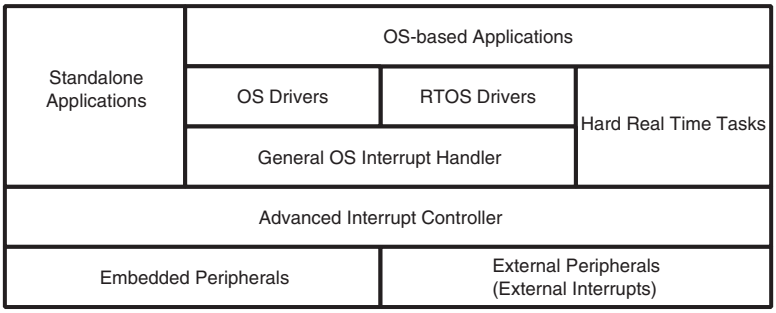
13.3 Block Diagram

Figure 13-1. Block Diagram



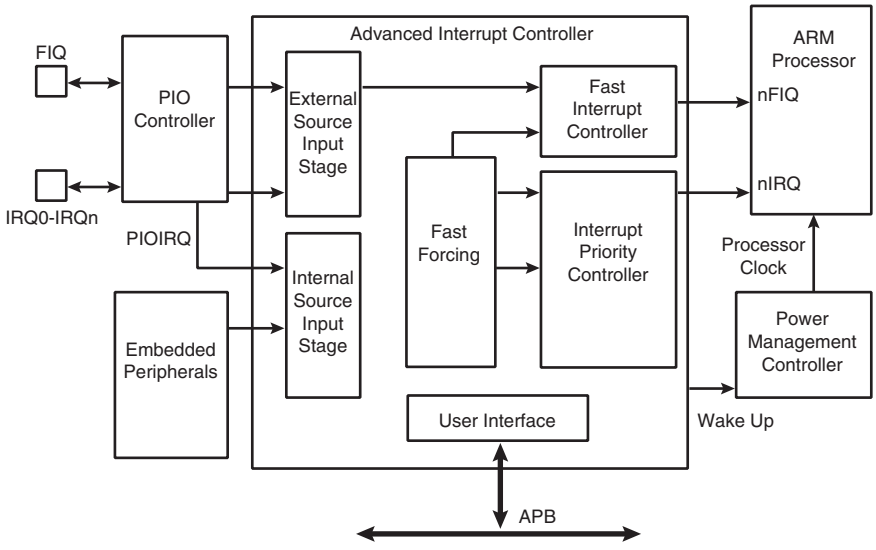
13.4 Application Block Diagram

Figure 13-2. Description of the Application Block



13.5 AIC Detailed Block Diagram

Figure 13-3. AIC Detailed Block Diagram



13.6 I/O Line Description

Table 13-1. I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0 - IRQn	Interrupt 0 - Interrupt n	Input

13.7 Product Dependencies

13.7.1 I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

Table 13-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
AIC	FIQ	PC31	A
AIC	IRQ	PB18	A

13.7.2 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.

13.7.3 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.

13.8 Functional Description

13.8.1 Interrupt Source Control

13.8.1.1 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.

13.8.1.2 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.

13.8.1.3 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 76.) 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 79.)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

13.8.1.4 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 76) 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.

Figure 13-4. Internal Interrupt Source Input Stage

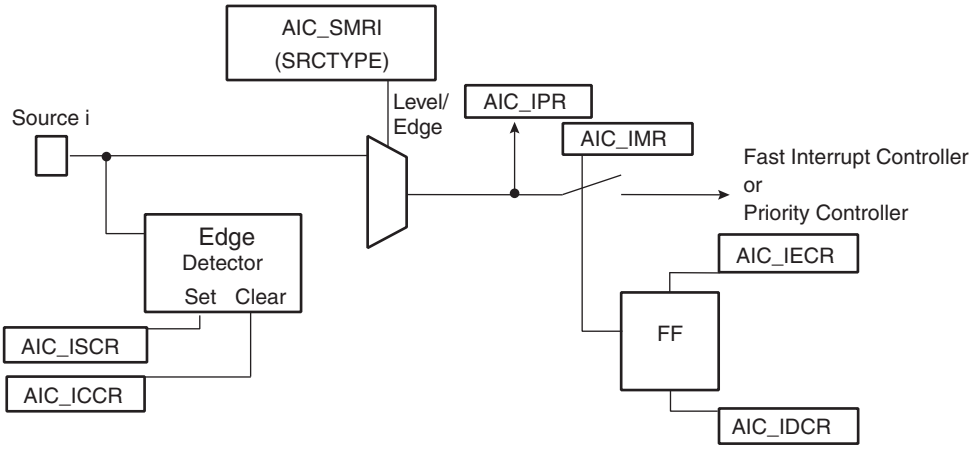
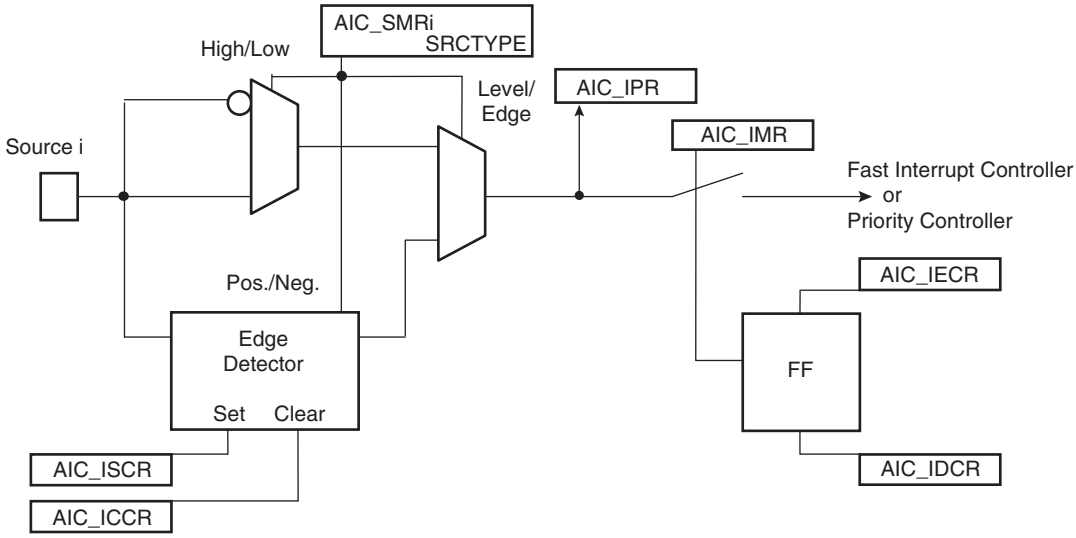


Figure 13-5. External Interrupt Source Input Stage



13.8.2 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.

Figure 13-6. External Interrupt Edge Triggered Source

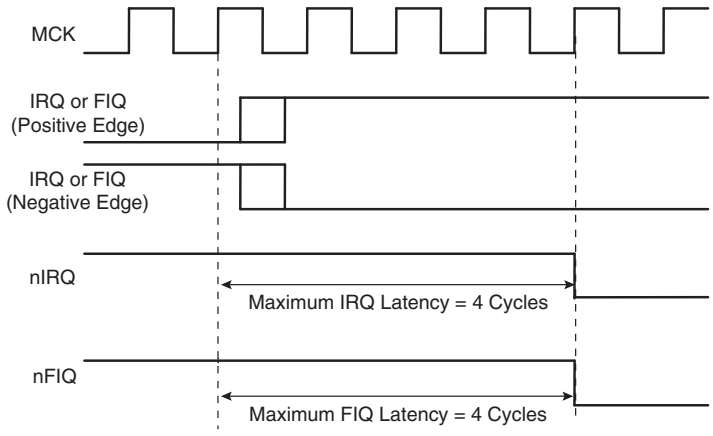


Figure 13-7. External Interrupt Level Sensitive Source

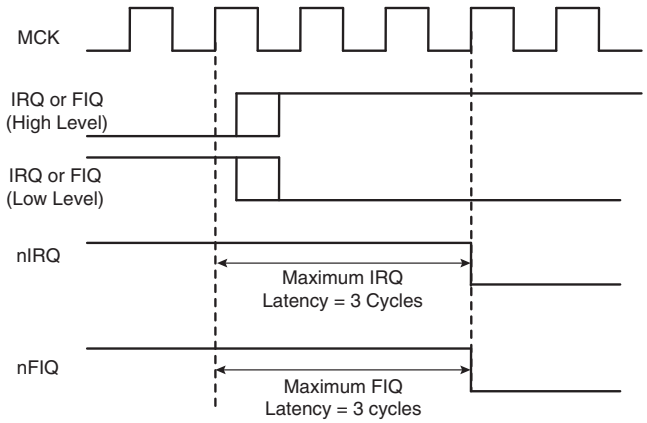


Figure 13-8. Internal Interrupt Edge Triggered Source

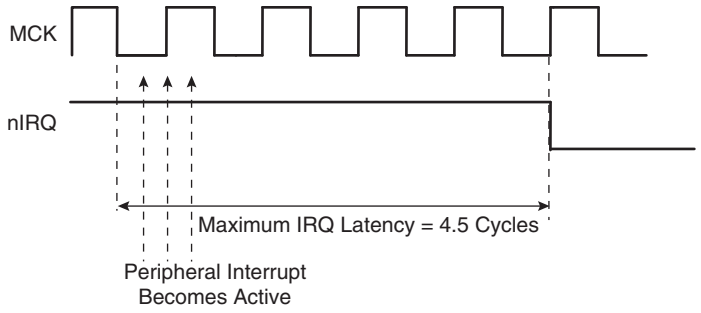
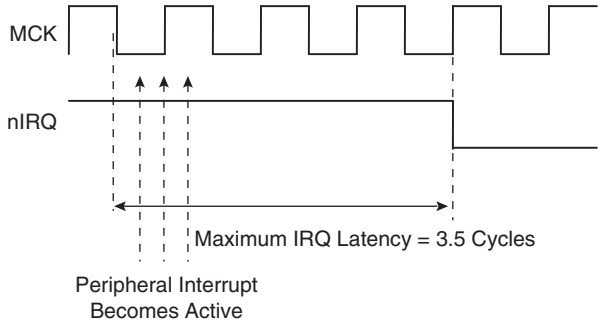


Figure 13-9. Internal Interrupt Level Sensitive Source



13.8.3 Normal Interrupt

13.8.3.1 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.

13.8.3.2 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.

13.8.3.3 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.

13.8.3.4 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).

13.8.4 Fast Interrupt

13.8.4.1 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.

13.8.4.2 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.

13.8.4.3 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.

13.8.4.4 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.

13.8.4.5 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).

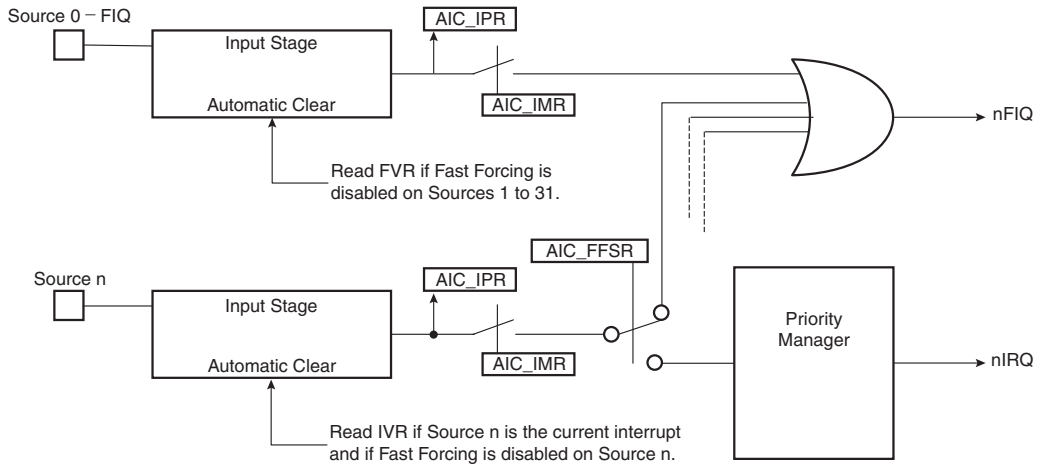
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt 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 13-10.Fast Forcing



13.8.5 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.

13.8.6 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.

13.8.7 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.

13.9 Write Protection Registers

To prevent any single software error that may corrupt AIC behavior, the registers listed below can be write-protected by setting the WPEN bit in the AIC Write Protect Mode Register (AIC_WPMR).

If a write access in a write-protected register is detected, then the WPVS flag in the AIC Write Protect Status Register (AIC_WPSR) is set and the WPVSRC field indicates in which register the write access has been attempted.

The WPVS flag is automatically reset after reading the AIC Write Protect Status Register.

The protected registers are:

- “AIC Source Mode Register” on page 84
- “AIC Source Vector Register” on page 85
- “AIC Spurious Interrupt Vector Register” on page 97
- “AIC Debug Control Register” on page 98

13.10 Advanced Interrupt Controller (AIC) User Interface

13.10.1 Base Address

The AIC is mapped at the address 0xFFFFF000. 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 13-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Source Mode Register 0	AIC_SMR0	Read-write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read-write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read-write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read-write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read-write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read-write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register(2)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(2)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118 - 0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register(2)	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register(2)	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register(2)	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register(2)	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read-write	0x0
0x138	Debug Control Register	AIC_DCR	Read-write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register(2)	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register(2)	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register(2)	AIC_FFSR	Read-only	0x0
0x14C - 0x1E0	Reserved	---	---	---
0x1E4	Write Protect Mode Register	AIC_WPMR	Read-write	0x0
0x1E8	Write Protect Status Register	AIC_WPSR	Read-only	0x0
0x1EC - 0x1FC	Reserved			

Notes: 1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.

2. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.

13.10.2 AIC Source Mode Register

Name: AIC_SMR0..AIC_SMR31

Address: 0xFFFFF000

Access Read-write

Reset: 0x0

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

This register can only be written if the WPEN bit is cleared in AIC Write Protect Mode Register

- **PRIOR: Priority Level**

The priority level is programmable from 0 (lowest priority) to 7 (highest priority).

The priority level is not used for the FIQ in the related SMR register AIC_SMR0.

- **SRCTYPE: Interrupt Source Type**

The active level or edge is not programmable for the internal interrupt sources.

Value	Name	Description
0x0	INT_LEVEL_SENSITIVE	High level Sensitive for internal source Low level Sensitive for external source
0x1	INT_EDGE_TRIGGERED	Positive edge triggered for internal source Negative edge triggered for external source
0x2	EXT_HIGH_LEVEL	High level Sensitive for internal source High level Sensitive for external source
0x3	EXT_POSITIVE_EDGE	Positive edge triggered for internal source Positive edge triggered for external source

13.10.3 AIC Source Vector Register

Name: AIC_SVR0..AIC_SVR31

Address: 0xFFFFF080

Access: Read-write

Reset: 0x0

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

This register can only be written if the WPEN bit is cleared in AIC Write Protect Mode Register

- **VECTOR: Source Vector**

The user may store in these registers the addresses of the corresponding handler for each interrupt source.

13.10.4 AIC Interrupt Vector Register

Name: AIC_IVR

Address: 0xFFFFF100

Access: Read-only

Reset: 0x0

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

- **IRQV: Interrupt Vector Register**

The Interrupt Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC_SPU.

13.10.5 AIC FIQ Vector Register

Name: AIC_FVR

Address: 0xFFFFF104

Access: Read-only

Reset: 0x0

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

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

13.10.6 AIC Interrupt Status Register

Name: AIC_ISR

Address: 0xFFFFF108

Access: Read-only

Reset: 0x0

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

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

13.10.7 AIC Interrupt Pending Register

Name: AIC_IPR

Address: 0xFFFFF10C

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

13.10.8 AIC Interrupt Mask Register

Name: AIC_IMR

Address: 0xFFFFF110

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

13.10.9 AIC Core Interrupt Status Register

Name: AIC_CISR

Address: 0xFFFFF114

Access: Read-only

Reset: 0x0

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

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.

13.10.10AIC Interrupt Enable Command Register

Name: AIC_IECR

Address: 0xFFFFF120

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

13.10.11AIC Interrupt Disable Command Register

Name: AIC_IDCR

Address: 0xFFFFF124

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

13.10.12AIC Interrupt Clear Command Register

Name: AIC_ICCR

Address: 0xFFFFF128

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

13.10.13AIC Interrupt Set Command Register

Name: AIC_ISCR

Address: 0xFFFFF12C

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

13.10.14AIC End of Interrupt Command Register

Name: AIC_EOICR

Address: 0xFFFFF130

Access: Write-only

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

The End of Interrupt Command Register is used by the interrupt routine to indicate that the interrupt treatment is complete. Any value can be written because it is only necessary to make a write to this register location to signal the end of interrupt treatment.

13.10.15AIC Spurious Interrupt Vector Register

Name: AIC_SPU

Address: 0xFFFFF134

Access: Read-write

Reset: 0x0

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

This register can only be written if the WPEN bit is cleared in AIC Write Protect Mode Register

- **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.

13.10.16AIC Debug Control Register

Name: AIC_DCR

Address: 0xFFFFF138

Access: Read-write

Reset: 0x0

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

This register can only be written if the WPEN bit is cleared in AIC Write Protect Mode Register

- **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.

13.10.17AIC Fast Forcing Enable Register

Name: AIC_FFER

Address: 0xFFFFF140

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the corresponding interrupt.

13.10.18AIC Fast Forcing Disable Register

Name: AIC_FFDR

Address: 0xFFFFF144

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the corresponding interrupt.

13.10.19AIC Fast Forcing Status Register

Name: AIC_FFSR

Address: 0xFFFFF148

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the corresponding interrupt.

1 = The Fast Forcing feature is enabled on the corresponding interrupt.

13.10.20AIC Write Protect Mode Register

Name: AIC_WPMR

Address: 0xFFFFF1E4

Access: Read-write

Reset: See Table 13-3

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x414943 ("AIC" in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x414943 ("AIC" in ASCII).

Protects the registers:

- “AIC Source Mode Register” on page 84
- “AIC Source Vector Register” on page 85
- “AIC Spurious Interrupt Vector Register” on page 97
- “AIC Debug Control Register” on page 98

- **WPKEY: Write Protect KEY**

Should be written at value 0x414943 ("AIC" in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

13.10.21AIC Write Protect Status Register

Name: AIC_WPSR

Address: 0xFFFFF1E8

Access: Read-only

Reset: See Table 13-3

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the AIC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the AIC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading AIC_WPSR automatically clears all fields.

14. Reset Controller (RSTC)

14.1 Description

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

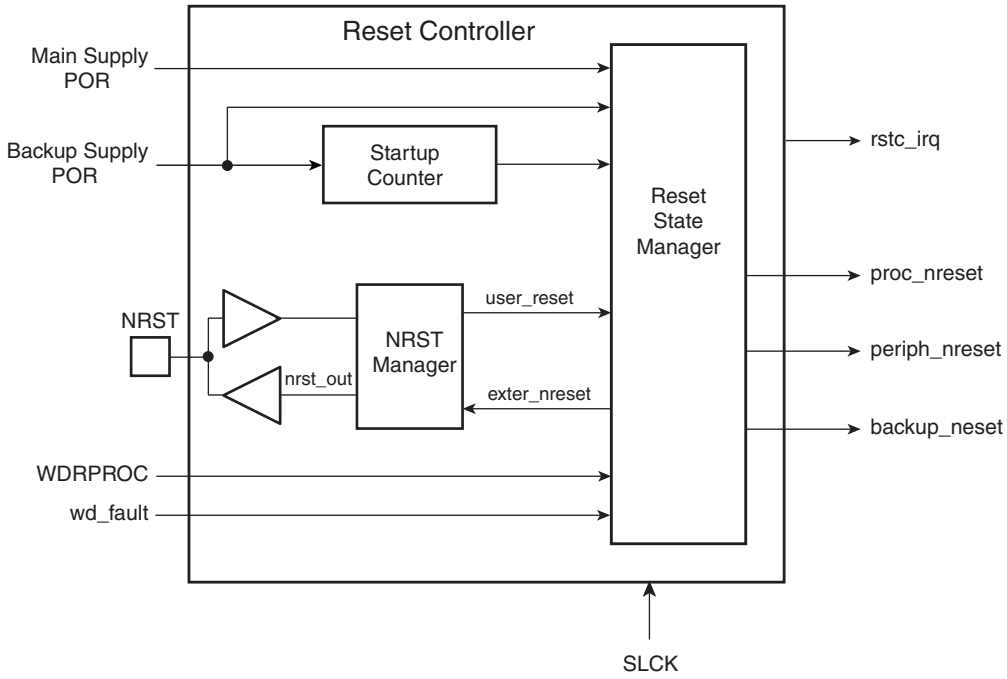
The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

14.2 Embedded Characteristics

- Manages All Resets of the System, Including
  - External Devices Through the NRST Pin
  - Processor Reset
  - Peripheral Set Reset
  - Backed-up Peripheral Reset
- Based on 2 Embedded Power-on Reset Cells
- Reset Source Status
  - Status of the Last Reset
  - Either General Reset, Wake-up Reset, Software Reset, User Reset, Watchdog Reset
- External Reset Signal Shaping
- AMBA™-compliant Interface
  - Interfaces to the ARM® Advanced Peripheral Bus

14.3 Block Diagram

Figure 14-1. Reset Controller Block Diagram



14.4 Functional Description

14.4.1 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.

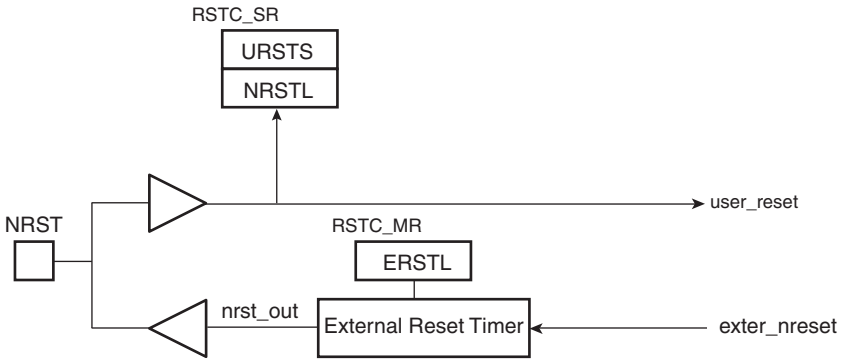
The Reset Controller Mode Register (RSTC_MR), allowing the configuration of the Reset Controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU is on.

14.4.2 NRST Manager

After power-up, NRST is an output during the ERSTL time defined in the RSTC. When ERSTL elapsed, the pin behaves as an input and all the system is held in reset if NRST is tied to GND by an external signal.

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



14.4.2.1 NRST Signal

The NRST Manager handles the NRST input line asynchronously. When the line is low, a User Reset is immediately reported to the Reset State Manager. When the NRST goes from low to high, the internal reset is synchronized with the Slow Clock to provide a safe internal de-assertion of reset.

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.

14.4.2.2 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.

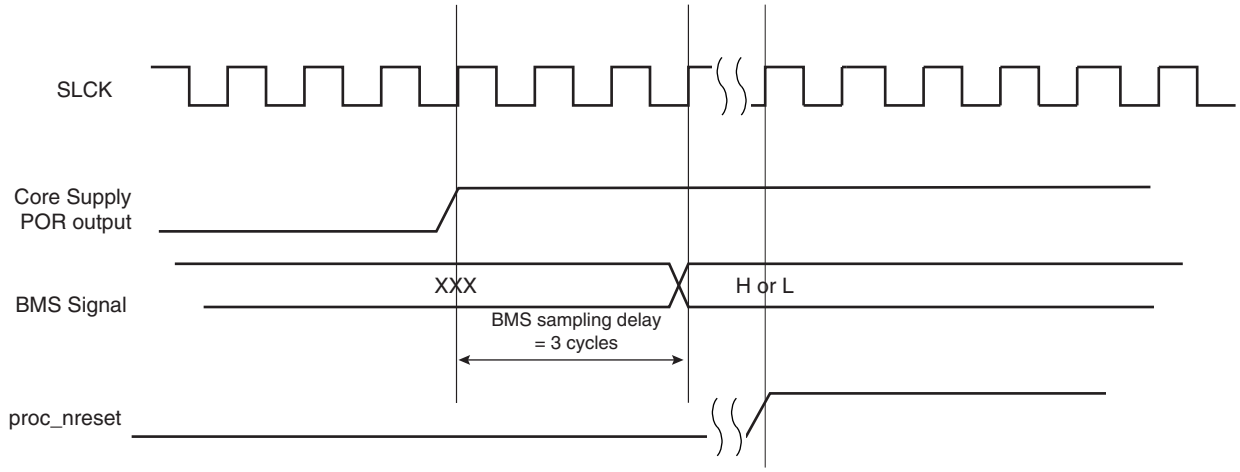
This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

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.

14.4.3 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



14.4.4 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.

14.4.4.1 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.

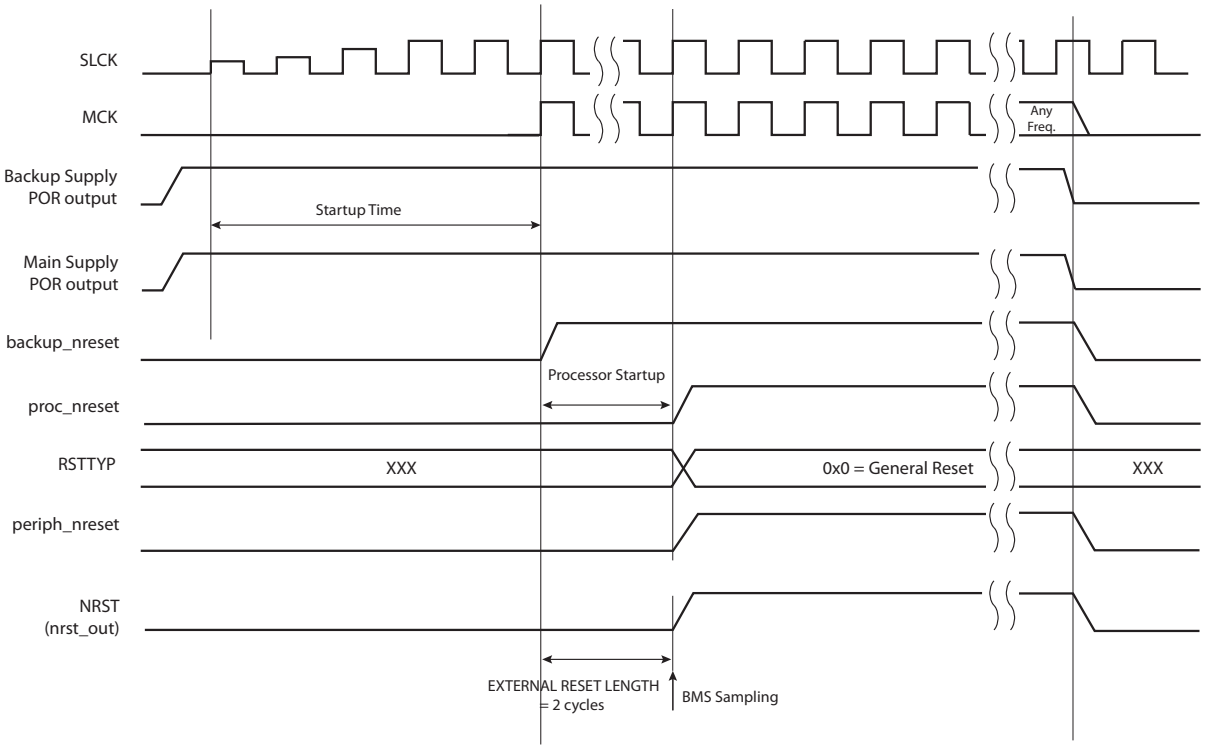
When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shutdown.

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



14.4.4.2 Wake-up Reset

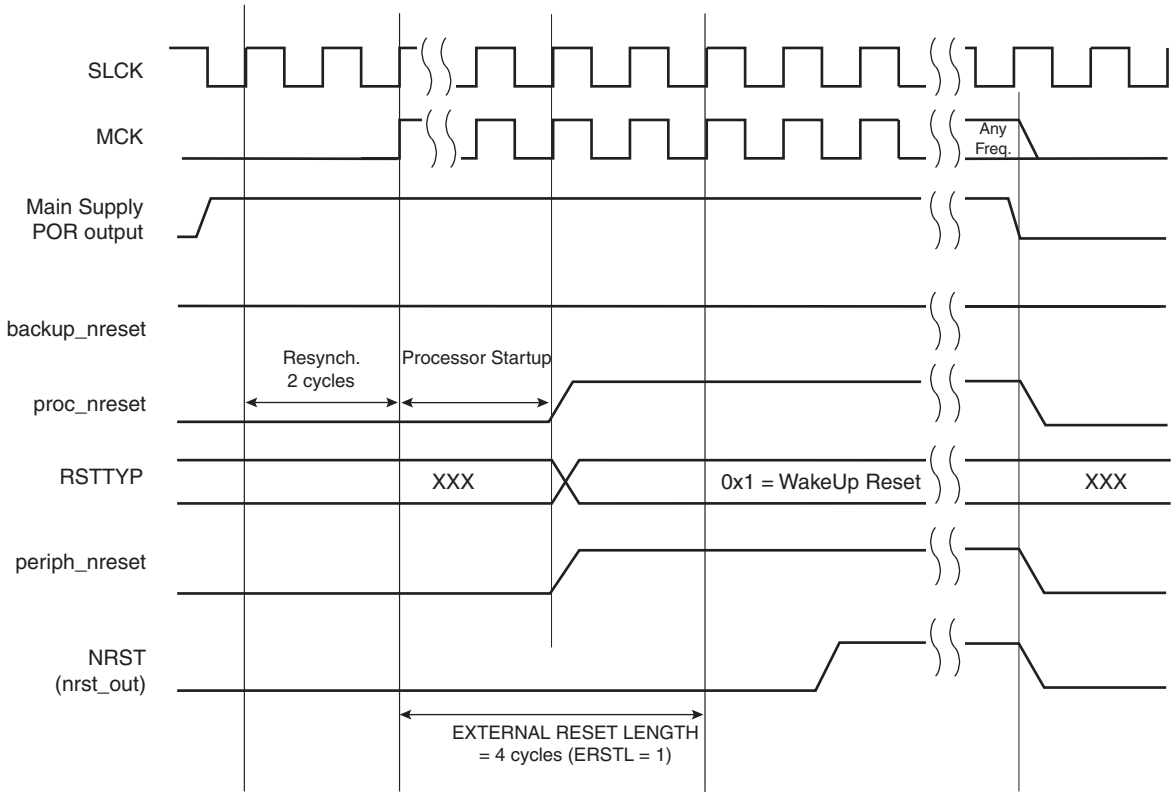
The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 3 Slow Clock cycles,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



14.4.4.3 User Reset

The User Reset is entered when a low level is detected on the NRST pin When a falling edge occurs on NRST (reset activation), internal reset lines are immediately asserted.

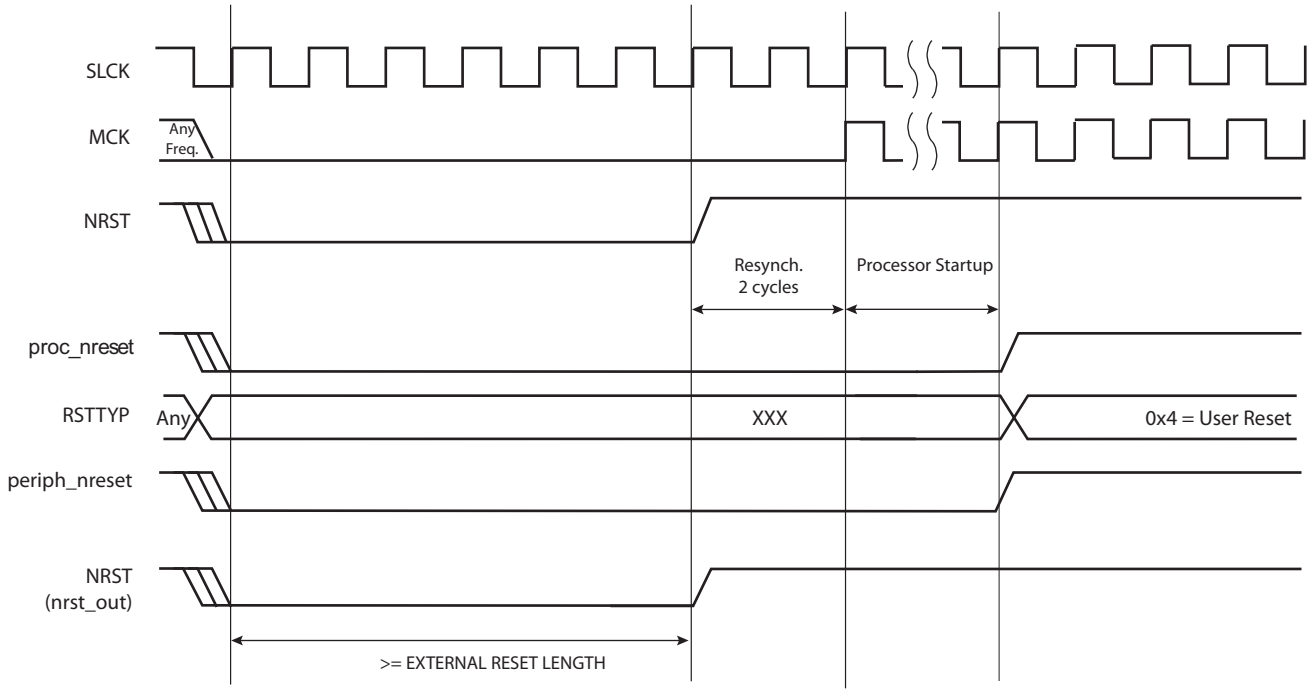
The Processor Reset and the Peripheral Reset are asserted.

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



14.4.4.4 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.

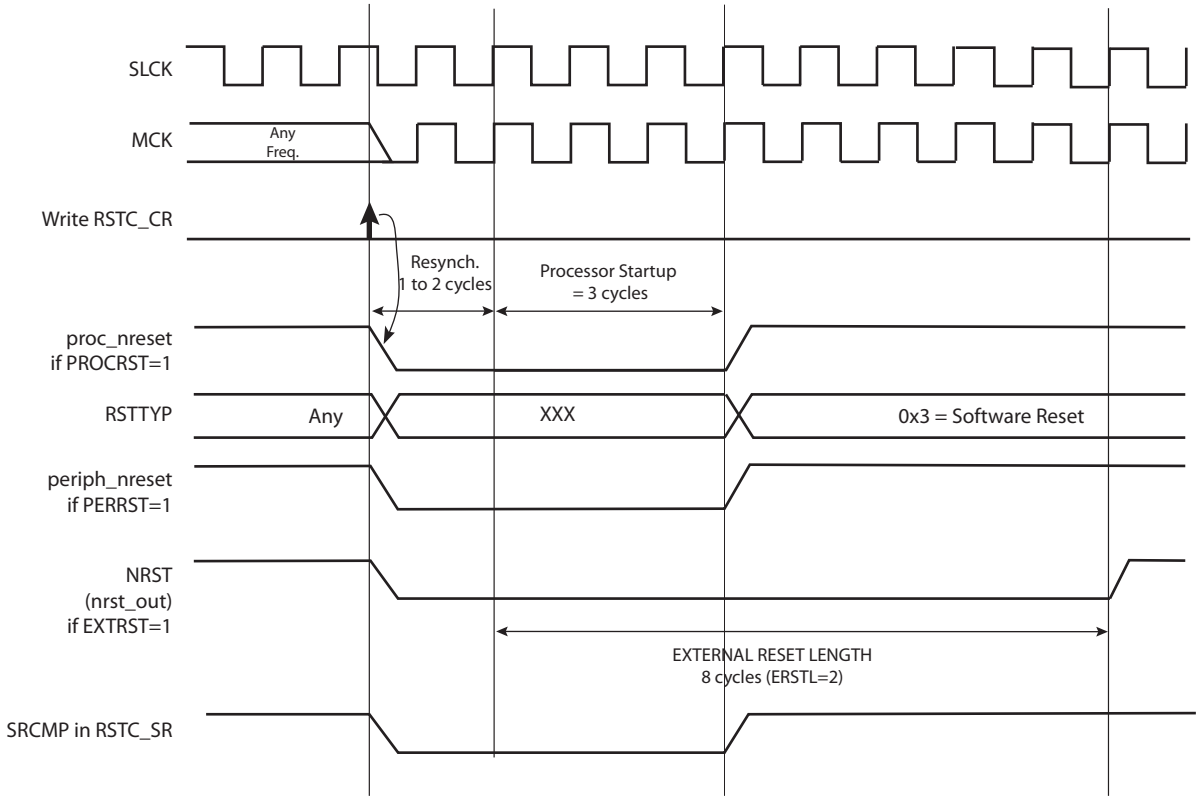
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

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



14.4.4.5 Watchdog Reset

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

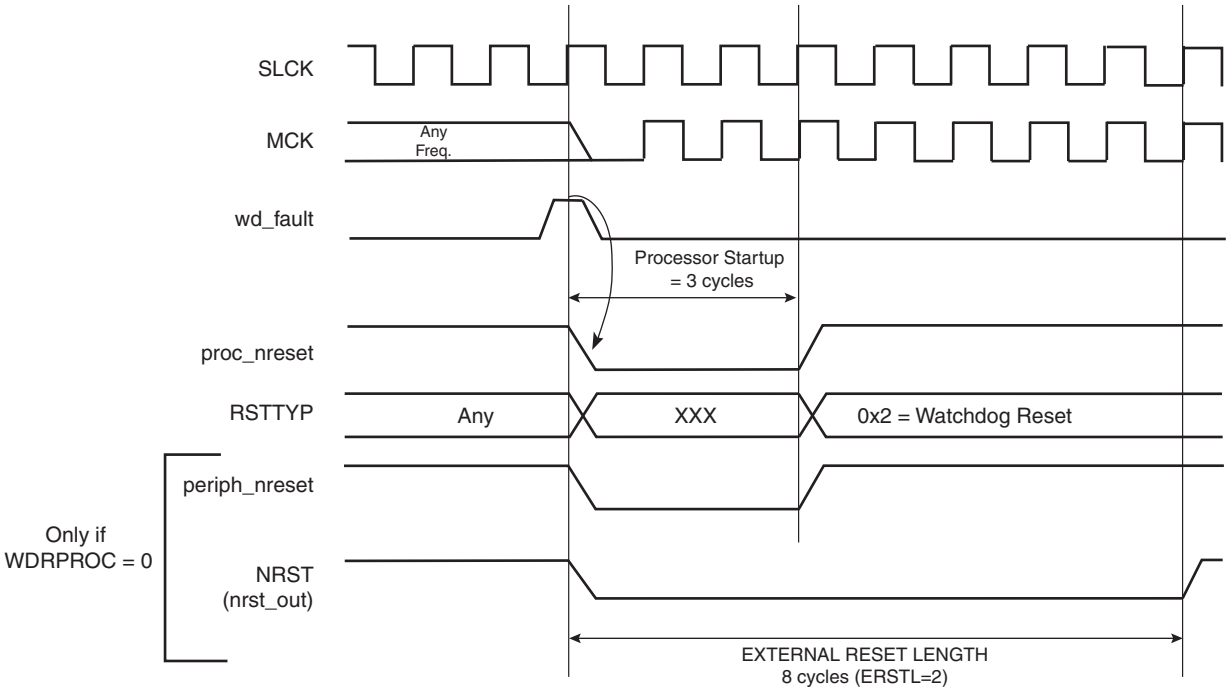
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- 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



14.4.5 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
- User Reset
- Watchdog Reset
- Software Reset

Particular cases are listed below:

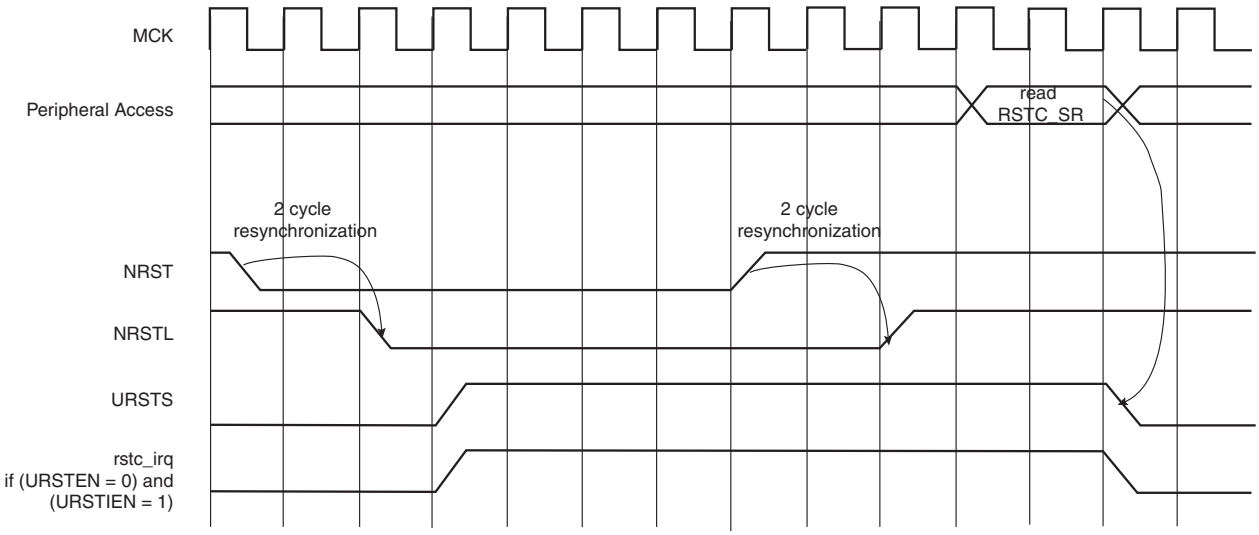
- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST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

14.4.6 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). Reading the RSTC_SR status register resets the URSTS bit.

Figure 14-9. Reset Controller Status and Interrupt



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

14.5.1 Reset Controller Control Register

Name: RSTC_CR

Address: 0xFFFFFE00

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 and resets the processor and the peripherals.

- **KEY: Password**

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

14.5.2 Reset Controller Status Register

Name: RSTC_SR

Address: 0xFFFFFE04

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.

14.5.3 Reset Controller Mode Register

Name: RSTC_MR

Address: 0xFFFFFE08

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
–	–		–	–	–	–	–

- **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 Clock (RTC)

15.1 Description

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

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

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

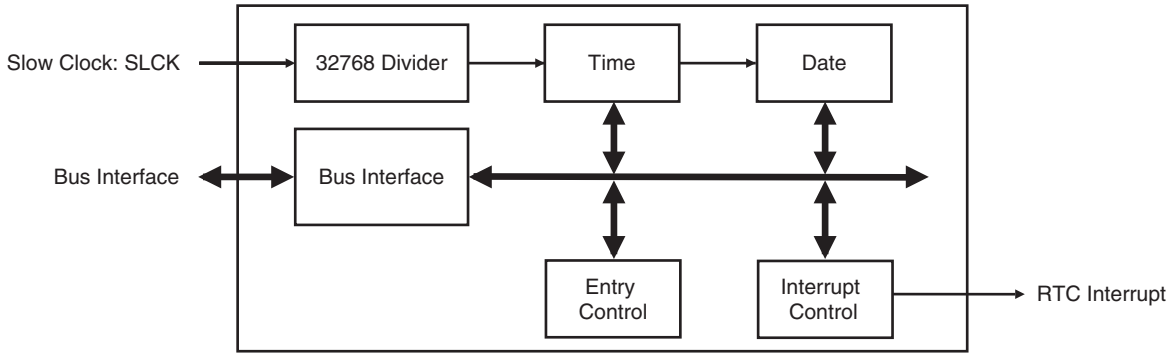
Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

15.2 Embedded Characteristics

- Ultra Low Power Consumption
- Full Asynchronous Design
- Gregorian Calendar up to 2099
- Programmable Periodic Interrupt
- Valid Time and Date Programmation Check

15.3 Block Diagram

Figure 15-1. RTC Block Diagram



15.4 Product Dependencies

15.4.1 Power Management

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

15.4.2 Interrupt

Within the System Controller, the RTC interrupt is OR-wired with all the other module interrupts.

Only one System Controller interrupt line is connected on one of the internal sources of the interrupt controller.

RTC interrupt requires the interrupt controller to be programmed first.

When a System Controller interrupt occurs, the service routine must first determine the cause of the interrupt. This is done by reading each status register of the System Controller peripherals successively.

15.5 Functional Description

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

The valid year range is 1900 to 2099 in Gregorian mode, a two-hundred-year calendar.

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

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

15.5.1 Reference Clock

The reference clock is Slow Clock (SLCK). It can be driven internally or by an external 32.768 kHz crystal.

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

15.5.2 Timing

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

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

15.5.3 Alarm

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

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

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

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

15.5.4 Error Checking when Programming

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

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

The following checks are performed:

1. Century (check if it is in range 19 - 20)
2. Year (BCD entry check)
3. Date (check range 01 - 31)
4. Month (check if it is in BCD range 01 - 12, check validity regarding “date”)
5. Day (check range 1 - 7)
6. Hour (BCD checks: in 24-hour mode, check range 00 - 23 and check that AM/PM flag is not set if RTC is set in 24-hour mode; in 12-hour mode check range 01 - 12)
7. Minute (check BCD and range 00 - 59)
8. Second (check BCD and range 00 - 59)

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

15.5.5 Updating Time/Calendar

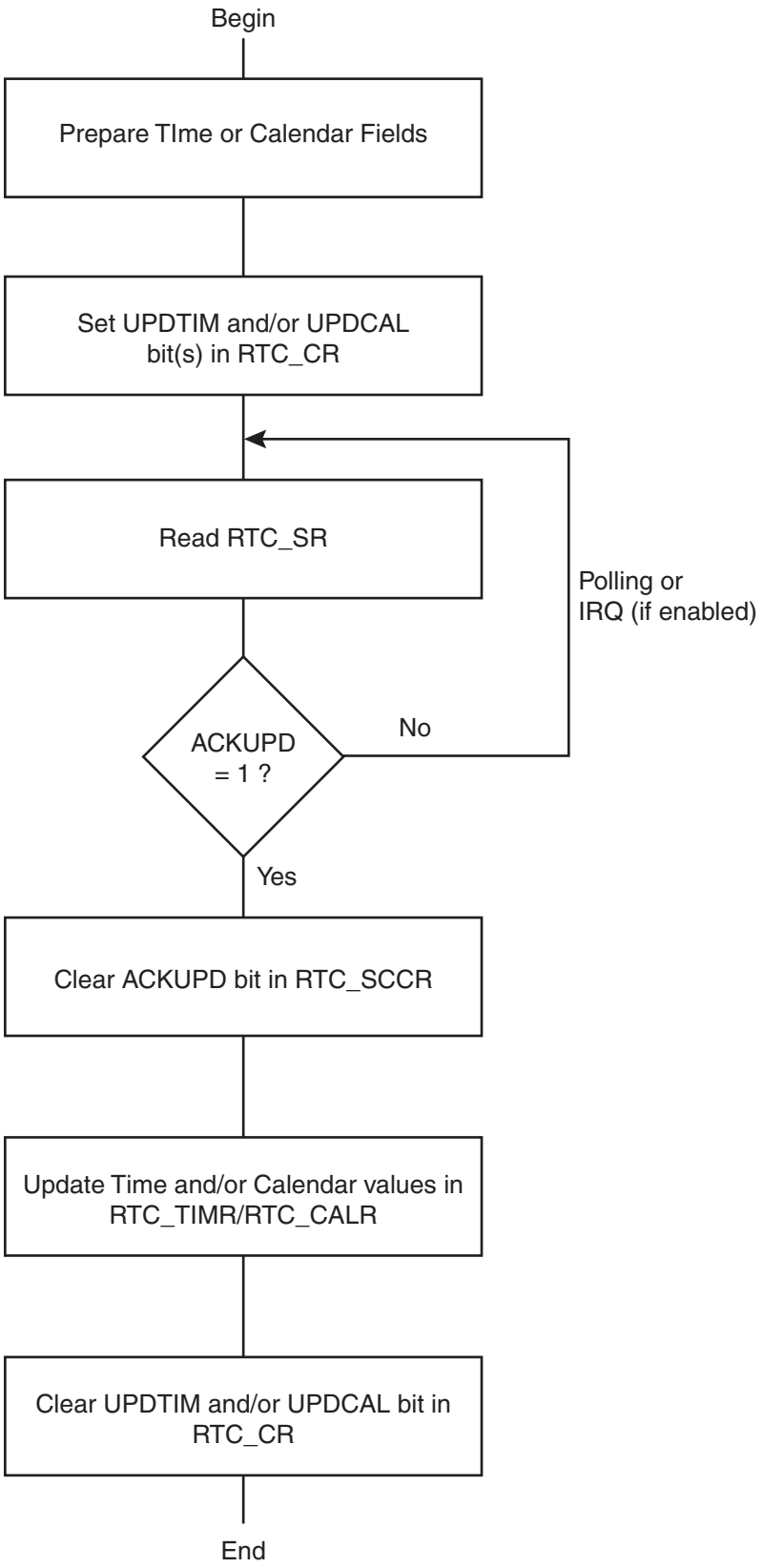
To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register. Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

Then the user must poll or wait for the interrupt (if enabled) of bit ACKUPD in the Status Register. Once the bit reads 1, it is mandatory to clear this flag by writing the corresponding bit in RTC_SCCR. The user can now write to the appropriate Time and Calendar register.

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

When entering programming mode of the calendar fields, the time fields remain enabled. When entering the programming mode of the time fields, both time and calendar fields are stopped. This is due to the location of the calendar logic circuity (downstream for low-power considerations). It is highly recommended to prepare all the fields to be updated before entering programming mode. In successive update operations, the user must wait at least one second after resetting the UPDTIM/UPDCAL bit in the RTC_CR (Control Register) before setting these bits again. This is done by waiting for the SEC flag in the Status Register before setting UPDTIM/UPDCAL bit. After resetting UPDTIM/UPDCAL, the SEC flag must also be cleared.

Figure 15-2. Update Sequence



15.6 Real-time Clock (RTC) User Interface

Table 15-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	RTC_CR	Read-write	0x0
0x04	Mode Register	RTC_MR	Read-write	0x0
0x08	Time Register	RTC_TIMR	Read-write	0x0
0x0C	Calendar Register	RTC_CALR	Read-write	0x01210720
0x10	Time Alarm Register	RTC_TIMALR	Read-write	0x0
0x14	Calendar Alarm Register	RTC_CALALR	Read-write	0x01010000
0x18	Status Register	RTC_SR	Read-only	0x0
0x1C	Status Clear Command Register	RTC_SCCR	Write-only	–
0x20	Interrupt Enable Register	RTC_IER	Write-only	–
0x24	Interrupt Disable Register	RTC_IDR	Write-only	–
0x28	Interrupt Mask Register	RTC_IMR	Read-only	0x0
0x2C	Valid Entry Register	RTC_VER	Read-only	0x0
0x30–0xC4	Reserved Register	–	–	–
0xC8–0xF8	Reserved Register	–	–	–
0xFC	Reserved Register	–	–	–

Note: If an offset is not listed in the table, it must be considered as reserved.

15.6.1 RTC Control Register

Name: RTC_CR

Address: 0xFFFFFEB0

Access: Read-write

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

- **UPDTIM: Update Request Time Register**

0 = No effect.

1 = Stops the RTC time counting.

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

- **UPDCAL: Update Request Calendar Register**

0 = No effect.

1 = Stops the RTC calendar counting.

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

- **TIMEVSEL: Time Event Selection**

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

Value	Name	Description
0	MINUTE	Minute change
1	HOUR	Hour change
2	MIDNIGHT	Every day at midnight
3	NOON	Every day at noon

- **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC_SR depends on the value of CALEVSEL

Value	Name	Description
0	WEEK	Week change (every Monday at time 00:00:00)
1	MONTH	Month change (every 01 of each month at time 00:00:00)
2	YEAR	Year change (every January 1 at time 00:00:00)
3	–	

15.6.2 RTC Mode Register

Name: RTC_MR

Address: 0xFFFFFEB4

Access: Read-write

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

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

0 = 24-hour mode is selected.

1 = 12-hour mode is selected.

15.6.3 RTC Time Register

Name: RTC_TIMR

Address: 0xFFFFFEB8

Access: Read-write

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

23	22	21	20	19	18	17	16
–	AMPM	HOUR					

15	14	13	12	11	10	9	8
–	MIN						

7	6	5	4	3	2	1	0
–	SEC						

- **SEC: Current Second**

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

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

- **MIN: Current Minute**

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

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

- **HOUR: Current Hour**

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

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

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

0 = AM.

1 = PM.

All non-significant bits read zero.

15.6.4 RTC Calendar Register

Name: RTC_CALR

Address: 0xFFFFFEBC

Access: Read-write

31	30	29	28	27	26	25	24
–	–	DATE					

23	22	21	20	19	18	17	16
DAY			MONTH				

15	14	13	12	11	10	9	8
YEAR							

7	6	5	4	3	2	1	0
–	CENT						

- **CENT: Current Century**

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

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

- **YEAR: Current Year**

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

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

- **MONTH: Current Month**

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

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

- **DAY: Current Day in Current Week**

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

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

- **DATE: Current Day in Current Month**

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

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

All non-significant bits read zero.

15.6.5 RTC Time Alarm Register

Name: RTC_TIMALR

Address: 0xFFFFFEC0

Access: Read-write

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

23	22	21	20	19	18	17	16
HOUREN	AMPM	HOUR					

15	14	13	12	11	10	9	8
MINEN	MIN						

7	6	5	4	3	2	1	0
SECEN	SEC						

- **SEC: Second Alarm**

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

- **SECEN: Second Alarm Enable**

0 = The second-matching alarm is disabled.

1 = The second-matching alarm is enabled.

- **MIN: Minute Alarm**

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

- **MINEN: Minute Alarm Enable**

0 = The minute-matching alarm is disabled.

1 = The minute-matching alarm is enabled.

- **HOUR: Hour Alarm**

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

- **AMPM: AM/PM Indicator**

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

- **HOUREN: Hour Alarm Enable**

0 = The hour-matching alarm is disabled.

1 = The hour-matching alarm is enabled.

15.6.6 RTC Calendar Alarm Register

Name: RTC_CALALR

Address: 0xFFFFFEC4

Access: Read-write

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

- **MONTH: Month Alarm**

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

- **MTHEN: Month Alarm Enable**

0 = The month-matching alarm is disabled.

1 = The month-matching alarm is enabled.

- **DATE: Date Alarm**

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

- **DATEEN: Date Alarm Enable**

0 = The date-matching alarm is disabled.

1 = The date-matching alarm is enabled.

15.6.7 RTC Status Register

Name: RTC_SR

Address: 0xFFFFFEC8

Access: Read-only

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

- **ACKUPD: Acknowledge for Update**

0 (FREERUN) = Time and calendar registers cannot be updated.

1 (UPDATE) = Time and calendar registers can be updated.

- **ALARM: Alarm Flag**

0 (NO_ALARMEVENT) = No alarm matching condition occurred.

1 (ALARMEVENT) = An alarm matching condition has occurred.

- **SEC: Second Event**

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

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

- **TIMEV: Time Event**

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

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

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

- **CALEV: Calendar Event**

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

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

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

15.6.8 RTC Status Clear Command Register

Name: RTC_SCCR

Address: 0xFFFFFECC

Access: Write-only

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

- **ACKCLR: Acknowledge Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **ALRCLR: Alarm Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **SECCLR: Second Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **TIMCLR: Time Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **CALCLR: Calendar Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

15.6.9 RTC Interrupt Enable Register

Name: RTC_IER

Address: 0xFFFFFED0

Access: Write-only

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

- **ACKEN: Acknowledge Update Interrupt Enable**

0 = No effect.

1 = The acknowledge for update interrupt is enabled.

- **ALREN: Alarm Interrupt Enable**

0 = No effect.

1 = The alarm interrupt is enabled.

- **SECEN: Second Event Interrupt Enable**

0 = No effect.

1 = The second periodic interrupt is enabled.

- **TIMEN: Time Event Interrupt Enable**

0 = No effect.

1 = The selected time event interrupt is enabled.

- **CALEN: Calendar Event Interrupt Enable**

0 = No effect.

1 = The selected calendar event interrupt is enabled.

15.6.10 RTC Interrupt Disable Register

Name: RTC_IDR

Address: 0xFFFFFED4

Access: Write-only

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

- **ACKDIS: Acknowledge Update Interrupt Disable**

0 = No effect.

1 = The acknowledge for update interrupt is disabled.

- **ALRDIS: Alarm Interrupt Disable**

0 = No effect.

1 = The alarm interrupt is disabled.

- **SECDIS: Second Event Interrupt Disable**

0 = No effect.

1 = The second periodic interrupt is disabled.

- **TIMDIS: Time Event Interrupt Disable**

0 = No effect.

1 = The selected time event interrupt is disabled.

- **CALDIS: Calendar Event Interrupt Disable**

0 = No effect.

1 = The selected calendar event interrupt is disabled.

15.6.11 RTC Interrupt Mask Register

Name: RTC_IMR

Address: 0xFFFFFED8

Access: Read-only

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

- **ACK: Acknowledge Update Interrupt Mask**

0 = The acknowledge for update interrupt is disabled.

1 = The acknowledge for update interrupt is enabled.

- **ALR: Alarm Interrupt Mask**

0 = The alarm interrupt is disabled.

1 = The alarm interrupt is enabled.

- **SEC: Second Event Interrupt Mask**

0 = The second periodic interrupt is disabled.

1 = The second periodic interrupt is enabled.

- **TIM: Time Event Interrupt Mask**

0 = The selected time event interrupt is disabled.

1 = The selected time event interrupt is enabled.

- **CAL: Calendar Event Interrupt Mask**

0 = The selected calendar event interrupt is disabled.

1 = The selected calendar event interrupt is enabled.

15.6.12 RTC Valid Entry Register

Name: RTC_VER

Address: 0xFFFFFEDC

Access: Read-only

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

- **NVTIM: Non-valid Time**

0 = No invalid data has been detected in RTC_TIMR (Time Register).

1 = RTC_TIMR has contained invalid data since it was last programmed.

- **NVCAL: Non-valid Calendar**

0 = No invalid data has been detected in RTC_CALR (Calendar Register).

1 = RTC_CALR has contained invalid data since it was last programmed.

- **NVTIMALR: Non-valid Time Alarm**

0 = No invalid data has been detected in RTC_TIMALR (Time Alarm Register).

1 = RTC_TIMALR has contained invalid data since it was last programmed.

- **NVCALALR: Non-valid Calendar Alarm**

0 = No invalid data has been detected in RTC_CALALR (Calendar Alarm Register).

1 = RTC_CALALR has contained invalid data since it was last programmed.

16. Periodic Interval Timer (PIT)

16.1 Description

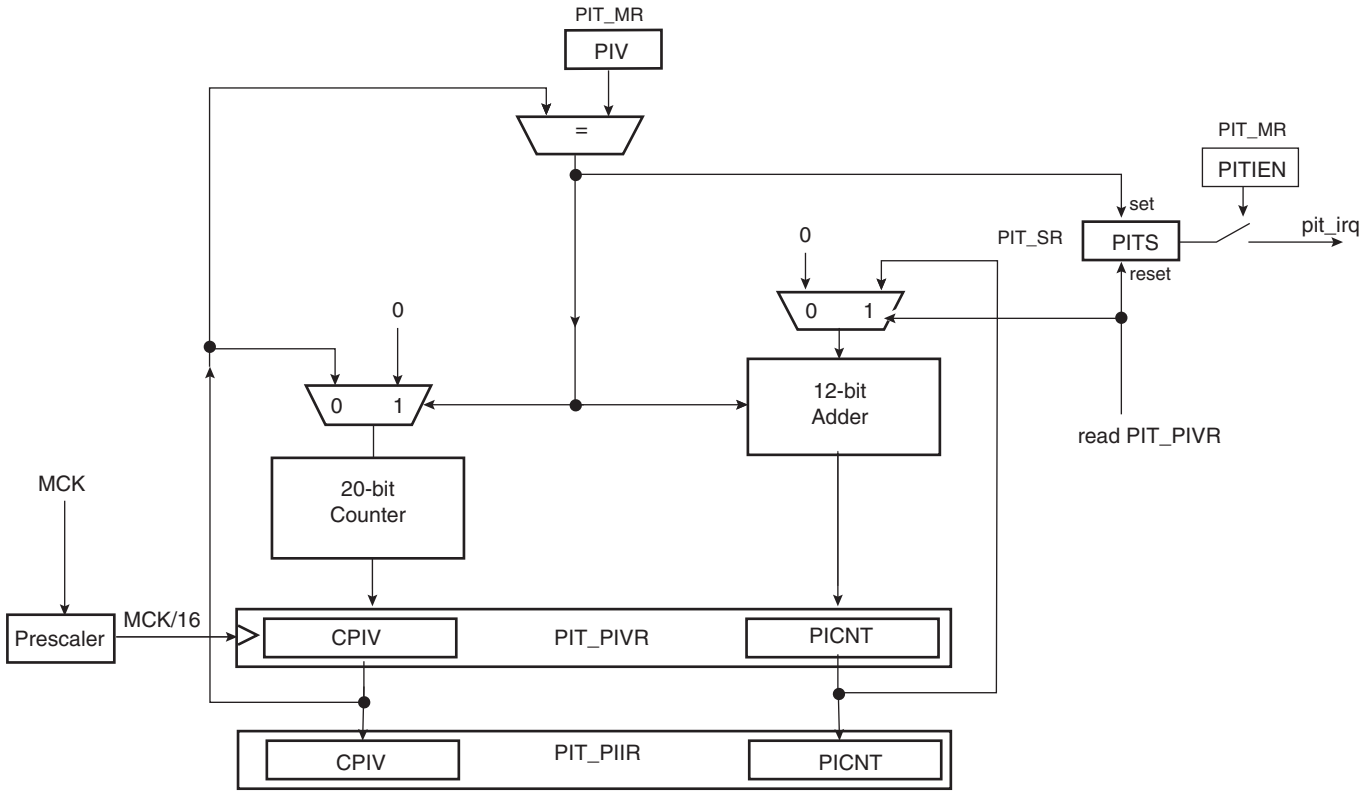
The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

16.2 Embedded Characteristics

- 20-bit Programmable Counter plus 12-bit Interval Counter
- Reset-on-read Feature
- Both Counters Work on Master Clock/16
- Real Time OS or Linux®/WinCE® compliant tick generator
- AMBA™-compliant Interface
  - Interfaces to the ARM Advanced Peripheral Bus

16.3 Block Diagram

Figure 16-1. Periodic Interval Timer



16.4 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.

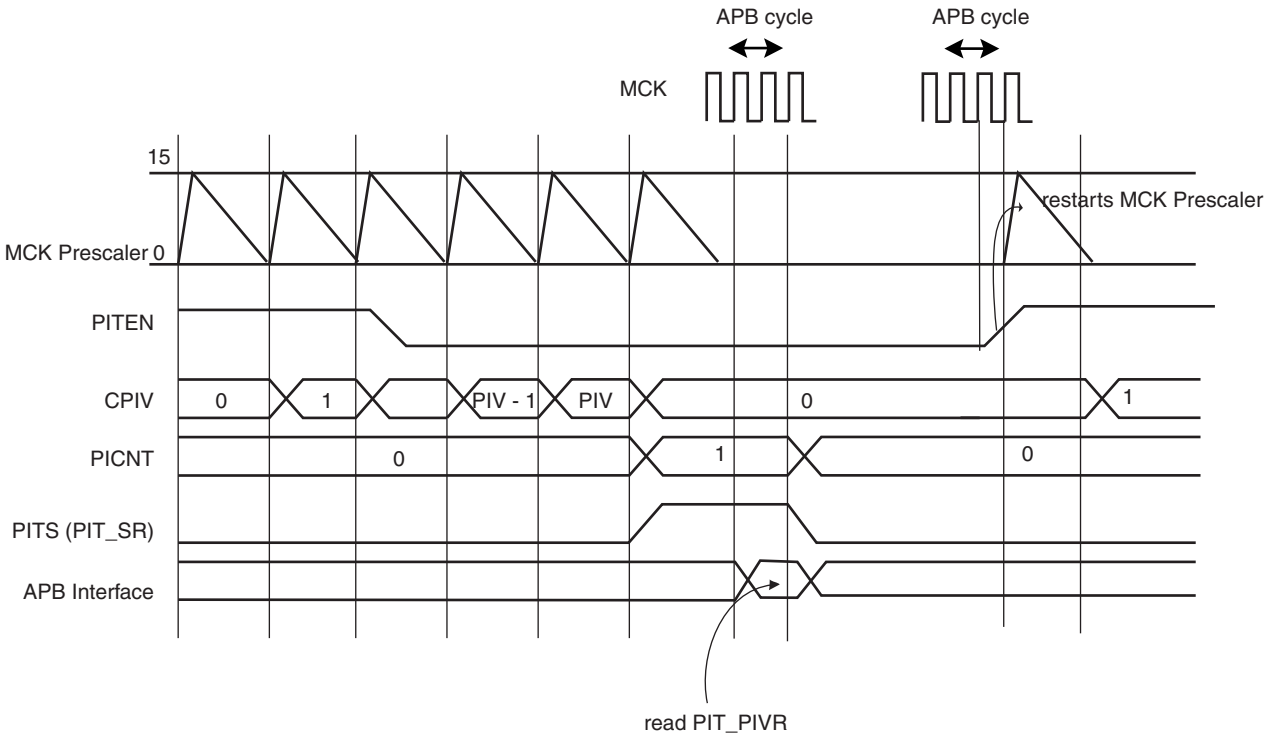
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

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



16.5 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

16.5.1 Periodic Interval Timer Mode Register

Name: PIT_MR

Address: 0xFFFFFE30

Access: Read-write

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

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

15	14	13	12	11	10	9	8
PIV							

7	6	5	4	3	2	1	0
PIV							

- **PIV: Periodic Interval Value**

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

- **PITEN: Period Interval Timer Enabled**

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

1 = The Periodic Interval Timer is enabled.

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

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

1 = The bit PITS in PIT_SR asserts interrupt.

16.5.2 Periodic Interval Timer Status Register

Name: PIT_SR

Address: 0xFFFFFE34

Access: Read-only

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

- **PITS: Periodic Interval Timer Status**

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

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

16.5.3 Periodic Interval Timer Value Register

Name: PIT_PIVR

Address: 0xFFFFFE38

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

16.5.4 Periodic Interval Timer Image Register

Name: PIT_PIIR

Address: 0xFFFFFE3C

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

17. Watchdog Timer (WDT)

17.1 Description

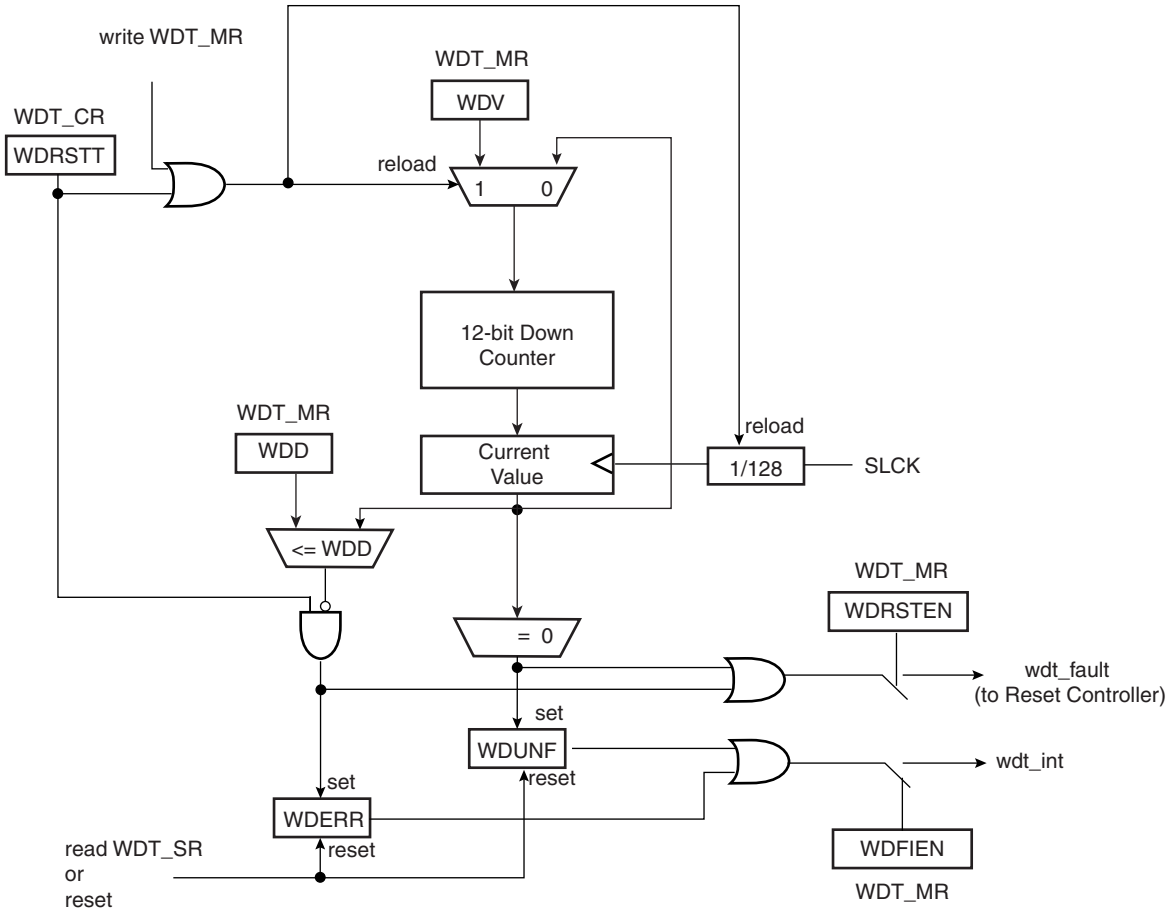
The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock at 32.768 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped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

17.2 Embedded Characteristics

- 12-bit Key-protected Programmable Counter
- Provides Reset or Interrupt Signals to the System
- Counter May Be Stopped While the Processor is in Debug State or in Idle Mode

17.3 Block Diagram

Figure 17-1. Watchdog Timer Block Diagram



17.4 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.

If the watchdog is restarted by writing into the WDT_CR register, the WDT_MR register must not be programmed during a period of time of 3 slow clock periods following the WDT_CR write access. In any case, programming a new value in the WDT_MR register automatically initiates a restart instruction.

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).

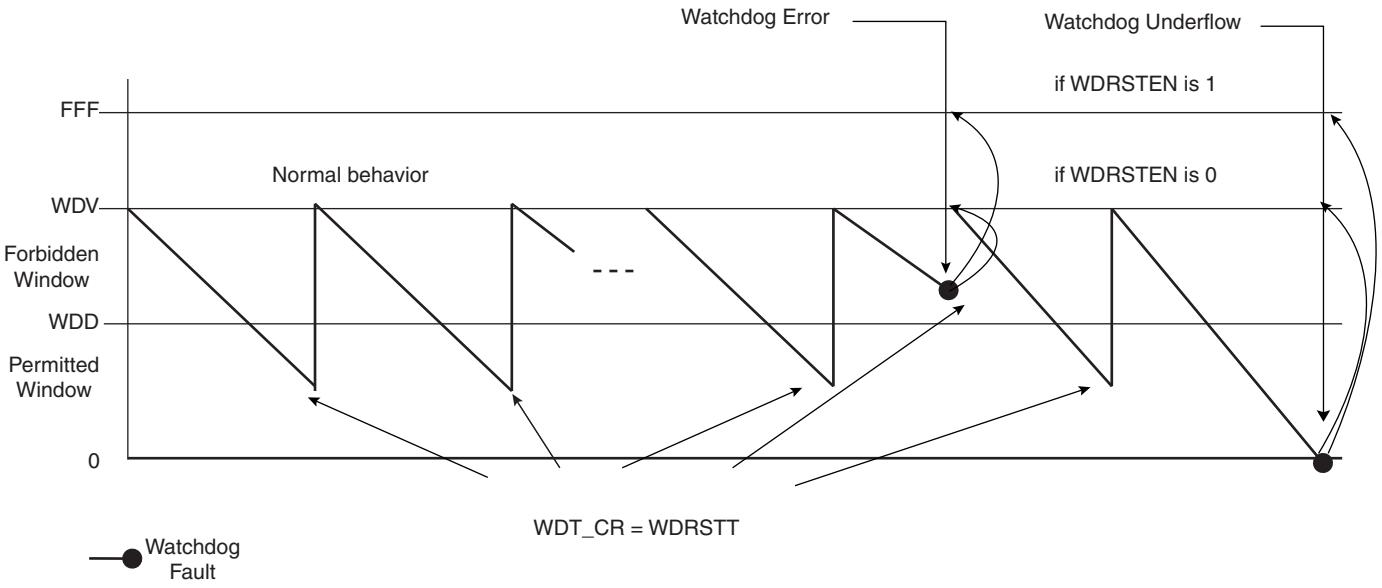
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet.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



17.5 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

17.5.1 Watchdog Timer Control Register

Register Name: WDT_CR

Address: 0xFFFFFE40

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.

17.5.2 Watchdog Timer Mode Register

Register Name: WDT_MR

Address: 0xFFFFFE44

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.

17.5.3 Watchdog Timer Status Register

Register Name: WDT_SR

Address: 0xFFFFFE48

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.

Note: The WDD and WDV values must not be modified within a period of time of 3 slow clock periods following a restart of the watchdog performed by means of a write access in the WDT_CR register, else the watchdog may trigger an end of period earlier than expected.

18. Shutdown Controller (SHDWC)

18.1 Description

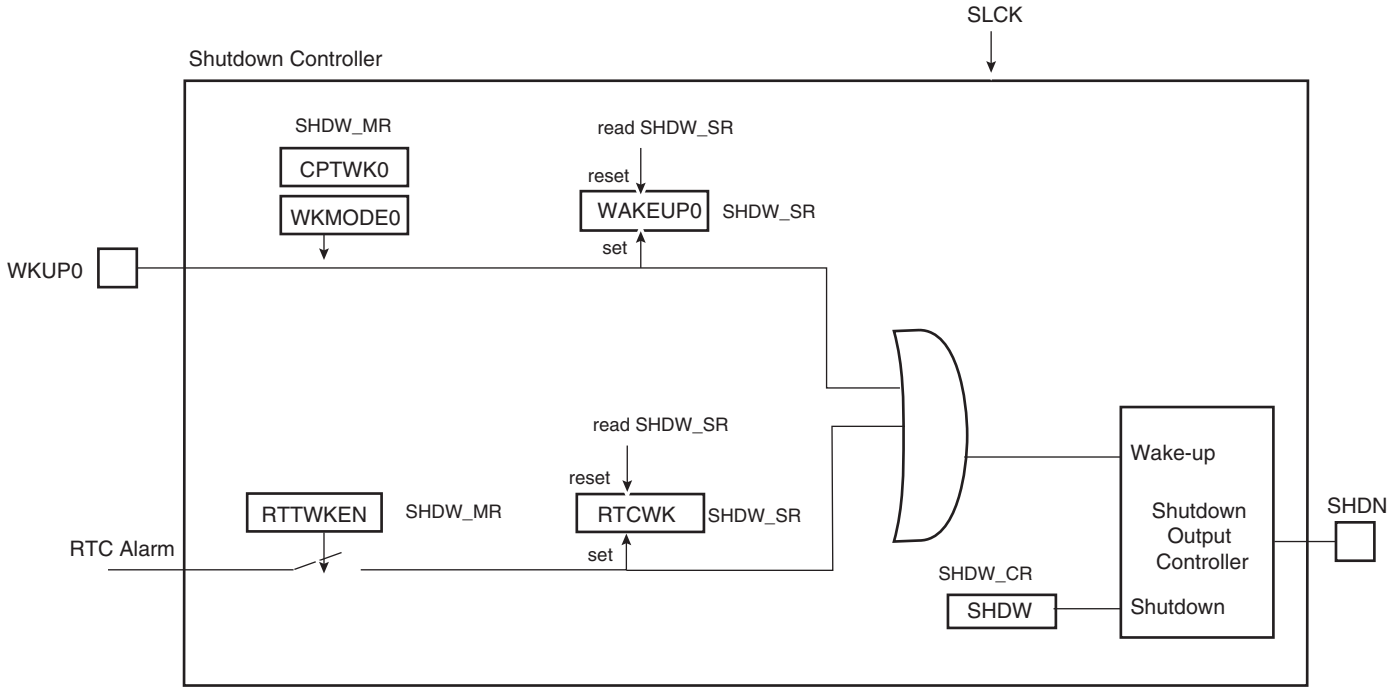
The Shutdown Controller controls the power supplies VDDIO and VDDCORE and the wake-up detection on debounced input lines.

18.2 Embedded Characteristics

- Shutdown and Wake-up Logic
  - Software Assertion of the SHDW Output Pin
  - Programmable De-assertion from the WKUP Input Pins

18.3 Block Diagram

Figure 18-1. Shutdown Controller Block Diagram



18.4 I/O Lines Description

Table 18-1. I/O Lines Description

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

18.5 Product Dependencies

18.5.1 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.

18.6 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 RTC alarm (the detection of the rising edge of the RTC alarm is synchronized with SLCK). This is done by writing the SHDW_MR register using the RTCWKEN field. When enabled, the detection of the RTC alarm is reported in the RTCWK bit of the SHDW_SR Status register. It is reset after the read of SHDW_SR. When using the RTC alarm to wake up the system, the user must ensure that the RTC alarm status flag is cleared before shutting down the system.Otherwise, no rising edge of the status flag may be detected and the wake-up fails fail.

18.7 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_0003
0x08	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

18.7.1 Shutdown Control Register

Name: SHDW_CR

Address: 0xFFFFFE10

Access: Write-only

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

- **SHDW: Shutdown Command**

0 = No effect.

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

- **KEY: Password**

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

18.7.2 Shutdown Mode Register

Name: SHDW_MR

Address: 0xFFFFFE14

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RTCWKEN	–
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.

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

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

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

18.7.3 Shutdown Status Register

Name: SHDW_SR

Address: 0xFFFFFE18

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RTCWK	–
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.

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

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

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

19. General Purpose Backup Registers (GPBR)

19.1 Description

The System Controller embeds Four General-purpose Backup Registers.

19.2 Embedded Characteristics

- Four 32-bit General Purpose Backup Registers

19.3 General Purpose Backup Registers (GPBR) User Interface

Table 19-1. Register Mapping

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

19.3.1 General Purpose Backup Register x

Name: SYS_GPBRx

Address: 0xFFFFFE60

Access: Read-write

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

- **GPBR_VALUE: Value of GPBR x**

20. Slow Clock Controller (SCKC)

20.1 Description

The System Controller embeds a Slow Clock Controller.

The slow clock can be generated either by an external 32768 Hz crystal oscillator or by the on-chip 32 kHz RC oscillator. The 32768 Hz crystal oscillator can be bypassed by setting the OSC32BYP bit to accept an external slow clock on XIN32.

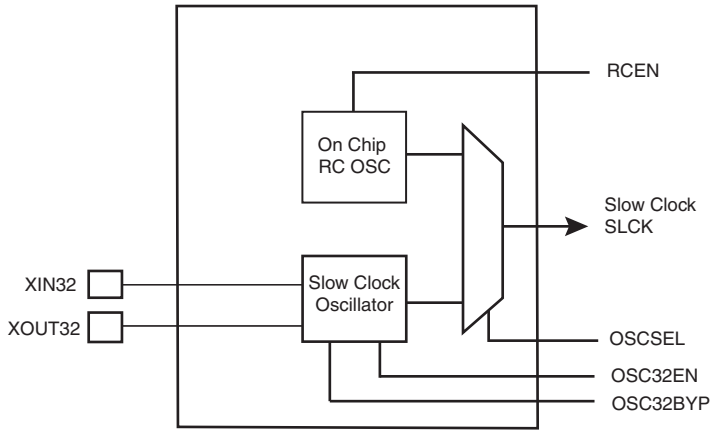
The internal 32 kHz RC oscillator and the 32768 Hz oscillator can be enabled by setting to 1, respectively, RCEN bit and OSC32EN bit in the System Controller user interface. The OSCSEL command selects the slow clock source.

20.2 Embedded Characteristics

- 32 kHz RC Oscillator or 32768 Hz Crystal Oscillator Selector
- VDDBU Powered

20.3 Block Diagram

Figure 20-1. Block Diagram



RCEN, OSC32EN, OSCSEL and OSC32BYP bits are located in the Slow Clock Configuration Register (SCKC_CR) located at the address 0xFFFFFE50 in the backed up part of the System Controller and, thus, they are preserved while VDDBU is present.

After a VDDBU power on reset, the default configuration is RCEN = 1, OSC32EN = 0 and OSCSEL = 0, allowing the system to start on the internal 32 kHz RC oscillator.

The programmer controls the slow clock switching by software and so must take precautions during the switching phase.

20.3.1 Switch from Internal 32 kHz RC Oscillator to 32768 Hz Crystal Oscillator

To switch from the internal 32 kHz RC oscillator to the 32768 Hz crystal oscillator, the programmer must execute the following sequence:

- Switch the master clock to a source different from slow clock (PLL or Main Oscillator) through the Power Management Controller.
- Enable the 32768 Hz oscillator by setting the bit OSC32EN to 1.
- Wait 32768 Hz Startup Time for clock stabilization (software loop).
- Switch from internal 32 kHz RC oscillator to 32768 Hz oscillator by setting the bit OSCSEL to 1.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32 kHz RC oscillator by setting the bit RCEN to 0.

20.3.2 Bypass the 32768 Hz Oscillator

The following steps must be added to bypass the 32768 Hz oscillator:

- An external clock must be connected on XIN32.
- Enable the bypass path OSC32BYP bit set to 1.
- Disable the 32768 Hz oscillator by setting the OSC32EN bit to 0.

20.3.3 Switch from 32768 Hz Crystal Oscillator to Internal 32 kHz RC Oscillator

The same procedure must be followed to switch from the 32768 Hz crystal oscillator to the internal 32 kHz RC oscillator:

- Switch the master clock to a source different from slow clock (PLL or Main Oscillator).
- Enable the internal 32 kHz RC oscillator for low power by setting the bit RCEN to 1
- Wait internal 32 kHz RC Startup Time for clock stabilization (software loop).
- Switch from 32768 Hz oscillator to internal RC by setting the bit OSCSEL to 0.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32768 Hz oscillator by setting the bit OSC32EN to 0.

20.4 Slow Clock Configuration (SCKC) User Interface

Table 20-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Slow Clock Configuration Register	SCKC_CR	Read-write	0x0000_0001

20.4.1 Slow Clock Configuration Register

Name: SCKC_CR

Address: 0xFFFFFE50

Access: Read-write

Reset: 0x0000_0001

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCSEL	OSC32BYP	OSC32EN	RCEN

- **RCEN: Internal 32 kHz RC Oscillator**

0: 32 kHz RC oscillator is disabled.

1: 32 kHz RC oscillator is enabled.

- **OSC32EN: 32768 Hz Oscillator**

0: 32768 Hz oscillator is disabled.

1: 32768 Hz oscillator is enabled.

- **OSC32BYP: 32768 Hz Oscillator Bypass**

0: 32768 Hz oscillator is not bypassed.

1: 32768 Hz oscillator is bypassed, accept an external slow clock on XIN32.

- **OSCSEL: Slow Clock Selector**

0 (RC): Slow clock is internal 32 kHz RC oscillator.

1 (XTAL): Slow clock is 32768 Hz oscillator.

21. Clock Generator (CKGR)

21.1 Description

The Clock Generator User Interface is embedded within the Power Management Controller and is described in Section 22.13 ”Power Management Controller (PMC) User Interface”. However, the Clock Generator registers are named CKGR_.

21.2 Embedded Characteristics

The Clock Generator is made up of:

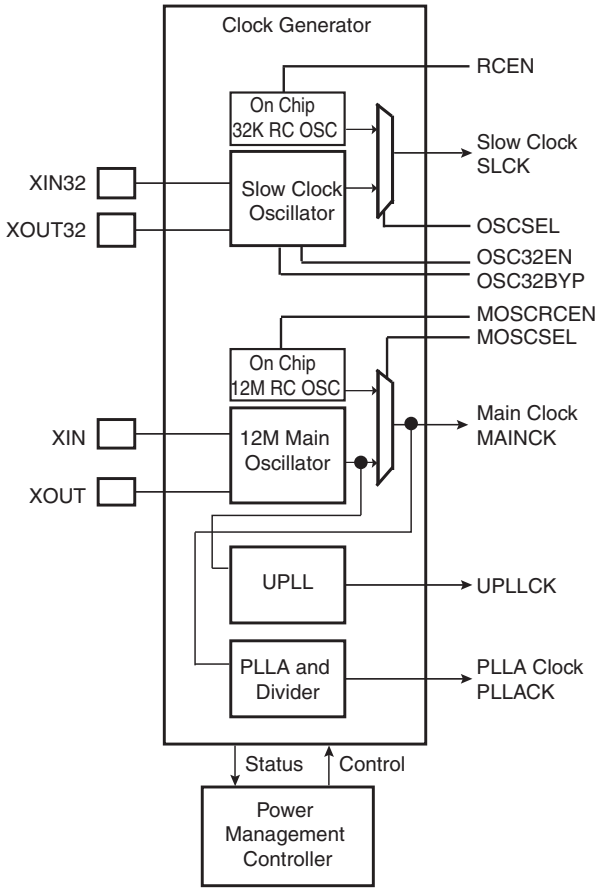
- A Low Power 32768 Hz Slow Clock Oscillator with bypass mode
- A Low Power RC Oscillator
- A 12 to 16 MHz Crystal Oscillator, which can be bypassed (12 MHz needed in case of USB)
- A Fast RC Oscillator, at 12 MHz.
- A 480 MHz UTMI PLL providing a clock for the USB High Speed Device Controller
- A 400 to 800 MHz programmable PLL (input from 8 to 16 MHz), capable of providing the clock MCK to the processor and to the peripherals.

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 Clock Oscillator selection: either Crystal Oscillator or 12 MHz Fast RC Oscillator
- PLLACK is the output of the Divider and 400 to 800 MHz programmable PLL (PLLA)
- UPLLCK is the output of the 480 MHz UTMI PLL (UPLL)

21.3 CKGR Block Diagram

Figure 21-1. Clock Generator Block Diagram

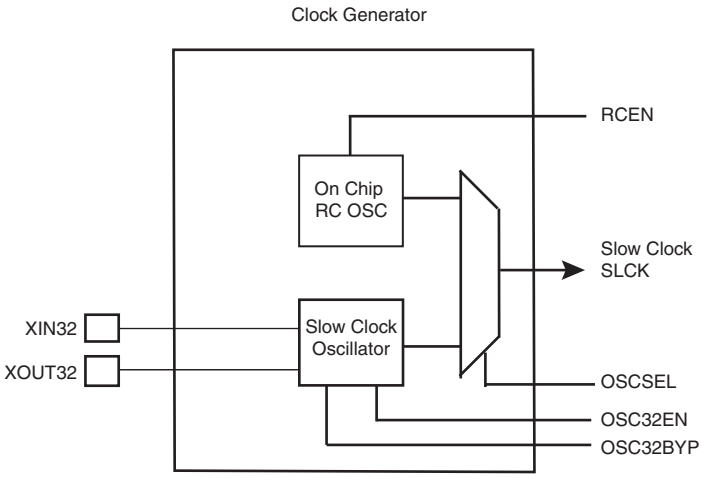


21.4 Slow Clock Selection

The slow clock can be generated either by an external 32768 Hz crystal or by the on-chip 32 kHz RC oscillator. The 32768 Hz crystal oscillator can be bypassed by setting the bit OSC32BYP to accept an external slow clock on XIN32.

The internal 32 kHz RC oscillator and the 32768 Hz oscillator can be enabled by setting to 1, respectively, RCEN bit and OSC32EN bit in the System Controller user interface. The OSCSEL command selects the slow clock source.

Figure 21-2. Slow Clock



RCEN, OSC32EN,OSCSEL and OSC32BYP bits are located in the Slow Clock Control Register (SCKCR) located at address 0xFFFFFE50 in the backed up part of the System Controller and so are preserved while VDDBU is present.

After a VDDBU power on reset, the default configuration is RCEN = 1, OSC32EN = 0 and OSCSEL = 0, BYPASS = 0, allowing the system to start on the internal 32 kHz RC oscillator.

The programmer controls the slow clock switching by software and so must take precautions during the switching phase.

21.4.1 Switch from Internal 32 kHz RC Oscillator to the 32768 Hz Crystal

To switch from internal 32 kHz RC oscillator to the 32768 Hz crystal, the programmer must execute the following sequence:

- Switch the master clock to a source different from slow clock (PLL or Main Oscillator) through the Power Management Controller.
- Enable the 32768 Hz oscillator by setting the bit OSC32EN to 1.
- Wait 32768 Hz Startup Time for clock stabilization (software loop).
- Switch from internal 32 kHz RC to 32768 Hz oscillator by setting the bit OSCSEL to 1.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32 kHz RC oscillator by setting the bit RCEN to 0.
- Switch the master clock back to the slow clock domain

21.4.2 Bypass the 32768 Hz Oscillator

The following step must be added to bypass the 32768 Hz Oscillator.

- An external clock must be connected on XIN32.
- Enable the bypass path OSC32BYP bit set to 1.
- Disable the 32768 Hz oscillator by setting the bit OSC32EN to 0.

21.4.3 Switch from the 32768 Hz Crystal to Internal 32 kHz RC Oscillator

The same procedure must be followed to switch from a 32768 Hz crystal to the internal 32 kHz RC oscillator.

- Switch the master clock to a source different from slow clock (PLL or Main Oscillator).
- Enable the internal 32 kHz RC oscillator for low power by setting the bit RCEN to 1
- Wait internal 32 kHz RC Startup Time for clock stabilization (software loop).
- Switch from 32768 Hz oscillator to internal RC by setting the bit OSCSEL to 0.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32768 Hz oscillator by setting the bit OSC32EN to 0.
- Switch the master clock back to the slow clock domain

21.4.4 Slow Clock Configuration Register

Name: SCKCR

Address: 0xFFFFFE50

Access: Read-write

Reset Value: 0x0000_0001

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCSEL	OSC32BYP	OSC32EN	RCEN

- **RCEN: Internal 32 kHz RC**

0: 32 kHz RC is disabled

1: 32 kHz RC is enabled

- **OSC32EN: 32768 Hz oscillator**

0: 32768 Hz oscillator is disabled

1: 32768 Hz oscillator is enabled

- **OSC32BYP: 32768 Hz oscillator bypass**

0: 32768 Hz oscillator is not bypassed

1: 32768 Hz oscillator is bypassed, accept an external slow clock on XIN32

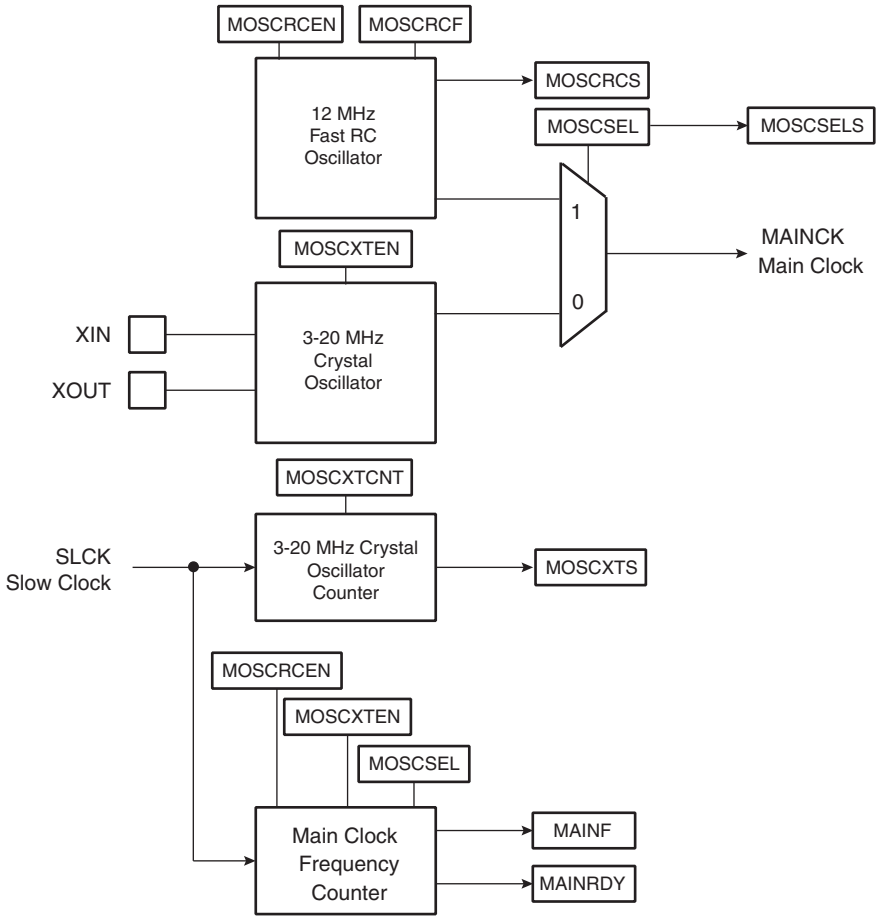
- **OSCSEL: Slow clock selector**

0: Slow clock is internal 32 kHz RC

1: Slow clock is 32768 Hz oscillator

21.5 Main Clock

Figure 21-3. Main Clock Block Diagram



The Main Clock has two sources:

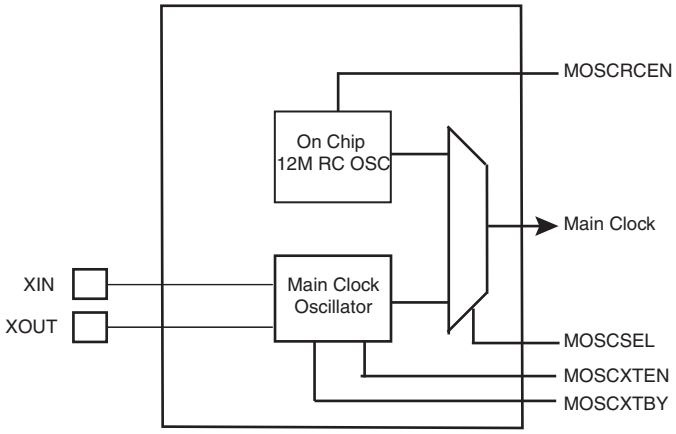
- 12 MHz Fast RC Oscillator which starts very quickly and is used at startup
- 12 to 16 MHz Crystal Oscillator, which can be bypassed

21.6 Main Clock Selection

The main clock can be generated either by an external 12 MHz crystal oscillator or by the on-chip 12 MHz RC oscillator. This fast RC oscillator allows the processor to start or restart in a few microseconds when 12 MHz internal RC is selected.

The 12 MHz crystal oscillator can be bypassed by setting the bit MOSCXTBY to accept an external main clock on XIN.

Figure 21-4. Main Clock Selection



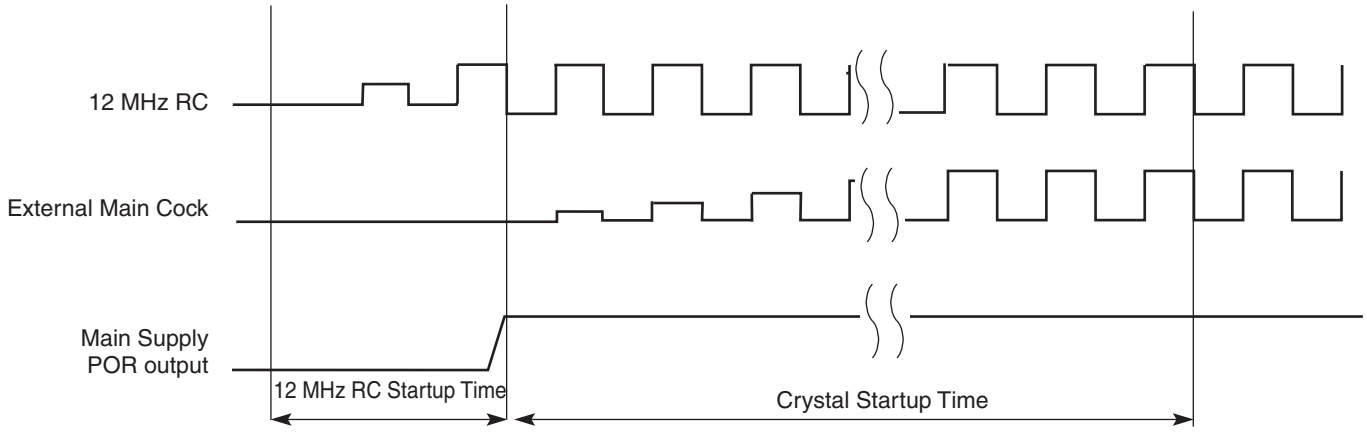
MOSCRCEN, MOSCXTEN, MOSCSEL and MOSCXTBY bits are located in the PMC Clock Generator Main Oscillator Register (CKGR_MOR).

After a VDDBU power on reset, the default configuration is MOSCRCEN = 1, MOSCXTEN = 0 and MOSCSEL = 0, the 12 MHz RC oscillator is started as Main clock.

21.6.1 Fast wake-up

To speed up the wake-up phase, the system boots on 12 MHz RC (Main Clock). This allows the user to perform system configuration (PLL, DDR2, etc.) at 12 MHz instead of 32 kHz during 12 MHz oscillator start-up.

Figure 21-5. PMC Startup



21.6.2 Switch from Internal 12 MHz RC Oscillator to the 12 MHz Crystal

For USB operations an external 12 MHz crystal is required for better accuracy.

The programmer controls the main clock switching by software and so must take precautions during the switching phase.

To switch from internal 12 MHz RC oscillator to the 12 MHz crystal, the programmer must execute the following sequence:

- Enable the 12 MHz oscillator by setting the bit MOSCXTEN to 1.
- Wait that the 12 MHz oscillator status bit MOSCXTS is 1.
- Switch from internal 12 MHz RC oscillator to the 12 MHz oscillator by setting the bit MOSCSEL to 1.
- If not the bit MOSCSEL is set to 0 by the PMC.
- Disable the 12 MHz RC oscillator by setting the bit MOSCRCEN to 0.

21.6.3 Bypass the 12 MHz Oscillator

Following step must be added to bypass the 12 MHz Oscillator.

- An external clock must be connected on XIN.
- Enable the bypass path MOSCXTBY bit set to 1.
- Disable the 12 MHz oscillator by setting the bit MOSCXTEN to 0.

21.6.4 Switch from the 12 MHz Crystal to Internal 12 MHz RC Oscillator

The same procedure must be followed to switch from a 12 MHz crystal to the internal 12 MHz RC oscillator.

- Enable the internal 12 MHz RC oscillator for low power by setting the bit MOSCRCEN to 1
- Wait internal 12 MHz RC Startup Time for clock stabilization (software loop).
- Switch from 12 MHz oscillator to internal 12 MHz RC oscillator by setting the bit MOSCSEL to 0.
- Disable the 12 MHz oscillator by setting the bit MOSCXTEN to 0.

21.6.5 12 MHz Fast RC Oscillator

After reset, the 12 MHz Fast RC Oscillator is enabled and it is selected as the source of MCK. MCK is the default clock selected to start up the system.

Please refer to the “DC Characteristics” section of the product datasheet.

The software can disable or enable the 12 MHz Fast RC Oscillator with the MOSCRCEN bit in the Clock Generator Main Oscillator Register (CKGR_MOR).

When disabling the Main Clock by clearing the MOSCRCEN bit in CKGR_MOR, the MOSCRCS bit in the Power Management Controller Status Register (PMC_SR) is automatically cleared, indicating the Main Clock is off.

Setting the MOSCRCS bit in the Power Management Controller Interrupt Enable Register (PMC_IER) can trigger an interrupt to the processor.

21.6.6 12 to 16 MHz Crystal Oscillator

After reset, the 12 to 16 MHz Crystal Oscillator is disabled and it is not selected as the source of MAINCK.

The user can select the 12 to 16 MHz crystal oscillator to be the source of MAINCK, as it provides a more accurate frequency. The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCXTEN bit in the Main Oscillator Register (CKGR_MOR).

When disabling the main oscillator by clearing the MOSCXTEN bit in CKGR_MOR, the MOSCXTS 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 oscillator.

When the MOSCXTEN bit and the MOSCXTCNT are written in CKGR_MOR to enable the main oscillator, the MOSCXTS bit in the Power Management Controller Status Register (PMC_SR) is cleared and the counter starts

counting down on the slow clock divided by 8 from the MOSCXTCNT value. Since the MOSCXTCNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCXTS bit is set, indicating that the main clock is valid. Setting the MOSCXTS bit in PMC_IMR can trigger an interrupt to the processor.

21.6.7 Main Clock Oscillator Selection

The user can select either the 12 MHz Fast RC Oscillator or the 12 to 16 MHz Crystal Oscillator to be the source of Main Clock.

The advantage of the 12 MHz Fast RC Oscillator is to have fast startup time, this is why it is selected by default (to start up the system) and when entering in Wait Mode.

The advantage of the 12 to 16 MHz Crystal Oscillator is that it is very accurate.

The selection is made by writing the MOSCSEL bit in the Main Oscillator Register (CKGR_MOR). The switch of the Main Clock source is glitch free, so there is no need to run out of SLCK, PLLACK or UPLLCK in order to change the selection. The MOSCSELS bit of the Power Management Controller Status Register (PMC_SR) allows knowing when the switch sequence is done.

Setting the MOSCSELS bit in PMC_IMR can trigger an interrupt to the processor.

21.6.8 Main Clock Frequency Counter

The device features a Main Clock frequency counter that provides the frequency of the Main Clock.

The Main Clock frequency counter is reset and starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock in the following cases:

- When the 12 MHz Fast RC Oscillator clock is selected as the source of Main Clock and when this oscillator becomes stable (i.e., when the MOSCRCS bit is set)
- When the 12 to 16 MHz Crystal Oscillator is selected as the source of Main Clock and when this oscillator becomes stable (i.e., when the MOSCXTS bit is set)
- When the Main Clock Oscillator selection is modified

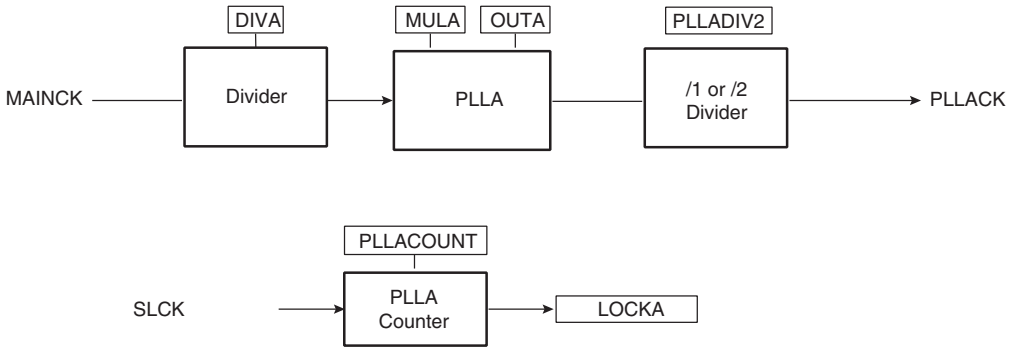
Then, at the 16th falling edge of Slow Clock, the MAINFRDY bit in the Clock Generator Main Clock Frequency Register (CKGR_MCFR) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the 12 MHz Fast RC Oscillator or 12 to 16 MHz Crystal Oscillator can be determined.

21.7 Divider and PLLA Block

The PLLA embeds an input divider to increase the accuracy of the resulting clock signals. However, the user must respect the PLLA minimum input frequency when programming the divider.

Figure 21-6 shows the block diagram of the divider and PLLA block.

Figure 21-6. Divider and PLLA Block Diagram



21.7.1 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 PLLA allows multiplication of the divider's outputs. The PLLA clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIVA and MULA. The factor applied to the source signal frequency is (MULA + 1)/DIVA. When MULA is written to 0, the PLLA is disabled and its power consumption is saved. Re-enabling the PLLA can be performed by writing a value higher than 0 in the MUL field.

Whenever the PLLA is re-enabled or one of its parameters is changed, the LOCKA bit in PMC_SR is automatically cleared. The values written in the PLLACOUNT field in CKGR_PLLAR are loaded in the PLLA counter. The PLLA 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 PLLA transient time into the PLLACOUNT field.

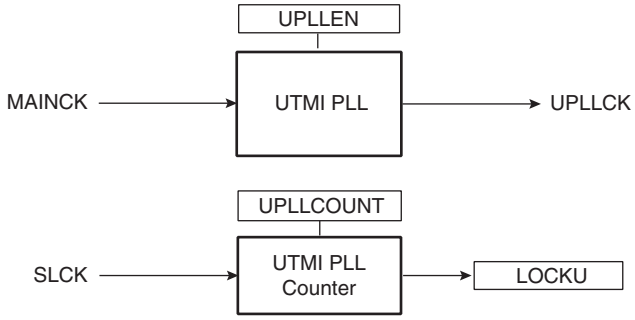
The PLLA clock can be divided by 2 by writing the PLLADIV2 bit in PMC_MCKR register.

21.8 UTMI Phase Lock Loop Programming

The source clock of the UTMI PLL is the Main Clock MAINCK. When the 12 MHz Fast RC Oscillator is selected as the source of MAINCK, the 12 MHz frequency must also be selected because the UTMI PLL multiplier contains a built-in multiplier of x 40 to obtain the USB High Speed 480 MHz.

A 12 MHz crystal is needed to use the USB.

Figure 21-7. UTMI PLL Block Diagram



Whenever the UTMI PLL is enabled by writing UPLLEN in CKGR_UCKR, the LOCKU bit in PMC_SR is automatically cleared. The values written in the PLLCOUNT field in CKGR_UCKR are loaded in the UTMI PLL counter. The UTMI PLL counter then decrements at the speed of the Slow Clock divided by 8 until it reaches 0. At this time, the LOCKU bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the UTMI PLL transient time into the PLLCOUNT field.

22. Power Management Controller (PMC)

22.1 Description

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the Core.

22.2 Embedded Characteristics

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

PMC input clocks:

- UPLLCK : From UTMI PLL
- PLLACK : From PLLA
- SLCK: slow clock from external 32 kHz oscillator or internal 32 kHz RC oscillator
- MAINCK: Main Clock from external 12 MHz oscillator or internal 12 MHz RC Oscillator

PMC output clocks:

- Processor Clock PCK.
- Master Clock MCK, in particular to the Matrix, the memory interfaces, the peripheral bridge. The divider can be 2, 3 or 4.
- Each peripheral embeds its own divider, programmable in the PMC User Interface.
- 133 MHz DDR clock

Note: DDR clock is not available when Master Clock (MCK) equals Processor Clock (PCK).

- USB Host EHCI High speed clock (UPLLCK)
- USB OHCI clocks (UHP48M and UHP12M)
- Two programmable clock outputs: PCK0 and PCK1
- SMD clock

This allows software control of five flexible operating modes:

- Normal Mode, processor and peripherals running at a programmable frequency
- Idle Mode, processor stopped waiting for an interrupt
- Slow Clock Mode, processor and peripherals running at low frequency
- Standby Mode, mix of Idle and Backup Mode, peripheral running at low frequency, processor stopped waiting for an interrupt
- Backup Mode, Main Power Supplies off, VDDBU powered by a battery

22.3 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.

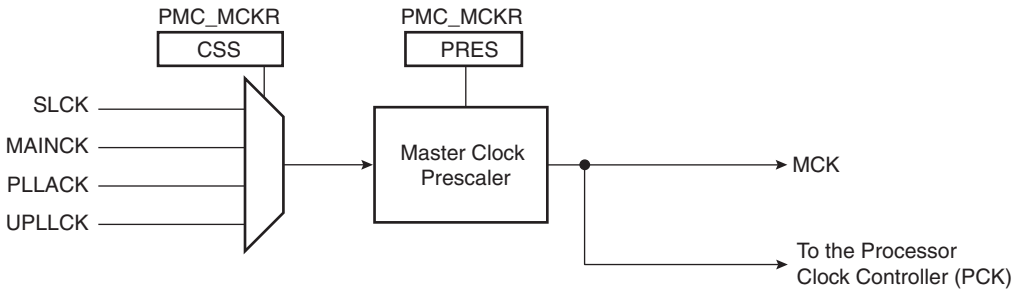
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

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, and the division by 3. The PRES field in PMC_MCKR programs the prescaler.

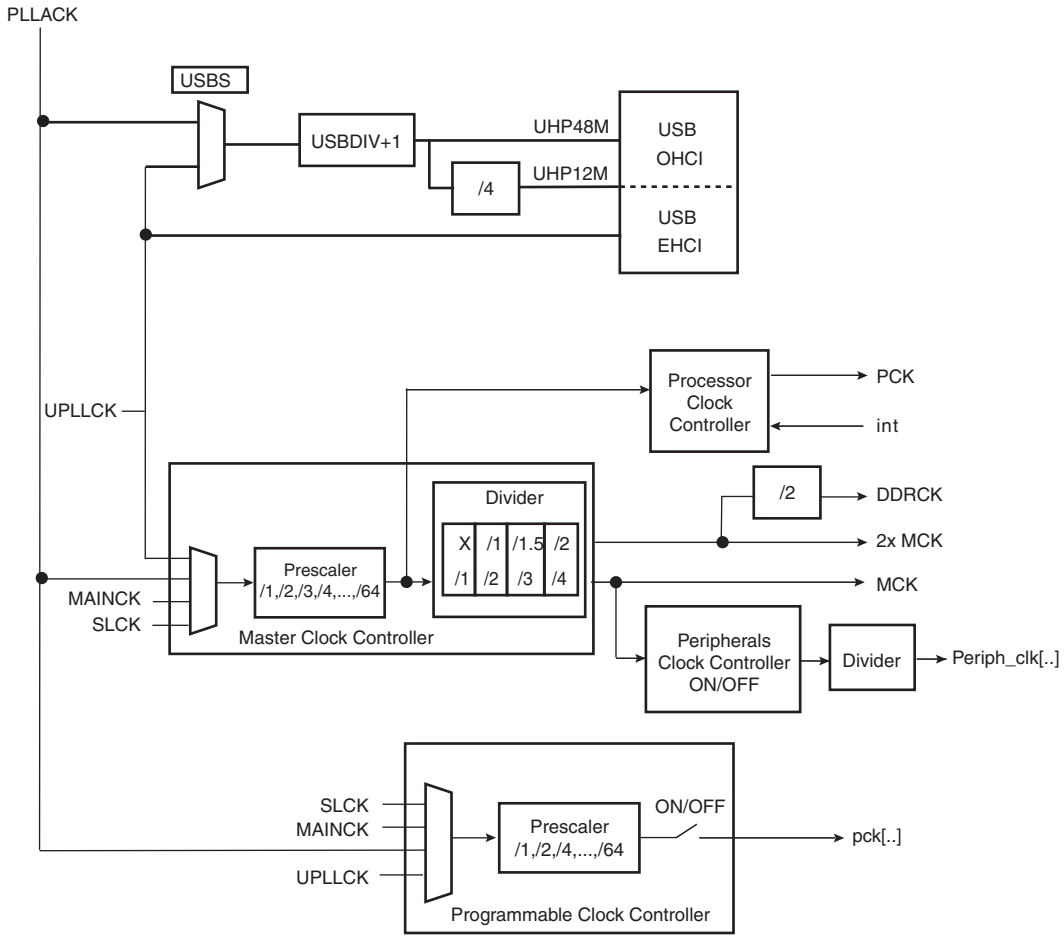
Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

Figure 22-1. Master Clock Controller



22.4 Block Diagram

Figure 22-2. General Clock Block Diagram



22.5 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the Processor Idle Mode. The Processor Clock can be disabled by writing the System Clock Disable Register (PMC_SCDR). The status of this clock (at least for debug purpose) can be read in the System Clock Status Register (PMC_SCSR).

The Processor Clock PCK is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock and entering Wait for Interrupt Mode. The Processor Clock is automatically re-enabled by any enabled fast or normal interrupt, or by reset of the product.

Note: The ARM Wait for Interrupt mode is entered by means of CP15 coprocessor operation. Refer to the Atmel application note, Optimizing Power Consumption for 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.

22.6 USB Device and Host Clocks

The USB Device and Host High Speed ports clocks are controlled by the UDPHS and UHPHS bits in PMC_PCER. To save power on this peripheral when they are is not used, the user can set these bits in PMC_PCDR. The UDPHS and UHPHS bits in PMC_PCR give the activity of these clocks.

The PMC also provides the clocks UHP48M and UHP12M to the USB Host OHCI. The USB Host OHCI clocks are controlled by 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 OHCI requires both the 12/48 MHz signal and the Master Clock. USBDIV field in PMC_USB register is to be programmed to 9 (division by 10) for normal operations.

To save more power consumption the user can stop UTMI PLL, in this case USB high-speed operations are not possible. Nevertheless, as the USB OHCI Input clock can be selected with USBS bit (PLLA or UTMI PLL) in PMC_USB register, OHCI full-speed operation remain possible.

The user must program the USB OHCI Input Clock and the USBDIV divider in PMC_USB register to generate a 48 MHz and a 12 MHz signal with an accuracy of ± 0.25%.

22.7 LP-DDR/DDR2 Clock

The Power Management Controller controls the clocks of the DDR memory.

The DDR clock can be enabled and disabled with DDRCK bit respectively in PMC_SCER and PMC_SDER registers. At reset DDR clock is disabled to save power consumption.

In the case MDIV = ‘00’, (PCK = MCK) and DDRCK clock is not available.

If Input clock is PLLACK/PLLADIV2 the DDR Controller can drive DDR2 and LP-DDR at up to 133 MHz with MDIV = ‘11’.

To save PLLA power consumption, the user can choose UPLLCK an Input clock for the system. In this case the DDR Controller can drive LD-DDR at up to 120 MHz.

22.8 Software Modem Clock

The Power Management Controller controls the clocks of the Software Modem.

SMDCK is a division of UPLL or PLLA.

22.9 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by means of the Peripheral Clock Controller. The user can individually enable and disable the clock on the peripherals and select a division factor from MCK. This is done through the Peripheral Control Register (PMC_PCR).

In order to save power consumption, the division factor can be 1, 2, 4 or 8. PMC_PCR is a register that features a command and acts like a mailbox. To write the division factor on a particular peripheral, the user needs to write a WRITE command, the peripheral ID and the chosen division factor. To read the current division factor on a particular peripheral, the user just needs to write the READ command and the peripheral ID.

Code Example to select divider 8 for peripheral 2 and enable its clock:

```
write_register(PMC_PCR,0x010031002)
```

Code Example to read the divider of peripheral 4:

```
write_register(PMC_PCR,0x00000004)
read_register(PMC_PCR)
```

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 Control registers is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

22.10 Programmable Clock Output Controller

The PMC controls 2 signals to be output on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

PCKx can be independently selected between the Slow clock, the Master Clock, the PLLACK/PLLADIV2, the UTMI PLL output and the main clock by writing the CSS field in PMC_PCKx. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

Each output signal can be enabled and disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

Moreover, like the PCK, a status bit in PMC_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

22.11 Programming Sequence

1. Enabling the 12 MHz Main Oscillator:

   The main oscillator is enabled by setting the MOSCEN field in the CKGR_MOR register. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the OSCOUNT field in the CKGR_MOR register.

   Once this register has been correctly configured, the user must wait for MOSCS field in the PMC_SR register to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to MOSCS has been enabled in the PMC_IER register.

2. Setting PLLA and divider:

   All parameters needed to configure PLLA and the divider are located in the CKGR_PLLAR register.

   The DIVA field is used to control the divider itself. A value between 0 and 255 can be programmed. Divider output is divider input divided by DIVA parameter. By default DIVA parameter is set to 0 which means that divider is turned off.

   The OUTA field is used to select the PLLA output frequency range.

   The MULA field is the PLLA multiplier factor. This parameter can be programmed between 0 and 254. If MULA is set to 0, PLLA will be turned off, otherwise the PLLA output frequency is PLLA 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 the PMC_PLLAR register has been written, the user must 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, MULA, DIVA is modified, LOCKA bit will go low to indicate that PLLA is not ready yet. When PLLA is locked, LOCKA will be set again. The user is constrained to wait for LOCKA bit to be set before using the PLLA output clock.

   Code Example:

```
write_register(CKGR_PLLAR,0x00040805)
```

If PLLA and divider are enabled, the PLLA input clock is the main clock. PLLA output clock is PLLA input clock multiplied by 5. Once CKGR_PLLAR has been written, LOCKA bit will be set after eight slow clock cycles.

3. Setting Bias and High Speed PLL (UPLL) for UTMI

   The UTMI PLL is enabled by setting the UPLLEN field in the CKGR_UCKR register. The UTMI Bias must is enabled by setting the BIASEN field in the CKGR_UCKR register in the same time. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the PLLCOUNT field in the CKGR_UCKR register.

   Once this register has been correctly configured, the user must wait for LOCKU field in the PMC_SR register to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKU has been enabled in the PMC_IER register.

4. Selection of Master Clock and Processor Clock

   The Master Clock and the Processor Clock are configurable via the PMC_MCKR register.

   The CSS field is used to select the clock source of the Master Clock and Processor Clock dividers. By default, the selected clock source is slow clock.

   The PRES field is used to control the Master/Processor Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Prescaler output is the selected clock source divided by PRES parameter. By default, PRES parameter is set to 1 which means that the input clock of the Master Clock and Processor Clock dividers 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, 3). The Master Clock output is Master/Processor Clock Prescaler output divided by 1, 2, 4 or 3, depending on the value programmed in MDIV.

   The PLLADIV2 field is used to control the PLLA Clock divider. It is possible to choose between different values (0, 1). The PMC PLLA Clock input is divided by 1 or 2, depending on the value programmed in PLLADIV2.

   By default, MDIV and PLLLADIV2 are set to 0, which indicates that 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 PLLA 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 PLLA clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLAR, the MCKRDY flag will go low while PLLA is unlocked. Once PLLA is locked again, LOCK goes high and MCKRDY is set.

While PLLA is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 22.12.2. “Clock Switching Waveforms” on page 185.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
```

wait (MCKRDY=1)

```
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

5. Selection of Programmable clocks

Programmable clocks are controlled via registers; PMC_SCER, PMC_SCDR and PMC_SCSR.

Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR registers. Depending on the system used, 2 programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which Programmable clock is enabled. By default all Programmable clocks are disabled.

PMC_PCKx registers are used to configure programmable clocks.

The CSS and CSSMCK fields are used to select the programmable clock divider source. Five clock options are available: main clock, slow clock, master clock, PLLACK, UPLLCK. By default, the clock source selected is slow clock.

The PRES field is used to control the programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 1 which means that master clock is equal to slow clock.

Once the PMC_PCKx register has been programmed, The corresponding programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER register. All parameters in PMC_PCKx can be programmed in a single write operation.

If the CSS and PRES parameters are to be modified, the corresponding programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the programmable clock and wait for the PCKRDYx bit to be set.

Code Example:

```
write_register(PMC_PCK0,0x00000015)
```

Programmable clock 0 is main clock divided by 32.

6. Enabling Peripheral Clocks

Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCER and PMC_PCDR.

Depending on the system used, 19 peripheral clocks can be enabled or disabled. The PMC_PCR 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.

22.12 Clock Switching Details

22.12.1 Master Clock Switching Timings

Table 22-1 and Table 22-2 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 22-1. 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 PLLA or the UPLL Clock.
2. PLLCOUNT designates either PLLACOUNT or UPLLCOUNT.

Table 22-2. Clock Switching Timings between Two PLLs (Worst Case)

From / To	PLLA Clock	UPLL 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
UPLL Clock	3 x UPLL Clock + 4 x SLCK + 1.5 x UPLL Clock	2.5 x UPLL Clock + 4 x SLCK + UPLLCOUNT x SLCK

22.12.2 Clock Switching Waveforms

Figure 22-3. Switch Master Clock from Slow Clock to PLL Clock

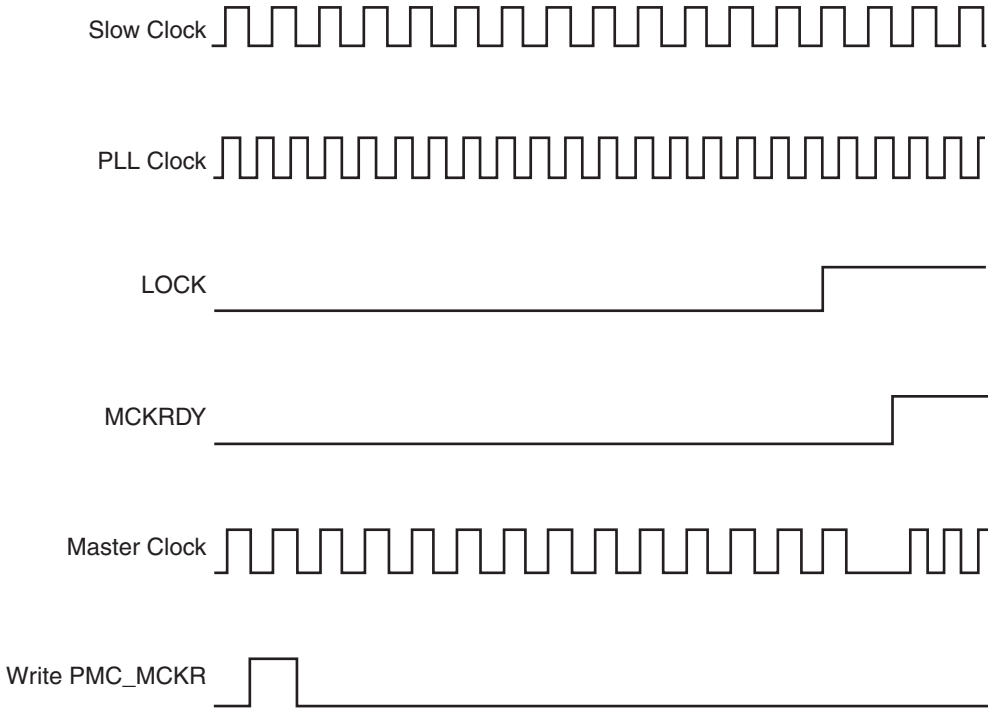


Figure 22-4. Switch Master Clock from Main Clock to Slow Clock

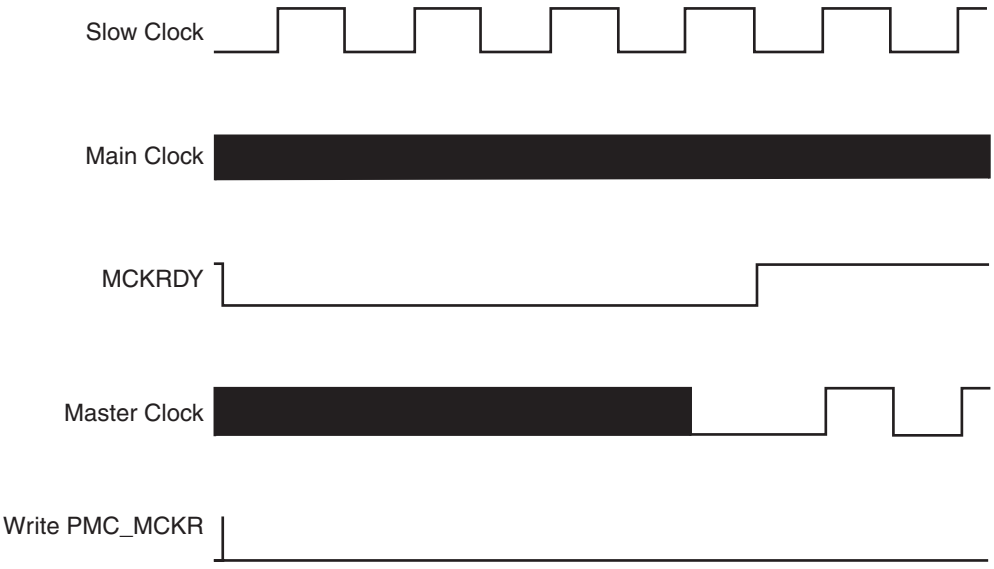


Figure 22-5. Change PLLA Programming

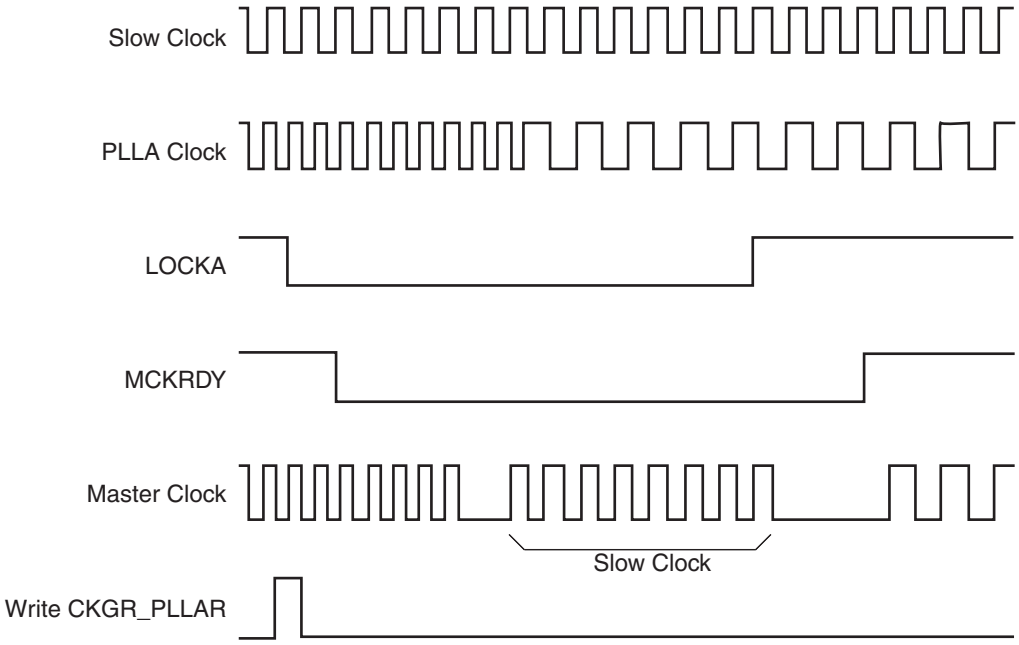
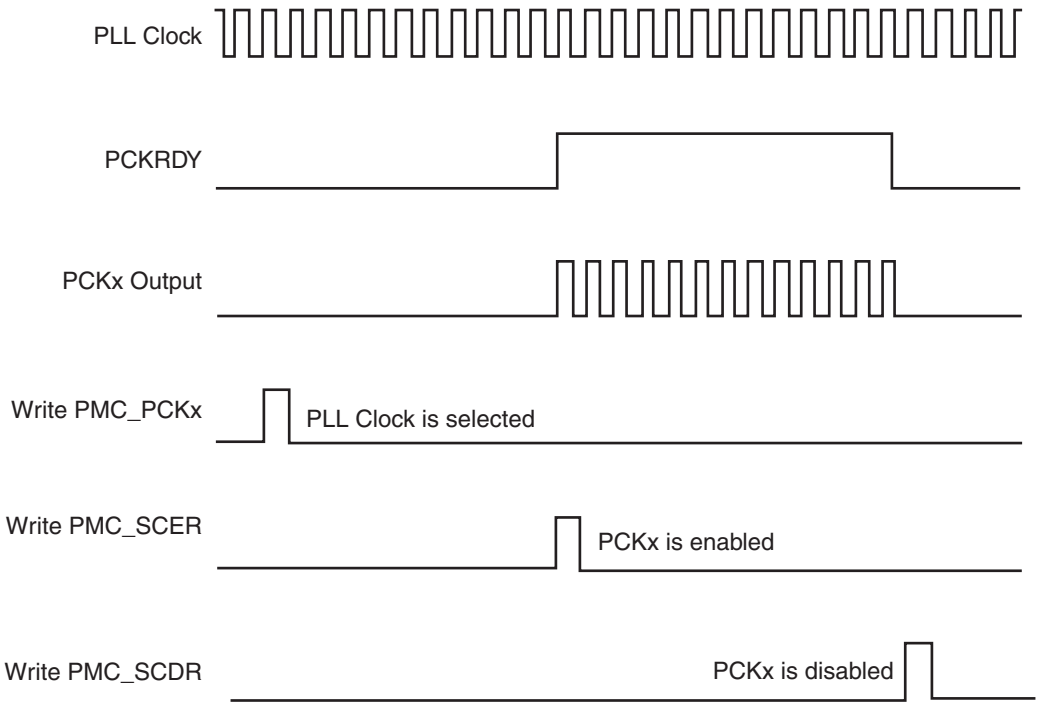


Figure 22-6. Programmable Clock Output Programming



22.13 Power Management Controller (PMC) User Interface

Table 22-3. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	System Clock Enable Register	PMC_SCER	Write-only	N.A.
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	N.A.
0x0008	System Clock Status Register	PMC_SCSR	Read-only	0x0000_0005
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	N.A.
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0000_0000
0x000C - 0x0018	Reserved	–	–	–
0x001C	UTMI Clock Register	CKGR_UCKR	Read-write	0x1020_0000
0x0020	Main Oscillator Register	CKGR_MOR	Read-write	0x0000_0008
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0000_0000
0x0028	PLLA Register	CKGR_PLLAR	Read-write	0x0000_3F00
0x002C	Reserved	–	–	–
0x0030	Master Clock Register	PMC_MCKR	Read-write	0x0000_0001
0x0034	Reserved	–	–	–
0x0038	USB Clock Register	PMC_USB	Read-write	0x0000_0000
0x003C	Soft Modem Clock Register	PMC_SMD	Read-write	0x0000_0000
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read-write	0x0000_0000
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read-write	0x0000_0000
0x0048 - 0x005C	Reserved	–	–	–
0x0060	Interrupt Enable Register	PMC_IER	Write-only	N.A.
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	N.A.
0x0068	Status Register	PMC_SR	Read-only	0x0001_0008
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0000_0000
0x0070 - 0x0078	Reserved	–	–	–
0x0080	PLL Charge Pump Current Register	PMC_PLLICPR	Write-only	0x0100_0100
0x0084-0x00E0	Reserved	–	–	–
0x00E4	Write Protect Mode Register	PMC_WPMR	Read-write	0x0000_0000
0x00E8	Write Protect Status Register	PMC_WPSR	Read-only	0x0000_0000
0x00EC-0x0108	Reserved	–	–	–
0x010C	Peripheral Control Register	PMC_PCR	Read-write	0x0000_0000

22.13.1 PMC System Clock Enable Register

Name: PMC_SCER

Address: 0xFFFFFC00

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	SMDCK	–	DDRCK	–	–

- **DDRCK: DDR Clock Enable**

0 = No effect.

1 = Enables the DDR clock.

- **SMDCK: SMD Clock Enable**

0 = No effect.

1 = Enables the soft modem clock.

- **UHP: USB Host OHCI Clocks Enable**

0 = No effect.

1 = Enables the UHP48M and UHP12M OHCI clocks.

- **UDP: USB Device Clock Enable**

0 = No effect.

1 = Enables the USB Device clock.

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

22.13.2 PMC System Clock Disable Register

Name: PMC_SCDR

Address: 0xFFFFFC04

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	SMDCK	–	DDRCK	–	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

- **DDRCK: DDR Clock Disable**

0 = No effect.

1 = Disables the DDR clock.

- **SMDCK: SMD Clock Disable**

0 = No effect.

1 = Disables the soft modem clock.

- **UHP: USB Host OHCI Clock Disable**

0 = No effect.

1 = Disables the UHP48M and UHP12M OHCI clocks.

- **UDP: USB Device Clock Enable**

0 = No effect.

1 = Disables the USB Device clock.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

22.13.3 PMC System Clock Status Register

Name: PMC_SCSR

Address: 0xFFFFFC08

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	SMDCK	–	DDRCK	–	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **DDRCK: DDR Clock Status**

0 = The DDR clock is disabled.

1 = The DDR clock is enabled.

- **SMDCK: SMD Clock Status**

0 = The soft modem clock is disabled.

1 = The soft modem clock is enabled.

- **UHP: USB Host Port Clock Status**

0 = The UHP48M and UHP12M OHCI clocks are disabled.

1 = The UHP48M and UHP12M OHCI clocks are enabled.

- **UDP: USB Device Port Clock Status**

0 = The USB Device clock is disabled.

1 = The USB Device clock is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

22.13.4 PMC Peripheral Clock Enable Register

Name: PMC_PCER

Address: 0xFFFFFC10

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Notes: 1. PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

2. Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

22.13.5 PMC Peripheral Clock Disable Register

Name: PMC_PCDR

Address: 0xFFFFFC14

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

22.13.6 PMC Peripheral Clock Status Register

Name: PMC_PCSR

Address: 0xFFFFFC18

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

22.13.7 PMC UTMI Clock Configuration Register

Name: CKGR_UCKR

Address: 0xFFFFFC1C

Access: Read-write

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

- **UPLLEN: UTMI PLL Enable**

0 = The UTMI PLL is disabled.

1 = The UTMI PLL is enabled.

When UPLLEN is set, the LOCKU flag is set once the UTMI PLL startup time is achieved.

- **UPLLCOUNT: UTMI PLL Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the UTMI PLL start-up time.

- **BIASEN: UTMI BIAS Enable**

0 = The UTMI BIAS is disabled.

1 = The UTMI BIAS is enabled.

- **BIASCOUNT: UTMI BIAS Start-up Time**

Specifies the number of Slow Clock cycles for the UTMI BIAS start-up time.

22.13.8 PMC Clock Generator Main Oscillator Register

Name: CKGR_MOR

Address: 0xFFFFFC20

Access: Read-write

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

23	22	21	20	19	18	17	16
KEY							

15	14	13	12	11	10	9	8
MOSCXTST							

7	6	5	4	3	2	1	0
–	–	–	–	MOSCRCEN	–	MOSCXTBY	MOSCXTEN

- **KEY: Password**

Should be written at value 0x37. Writing any other value in this field aborts the write operation.

- **MOSCXTEN: Main Crystal Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0 = The Main Crystal Oscillator is disabled.

1 = The Main Crystal Oscillator is enabled. MOSCXTBY must be set to 0.

When MOSCXTEN is set, the MOSCXTS flag is set once the Main Crystal Oscillator startup time is achieved.

- **MOSCXTBY: Main Crystal Oscillator Bypass**

0 = No effect.

1 = The Main Crystal Oscillator is bypassed. MOSCXTEN must be set to 0. An external clock must be connected on XIN.

When MOSCXTBY is set, the MOSCXTS flag in PMC_SR is automatically set.

Clearing MOSCXTEN and MOSCXTBY bits allows resetting the MOSCXTS flag.

- **MOSCRCEN: Main On-Chip RC Oscillator Enable**

0 = The Main On-Chip RC Oscillator is disabled.

1 = The Main On-Chip RC Oscillator is enabled.

When MOSCRCEN is set, the MOSCRCS flag is set once the Main On-Chip RC Oscillator startup time is achieved.

- **MOSCXTST: Main Crystal Oscillator Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the Main Crystal Oscillator start-up time.

- **MOSCSEL: Main Oscillator Selection**

0 = The Main On-Chip RC Oscillator is selected.

1 = The Main Crystal Oscillator is selected.

- **CFDEN: Clock Failure Detector Enable**

0 = The Clock Failure Detector is disabled.

1 = The Clock Failure Detector is enabled.

22.13.9 PMC Clock Generator Main Clock Frequency Register

Name: CKGR_MCFR

Address: 0xFFFFFC24

Access: Read-only

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

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

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.

- **MAINFRDY: 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.

22.13.10PMC Clock Generator PLLA Register

Name: CKGR_PLLAR

Address: 0xFFFFFC28

Access: 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 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**

Value	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the selected clock divided by DIVA.

- **PLLACOUNT: PLLA Counter**

Specifies the number of slow clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **OUTA: PLLA 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: PLLA Multiplier**

0 = The PLLA is deactivated.

1 up to 254 = The PLLA Clock frequency is the PLLA input frequency multiplied by MULA+ 1.

22.13.11PMC Master Clock Register

Name: PMC_MCKR

Address: 0xFFFFFC30

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	PLLADIV2	–	–	MDIV	
7	6	5	4	3	2	1	0
–	PRES			–	–	CSS	

- **CSS: Master/Processor Clock Source Selection**

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLACK/PLLADIV2 is selected
3	UPLL_CLK	UPLL Clock is selected

- **PRES: Master/Processor Clock Prescaler**

Value	Name	Description
0	CLOCK	Selected clock
1	CLOCK_DIV2	Selected clock divided by 2
2	CLOCK_DIV4	Selected clock divided by 4
3	CLOCK_DIV8	Selected clock divided by 8
4	CLOCK_DIV16	Selected clock divided by 16
5	CLOCK_DIV32	Selected clock divided by 32
6	CLOCK_DIV64	Selected clock divided by 64
7	CLOCK_DIV3	Selected clock divided by 3

- **MDIV: Master Clock Division**

Value	Name	Description
0	EQ_PCK	Master Clock is Prescaler Output Clock divided by 1. Warning: DDRCK is not available.
1	PCK_DIV2	Master Clock is Prescaler Output Clock divided by 2. DDRCK is equal to MCK.
2	PCK_DIV4	Master Clock is Prescaler Output Clock divided by 4. DDRCK is equal to MCK.
3	PCK_DIV3	Master Clock is Prescaler Output Clock divided by 3. DDRCK is equal to MCK.

- **PLLADIV2: PLLA divisor by 2**

Value	Name	Description
0	NOT_DIV2	PLLA clock frequency is divided by 1.
1	DIV2	PLLA clock frequency is divided by 2.

22.13.12PMC USB Clock Register

Name: PMC_USB

Address: 0xFFFFFC38

Access: Read-write

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

- **USBS: USB OHCI Input Clock Selection**

0 = USB Clock Input is PLLA

1 = USB Clock Input is UPLL

- **USBDIV: Divider for USB OHCI Clock.**

USB Clock is Input clock divided by USBDIV+1

22.13.13PMC SMD Clock Register

Name: PMC_SMD

Address: 0xFFFFFC3C

Access : Read-write

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

- **SMDS: SMD input clock selection**

0 = SMD Clock Input is PLLA

1 = SMD Clock Input is UPLL

- **SMDDIV: Divider for SMD Clock.**

SMD Clock is Input clock divided by SMD +1

22.13.14PMC Programmable Clock Register

Name: PMC_PCKx

Address: 0xFFFFFC40

Access: Read-write

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

- **CSS: Master Clock Source Selection**

Value	name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLACK/PLLADIV2 is selected
3	UPLL_CLK	UPLL Clock is selected
4	MCK_CLK	Master Clock is selected

- **PRES: Programmable Clock Prescaler**

Value	name	Description
0	CLOCK	Selected clock
1	CLOCK_DIV2	Selected clock divided by 2
2	CLOCK_DIV4	Selected clock divided by 4
3	CLOCK_DIV8	Selected clock divided by 8
4	CLOCK_DIV16	Selected clock divided by 16
5	CLOCK_DIV32	Selected clock divided by 32
6	CLOCK_DIV64	Selected clock divided by 64
7	Reserved	Reserved

22.13.15PMC Interrupt Enable Register

Name: PMC_IER

Address: 0xFFFFFC60

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	_	LOCKA	MOSCXTS

- **MOSCXTS: Main Crystal Oscillator Status Interrupt Enable**
- **LOCKA: PLLA Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **LOCKU: UTMI PLL Lock Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**
- **MOSCSELS: Main Oscillator Selection Status Interrupt Enable**
- **MOSCRCS: Main On-Chip RC Status Interrupt Enable**
- **CFDEV: Clock Failure Detector Event Interrupt Enable**

22.13.16PMC Interrupt Disable Register

Name: PMC_IDR

Address: 0xFFFFFC64

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	_	LOCKA	MOSCXTS

- **MOSCXTS: Main Crystal Oscillator Status Interrupt Disable**
- **LOCKA: PLLA Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **LOCKU: UTMI PLL Lock Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**
- **MOSCSELS: Main Oscillator Selection Status Interrupt Disable**
- **MOSCRCS: Main On-Chip RC Status Interrupt Disable**
- **CFDEV: Clock Failure Detector Event Interrupt Disable**

22.13.17PMC Status Register

Name: PMC_SR

Address: 0xFFFFFC68

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	FOS	CFDS	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
OSCSELS	LOCKU	–	–	MCKRDY	_	LOCKA	MOSCXTS

- **MOSCXTS: Main XTAL Oscillator Status**

0 = Main XTAL oscillator is not stabilized.

1 = Main XTAL oscillator is stabilized.

- **LOCKA: PLLA Lock Status**

0 = PLLA is not locked

1 = PLLA is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **LOCKU: UPLL Clock Status**

0 = UPLL Clock is not ready.

1 = UPLL Clock is ready.

- **OSCSELS: Slow Clock Oscillator Selection**

0 = Internal slow clock RC oscillator is selected.

1 = External slow clock 32 kHz oscillator is selected.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

- **MOSCSELS: Main Oscillator Selection Status**

0 = Selection is in progress.

1 = Selection is done.

- **MOSCRCS: Main On-Chip RC Oscillator Status**

0 = Main on-chip RC oscillator is not stabilized.

1 = Main on-chip RC oscillator is stabilized.

- **CFDEV: Clock Failure Detector Event**

0 = No clock failure detection of the main on-chip RC oscillator clock has occurred since the last read of PMC_SR.

1 = At least one clock failure detection of the main on-chip RC oscillator clock has occurred since the last read of PMC_SR.

- **CFDS: Clock Failure Detector Status**

0 = A clock failure of the main on-chip RC oscillator clock is not detected.

1 = A clock failure of the main on-chip RC oscillator clock is detected.

- **FOS: Clock Failure Detector Fault Output Status**

0 = The fault output of the clock failure detector is inactive.

1 = The fault output of the clock failure detector is active.

22.13.18PMC Interrupt Mask Register

Name: PMC_IMR

Address: 0xFFFFFC6C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	_	LOCKA	MOSCXTS

- **MOSCXTS: Main Crystal Oscillator Status Interrupt Mask**
- **LOCKA: PLLA Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**
- **MOSCSELS: Main Oscillator Selection Status Interrupt Mask**
- **MOSCRCS: Main On-Chip RC Status Interrupt Mask**
- **CFDEV: Clock Failure Detector Event Interrupt Mask**

22.13.19PLL Charge Pump Current Register

Name: PMC_PLLICPR

Address: 0xFFFFFC80

Access: Write-only

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

- **ICPLLA: Charge Pump Current**

To optimize clock performance, this field must be programmed as specified in “PLL A Characteristics” in the Electrical Characteristics section of the product datasheet.

22.13.20PMC Write Protect Mode Register

Name: PMC_WPMR

Address: 0xFFFFFCE4

Access: Read-write

Reset: See Table 22-3

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

Protects the registers:

- “PMC System Clock Enable Register” on page 188
- “PMC System Clock Disable Register” on page 189
- “PMC Clock Generator Main Clock Frequency Register” on page 196
- “PMC Clock Generator PLLA Register” on page 197
- “PMC Master Clock Register” on page 198
- “PMC USB Clock Register” on page 200
- “PMC Programmable Clock Register” on page 202
- “PLL Charge Pump Current Register” on page 208

- **WPKEY: Write Protect KEY**

Should be written at value 0x504D43 (“PMC” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

22.13.21PMC Write Protect Status Register

Name: PMC_WPSR

Address: 0xFFFFFCE8

Access: Read-only

Reset: See Table 22-3

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the PMC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the PMC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Reading PMC_WPSR automatically clears all fields.

22.13.22PMC Peripheral Control Register

Name: PMC_PCR

Address: 0xFFFFFD0C

Access: Read-write

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

- **PID: Peripheral ID**

Only the following Peripheral IDs can have a DIV value other than 0: PID2, PID3, PID5 to PID11, PID13 to PID19, PID28 to PID30.

PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

- **CMD: Command**

0: Read mode

1: Write mode

- **DIV: Divisor Value**

Value	Name	Description
0	PERIPH_DIV_MCK	Peripheral clock is MCK
1	PERIPH_DIV2_MCK	Peripheral clock is MCK/2
2	PERIPH_DIV4_MCK	Peripheral clock is MCK/4
3	PERIPH_DIV8_MCK	Peripheral clock is MCK/8

- **EN: Enable**

0: Selected Peripheral clock is disabled

1: Selected Peripheral clock is enabled

23. Parallel Input/Output (PIO) Controller

23.1 Description

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- Additional Interrupt modes enabling rising edge, falling edge, low level or high level detection on any I/O line.
- A glitch filter providing rejection of glitches lower than one-half of PIO clock cycle.
- A debouncing filter providing rejection of unwanted pulses from key or push button operations.
- Multi-drive capability similar to an open drain I/O line.
- Control of the pull-up and pull-down 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.

23.2 Embedded Characteristics

- Up to 32 Programmable I/O Lines
- Fully Programmable through Set/Clear Registers
- Multiplexing of Four 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
  - Programmable Glitch Filter
  - Programmable Debouncing 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
  - Additional Interrupt Modes on a Programmable Event: Rising Edge, Falling Edge, Low Level or High Level
  - Lock of the Configuration by the Connected Peripheral
- Synchronous Output, Provides Set and Clear of Several I/O lines in a Single Write
- Write Protect Registers
- Programmable Schmitt Trigger Inputs
- Programmable I/O Delay
- Programmable I/O Drive

23.3 Block Diagram

Figure 23-1. Block Diagram

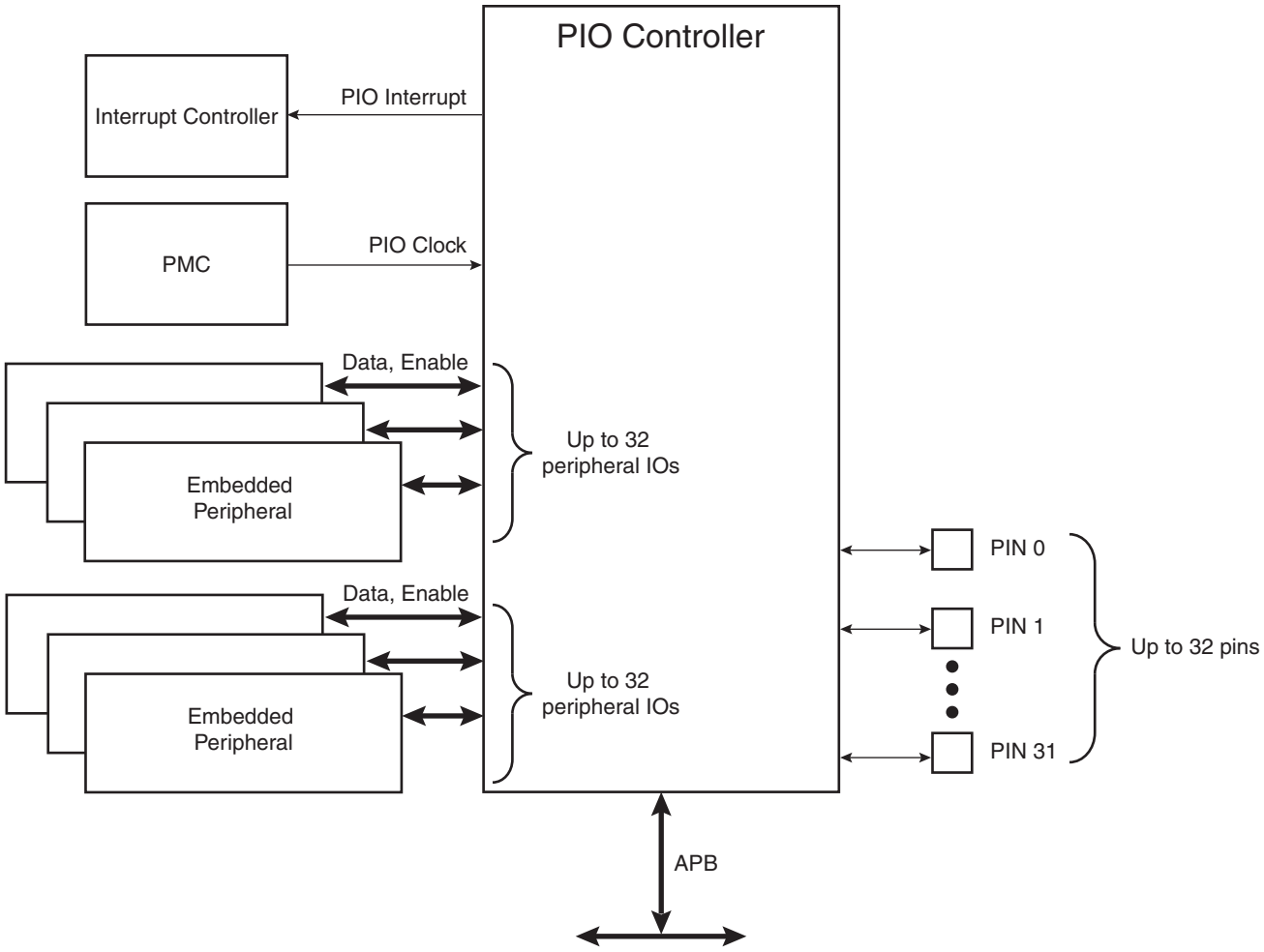
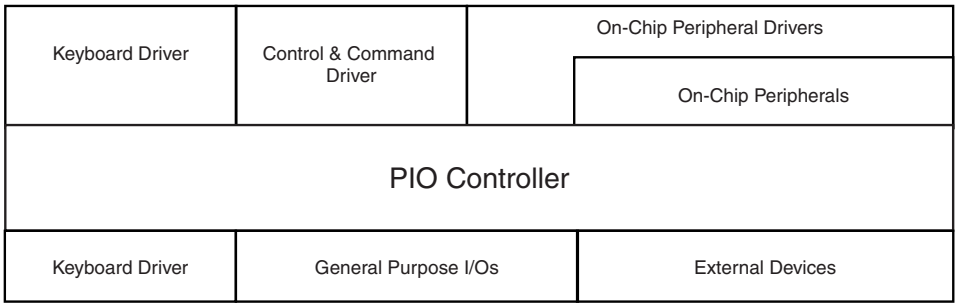


Figure 23-2. Application Block Diagram



23.4 Product Dependencies

23.4.1 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.

23.4.2 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.

23.4.3 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, including glitch filtering. Note that the Input Change Interrupt, Interrupt Modes on a programmable event 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.

23.4.4 Interrupt Generation

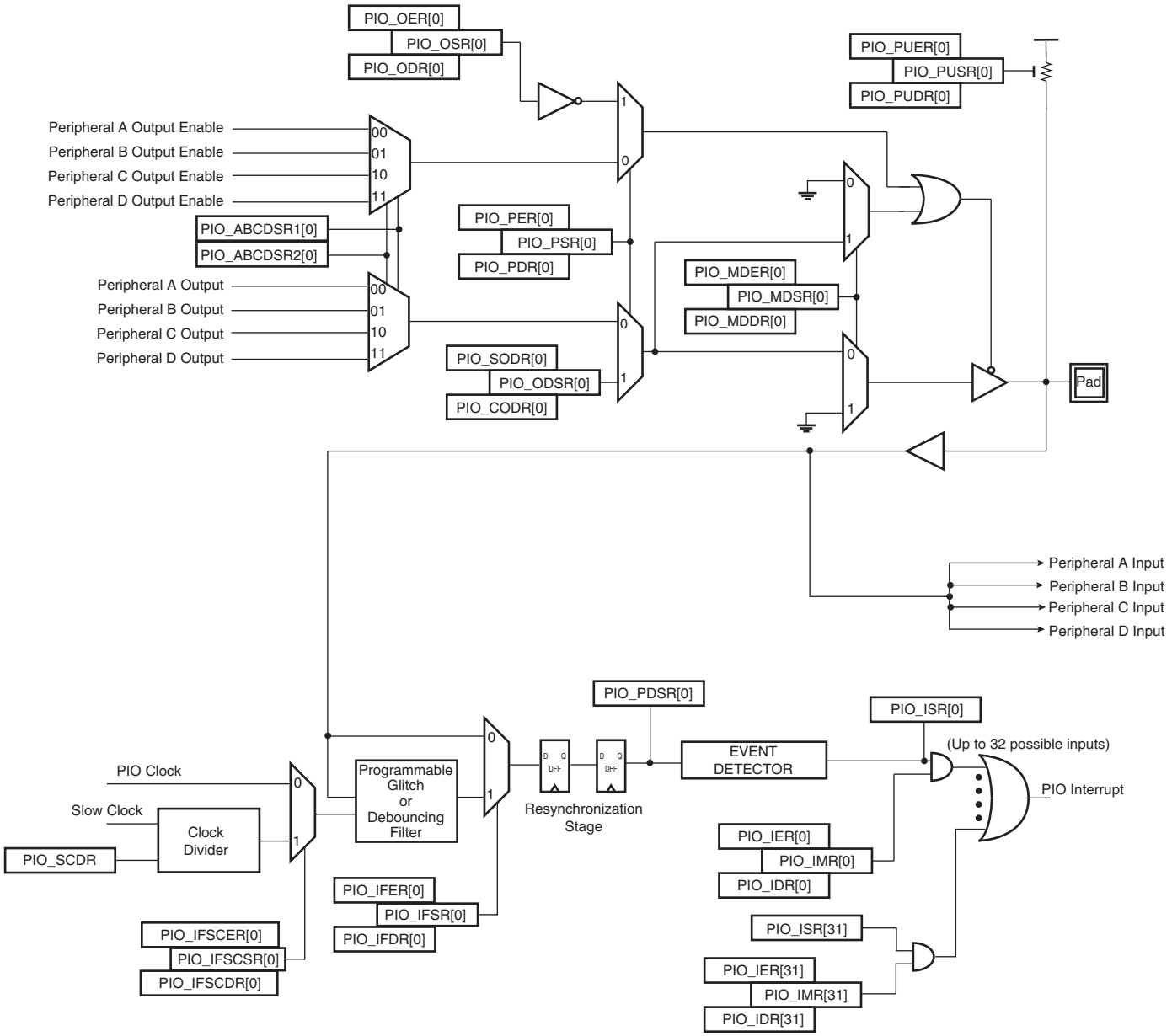
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers. Using the PIO Controller requires the Interrupt Controller to be programmed first.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

23.5 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 23-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 23-3. I/O Line Control Logic



23.5.1 Pull-up and Pull-down Resistor Control

Each I/O line is designed with an embedded pull-up resistor and an embedded pull-down 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. The pull-down resistor can be enabled or disabled by writing respectively PIO_PPDER (Pull-down Enable Register) and PIO_PPDDR (Pull-down Disable Resistor). Writing in these registers results in setting or clearing the corresponding bit in PIO_PPDSR (Pull-down Status Register). Reading a 1 in PIO_PPDSR means the pull-up is disabled and reading a 0 means the pull-down is enabled.

Enabling the pull-down resistor while the pull-up resistor is still enabled is not possible. In this case, the write of PIO_PPDER for the concerned I/O line is discarded. Likewise, enabling the pull-up resistor while the pull-down resistor is still enabled is not possible. In this case, the write of PIO_PUER for the concerned I/O line is discarded.

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, and all the pull-downs are disabled, i.e. PIO_PPDSR resets at the value 0xFFFFFFFF.

23.5.2 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_ABCDSR1 and PIO_ABCDSR2 (ABCD Select Registers). 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.

23.5.3 Peripheral A or B or C or D Selection

The PIO Controller provides multiplexing of up to four peripheral functions on a single pin. The selection is performed by writing PIO_ABCDSR1 and PIO_ABCDSR2 (ABCD Select Registers).

For each pin:

- The corresponding bit at level 0 in PIO_ABCDSR1 and the corresponding bit at level 0 in PIO_ABCDSR2 means peripheral A is selected.
- The corresponding bit at level 1 in PIO_ABCDSR1 and the corresponding bit at level 0 in PIO_ABCDSR2 means peripheral B is selected.
- The corresponding bit at level 0 in PIO_ABCDSR1 and the corresponding bit at level 1 in PIO_ABCDSR2 means peripheral C is selected.
- The corresponding bit at level 1 in PIO_ABCDSR1 and the corresponding bit at level 1 in PIO_ABCDSR2 means peripheral D is selected.

Note that multiplexing of peripheral lines A, B, C and D only affects the output line. The peripheral input lines are always connected to the pin input.

Writing in PIO_ABCDSR1 and PIO_ABCDSR2 manages the multiplexing regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the peripheral selection registers (PIO_ABCDSR1 and PIO_ABCDSR2) in addition to a write in PIO_PDR.

After reset, PIO_ABCDSR1 and PIO_ABCDSR2 are 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.

23.5.4 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 or C or D depending on the value in PIO_ABCDSR1 and PIO_ABCDSR2 (ABCD Select Registers) 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.

23.5.5 Synchronous Data Output

Clearing one (or more) PIO line(s) and setting another one (or more) PIO line(s) synchronously cannot be done by using PIO_SODR and PIO_CODR registers. It requires two successive write operations into two different registers. To overcome this, 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 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.

23.5.6 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.

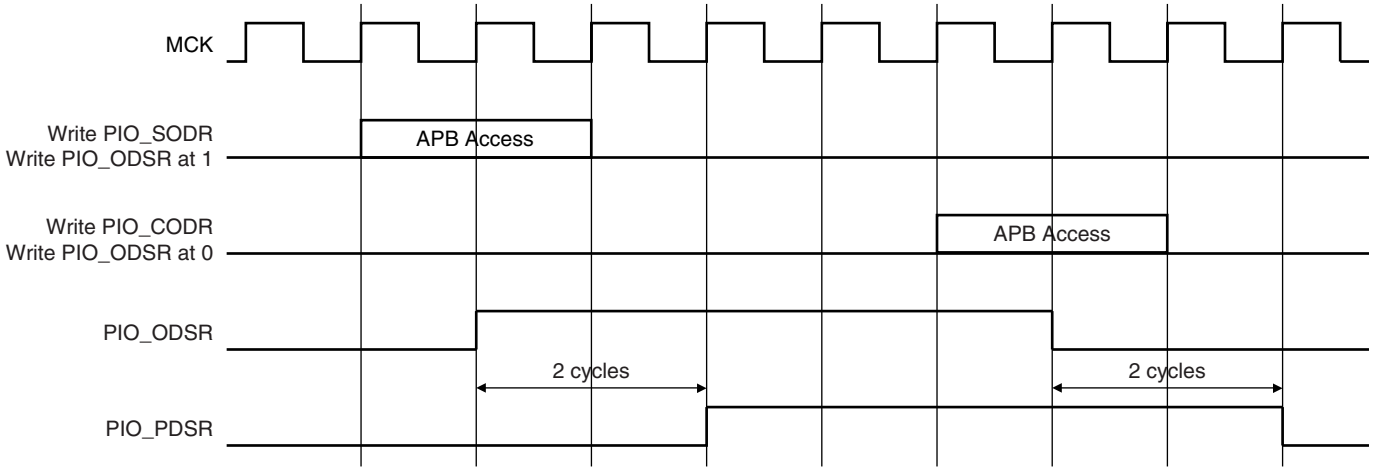
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates 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.

23.5.7 Output Line Timings

Figure 23-4 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 23-4 also shows when the feedback in PIO_PDSR is available.

Figure 23-4. Output Line Timings



23.5.8 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.

23.5.9 Input Glitch and Debouncing Filters

Optional input glitch and debouncing filters are independently programmable on each I/O line.

The glitch filter can filter a glitch with a duration of less than 1/2 Master Clock (MCK) and the debouncing filter can filter a pulse of less than 1/2 Period of a Programmable Divided Slow Clock.

The selection between glitch filtering or debounce filtering is done by writing in the registers PIO_IFSCDR (PIO Input Filter Slow Clock Disable Register) and PIO_IFSCER (PIO Input Filter Slow Clock Enable Register). Writing PIO_IFSCDR and PIO_IFSCER respectively, sets and clears bits in PIO_IFSCSR.

The current selection status can be checked by reading the register PIO_IFSCSR (Input Filter Slow Clock Status Register).

- If PIO_IFSCSR[i] = 0: The glitch filter can filter a glitch with a duration of less than 1/2 Period of Master Clock.
- If PIO_IFSCSR[i] = 1: The debouncing filter can filter a pulse with a duration of less than 1/2 Period of the Programmable Divided Slow Clock.

For the debouncing filter, the Period of the Divided Slow Clock is performed by writing in the DIV field of the PIO_SCDR (Slow Clock Divider Register)

Tdiv_slclk = ((DIV+1)*2).Tslow_clock

When the glitch or debouncing filter is enabled, a glitch or pulse with a duration of less than 1/2 Selected Clock Cycle (Selected Clock represents MCK or Divided Slow Clock depending on PIO_IFSCDR and PIO_IFSCER programming) is automatically rejected, while a pulse with a duration of 1 Selected Clock (MCK or Divided Slow Clock) cycle or more is accepted. For pulse durations between 1/2 Selected Clock cycle and 1 Selected 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 Selected Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Selected Clock cycle.

The filters also introduce some latencies, this is illustrated in Figure 23-5 and Figure 23-6.

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 and/or debouncing 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 and debouncing filters require that the PIO Controller clock is enabled.

Figure 23-5. Input Glitch Filter Timing

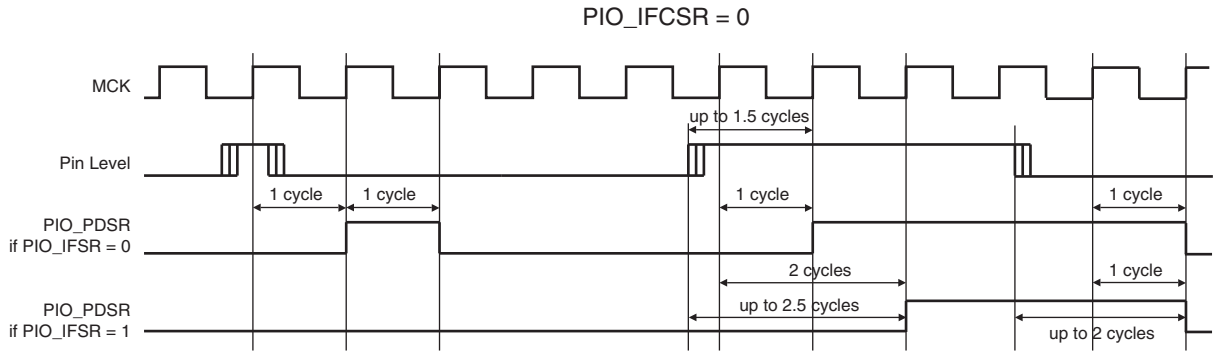
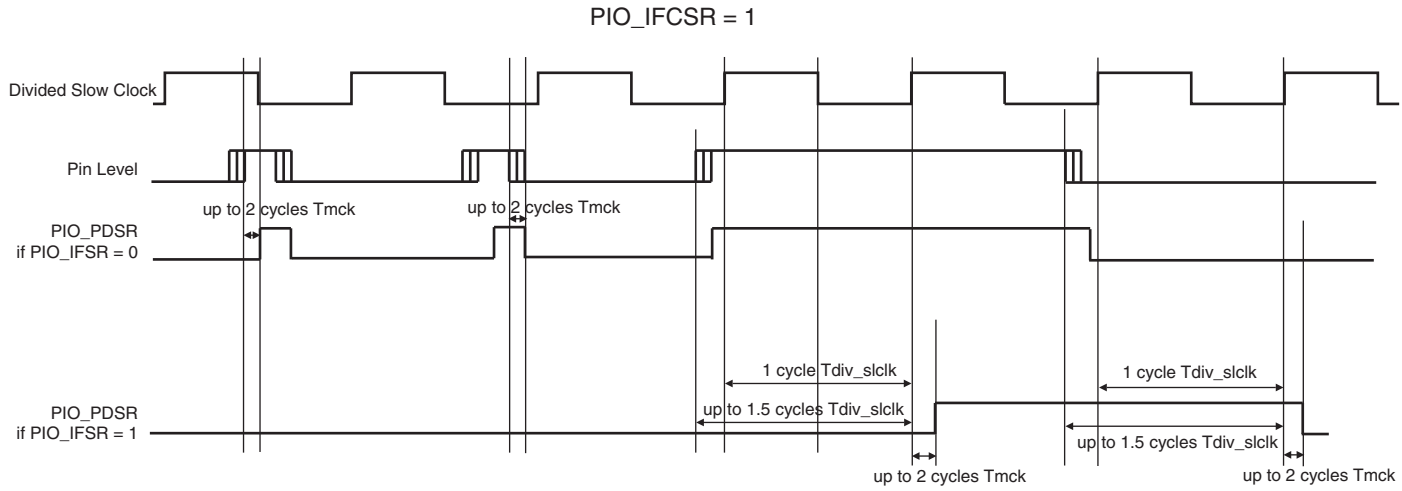


Figure 23-6. Input Debouncing Filter Timing



23.5.10 Input Edge/Level Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an edge or a level on an I/O line. The Input Edge/Level 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.

By default, the interrupt can be generated at any time an edge is detected on the input.

Some additional Interrupt modes can be enabled/disabled by writing in the PIO_AIMER (Additional Interrupt Modes Enable Register) and PIO_AIMDR (Additional Interrupt Modes Disable Register). The current state of this selection can be read through the PIO_AIMMR (Additional Interrupt Modes Mask Register)

These Additional Modes are:

- Rising Edge Detection
- Falling Edge Detection
- Low Level Detection
- High Level Detection

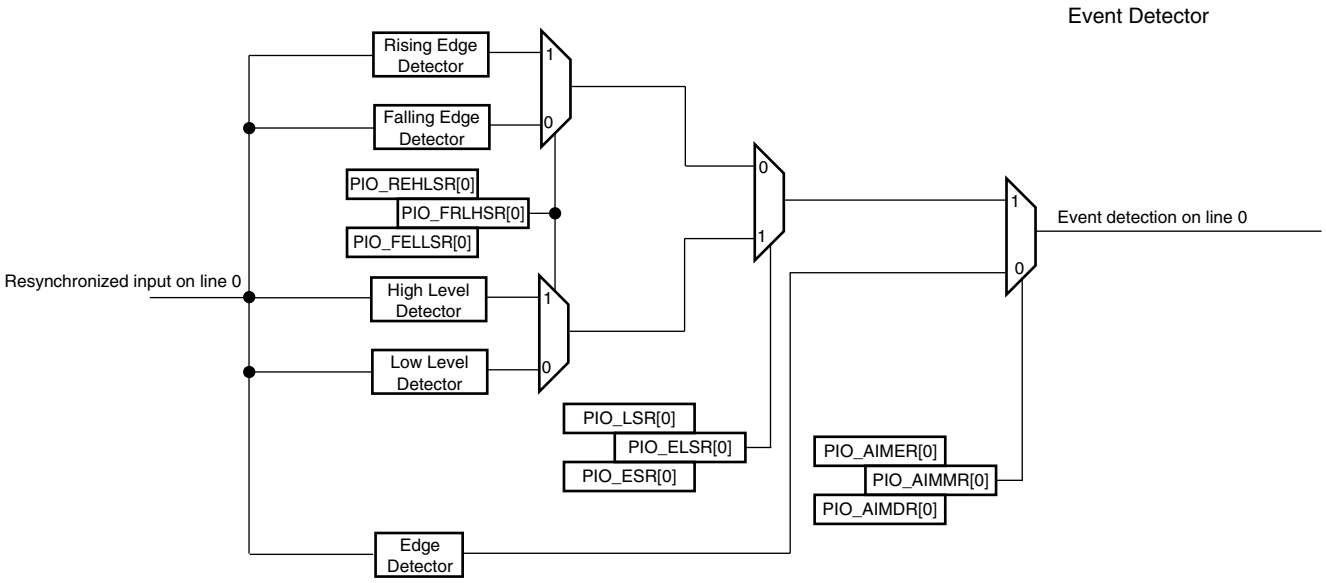
In order to select an Additional Interrupt Mode:

- The type of event detection (Edge or Level) must be selected by writing in the set of registers; PIO_ESR (Edge Select Register) and PIO_LSR (Level Select Register) which enable respectively, the Edge and Level Detection. The current status of this selection is accessible through the PIO_ELSR (Edge/Level Status Register).
- The Polarity of the event detection (Rising/Falling Edge or High/Low Level) must be selected by writing in the set of registers; PIO_FELLSR (Falling Edge /Low Level Select Register) and PIO_REHLSR (Rising Edge/High Level Select Register) which allow to select Falling or Rising Edge (if Edge is selected in the PIO_ELSR), Edge or High or Low Level Detection (if Level is selected in the PIO_ELSR). The current status of this selection is accessible through the PIO_FRLHSR (Fall/Rise - Low/High Status Register).

When an input Edge or Level 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 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. When an Interrupt is enabled on a “Level”, the interrupt is generated as long as the interrupt source is not cleared, even if some read accesses in PIO_ISR are performed.

Figure 23-7. Event Detector on Input Lines (Figure represents line 0)



23.5.10.1 Example

If generating an interrupt is required on the following:

- Rising edge on PIO line 0
- Falling edge on PIO line 1
- Rising edge on PIO line 2
- Low Level on PIO line 3
- High Level on PIO line 4
- High Level on PIO line 5
- Falling edge on PIO line 6

- Rising edge on PIO line 7
- Any edge on the other lines

The configuration required is described below.

23.5.10.2 Interrupt Mode Configuration

All the interrupt sources are enabled by writing 32'hFFFF_FFFF in PIO_IER.

Then the Additional Interrupt Mode is enabled for line 0 to 7 by writing 32'h0000_00FF in PIO_AIMER.

23.5.10.3 Edge or Level Detection Configuration

Lines 3, 4 and 5 are configured in Level detection by writing 32'h0000_0038 in PIO_LSR.

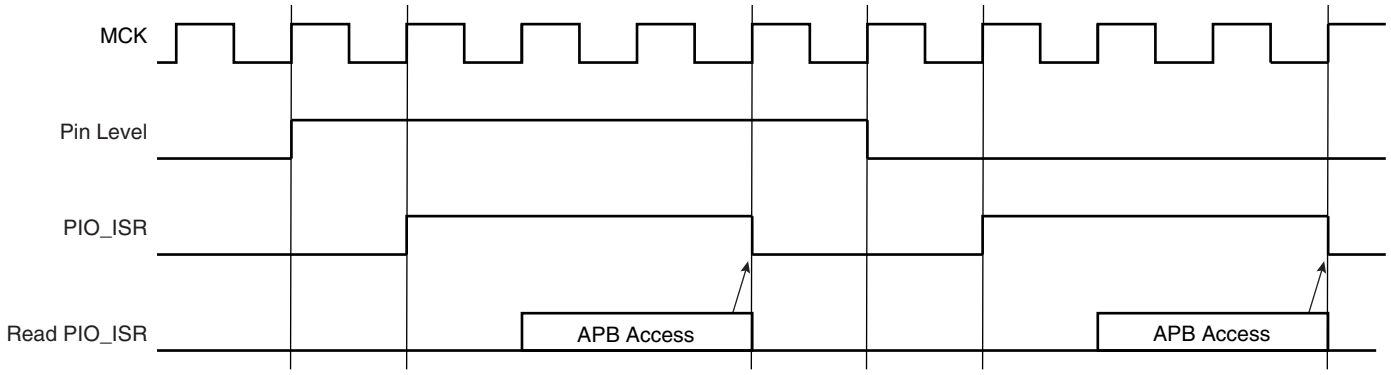
The other lines are configured in Edge detection by default, if they have not been previously configured. Otherwise, lines 0, 1, 2, 6 and 7 must be configured in Edge detection by writing 32'h0000_00C7 in PIO_ESR.

23.5.10.4 Falling/Rising Edge or Low/High Level Detection Configuration.

Lines 0, 2, 4, 5 and 7 are configured in Rising Edge or High Level detection by writing 32'h0000_00B5 in PIO_REHLSR.

The other lines are configured in Falling Edge or Low Level detection by default, if they have not been previously configured. Otherwise, lines 1, 3 and 6 must be configured in Falling Edge/Low Level detection by writing 32'h0000_004A in PIO_FELLSR.

Figure 23-8. Input Change Interrupt Timings if there are no Additional Interrupt Modes



23.5.11 I/O Lines Lock

When an I/O line is controlled by a peripheral (particularly the Pulse Width Modulation Controller PWM), it can become locked by the action of this peripheral via an input of the PIO controller. When an I/O line is locked, the write of the corresponding bit in the registers PIO_PER, PIO_PDR, PIO_MDER, PIO_MDDR, PIO_PUDR, PIO_PUER, PIO_ABCDSR1 and PIO_ABCDSR2 is discarded in order to lock its configuration. The user can know at anytime which I/O line is locked by reading the PIO Lock Status register PIO_LOCKSR. Once an I/O line is locked, the only way to unlock it is to apply a hardware reset to the PIO Controller.

23.5.12 Programmable I/O Delays

The PIO interface consists of a series of signals driven by peripherals or directly by software. The simultaneous switching outputs on these busses may lead to a peak of current in the internal and external power supply lines.

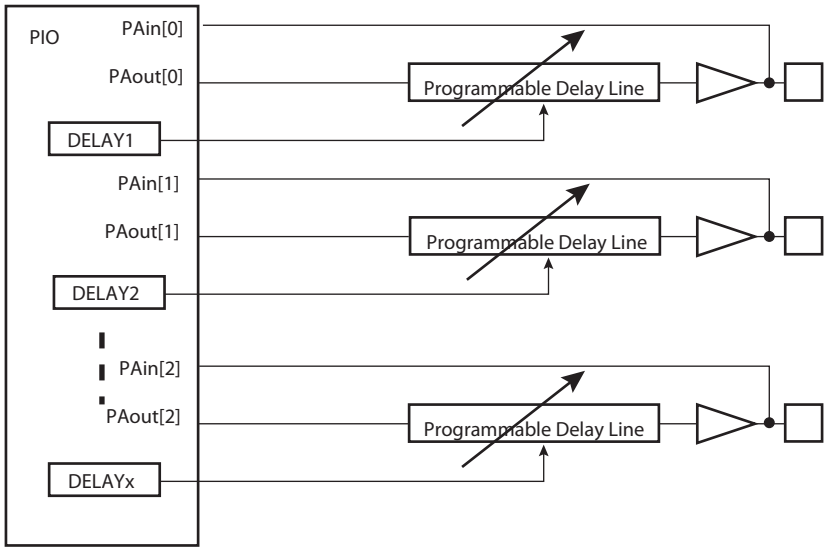
In order to reduce the current peak in such cases, additional propagation delays can be adjusted independently for pad buffers by means of configuration registers, PIO_DELAY.

The additional programmable delays for each supporting range from 0 to 4 ns (Worst Case PVT). The delay can differ between I/Os supporting this feature. Delay can be modified per programming for each I/O. The minimal additional delay that can be programmed on a PAD supporting this feature is 1/16 of the maximum programmable delay.

Only PADs PA[20:15], PA[13:11] and PA[4:2] can be configured.

When programming 0x0 in fields, no delay is added (reset value) and the propagation delay of the pad buffers is the inherent delay of the pad buffer. When programming 0xF in fields, the propagation delay of the corresponding pad is maximal.

Figure 23-9. Programmable I/O Delays



23.5.13 Programmable I/O Drive

It is possible to configure the I/O drive for pads PA[20:15], PA[13:11] and PA[4:2]. For any details, refer to the product electrical characteristics.

23.5.14 Programmable Schmitt Trigger

It is possible to configure each input for the Schmitt Trigger. By default the Schmitt trigger is active. Disabling the Schmitt Trigger is requested when using the QTouch™ Library.

23.5.15 Write Protection Registers

To prevent any single software error that may corrupt PIO behavior, certain address spaces can be write-protected by setting the WPEN bit in the "PIO Write Protect Mode Register" (PIO_WPMR).

If a write access to the protected registers is detected, then the WPVS flag in the PIO Write Protect Status Register (PIO_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the PIO Write Protect Mode Register (PIO_WPMR) with the appropriate access key, WPKEY.

The protected registers are:

- "PIO Enable Register" on page 228
- "PIO Disable Register" on page 229
- "PIO Output Enable Register" on page 231
- "PIO Output Disable Register" on page 232
- "PIO Input Filter Enable Register" on page 234
- "PIO Input Filter Disable Register" on page 235
- "PIO Multi-driver Enable Register" on page 245
- "PIO Multi-driver Disable Register" on page 246
- "PIO Pull Up Disable Register" on page 248
- "PIO Pull Up Enable Register" on page 249
- "PIO Peripheral ABCD Select Register 1" on page 251
- "PIO Peripheral ABCD Select Register 2" on page 252
- "PIO Output Write Enable Register" on page 260
- "PIO Output Write Disable Register" on page 261
- "PIO Pad Pull Down Disable Register" on page 257
- "PIO Pad Pull Down Status Register" on page 259

23.6 I/O Lines Programming Example

The programing example as shown in Table 23-1 below is used to obtain 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, no pull-down 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 with pull-down resistor
- I/O line 24 to 27 assigned to peripheral C with Input Change Interrupt, no pull-up resistor and no pull-down resistor
- I/O line 28 to 31 assigned to peripheral D, no pull-up resistor and no pull-down resistor

Table 23-1. Programming Example

Register	Value to be Written
PIO_PER	0x0000_FFFF
PIO_PDR	0xFFFF_0000
PIO_OER	0x0000_00FF
PIO_ODR	0xFFFF_FF00
PIO_IFER	0x0000_0F00
PIO_IFDR	0xFFFF_F0FF
PIO_SODR	0x0000_0000
PIO_CODR	0x0FFF_FFFF
PIO_IER	0x0F00_0F00
PIO_IDR	0xF0FF_F0FF
PIO_MDER	0x0000_000F
PIO_MDDR	0xFFFF_FFF0
PIO_PUDR	0xFFF0_00F0
PIO_PUER	0x000F_FF0F
PIO_PPDDR	0xFF0F_FFFF
PIO_PPDER	0x00F0_0000
PIO_ABCDSR1	0xF0F0_0000
PIO_ABCDSR2	0xFF00_0000
PIO_OWER	0x0000_000F
PIO_OWDR	0x0FFF_ FFF0

23.7 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 notmultiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 23-2. 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	(1)
0x006C	Reserved			

Table 23-2. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0070	Peripheral Select Register 1	PIO_ABCDSR1	Read-write	0x00000000
0x0074	Peripheral Select Register 2	PIO_ABCDSR2	Read-write	0x00000000
0x0078 to 0x007C	Reserved			
0x0080	Input Filter Slow Clock Disable Register	PIO_IFSCDR	Write-only	–
0x0084	Input Filter Slow Clock Enable Register	PIO_IFSCER	Write-only	–
0x0088	Input Filter Slow Clock Status Register	PIO_IFSCSR	Read-only	0x00000000
0x008C	Slow Clock Divider Debouncing Register	PIO_SCDR	Read-write	0x00000000
0x0090	Pad Pull-down Disable Register	PIO_PPDDR	Write-only	–
0x0094	Pad Pull-down Enable Register	PIO_PPDER	Write-only	–
0x0098	Pad Pull-down Status Register	PIO_PPDSR	Read-only	(1)
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			
0x00B0	Additional Interrupt Modes Enable Register	PIO_AIMER	Write-only	–
0x00B4	Additional Interrupt Modes Disables Register	PIO_AIMDR	Write-only	–
0x00B8	Additional Interrupt Modes Mask Register	PIO_AIMMR	Read-only	0x00000000
0x00BC	Reserved			
0x00C0	Edge Select Register	PIO_ESR	Write-only	–
0x00C4	Level Select Register	PIO_LSR	Write-only	–
0x00C8	Edge/Level Status Register	PIO_ELSR	Read-only	0x00000000
0x00CC	Reserved			
0x00D0	Falling Edge/Low Level Select Register	PIO_FELLSR	Write-only	–
0x00D4	Rising Edge/ High Level Select Register	PIO_REHLSR	Write-only	–
0x00D8	Fall/Rise - Low/High Status Register	PIO_FRLHSR	Read-only	0x00000000
0x00DC	Reserved			
0x00E0	Lock Status	PIO_LOCKSR	Read-only	0x00000000
0x00E4	Write Protect Mode Register	PIO_WPMR	Read-write	0x0
0x00E8	Write Protect Status Register	PIO_WPSR	Read-only	0x0
0x00EC to 0x00F8	Reserved			
0x0100	Schmitt Trigger Register	PIO_SCHMITT	Read-write	0x00000000
0x0104-0x010C	Reserved			
0x0110	IO Delay Register	PIO_DELAYR	Read-write	0x00000000
0x0114	I/O Drive Register 1	PIO_DRIVER1	Read-write	0x00000000

Table 23-2. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0118	I/O Drive Register 2	PIO_DRIVER2	Read-write	0x00000000
0x011C	Reserved			
0x0120 to 0x014C	Reserved			

Notes:
1. Reset value 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.

Note: If an offset is not listed in the table it must be considered as reserved.

23.7.1 PIO Enable Register

Name: PIO_PER

Address: 0xFFFFF400 (PIOA), 0xFFFFF600 (PIOB), 0xFFFFF800 (PIOC), 0xFFFFFA00 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: PIO Enable**

0: No effect.

1: Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

23.7.2 PIO Disable Register

Name: PIO_PDR

Address: 0xFFFFF404 (PIOA), 0xFFFFF604 (PIOB), 0xFFFFF804 (PIOC), 0xFFFFFA04 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: PIO Disable**

0: No effect.

1: Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

23.7.3 PIO Status Register

Name: PIO_PSR

Address: 0xFFFFF408 (PIOA), 0xFFFFF608 (PIOB), 0xFFFFF808 (PIOC), 0xFFFFFA08 (PIOD)

Access: Read-only

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

- **P0-P31: PIO Status**

0: PIO is inactive on the corresponding I/O line (peripheral is active).

1: PIO is active on the corresponding I/O line (peripheral is inactive).

23.7.4 PIO Output Enable Register

Name: PIO_OER

Address: 0xFFFFF410 (PIOA), 0xFFFFF610 (PIOB), 0xFFFFF810 (PIOC), 0xFFFFFA10 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Output Enable**

0: No effect.

1: Enables the output on the I/O line.

23.7.5 PIO Output Disable Register

Name: PIO_ODR

Address: 0xFFFFF414 (PIOA), 0xFFFFF614 (PIOB), 0xFFFFF814 (PIOC), 0xFFFFFA14 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Output Disable**

0: No effect.

1: Disables the output on the I/O line.

23.7.6 PIO Output Status Register

Name: PIO_OSR

Address: 0xFFFFF418 (PIOA), 0xFFFFF618 (PIOB), 0xFFFFF818 (PIOC), 0xFFFFFA18 (PIOD)

Access: Read-only

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

- **P0-P31: Output Status**

0: The I/O line is a pure input.

1: The I/O line is enabled in output.

23.7.7 PIO Input Filter Enable Register

Name: PIO_IFER

Address: 0xFFFFF420 (PIOA), 0xFFFFF620 (PIOB), 0xFFFFF820 (PIOC), 0xFFFFFA20 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Input Filter Enable**

0: No effect.

1: Enables the input glitch filter on the I/O line.

23.7.8 PIO Input Filter Disable Register

Name: PIO_IFDR

Address: 0xFFFFF424 (PIOA), 0xFFFFF624 (PIOB), 0xFFFFF824 (PIOC), 0xFFFFFA24 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Input Filter Disable**

0: No effect.

1: Disables the input glitch filter on the I/O line.

23.7.9 PIO Input Filter Status Register

Name: PIO_IFSR

Address: 0xFFFFF428 (PIOA), 0xFFFFF628 (PIOB), 0xFFFFF828 (PIOC), 0xFFFFFA28 (PIOD)

Access: Read-only

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

- **P0-P31: Input Filer Status**

0: The input glitch filter is disabled on the I/O line.

1: The input glitch filter is enabled on the I/O line.

23.7.10 PIO Set Output Data Register

Name: PIO_SODR

Address: 0xFFFFF430 (PIOA), 0xFFFFF630 (PIOB), 0xFFFFF830 (PIOC), 0xFFFFFA30 (PIOD)

Access: Write-only

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

- **P0-P31: Set Output Data**

0: No effect.

1: Sets the data to be driven on the I/O line.

23.7.11 PIO Clear Output Data Register

Name: PIO_CODR

Address: 0xFFFFF434 (PIOA), 0xFFFFF634 (PIOB), 0xFFFFF834 (PIOC), 0xFFFFFA34 (PIOD)

Access: Write-only

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

- **P0-P31: Clear Output Data**

0: No effect.

1: Clears the data to be driven on the I/O line.

23.7.12 PIO Output Data Status Register

Name: PIO_ODSR

Address: 0xFFFFF438 (PIOA), 0xFFFFF638 (PIOB), 0xFFFFF838 (PIOC), 0xFFFFFA38 (PIOD)

Access: Read-only or Read-write

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

- **P0-P31: Output Data Status**

0: The data to be driven on the I/O line is 0.

1: The data to be driven on the I/O line is 1.

23.7.13 PIO Pin Data Status Register

Name: PIO_PDSR

Address: 0xFFFFF43C (PIOA), 0xFFFFF63C (PIOB), 0xFFFFF83C (PIOC), 0xFFFFFA3C (PIOD)

Access: Read-only

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

- **P0-P31: Output Data Status**

0: The I/O line is at level 0.

1: The I/O line is at level 1.

23.7.14 PIO Interrupt Enable Register

Name: PIO_IER

Address: 0xFFFFF440 (PIOA), 0xFFFFF640 (PIOB), 0xFFFFF840 (PIOC), 0xFFFFFA40 (PIOD)

Access: Write-only

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

- **P0-P31: Input Change Interrupt Enable**

0: No effect.

1: Enables the Input Change Interrupt on the I/O line.

23.7.15 PIO Interrupt Disable Register

Name: PIO_IDR

Address: 0xFFFFF444 (PIOA), 0xFFFFF644 (PIOB), 0xFFFFF844 (PIOC), 0xFFFFFA44 (PIOD)

Access: Write-only

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

- **P0-P31: Input Change Interrupt Disable**

0: No effect.

1: Disables the Input Change Interrupt on the I/O line.

23.7.16 PIO Interrupt Mask Register

Name: PIO_IMR

Address: 0xFFFFF448 (PIOA), 0xFFFFF648 (PIOB), 0xFFFFF848 (PIOC), 0xFFFFFA48 (PIOD)

Access: Read-only

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

- **P0-P31: Input Change Interrupt Mask**

0: Input Change Interrupt is disabled on the I/O line.

1: Input Change Interrupt is enabled on the I/O line.

23.7.17 PIO Interrupt Status Register

Name: PIO_ISR

Address: 0xFFFFF44C (PIOA), 0xFFFFF64C (PIOB), 0xFFFFF84C (PIOC), 0xFFFFFA4C (PIOD)

Access: Read-only

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

- **P0-P31: Input Change Interrupt Status**

0: No Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

1: At least one Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

23.7.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Address: 0xFFFFF450 (PIOA), 0xFFFFF650 (PIOB), 0xFFFFF850 (PIOC), 0xFFFFFA50 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Multi Drive Enable.**

0: No effect.

1: Enables Multi Drive on the I/O line.

23.7.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Address: 0xFFFFF454 (PIOA), 0xFFFFF654 (PIOB), 0xFFFFF854 (PIOC), 0xFFFFFA54 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Multi Drive Disable.**

0: No effect.

1: Disables Multi Drive on the I/O line.

23.7.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Address: 0xFFFFF458 (PIOA), 0xFFFFF658 (PIOB), 0xFFFFF858 (PIOC), 0xFFFFFA58 (PIOD)

Access: Read-only

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

- **P0-P31: Multi Drive Status.**

0: The Multi Drive is disabled on the I/O line. The pin is driven at high and low level.

1: The Multi Drive is enabled on the I/O line. The pin is driven at low level only.

23.7.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Address: 0xFFFFF460 (PIOA), 0xFFFFF660 (PIOB), 0xFFFFF860 (PIOC), 0xFFFFFA60 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Pull Up Disable.**

0: No effect.

1: Disables the pull up resistor on the I/O line.

23.7.22 PIO Pull Up Enable Register

Name: PIO_PUER

Address: 0xFFFFF464 (PIOA), 0xFFFFF664 (PIOB), 0xFFFFF864 (PIOC), 0xFFFFFA64 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Pull Up Enable.**

0: No effect.

1: Enables the pull up resistor on the I/O line.

23.7.23 PIO Pull Up Status Register

Name: PIO_PUSR

Address: 0xFFFFF468 (PIOA), 0xFFFFF668 (PIOB), 0xFFFFF868 (PIOC), 0xFFFFFA68 (PIOD)

Access: Read-only

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

- **P0-P31: Pull Up Status.**

0: Pull Up resistor is enabled on the I/O line.

1: Pull Up resistor is disabled on the I/O line.

23.7.24 PIO Peripheral ABCD Select Register 1

Name: PIO_ABCDSR1

Access: 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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Peripheral Select.**
  If the same bit is set to 0 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral B function.

If the same bit is set to 1 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral C function.

1: Assigns the I/O line to the Peripheral D function.

23.7.25 PIO Peripheral ABCD Select Register 2

Name: PIO_ABCDSR2

Access: 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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Peripheral Select.**

If the same bit is set to 0 in PIO_ABCDSR1:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral C function.

If the same bit is set to 1 in PIO_ABCDSR1:

0: Assigns the I/O line to the Peripheral B function.

1: Assigns the I/O line to the Peripheral D function.

23.7.26 PIO Input Filter Slow Clock Disable Register

Name: PIO_IFSCDR

Address: 0xFFFFF480 (PIOA), 0xFFFFF680 (PIOB), 0xFFFFF880 (PIOC), 0xFFFFFA80 (PIOD)

Access: Write-only

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

- **P0-P31: PIO Clock Glitch Filtering Select.**

0: No Effect.

1: The Glitch Filter is able to filter glitches with a duration < Tmck/2.

23.7.27 PIO Input Filter Slow Clock Enable Register

Name: PIO_IFSCER

Address: 0xFFFFF484 (PIOA), 0xFFFFF684 (PIOB), 0xFFFFF884 (PIOC), 0xFFFFFA84 (PIOD)

Access: Write-only

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

- **P0-P31: Debouncing Filtering Select.**

0: No Effect.

1: The Debouncing Filter is able to filter pulses with a duration < Tdiv_slclk/2.

23.7.28 PIO Input Filter Slow Clock Status Register

Name: PIO_IFSCSR

Address: 0xFFFFF488 (PIOA), 0xFFFFF688 (PIOB), 0xFFFFF888 (PIOC), 0xFFFFFA88 (PIOD)

Access: Read-only

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

- **P0-P31: Glitch or Debouncing Filter Selection Status**

0: The Glitch Filter is able to filter glitches with a duration < Tmck2.

1: The Debouncing Filter is able to filter pulses with a duration < Tdiv_slclk/2.

23.7.29 PIO Slow Clock Divider Debouncing Register

Name: PIO_SCDR

Address: 0xFFFFF48C (PIOA), 0xFFFFF68C (PIOB), 0xFFFFF88C (PIOC), 0xFFFFFA8C (PIOD)

Access: Read-write

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

- **DIVx: Slow Clock Divider Selection for Debouncing**

Tdiv_slclk = 2*(DIV+1)*Tslow_clock.

23.7.30 PIO Pad Pull Down Disable Register

Name: PIO_PPDDR

Address: 0xFFFFF490 (PIOA), 0xFFFFF690 (PIOB), 0xFFFFF890 (PIOC), 0xFFFFFA90 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Pull Down Disable.**

0: No effect.

1: Disables the pull down resistor on the I/O line.

23.7.31 PIO Pad Pull Down Enable Register

Name: PIO_PPDER

Address: 0xFFFFF494 (PIOA), 0xFFFFF694 (PIOB), 0xFFFFF894 (PIOC), 0xFFFFFA94 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Pull Down Enable.**

0: No effect.

1: Enables the pull down resistor on the I/O line.

23.7.32 PIO Pad Pull Down Status Register

Name: PIO_PPDSR

Address: 0xFFFFF498 (PIOA), 0xFFFFF698 (PIOB), 0xFFFFF898 (PIOC), 0xFFFFFA98 (PIOD)

Access: Read-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Pull Down Status.**

0: Pull Down resistor is enabled on the I/O line.

1: Pull Down resistor is disabled on the I/O line.

23.7.33 PIO Output Write Enable Register

Name: PIO_OWER

Address: 0xFFFFF4A0 (PIOA), 0xFFFFF6A0 (PIOB), 0xFFFFF8A0 (PIOC), 0xFFFFFAA0 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Output Write Enable.**

0: No effect.

1: Enables writing PIO_ODSR for the I/O line.

23.7.34 PIO Output Write Disable Register

Name: PIO_OWDR

Address: 0xFFFFF4A4 (PIOA), 0xFFFFF6A4 (PIOB), 0xFFFFF8A4 (PIOC), 0xFFFFFAA4 (PIOD)

Access: Write-only

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

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Output Write Disable.**

0: No effect.

1: Disables writing PIO_ODSR for the I/O line.

23.7.35 PIO Output Write Status Register

Name: PIO_OWSR

Address: 0xFFFFF4A8 (PIOA), 0xFFFFF6A8 (PIOB), 0xFFFFF8A8 (PIOC), 0xFFFFFAA8 (PIOD)

Access: Read-only

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

- **P0-P31: Output Write Status.**

0: Writing PIO_ODSR does not affect the I/O line.

1: Writing PIO_ODSR affects the I/O line.

23.7.36 PIO Additional Interrupt Modes Enable Register

Name: PIO_AIMER

Address: 0xFFFFF4B0 (PIOA), 0xFFFFF6B0 (PIOB), 0xFFFFF8B0 (PIOC), 0xFFFFFAB0 (PIOD)

Access: Write-only

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

- **P0-P31: Additional Interrupt Modes Enable.**

0: No effect.

1: The interrupt source is the event described in PIO_ELSR and PIO_FRLHSR.

23.7.37 PIO Additional Interrupt Modes Disable Register

Name: PIO_AIMDR

Address: 0xFFFFF4B4 (PIOA), 0xFFFFF6B4 (PIOB), 0xFFFFF8B4 (PIOC), 0xFFFFFAB4 (PIOD)

Access: Write-only

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

- **P0-P31: Additional Interrupt Modes Disable.**

0: No effect.

1: The interrupt mode is set to the default interrupt mode (Both Edge detection).

23.7.38 PIO Additional Interrupt Modes Mask Register

Name: PIO_AIMMR

Address: 0xFFFFF4B8 (PIOA), 0xFFFFF6B8 (PIOB), 0xFFFFF8B8 (PIOC), 0xFFFFFAB8 (PIOD)

Access: Read-only

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

- **P0-P31: Peripheral CD Status.**

0: The interrupt source is a Both Edge detection event

1: The interrupt source is described by the registers PIO_ELSR and PIO_FRLHSR

23.7.39 PIO Edge Select Register

Name: PIO_ESR

Address: 0xFFFFF4C0 (PIOA), 0xFFFFF6C0 (PIOB), 0xFFFFF8C0 (PIOC), 0xFFFFFAC0 (PIOD)

Access: Write-only

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

- **P0-P31: Edge Interrupt Selection.**

0: No effect.

1: The interrupt source is an Edge detection event.

23.7.40 PIO Level Select Register

Name: PIO_LSR

Address: 0xFFFFF4C4 (PIOA), 0xFFFFF6C4 (PIOB), 0xFFFFF8C4 (PIOC), 0xFFFFFAC4 (PIOD)

Access: Write-only

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

- **P0-P31: Level Interrupt Selection.**

0: No effect.

1: The interrupt source is a Level detection event.

23.7.41 PIO Edge/Level Status Register

Name: PIO_ELSR

Address: 0xFFFFF4C8 (PIOA), 0xFFFFF6C8 (PIOB), 0xFFFFF8C8 (PIOC), 0xFFFFFAC8 (PIOD)

Access: Read-only

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

- **P0-P31: Edge/Level Interrupt source selection.**

0: The interrupt source is an Edge detection event.

1: The interrupt source is a Level detection event.

23.7.42 PIO Falling Edge/Low Level Select Register

Name: PIO_FELLSR

Address: 0xFFFFF4D0 (PIOA), 0xFFFFF6D0 (PIOB), 0xFFFFF8D0 (PIOC), 0xFFFFFAD0 (PIOD)

Access: Write-only

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

- **P0-P31: Falling Edge/Low Level Interrupt Selection.**

0: No effect.

1: The interrupt source is set to a Falling Edge detection or Low Level detection event, depending on PIO_ELSR.

23.7.43 PIO Rising Edge/High Level Select Register

Name: PIO_REHLSR

Address: 0xFFFFF4D4 (PIOA), 0xFFFFF6D4 (PIOB), 0xFFFFF8D4 (PIOC), 0xFFFFFAD4 (PIOD)

Access: Write-only

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

- **P0-P31: Rising Edge /High Level Interrupt Selection.**

0: No effect.

1: The interrupt source is set to a Rising Edge detection or High Level detection event, depending on PIO_ELSR.

23.7.44 PIO Fall/Rise - Low/High Status Register

Name: PIO_FRLHSR

Address: 0xFFFFF4D8 (PIOA), 0xFFFFF6D8 (PIOB), 0xFFFFF8D8 (PIOC), 0xFFFFFAD8 (PIOD)

Access: Read-only

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

- **P0-P31: Edge /Level Interrupt Source Selection.**

0: The interrupt source is a Falling Edge detection (if PIO_ELSR = 0) or Low Level detection event (if PIO_ELSR = 1).

1: The interrupt source is a Rising Edge detection (if PIO_ELSR = 0) or High Level detection event (if PIO_ELSR = 1).

23.7.45 PIO Lock Status Register

Name: PIO_LOCKSR

Address: 0xFFFFF4E0 (PIOA), 0xFFFFF6E0 (PIOB), 0xFFFFF8E0 (PIOC), 0xFFFFFAE0 (PIOD)

Access: Read-only

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

- **P0-P31: Lock Status.**

0: The I/O line is not locked.

1: The I/O line is locked.

23.7.46 PIO Write Protect Mode Register

Name: PIO_WPMR

Address: 0xFFFFF4E4 (PIOA), 0xFFFFF6E4 (PIOB), 0xFFFFF8E4 (PIOC), 0xFFFFFAE4 (PIOD)

Access: Read-write

Reset: See Table 23-2

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

For more information on Write Protection Registers, refer to Section 23.7 ”Parallel Input/Output Controller (PIO) User Interface”.

- **WPEN: Write Protect Enable**

0: Disables the Write Protect if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

1: Enables the Write Protect if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

Protects the registers:

“PIO Enable Register” on page 228

“PIO Disable Register” on page 229

“PIO Output Enable Register” on page 231

“PIO Output Disable Register” on page 232

“PIO Input Filter Enable Register” on page 234

“PIO Input Filter Disable Register” on page 235

“PIO Multi-driver Enable Register” on page 245

“PIO Multi-driver Disable Register” on page 246

“PIO Pull Up Disable Register” on page 248

“PIO Pull Up Enable Register” on page 249

“PIO Peripheral ABCD Select Register 1” on page 251

“PIO Peripheral ABCD Select Register 2” on page 252

“PIO Output Write Enable Register” on page 260

“PIO Output Write Disable Register” on page 261

“PIO Pad Pull Down Disable Register” on page 257

“PIO Pad Pull Down Status Register” on page 259

- **WPKEY: Write Protect KEY**

Should be written at value 0x50494F (“PIO” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

23.7.47 PIO Write Protect Status Register

Name: PIO_WPSR

Address: 0xFFFFF4E8 (PIOA), 0xFFFFF6E8 (PIOB), 0xFFFFF8E8 (PIOC), 0xFFFFFAE8 (PIOD)

Access: Read-only

Reset: See Table 23-2

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

- **WPVS: Write Protect Violation Status**

0: No Write Protect Violation has occurred since the last read of the PIO_WPSR register.

1: A Write Protect Violation has occurred since the last read of the PIO_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading PIO_WPSR automatically clears all fields.

23.7.48 PIO Schmitt Trigger Register

Name: PIO_SCHMITT

Address: 0xFFFFF500 (PIOA), 0xFFFFF700 (PIOB), 0xFFFFF900 (PIOC), 0xFFFFFB00 (PIOD)

Access: Read-write

Reset: See Table 23-2

31	30	29	28	27	26	25	24
SCHMITT31	SCHMITT30	SCHMITT29	SCHMITT28	SCHMITT27	SCHMITT26	SCHMITT25	SCHMITT24
23	22	21	20	19	18	17	16
SCHMITT23	SCHMITT22	SCHMITT21	SCHMITT20	SCHMITT19	SCHMITT18	SCHMITT17	SCHMITT16
15	14	13	12	11	10	9	8
SCHMITT15	SCHMITT14	SCHMITT13	SCHMITT12	SCHMITT11	SCHMITT10	SCHMITT9	SCHMITT8
7	6	5	4	3	2	1	0
SCHMITT7	SCHMITT6	SCHMITT5	SCHMITT4	SCHMITT3	SCHMITT2	SCHMITT1	SCHMITT0

- **SCHMITTx [x=0..31]:**

0: Schmitt Trigger is enabled.

1: Schmitt Trigger is disabled.

23.7.49 PIO I/O Delay Register

Name: PIO_DELAYR

Address: 0xFFFFF510 (PIOA), 0xFFFFF710 (PIOB), 0xFFFFF910 (PIOC), 0xFFFFFB10 (PIOD)

Access: Read-write

Reset: See Table 23-2

31	30	29	28	27	26	25	24
Delay7				Delay6			
23	22	21	20	19	18	17	16
Delay5				Delay4			
15	14	13	12	11	10	9	8
Delay3				Delay2			
7	6	5	4	3	2	1	0
Delay1				Delay0			

- **Delay x:**

Gives the number of elements in the delay line associated to pad x.

23.7.50 PIO I/O Drive Register 1

Name: PIO_DRIVER1

Address: 0xFFFFF514 (PIOA), 0xFFFFF714 (PIOB), 0xFFFFF914 (PIOC), 0xFFFFFB14 (PIOD)

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
LINE15		LINE14		LINE13		LINE12	
23	22	21	20	19	18	17	16
LINE11		LINE10		LINE9		LINE8	
15	14	13	12	11	10	9	8
LINE7		LINE6		LINE5		LINE4	
7	6	5	4	3	2	1	0
LINE3		LINE2		LINE1		LINE0	

- **LINEx [x=0..15]: Drive of PIO Line x**

Value	Name	Description
0	HI_DRIVE	High drive
1	ME_DRIVE	Medium drive
2	LO_DRIVE	Low drive
3		Reserved

23.7.51 PIO I/O Drive Register 2

Name: PIO_DRIVER2

Address: 0xFFFFF518 (PIOA), 0xFFFFF718 (PIOB), 0xFFFFF918 (PIOC), 0xFFFFFB18 (PIOD)

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
LINE31		LINE30		LINE29		LINE28	
23	22	21	20	19	18	17	16
LINE27		LINE26		LINE25		LINE24	
15	14	13	12	11	10	9	8
LINE23		LINE22		LINE21		LINE20	
7	6	5	4	3	2	1	0
LINE19		LINE18		LINE17		LINE16	

- **LINEx [x=16..31]: Drive of PIO line x**

Value	Name	Description
0	HI_DRIVE	High drive
1	ME_DRIVE	Medium drive
2	LO_DRIVE	Low drive
3		Reserved

24. Debug Unit (DBGU)

24.1 Description

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication. Moreover, the association with DMA 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.

24.2 Embedded Characteristics

- Composed of two functions
  - Two-pin UART
  - Debug Communication Channel (DCC) support
- Two-pin UART
  - Implemented features are 100% compatible with the standard Atmel USART
  - Independent receiver and transmitter with a common programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Support for two DMA channels with connection to receiver and transmitter
- Debug Communication Channel Support
  - Offers visibility of and interrupt trigger from COMMRX and COMMTX signals from the ARM Processor's ICE Interface

24.3 Block Diagram

Figure 24-1. Debug Unit Functional Block Diagram

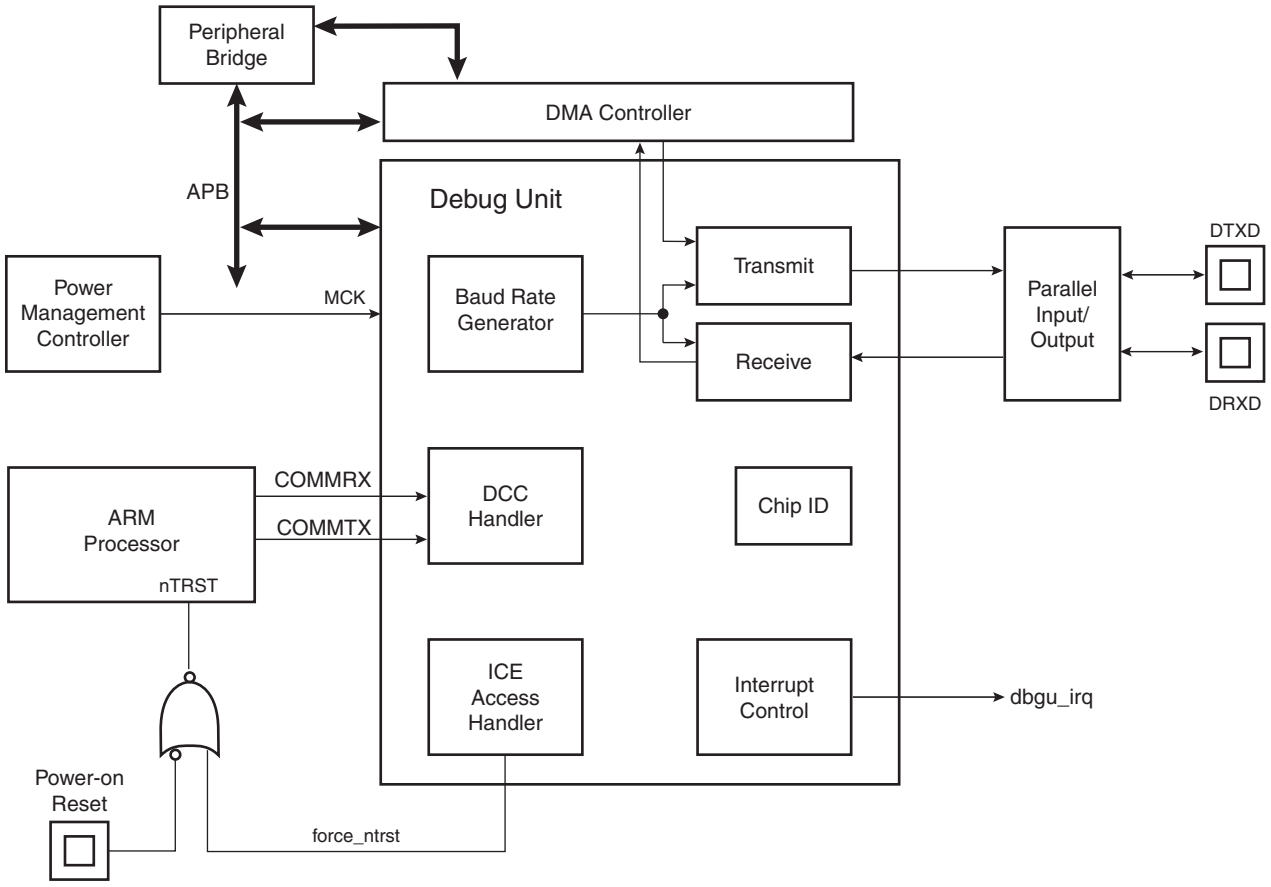
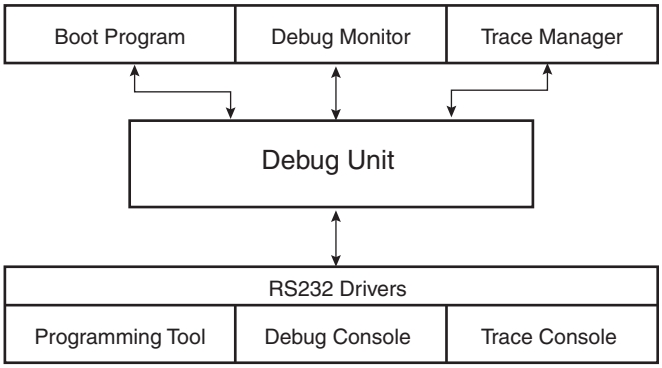


Table 24-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 24-2. Debug Unit Application Example



24.4 Product Dependencies

24.4.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

Table 24-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
DBGU	DRXD	PA9	A
DBGU	DTXD	PA10	A

24.4.2 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.

24.4.3 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 24-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

24.5 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.

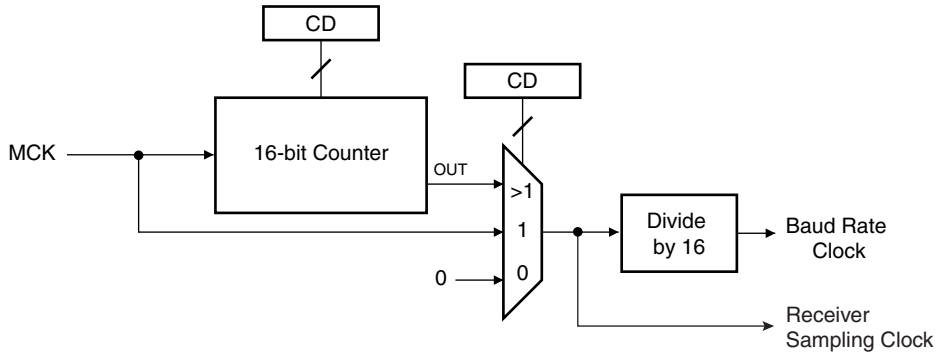
24.5.1 Baud Rate Generator

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 24-3. Baud Rate Generator



24.5.2 Receiver

24.5.2.1 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.

24.5.2.2 Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the DRXD at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after the falling edge of the start bit was detected.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 24-4. Start Bit Detection

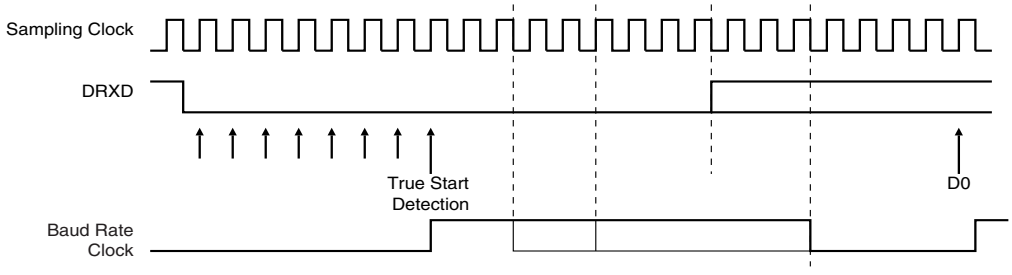
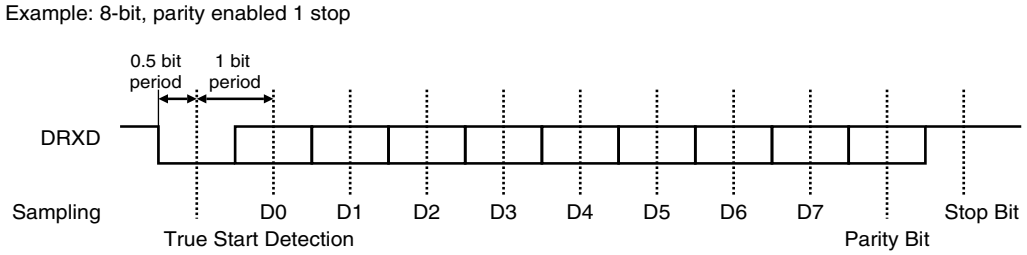


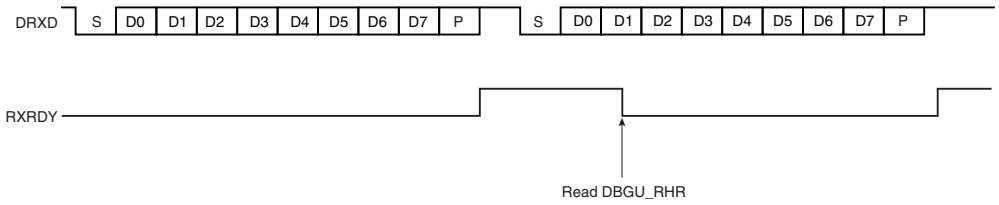
Figure 24-5. Character Reception



24.5.2.3 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.

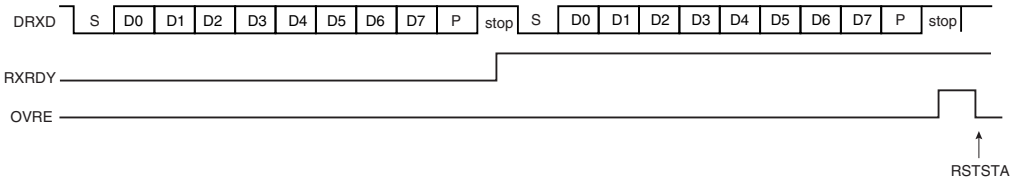
Figure 24-6. Receiver Ready



24.5.2.4 Receiver Overrun

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller or DMA 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.

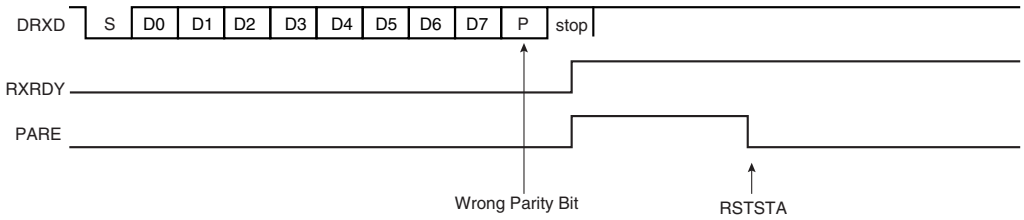
Figure 24-7. Receiver Overrun



24.5.2.5 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.

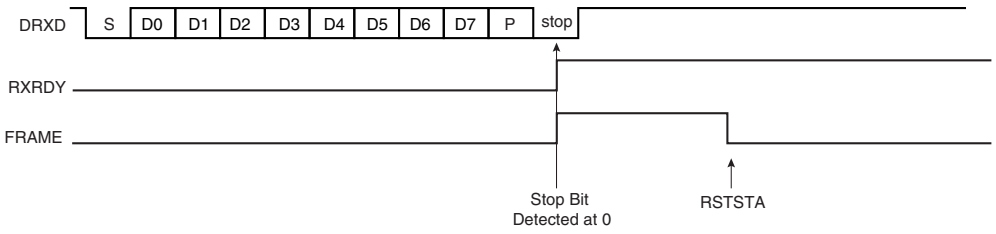
Figure 24-8. Parity Error



24.5.2.6 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 24-9. Receiver Framing Error



24.5.3 Transmitter

24.5.3.1 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.

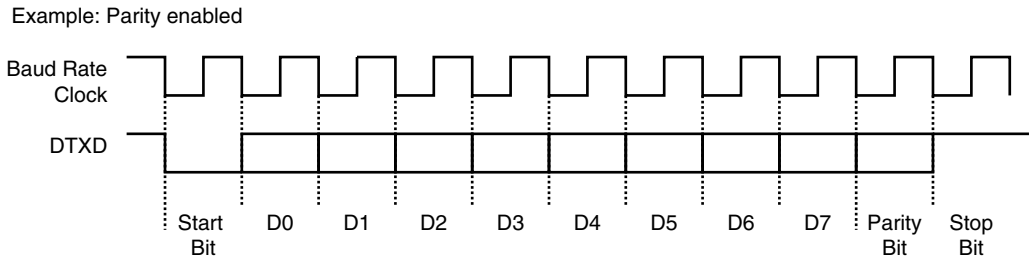
The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters 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.

24.5.3.2 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 24-10.Character Transmission

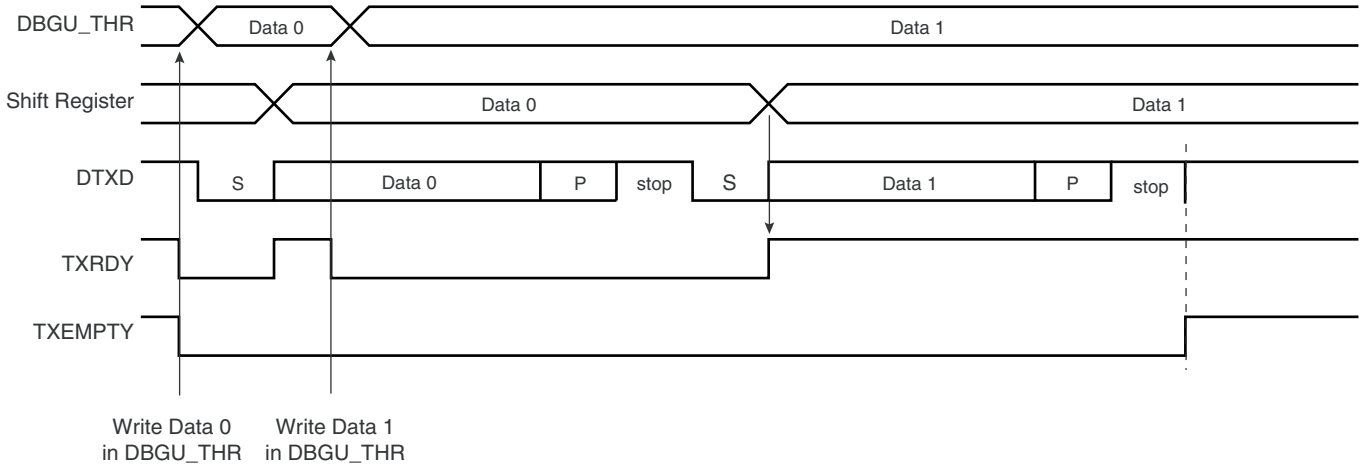


24.5.3.3 Transmitter Control

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 24-11.Transmitter Control



24.5.4 DMA Support

Both the receiver and the transmitter of the Debug Unit's UART are connected to a DMA Controller (DMAC) channel.

The DMA Controller channels are programmed via registers that are mapped within the DMAC user interface.

24.5.5 Test Modes

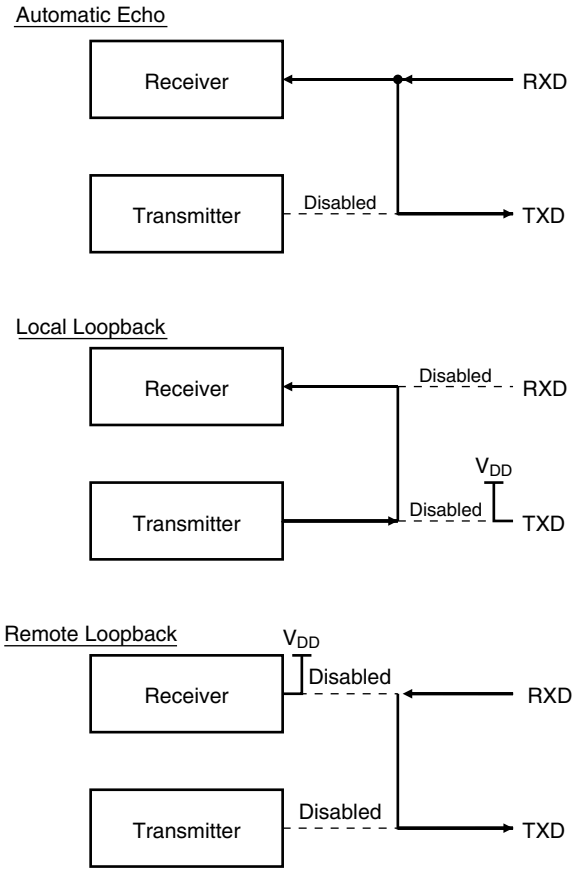
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

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 24-12.Test Modes



24.5.6 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.

24.5.7 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.

24.5.8 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.

24.6 Debug Unit (DBGU) User Interface

Table 24-3. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	DBGU_CR	Write-only	–
0x0004	Mode Register	DBGU_MR	Read-write	0x0
0x0008	Interrupt Enable Register	DBGU_IER	Write-only	–
0x000C	Interrupt Disable Register	DBGU_IDR	Write-only	–
0x0010	Interrupt Mask Register	DBGU_IMR	Read-only	0x0
0x0014	Status Register	DBGU_SR	Read-only	–
0x0018	Receive Holding Register	DBGU_RHR	Read-only	0x0
0x001C	Transmit Holding Register	DBGU_THR	Write-only	–
0x0020	Baud Rate Generator Register	DBGU_BRGR	Read-write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x0040	Chip ID Register	DBGU_CIDR	Read-only	–
0x0044	Chip ID Extension Register	DBGU_EXID	Read-only	–
0x0048	Force NTRST Register	DBGU_FNR	Read-write	0x0
0x004C - 0x00FC	Reserved	–	–	–

24.6.1 Debug Unit Control Register

Name: DBGU_CR

Address: 0xFFFFF200

Access: Write-only

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

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME and OVRE in the DBGU_SR.

24.6.2 Debug Unit Mode Register

Name: DBGU_MR

Address: 0xFFFFF204

Access: Read-write

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

- **PAR: Parity Type**

Value	Name	Description
0b000	EVEN	Even Parity
0b001	ODD	Odd Parity
0b010	SPACE	Space: Parity forced to 0
0b011	MARK	Mark: Parity forced to 1
0b1xx	NONE	No Parity

- **CHMODE: Channel Mode**

Value	Name	Description
0b00	NORM	Normal Mode
0b01	AUTO	Automatic Echo
0b10	LOCLOOP	Local Loopback
0b11	REMLOOP	Remote Loopback

24.6.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Address: 0xFFFFF208

Access: Write-only

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

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

24.6.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Address: 0xFFFFF20C

Access: Write-only

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

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

24.6.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Address: 0xFFFFF210

Access: Read-only

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

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

24.6.6 Debug Unit Status Register

Name: DBGU_SR

Address: 0xFFFFF214

Access: Read-only

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

- **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.

- **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.

24.6.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Address: 0xFFFFF218

Access: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

24.6.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Address: 0xFFFFF21C

Access: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

24.6.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Address: 0xFFFFF220

Access: Read-write

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

- **CD: Clock Divisor**

Value	Name	Description
0	DISABLED	DBGU Disabled
1	MCK	MCK
2 to 65535	–	MCK / (CD x 16)

24.6.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

Address: 0xFFFFF240

Access: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**

Values depend upon the version of the device.

- **EPROC: Embedded Processor**

Value	Name	Description
1	ARM946ES	ARM946ES
2	ARM7TDMI	ARM7TDMI
3	CM3	Cortex®-M3
4	ARM920T	ARM920T
5	ARM926EJS	ARM926EJS
6	CA5	Cortex®-A5

- **NVPSIZ: Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8K bytes
2	16K	16K bytes
3	32K	32K bytes
4	–	Reserved
5	64K	64K bytes
6	–	Reserved
7	128K	128K bytes
8	–	Reserved
9	256K	256K bytes
10	512K	512K bytes
11	–	Reserved
12	1024K	1024K bytes
13	–	Reserved
14	2048K	2048K bytes

Value	Name	Description
15	–	Reserved

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8K bytes
2	16K	16K bytes
3	32K	32K bytes
4	–	Reserved
5	64K	64K bytes
6		Reserved
7	128K	128K bytes
8	–	Reserved
9	256K	256K bytes
10	512K	512K bytes
11	–	Reserved
12	1024K	1024K bytes
13	–	Reserved
14	2048K	2048K bytes
15	–	Reserved

- **SRAMSIZ: Internal SRAM Size**

Value	Name	Description
0	–	Reserved
1	1K	1K bytes
2	2K	2K bytes
3	6K	6K bytes
4	112K	112K bytes
5	4K	4K bytes
6	80K	80K bytes
7	160K	160K bytes
8	8K	8K bytes
9	16K	16K bytes
10	32K	32K bytes
11	64K	64K bytes
12	128K	128K bytes
13	256K	256K bytes
14	96K	96K bytes
15	512K	512K bytes

- **ARCH: Architecture Identifier**

Value	Name	Description
0x19	AT91SAM9xx	AT91SAM9xx Series
0x29	AT91SAM9XExx	AT91SAM9XExx Series
0x34	AT91x34	AT91x34 Series
0x37	CAP7	CAP7 Series
0x39	CAP9	CAP9 Series
0x3B	CAP11	CAP11 Series
0x40	AT91x40	AT91x40 Series
0x42	AT91x42	AT91x42 Series
0x55	AT91x55	AT91x55 Series
0x60	AT91SAM7Axx	AT91SAM7Axx Series
0x61	AT91SAM7AQxx	AT91SAM7AQxx Series
0x63	AT91x63	AT91x63 Series
0x70	AT91SAM7Sxx	AT91SAM7Sxx Series
0x71	AT91SAM7XCxx	AT91SAM7XCxx Series
0x72	AT91SAM7SExx	AT91SAM7SExx Series
0x73	AT91SAM7Lxx	AT91SAM7Lxx Series
0x75	AT91SAM7Xxx	AT91SAM7Xxx Series
0x76	AT91SAM7SLxx	AT91SAM7SLxx Series
0x80	ATSAM3UxC	ATSAM3UxC Series (100-pin version)
0x81	ATSAM3UxE	ATSAM3UxE Series (144-pin version)
0x83	ATSAM3AxC	ATSAM3AxC Series (100-pin version)
0x84	ATSAM3XxC	ATSAM3XxC Series (100-pin version)
0x85	ATSAM3XxE	ATSAM3XxE Series (144-pin version)
0x86	ATSAM3XxG	ATSAM3XxG Series (208/217-pin version)
0x88	ATSAM3SxA	ATSAM3SxA Series (48-pin version)
0x89	ATSAM3SxB	ATSAM3SxB Series (64-pin version)
0x8A	ATSAM3SxC	ATSAM3SxC Series (100-pin version)
0x92	AT91x92	AT91x92 Series
0x93	ATSAM3NxA	ATSAM3NxA Series (48-pin version)
0x94	ATSAM3NxB	ATSAM3NxB Series (64-pin version)
0x95	ATSAM3NxC	ATSAM3NxC Series (100-pin version)
0x98	ATSAM3SDxA	ATSAM3SDxA Series (48-pin version)
0x99	ATSAM3SDxB	ATSAM3SDxB Series (64-pin version)
0x9A	ATSAM3SDxC	ATSAM3SDxC Series (100-pin version)
0xA5	–	Reserved
0xF0	AT75Cxx	AT75Cxx Series

- **NVPTYP: Nonvolatile Program Memory Type**

Value	Name	Description
0	ROM	ROM
1	ROMLESS	ROMless or on-chip Flash
4	SRAM	SRAM emulating ROM
2	FLASH	Embedded Flash Memory
3	ROM_FLASH	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.

24.6.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Address: 0xFFFFF244

Access: Read-only

31	30	29	28	27	26	25	24
EXID							

23	22	21	20	19	18	17	16
EXID							

15	14	13	12	11	10	9	8
EXID							

7	6	5	4	3	2	1	0
EXID							

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

24.6.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Address: 0xFFFFF248

Access: Read-write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor’s TAP controller is driven by the power_on_reset signal.

1 = NTRST of the ARM processor’s TAP controller is held low.

25. Bus Matrix (MATRIX)

25.1 Description

The Bus Matrix implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 16 AHB masters to up to 16 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.

25.2 Embedded Characteristics

- 12-layer Matrix, handling requests from 11 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 ROM boot, one for internal flash boot, one after remap
- Boot Mode Select
  - Non-volatile Boot Memory can be internal ROM or external memory on EBI_NCS0
  - Selection is made by General purpose NVM bit sampled at reset
- Remap Command
  - Allows Remapping of an Internal SRAM in Place of the Boot Non-Volatile Memory (ROM or External Flash)
  - Allows Handling of Dynamic Exception Vectors

25.2.1 Matrix Masters

The Bus Matrix manages 12 masters, which means that each master can perform an access concurrently with others, depending on whether the slave it accesses is available.

Each master has its own decoder, which can be defined specifically for each master. In order to simplify the addressing, all the masters have the same decodings.

Table 25-1. List of Bus Matrix Masters

Master 0	ARM926 Instruction
Master 1	ARM926 Data
Master 2 & 3	DMA Controller 0
Master 4 & 5	DMA Controller 1
Master 6	UDP HS DMA
Master 7	UHP EHCI DMA
Master 8	UHP OHCI DMA
Master 9	Reserved
Master 10	EMAC0 DMA
Master 11	EMAC1 DMA

25.2.2 Matrix Slaves

The Bus Matrix manages 10 slaves. Each slave has its own arbiter, thus allowing a different arbitration per slave to be programmed.

Table 25-2. List of Bus Matrix Slaves

Slave 0	Internal SRAM
Slave 1	Internal ROM
Slave 2	Soft Modem (SMD)
Slave 3	USB Device High Speed Dual Port RAM (DPR)
	USB Host EHCI registers
	USB Host OHCI registers
Slave 4	External Bus Interface
Slave 5	DDR2 port 1
Slave 6	DDR2 port 2
Slave 7	DDR2 port 3
Slave 8	Peripheral Bridge 0
Slave 9	Peripheral Bridge 1

25.2.3 Master to Slave Access

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

Table 25-3. Master to Slave Access

Masters		0	1	2 & 3	4 & 5	6	7	8	9	10	11
Slaves		ARM926 Instr.	ARM926 Data	DMA 0	DMA 1	USB Device HS DMA	USB Host HS EHCI	USB Host HS OHCI	Reserved	EMAC0 DMA	EMAC1 DMA
0	Internal SRAM	X	X	X	X	X	X	X	X	X	X
1	Internal ROM	X	X	X	X	–	–	–	–	–	–
2	SMD	X	X	–	X	–	–	–	–	–	–
3	USB Device High Speed DPR USB Host EHCI registers USB Host OHCI registers	X	X	–	–	–	–	–	–	–	–
4	External Bus Interface	X	X	X	X	X	X	X	X	X	X
5	DDR2 Port 1	X	–	X	–	–	–	–	–	–	–
6	DDR2 Port 2	–	X	–	X	–	–	–	–	–	–
7	DDR2 Port 3	–	–	–	–	–	–	–	X	–	–
8	Peripheral Bridge 0	X	X	X	X	–	–	–	–	–	–
9	Peripheral Bridge 1	X	X	X	X	–	–	–	–	–	–

25.3 Memory Mapping

The Bus Matrix provides one decoder for every AHB master interface. The decoder offers each AHB master several memory mappings. 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 the Master Remap Control Register (MATRIX_MRCR), that performs remap action for every master independently.

25.4 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from masters. This mechanism reduces latency at first access of a burst, or single transfer, as long as the slave is free from any other master access, but does not provide any benefit as soon as the slave is continuously accessed by more than one master, since arbitration is pipelined and has no negative effect on the slave bandwidth or access latency.

This bus granting mechanism sets a different default master for every slave.

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

- No default master
- Last access master
- Fixed default master

To change from one type of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for every slave, that set a default master for each slave. The Slave Configuration Register contains two fields: DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no

default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Refer to Section 25.7.2 “Bus Matrix Slave Configuration Registers”.

25.4.1 No Default Master

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

This configuration incurs one latency clock cycle for the first access of a burst after bus Idle. Arbitration without default master may be used for masters that perform significant bursts or several transfers with no Idle in between, or if the slave bus bandwidth is widely used by one or more masters.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput, irregardless of the number of requesting masters.

25.4.2 Last Access Master

After 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.

This allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. Other non-privileged masters still get one latency clock cycle if they want to access the same slave. This technique is useful for masters that mainly perform single accesses or short bursts with some Idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput irregardless of the number of requesting masters.

25.4.3 Fixed Default Master

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

This allows the Bus Matrix arbiters to remove the one latency clock cycle for the fixed default master of the slave. All requests attempted by the fixed default master do not cause any arbitration latency, whereas other non-privileged masters will get one latency cycle. This technique is useful for a master that mainly performs single accesses or short bursts with Idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput, irregardless of the number of requesting masters.

25.5 Arbitration

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

The Bus Matrix provides the user with the possibility of choosing between two arbitration types or mixing them for each slave:

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

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

When re-arbitration is required, specific conditions apply. See Section 25.5.1 “Arbitration Scheduling”.

25.5.1 Arbitration Scheduling

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

1. Idle Cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: When a slave is currently doing a single access.
3. End of Burst Cycles: When the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. See Section 25.5.1.1 “Undefined Length Burst Arbitration”
4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See Section 25.5.1.2 “Slot Cycle Limit Arbitration”

25.5.1.1 Undefined Length Burst Arbitration

In order to prevent long AHB burst lengths that can lock the access to the slave for an excessive period of time, the user can trigger the re-arbitration before the end of the incremental bursts. The re-arbitration period can be selected from the following Undefined Length Burst Type (ULBT) possibilities:

1. Unlimited: no predetermined end of burst is generated. This value enables 1-kbyte burst lengths.
2. 1-beat bursts: predetermined end of burst is generated at each single transfer during the INCR transfer.
3. 4-beat bursts: predetermined end of burst is generated at the end of each 4-beat boundary during INCR transfer.
4. 8-beat bursts: predetermined end of burst is generated at the end of each 8-beat boundary during INCR transfer.
5. 16-beat bursts: predetermined end of burst is generated at the end of each 16-beat boundary during INCR transfer.
6. 32-beat bursts: predetermined end of burst is generated at the end of each 32-beat boundary during INCR transfer.
7. 64-beat bursts: predetermined end of burst is generated at the end of each 64-beat boundary during INCR transfer.
8. 128-beat bursts: predetermined end of burst is generated at the end of each 128-beat boundary during INCR transfer.

Use of undefined length16-beat bursts, or less, is discouraged since this generally decreases significantly overall bus bandwidth due to arbitration and slave latencies at each first access of a burst.

If the master does not permanently and continuously request the same slave or has an intrinsically limited average throughput, the ULBT should be left at its default unlimited value, knowing that the AHB specification natively limits all word bursts to 256 beats and double-word bursts to 128 beats because of its 1 Kbyte address boundaries.

Unless duly needed, the ULBT should be left at its default value of 0 for power saving.

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

25.5.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as back-to-back undefined length bursts or very long bursts on a very slow slave (e.g., an external low speed memory). At each arbitration time 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 elapses, the arbiter has the ability to re-arbitrate at the end of the current AHB bus access cycle.

Unless a master has a very tight access latency constraint, which could lead to data overflow or underflow due to a badly undersized internal FIFO with respect to its throughput, the Slot Cycle Limit should be disabled (SLOT_CYCLE = 0) or set to its default maximum value in order not to inefficiently break long bursts performed by some Atmel masters.

However, the Slot Cycle Limit should not be disabled in the particular case of a master capable of accessing the slave by performing back-to-back undefined length bursts shorter than the number of ULBT beats with no Idle cycle in between, since in this case the arbitration could be frozen all along the burst sequence.

In most cases this feature is not needed and should be disabled for power saving.

Warning: This feature cannot prevent any slave from locking its access indefinitely.

25.5.2 Arbitration Priority Scheme

The bus Matrix arbitration scheme is organized in priority pools.

Round-robin priority is used in the highest and lowest priority pools, whereas fixed level priority is used between priority pools and in the intermediate priority pools.

For each slave, each master is assigned to one of the slave priority pools through the priority registers for slaves (MxPR fields of MATRIX_PRAS and MATRIX_PRBS). When evaluating master requests, this programmed priority level always takes precedence.

After reset, all the masters belong to the lowest priority pool (MxPR = 0) and are therefore granted bus access in a true round-robin order.

The highest priority pool must be specifically reserved for masters requiring very low access latency. If more than one master belongs to this pool, they will be granted bus access in a biased round-robin manner which allows tight and deterministic maximum access latency from AHB bus requests. At worst, any currently occurring high-priority master request will be granted after the current bus master access has ended and other high priority pool master requests, if any, have been granted once each.

The lowest priority pool shares the remaining bus bandwidth between AHB Masters.

Intermediate priority pools allow fine priority tuning. Typically, a moderately latency-critical master or a bandwidth-only critical master will use such a priority level. The higher the priority level (MxPR value), the higher the master priority.

All combinations of MxPR values are allowed for all masters and slaves. For example some masters might be assigned to the highest priority pool (round-robin) and the remaining masters to the lowest priority pool (round-robin), with no master for intermediate fix priority levels.

If more than one master requests the slave bus, irregardless of the respective masters priorities, no master will be granted the slave bus for two consecutive runs. A master can only get back-to-back grants so long as it is the only requesting master.

25.5.2.1 Fixed Priority Arbitration

Fixed priority arbitration algorithm is the first and only arbitration algorithm applied between masters from distinct priority pools. It is also used in priority pools other than the highest and lowest priority pools (intermediate priority pools).

Fixed priority arbitration 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 in the MxPR field for each master in the Priority Registers, MATRIX_PRAS and MATRIX_PRBS. If two or more master requests are active at the same time, the master with the highest priority MxPR number is serviced first.

In intermediate priority pools, if two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

25.5.2.2 Round-Robin Arbitration

This algorithm is only used in the highest and lowest priority pools. It allows the Bus Matrix arbiters to properly dispatch requests from different masters to the same slave. If two or more master requests are active at the same time in the priority pool, they are serviced in a round-robin increasing master number order.

25.6 Register Write Protection

To prevent any single software error from corrupting MATRIX behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the "Write Protection Mode Register" (MATRIX_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the "Write Protection Status Register" (MATRIX_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the MATRIX_WPSR.

The following registers can be write-protected:

- "Bus Matrix Master Configuration Registers"
- "Bus Matrix Slave Configuration Registers"
- "Bus Matrix Priority Registers A For Slaves"
- "Bus Matrix Priority Registers B For Slaves"
- "Bus Matrix Master Remap Control Register"

25.7 Bus Matrix (MATRIX) User Interface

Table 25-4. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read/Write	0x00000001
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	Reserved	–	–	–
0x0028	Master Configuration Register 10	MATRIX_MCFG10	Read/Write	0x00000000
0x002C	Master Configuration Register 11	MATRIX_MCFG11	Read/Write	0x00000000
0x0030–0x003C	Reserved	–	–	–
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x000001FF
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x000001FF
0x0048	Slave Configuration Register 2	MATRIX_SCFG2	Read/Write	0x000001FF
0x004C	Slave Configuration Register 3	MATRIX_SCFG3	Read/Write	0x000001FF
0x0050	Slave Configuration Register 4	MATRIX_SCFG4	Read/Write	0x000001FF
0x0054	Slave Configuration Register 5	MATRIX_SCFG5	Read/Write	0x000001FF
0x0058	Slave Configuration Register 6	MATRIX_SCFG6	Read/Write	0x000001FF
0x005C	Slave Configuration Register 7	MATRIX_SCFG7	Read/Write	0x000001FF
0x0060	Slave Configuration Register 8	MATRIX_SCFG8	Read/Write	0x000001FF
0x0064	Slave Configuration Register 9	MATRIX_SCFG9	Read/Write	0x000001FF
0x0068–0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read/Write	0x00000000
0x0084	Priority Register B for Slave 0	MATRIX_PRBS0	Read/Write	0x00000000
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read/Write	0x00000000
0x008C	Priority Register B for Slave 1	MATRIX_PRBS1	Read/Write	0x00000000
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read/Write	0x00000000
0x0094	Priority Register B for Slave 2	MATRIX_PRBS2	Read/Write	0x00000000
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read/Write	0x00000000
0x009C	Priority Register B for Slave 3	MATRIX_PRBS3	Read/Write	0x00000000
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read/Write	0x00000000
0x00A4	Priority Register B for Slave 4	MATRIX_PRBS4	Read/Write	0x00000000
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Read/Write	0x00000000

Table 25-4. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x00AC	Priority Register B for Slave 5	MATRIX_PRBS5	Read/Write	0x00000000
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Read/Write	0x00000000
0x00B4	Priority Register B for Slave 6	MATRIX_PRBS6	Read/Write	0x00000000
0x00B8	Priority Register A for Slave 7	MATRIX_PRAS7	Read/Write	0x00000000
0x00BC	Priority Register B for Slave 7	MATRIX_PRBS7	Read/Write	0x00000000
0x00C0	Priority Register A for Slave 8	MATRIX_PRAS8	Read/Write	0x00000000
0x00C4	Priority Register B for Slave 8	MATRIX_PRBS8	Read/Write	0x00000000
0x00C8	Priority Register A for Slave 9	MATRIX_PRAS9	Read/Write	0x00000000
0x00CC	Priority Register B for Slave 9	MATRIX_PRBS9	Read/Write	0x00000000
0x00D0–0x00FC	Reserved	–	–	–
0x0100	Master Remap Control Register	MATRIX_MRCR	Read/Write	0x00000000
0x0104–0x011C	Reserved	–	–	–
0x0120	EBI Chip Select Assignment Register	CCFG_EBICSA	Read/Write	0x00000200
0x0124–0x01FC	Reserved	–	–	–
0x01E4	Write Protection Mode Register	MATRIX_WPMR	Read/Write	0x00000000
0x01E8	Write Protection Status Register	MATRIX_WPSR	Read-only	0x00000000

25.7.1 Bus Matrix Master Configuration Registers

Name: MATRIX_MCFG0...MATRIX_MCFG11

Address: 0xFFFFDE00

Address: 0xFFFFDE04

Address: 0xFFFFDE08

Address: 0xFFFFDE0C

Address: 0xFFFFDE10

Address: 0xFFFFDE14

Address: 0xFFFFDE18

Address: 0xFFFFDE1C

Address: 0xFFFFDE20

Address: 0xFFFFDE28

Address: 0xFFFFDE2C

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in the “Write Protection Mode Register” .

- **ULBT: Undefined Length Burst Type**

0: Unlimited Length Burst

No predicted end of burst is generated, therefore INCR bursts coming from this master can only be broken if the Slave Slot Cycle Limit is reached. If the Slot Cycle Limit is not reached, the burst is normally completed by the master, at the latest, on the next AHB 1 Kbyte address boundary, allowing up to 256-beat word bursts or 128-beat double-word bursts.

1: Single Access

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

2: 4-beat Burst

The undefined length burst is split into 4-beat bursts, allowing re-arbitration at each 4-beat burst end.

3: 8-beat Burst

The undefined length burst is split into 8-beat bursts, allowing re-arbitration at each 8-beat burst end.

4: 16-beat Burst

The undefined length burst is split into 16-beat bursts, allowing re-arbitration at each 16-beat burst end.

5: 32-beat Burst

The undefined length burst is split into 32-beat bursts, allowing re-arbitration at each 32-beat burst end.

6: 64-beat Burst

The undefined length burst is split into 64-beat bursts, allowing re-arbitration at each 64-beat burst end.

7: 128-beat Burst

The undefined length burst is split into 128-beat bursts, allowing re-arbitration at each 128-beat burst end.

Unless duly needed, the ULBT should be left at its default 0 value for power saving.

25.7.2 Bus Matrix Slave Configuration Registers

Name: MATRIX_SCFG0...MATRIX_SCFG9

Address: 0xFFFFDE40[0], 0xFFFFDE44 [1], 0xFFFFDE48 [2], 0xFFFFDE4C [3], 0xFFFFDE50 [4], 0xFFFFDE54 [5], 0xFFFFDE58 [6], 0xFFFFDE5C [7], 0xFFFFDE60 [8], 0xFFFFDE64 [9]

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in the "Write Protection Mode Register" .

- **SLOT_CYCLE: Maximum Bus Grant Duration for Masters**

When SLOT_CYCLE AHB clock cycles have elapsed since the last arbitration, a new arbitration takes place so as to let another master access this slave. If another master is requesting the slave bus, then the current master burst is broken.

If SLOT_CYCLE = 0, the Slot Cycle Limit feature is disabled and bursts always complete unless broken according to the ULBT.

This limit has been placed in order to enforce arbitration so as to meet potential latency constraints of masters waiting for slave access or in the particular case of a master performing back-to-back undefined length bursts indefinitely freezing the arbitration.

This limit must not be too small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. The default maximum value is usually an optimal conservative choice.

In most cases this feature is not needed and should be disabled for power saving. See Section 25.5.1.2 on page 307.

- **DEFMSTR_TYPE: Default Master Type**

0: No Default Master

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

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

1: Last Default Master

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

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

2: Fixed Default Master

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

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

- **FIXED_DEFMSTR: Fixed Default Master**

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

25.7.3 Bus Matrix Priority Registers A For Slaves

Name: MATRIX_PRAS0...MATRIX_PRAS9

Address: 0xFFFFDE80 [0], 0xFFFFDE88 [1], 0xFFFFDE90 [2], 0xFFFFDE98 [3], 0xFFFFDEA0 [4], 0xFFFFDEA8 [5], 0xFFFFDEB0 [6], 0xFFFFDEB8 [7], 0xFFFFDEC0 [8], 0xFFFFDEC8 [9]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	M7PR		–	–	M6PR	
23	22	21	20	19	18	17	16
–	–	M5PR		–	–	M4PR	
15	14	13	12	11	10	9	8
–	–	M3PR		–	–	M2PR	
7	6	5	4	3	2	1	0
–	–	M1PR		–	–	M0PR	

This register can only be written if the WPEN bit is cleared in the "Write Protection Mode Register" .

- **MxPR: Master x Priority**

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

All the masters programmed with the same MxPR value for the slave make up a priority pool.

Round-robin arbitration is used in the lowest (MxPR = 0) and highest (MxPR = 3) priority pools.

Fixed priority is used in intermediate priority pools (MxPR = 1) and (MxPR = 2).

See "Arbitration Priority Scheme" on page 308 for details.

25.7.4 Bus Matrix Priority Registers B For Slaves

Name: MATRIX_PRBS0...MATRIX_PRBS9

Address: 0xFFFFDE84 [0], 0xFFFFDE8C [1], 0xFFFFDE94 [2], 0xFFFFDE9C [3], 0xFFFFDEA4 [4], 0xFFFFDEAC [5], 0xFFFFDEB4 [6], 0xFFFFDEBC [7], 0xFFFFDEC4 [8], 0xFFFFDECC [9]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	M11PR		–	–	M10PR	
7	6	5	4	3	2	1	0
–	–	–	–	–	–	M8PR	

This register can only be written if the WPEN bit is cleared in the “Write Protection Mode Register” .

- **MxPR: Master x Priority**

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

All the masters programmed with the same MxPR value for the slave make up a priority pool.

Round-robin arbitration is used in the lowest (MxPR = 0) and highest (MxPR = 3) priority pools.

Fixed priority is used in intermediate priority pools (MxPR = 1) and (MxPR = 2).

See “Arbitration Priority Scheme” on page 308 for details.

25.7.5 Bus Matrix Master Remap Control Register

Name: MATRIX_MRCR

Address: 0xFFFFDF00

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RCB11	RCB10	–	RCB8
7	6	5	4	3	2	1	0
RCB7	RCB6	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

This register can only be written if the WPEN bit is cleared in the "Write Protection Mode Register" .

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

0: Disable remapped address decoding for the selected Master

1: Enable remapped address decoding for the selected Master

25.7.6 EBI Chip Select Assignment Register

Name: CCFG_EBICSA

Address: 0xFFFFDF20

Access: Read/Write

Reset: 0x00000200

31	30	29	28	27	26	25	24
–	–	–	–	–	–	DDR_MP_EN	NFD0_ON_D16
23	22	21	20	19	18	17	16
–	–	–	–	–	–	EBI_DRIVE	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	EBI_DBPDC	EBI_DBPUC
7	6	5	4	3	2	1	0
–	–	–	–	EBI_CS3A	–	EBI_CS1A	–

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

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

1: EBI Chip Select 1 is assigned to the DDR2SDR Controller.

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

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

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

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

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

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

- **EBI_DBPDC: EBI Data Bus Pull-Down Configuration**

0: EBI D0–D15 Data Bus bits are internally pulled-down to the ground.

1: EBI D0–D15 Data Bus bits are not internally pulled-down.

- **EBI_DRIVE: EBI I/O Drive Configuration**

This allows to avoid overshoots and gives the best performance according to the bus load and external memories.

0: Low drive (default).

1: High drive.

- **NFD0_ON_D16: NAND Flash Databus Selection**

0: NAND Flash I/O are connected to D0–D15 (default).

1: NAND Flash I/O are connected to D16–D31.

NFD0_ON_D16	Signals	VDDIOM	VDDNF	External Memory
0	NFD0 = D0,..., NFD15 = D15	1.8V	1.8V	DDR2 or LP-DDR or LPSDR + NAND Flash 1.8V
0	NFD0 = D0,..., NFD15 = D15	3.3V	3.3V	32-bit SDRAM + NAND Flash 3.3V
1	NFD0 = D16,..., NFD15 = D31	1.8V	1.8V	DDR2 or LP-DDR or LPSDR + NAND Flash 1.8V
1	NFD0 = D16,..., NFD15 = D31	1.8V	3.3V	DDR2 or LP-DDR or LPSDR + NAND Flash 3.3V
1	NFD0 = D16,..., NFD15 = D31	3.3V	1.8V	16-bit SDR + NAND Flash 1.8V

- **DDR_MP_EN: DDR Multi-port Enable**

0: DDR Multi-port is disabled (default).

1: DDR Multi-port is enabled, performance is increased. **Warning:** Use only with NFDO0_ON_D16 = 0. The system behavior is unpredictable if ND0_ON_D16 is set to 1 at the same time.

Note: EBI Chip Select 1 is to be assigned to the DDR2SDR Controller.

25.7.7 Write Protection Mode Register

Name: MATRIX_WPMR

Address: 0xFFFFDFE4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

See Section 25.6 “Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x4D4154	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

25.7.8 Write Protection Status Register

Name: MATRIX_WPSR

Address: 0xFFFFDFE8

Access: Read-only

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the MATRIX_WPSR.

1: A write protection violation has occurred since the last read of the MATRIX_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

26. External Bus Interface (EBI)

26.1 Description

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

The Static Memory, DDR, SDRAM 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, DDR2 and SDRAM. The EBI operates with 1.8V or 3.3V Power Supply (VDDIOM).

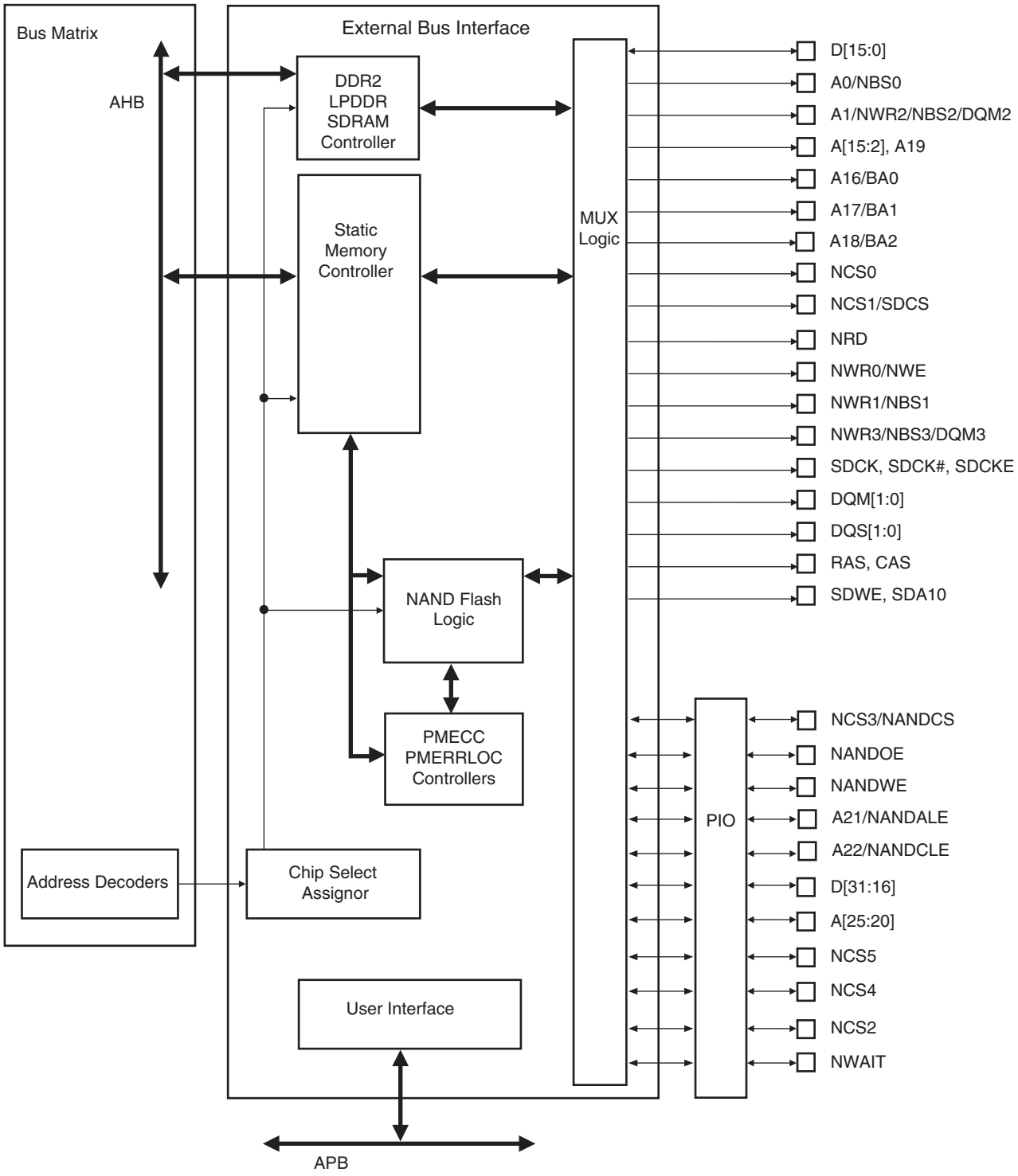
The EBI also supports the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded Memory Controller. Data transfers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

26.2 Embedded Characteristics

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - DDR2/SDRAM Controller
  - 8-bit NAND Flash ECC Controller
- Up to 26-bit Address Bus (up to 64 Mbytes linear per chip select)
- Up to 6 chips selects, Configurable Assignment:
  - Static Memory Controller on NCS0, NCS1, NCS2, NCS3, NCS4, NCS5
  - DDR2/SDRAM Controller (SDCS) or Static Memory Controller on NCS1
  - NAND Flash support on NCS3

26.3 EBI Block Diagram

Figure 26-1. Organization of the External Bus Interface



26.4 I/O Lines Description

Table 26-1. EBI I/O Lines Description

Name	Function	Type	Active Level
EBI			
EBI_D0–EBI_D31	Data Bus	I/O	
EBI_A0–EBI_A25	Address Bus	Output	
EBI_NWAIT	External Wait Signal	Input	Low
SMC			
EBI_NCS0–EBI_NCS5	Chip Select Lines	Output	Low
EBI_NWR0–EBI_NWR3	Write Signals	Output	Low
EBI_NRD	Read Signal	Output	Low
EBI_NWE	Write Enable	Output	Low
EBI_NBS0–EBI_NBS3	Byte Mask Signals	Output	Low
EBI for NAND Flash Support			
EBI_NANDCS	NAND Flash Chip Select Line	Output	Low
EBI_NANDOE	NAND Flash Output Enable	Output	Low
EBI_NANDWE	NAND Flash Write Enable	Output	Low
DDR2/SDRAM Controller			
EBI_SDCK, EBI_SDCK#	DDR2/SDRAM Differential Clock	Output	
EBI_SDCKE	DDR2/SDRAM Clock Enable	Output	High
EBI_SDCS	DDR2/SDRAM Controller Chip Select Line	Output	Low
EBI_BA0–2	Bank Select	Output	
EBI_SDWE	DDR2/SDRAM Write Enable	Output	Low
EBI_RAS - EBI_CAS	Row and Column Signal	Output	Low
EBI_SDA10	SDRAM Address 10 Line	Output	

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

Table 26-2 details the connections between the two Memory Controllers and the EBI pins.

Table 26-2. EBI Pins and Memory Controllers I/O Lines Connections

EBIx Pins	SDRAM I/O Lines	SMC I/O Lines
EBI_NWR1/NBS1/CFIOR	NBS1	NWR1
EBI_A0/NBS0	Not Supported	SMC_A0
EBI_A1/NBS2/NWR2	Not Supported	SMC_A1
EBI_A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
EBI_SDA10	SDRAMC_A10	Not Supported
EBI_A12	Not Supported	SMC_A12
EBI_A[15:13]	SDRAMC_A[13:11]	SMC_A[15:13]
EBI_A[25:16]	Not Supported	SMC_A[25:16]
EBI_D[31:0]	D[31:0]	D[31:0]

26.5 Application Example

26.5.1 Hardware Interface

Table 26-3 details the connections to be applied between the EBI pins and the external devices for each Memory Controller.

Table 26-3. EBI Pins and External Static Device Connections

Signals: EBI_	Pins of the Interfaced Device					
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
Controller	**SMC**					
D0–D7	D0–D7	D0–D7	D0–D7	D0–D7	D0–D7	D0–D7
D8–D15	–	D8–D15	D8–D15	D8–D15	D8–15	D8–15
D16–D23	–	–	–	D16–D23	D16–D23	D16–D23
D24–D31(5))	–	–	–	D24–D31	D24–D31	D24–D31
A0/NBS0	A0	–	NLB	–	NLB (3)	BE0
A1/NWR2/NBS2/DQM2	A1	A0	A0	WE (2)	NLB (4)	BE2
A2–A22(5)	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/DDRSDCS	CS	CS	CS	CS	CS	CS
NCS2(5)	CS	CS	CS	CS	CS	CS
NCS3/NANDCS	CS	CS	CS	CS	CS	CS
NCS4(5)	CS	CS	CS	CS	CS	CS
NCS5(5)	CS	CS	CS	CS	CS	CS
NRD	OE	OE	OE	OE	OE	OE
NWR0/NWE	WE	WE (1)	WE	WE (2)	WE	WE
NWR1/NBS1	–	WE (1)	NUB	W E(2)	NUB (3)	BE1
NWR3/NBS3/DQM3	–	–	–	WE (2)	NUB (4)	BE3

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. D24–31 and A20, A23–A25, NCS2, NCS4, NCS5 are multiplexed on PD15–PD31.

Table 26-4. EBI Pins and External Device Connections

Signals: EBI_		Pins of the Interfaced Device		
		DDR2/LPDDR	SDR/LPSDR	NAND Flash
Controller	Power supply	DDRC	SDRAMC	NFC
D0–D15	VDDIOM	D0–D15	D0–D15	NFD0–NFD15[1]
D16–D31	VDDNF	–	D16–D31	NFD0–NFD15[1]
A0/NBS0	VDDIOM	–	–	–
A1/NWR2/NBS2/DQM2	VDDIOM	–	DQM2	–
DQM0–DQM1	VDDIOM	DQM0–DQM1	DQM0–DQM1	–
DQS0–DQS1	VDDIOM	DQS0–DQS1	–	–
A2–A10	VDDIOM	A[0:8]	A[0:8]	–
A11	VDDIOM	A9	A9	–
SDA10	VDDIOM	A10	A10	–
A12	VDDIOM	–	–	–
A13–A14	VDDIOM	A[11:12]	A[11:12]	–
A15	VDDIOM	A13	A13	–
A16/BA0	VDDIOM	BA0	BA0	–
A17/BA1	VDDIOM	BA1	BA1	–
A18/BA2	VDDIOM	BA2	BA2	–
A19	VDDIOM	–	–	–
A20	VDDNF	–	–	–
A21/NANDALE	VDDNF	–	–	ALE
A22/NANDCLE	VDDNF	–	–	CLE
A23–A24	VDDNF	–	–	–
A25	VDDNF	–	–	–
NCS0	VDDIOM	–	–	–
NCS1/DDRSDCS	VDDIOM	DDRCS	SDCS	–
NCS2	VDDNF	–	–	–
NCS3/NANDCS	VDDNF	–	–	CE
NCS4	VDDNF	–	–	–
NCS5	VDDNF	–	–	–
NANDOE	VDDNF	–	–	OE
NANDWE	VDDNF	–	–	WE
NRD	VDDIOM	–	–	–
NWR0/NWE	VDDIOM	–	–	–
NWR1/NBS1	VDDIOM	–	–	–
NWR3/NBS3/DQM3	VDDIOM	–	DQM3	–
SDCK	VDDIOM	CK	CK	–

Table 26-4. EBI Pins and External Device Connections (Continued)

Signals: EBI_		Pins of the Interfaced Device		
		DDR2/LPDDR	SDR/LPSDR	NAND Flash
Controller	Power supply	DDRC	SDRAMC	NFC
SDCK#	VDDIOM	CK#	–	–
SDCKE	VDDIOM	CKE	CKE	–
RAS	VDDIOM	RAS	RAS	–
CAS	VDDIOM	CAS	CAS	–
SDWE	VDDIOM	WE	WE	–
Pxx	VDDNF	–	–	CE
Pxx	VDDNF	–	–	RDY

Note: 1. The switch NFD0_ON_D16 is used to select NAND Flash path on D0–D7 or D16–D23 depending on memory power supplies. This switch is located in the CCFG_EBICSA register in the Bus Matrix.

26.5.2 Product Dependencies

26.5.2.1 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.

26.5.3 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:

- Static Memory Controller (SMC)
- DDR2/SDRAM Controller (DDR2SDRC)
- Programmable Multibit ECC Controller (PMECC)
- 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 NAND Flash support logic

26.5.3.1 Bus Multiplexing

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

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the DDR2 and SDRAM are executed independently by the DDR2SDR Controller without delaying the other external Memory Controller accesses.

26.5.3.2 Pull-up and Pull-down Control

The EBI_CSA registers in the Chip Configuration User Interface enable on-chip pull-up and pull-down resistors on data bus lines not multiplexed with the PIO Controller lines. The pull-down resistors are enabled after reset. The bits, EBIx_DBPUC and EBI_DBPDC, control the pull-up and pull-down resistors on the D0–D15 lines. Pull-up or pull-down resistors on the D16–D31 lines can be performed by programming the appropriate PIO controller.

26.5.3.3 Drive Level and Delay Control

The EBI I/Os accept two drive levels, HIGH and LOW. This allows to avoid overshoots and give the best performance according to the bus load and external memories.

The slew rates are determined by programming EBI_DRIVE bit in the EBI Chip Select Assignment Register (CCFG_EBICSA) in the Bus Matrix.

At reset the selected current drive is LOW.

To reduce EMI, programmable delay has been inserted on high-speed lines. The control of these delays is as follows:

- EBI (DDR2SDRC\SMC\NAND Flash)

D[15:0] controlled by 2 registers DELAY1 and DELAY2 located in the SMC user interface.

- D[0] <=> DELAY1[3:0],
- D[1] <=> DELAY1[7:4],...,
- D[6] <=> DELAY1[27:24],
- D[7] <=> DELAY1[31:28]
- D[8] <=> DELAY2[3:0],
- D[9] <=> DELAY2[7:4],...,
- D[14] <=> DELAY2[27:24],
- D[15] <=> DELAY2[31:28]

D[31:16] on PIOD[21:6] controlled by 2 registers, DELAY3 and DELAY4 located in the SMC user interface.

- D[16] <=> DELAY3[3:0],
- D[17] <=> DELAY3[7:4],...,
- ...
- D[24] <=> DELAY4[3:0]
- D[25] <=> DELAY4[7:4](1)
- D[26] <=> DELAY4[11:8](1)
- D[27] <=> DELAY4[15:12](1)
- D[28] <=> DELAY4[19:16](1)
- D[29] <=> DELAY4[23:20]
- D[30] <=> DELAY4[27:24]
- D[31] <=> DELAY4[31:28]

Note: 1. A20, A23, A24 and A25 are multiplexed with D25, D26, D27 and D28 in PIOD, on PD15, PD16, PD17 and PD18 lines respectively. Delays applied on these IO lines are common to A20, A23, A24, A25 and D25, D26, D27, D28 respectively.

A[25:0], controlled by 4 registers DELAY5, DELAY6, DELAY7 and DELAY8 located in the SMC user interface.

- A[0] <=> DELAY5[3:0]
- A[1] <=> DELAY5[7:4],...,
- ...
- A[14] <=> DELAY6[27:24]
- A[15] <=> DELAY6[31:28]
- A[16] <=> DELAY7[3:0]
- A[17] <=> DELAY7[7:4]
- A[18] <=> DELAY7[11:8]

and

- A19 <=> DELAY7[15:12]
- A21 <=> PD[2] <=> DELAY7[23:20]
- A22 <=> PD[3] <=> DELAY7[27:24]

26.5.3.4 Power supplies

The product embeds a dual power supply for EBI: VDDNF for NAND Flash signals and VDDIOM for others. This makes it possible to use a 1.8V or 3.3V NAND Flash independently of the SDRAM power supply.

The switch NFD0_ON_D16 is used to select the NAND Flash path on D0–D15 or D16–D31 depending on memory power supplies. This switch is located in the CCFG_EBICSA register in the Bus Matrix.

Figure 26-2 illustrates an example of the NAND Flash and the external RAM (DDR2 or LP-DDR or 16-bit LP-SDR) in the same power supply range (NFD0_ON_D16 = default).

Figure 26-2. NAND Flash and External RAM in Same Power Supply Range (NFD0_ON_D16 = default)

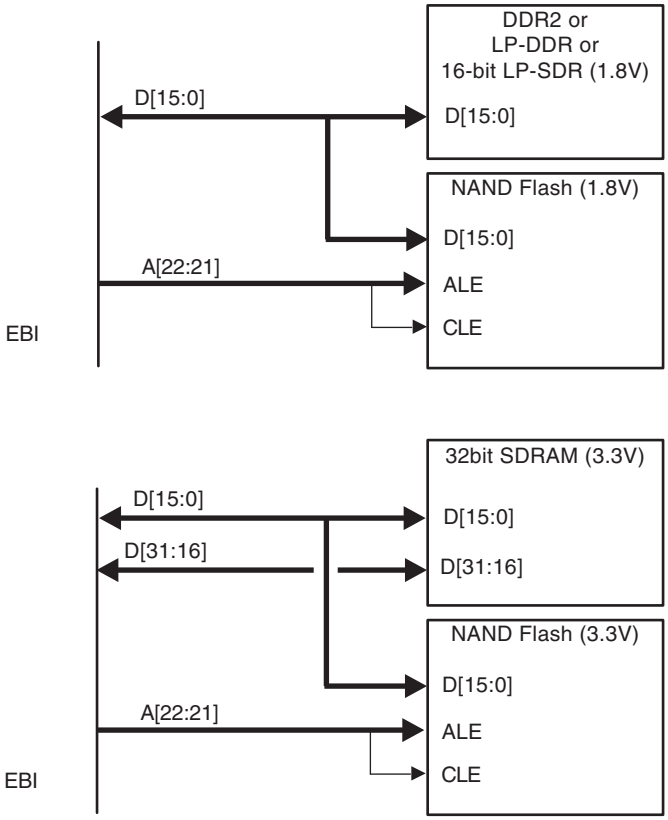
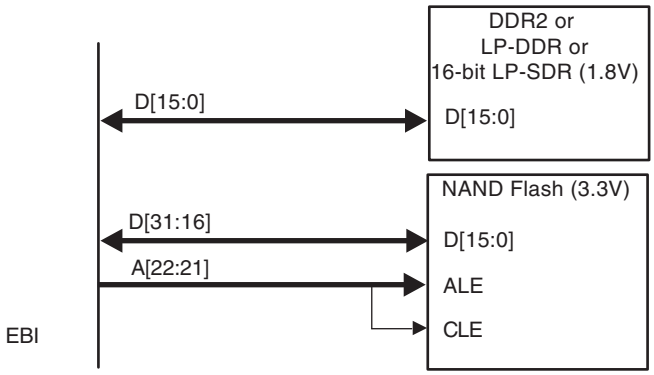


Figure 26-3 illustrates an example of the NAND Flash and the external RAM (DDR2 or LP-DDR or 16-bit LP-SDR) **not** in the same power supply range (NFD0_ON_D16 = 1).

This can be used if the SMC connects to the NAND Flash only. Using this function with another device on the SMC will lead to an unpredictable behavior of that device. In that case, the default value must be selected.

Figure 26-3. NAND Flash and External RAM Not in Same Power Supply Range (NFD0_ON_D16 = 1)



At reset NFD0_ON_D16 = 0 and the NAND Flash bus is connected to D0–D15.

26.5.3.5 Static Memory Controller

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

26.5.3.6 DDR2SDRAM Controller

The product embeds a multi-port DDR2SDR Controller. This allows to use three additional ports on DDR2SDRC to lessen the EBI load from a part of DDR2 or LP-DDR accesses. This increases the bandwidth when DDR2 and NAND Flash devices are used. This feature is NOT compatible with SDR or LP-SDR Memory.

It is controlled by DDR_MP_EN bit in EBI Chip Select Assignment Register.

Figure 26-4. DDR2SDRC Multi-port Enabled (DDR_MP_EN = 1)

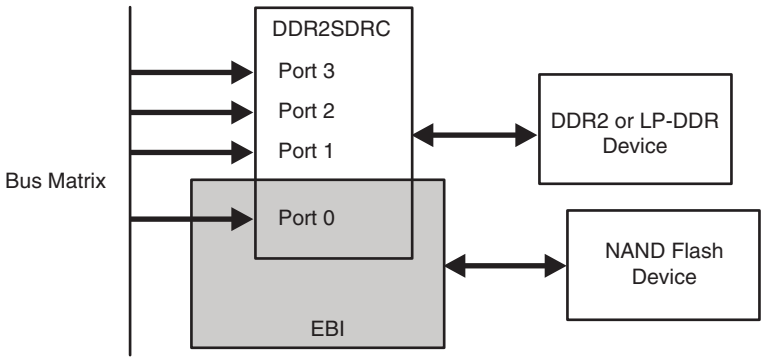
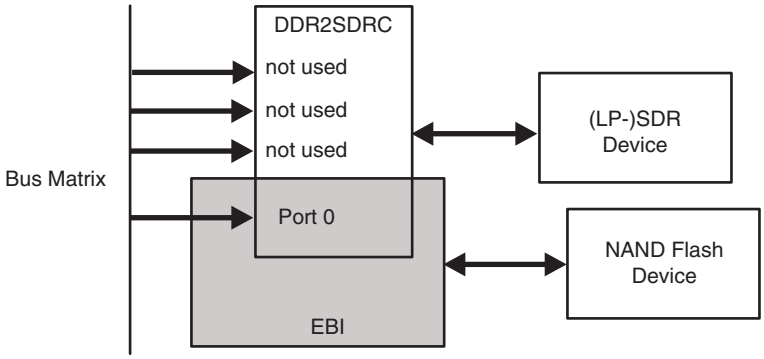


Figure 26-5. DDR2SDRC Multi-port Disabled (DDR_MP_EN = 0)



26.5.3.7 Programmable Multibit ECC Controller

For information on the PMECC Controller, refer to PMECC and PMERRLOC sections; also refer to Boot Strategies Section, NAND Flash Boot: PMECC Error Detection and Correction.

26.5.3.8 NAND Flash Support

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

External Bus Interface

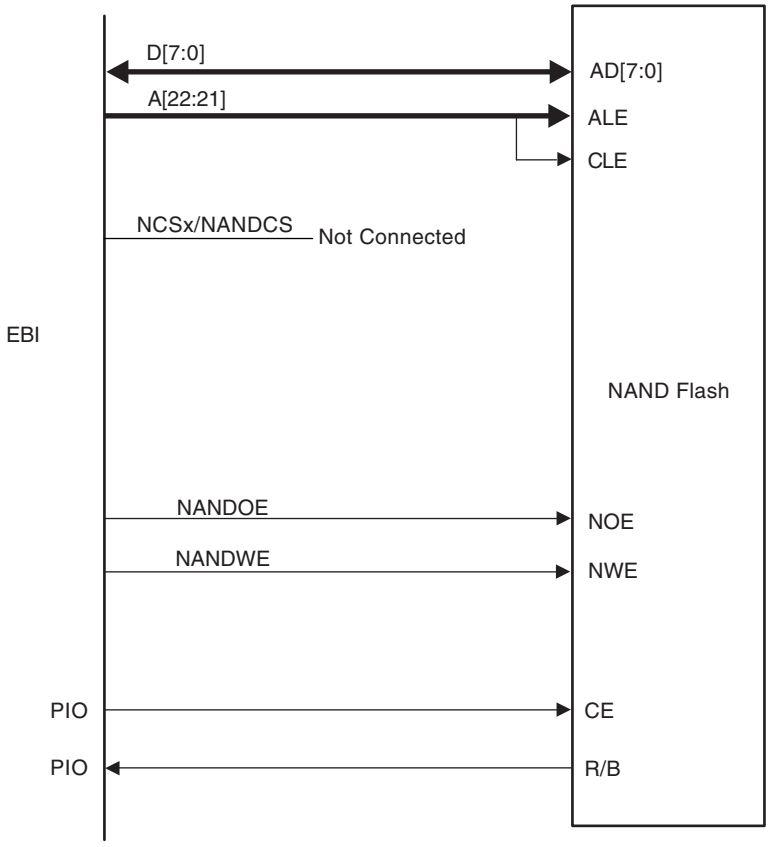
The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the EBI_CSA field in the EBI_CSA Register in the Chip Configuration User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to the Bus Matrix section.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

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. See Figure 26-6 for more information. For details on these waveforms, refer to the Static Memory Controller section.

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 26-6. NAND Flash Application Example

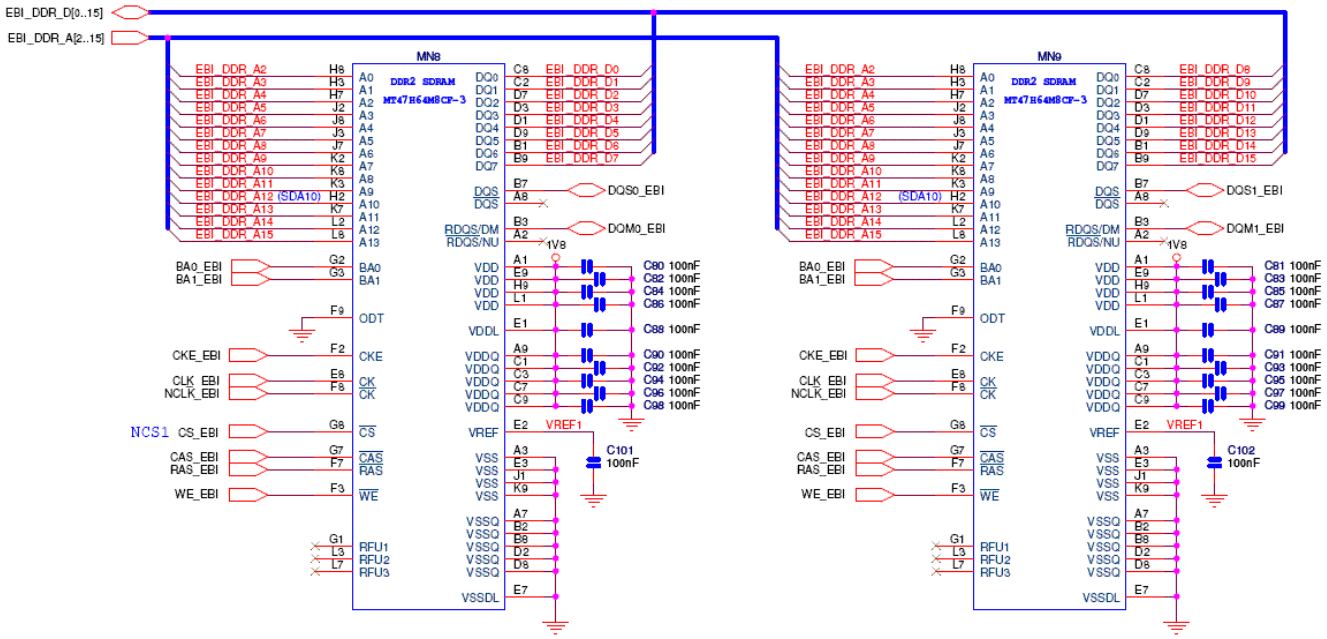


26.5.4 Implementation Examples

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

26.5.4.1 2x8-bit DDR2 on EBI

Hardware Configuration



Software Configuration

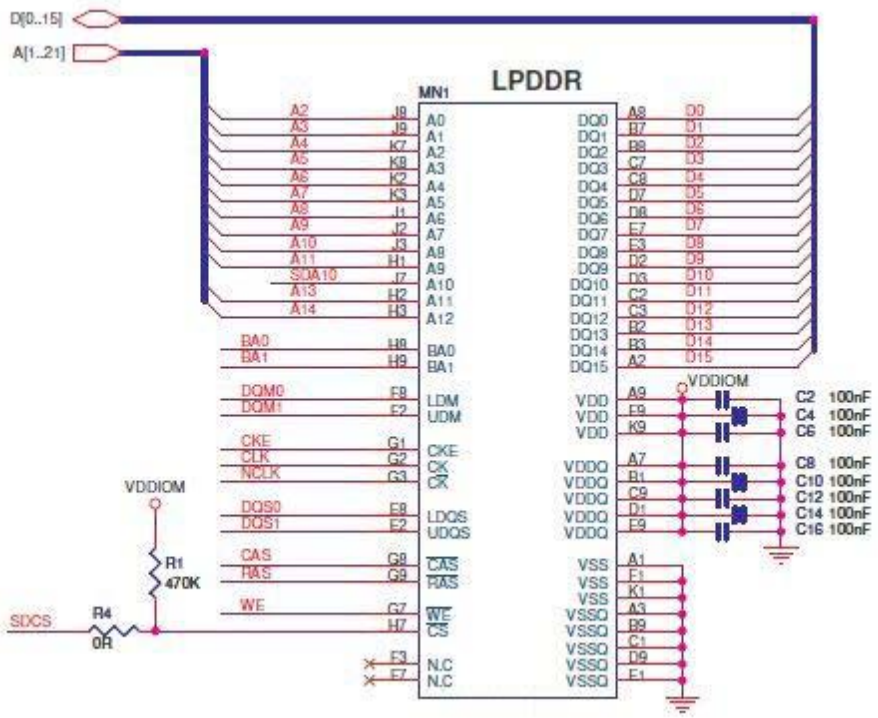
- Assign EBI_CS1 to the DDR2 controller by setting the EBI_CS1A bit in the EBI Chip Select Assignment Register (CCFG_EBICSA) in the Bus Matrix.
- Initialize the DDR2 Controller depending on the DDR2 device and system bus frequency.

The DDR2 initialization sequence is described in the subsection “DDR2 Device Initialization” of the DDRSDRC section.

In this case VDDNF can be different from VDDIOM. NAND Flash device can be 3.3V or 1.8V and wired on D16–D31 data bus. NFD0_ON_D16 is to be set to 1.

26.5.4.2 16-bit LPDDR on EBI

Hardware Configuration



Software Configuration

The following configuration has to be performed:

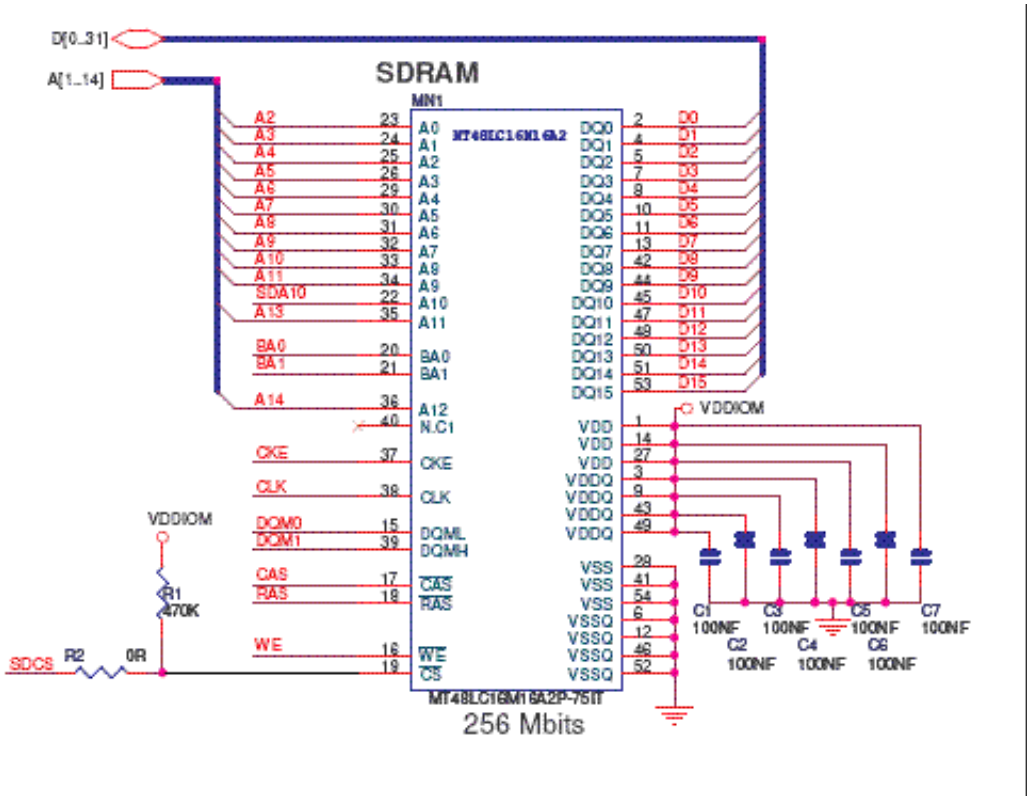
- Assign EBI_CS1 to the DDR2 controller by setting the bit EBI_CS1A bit in the EBI Chip Select Assignment Register (CCFG_EBICSA) in the Bus Matrix.
- Initialize the DDR2 Controller depending on the LP-DDR device and system bus frequency.

The LP-DDR initialization sequence is described in the section “Low-power DDR1-SDRAM Initialization” in “DDR/SDR SDRAM Controller (DDRSDRC)”.

In this case VDDNF can be different from VDDIOM. NAND Flash device can be 3.3V or 1.8V and wired on D16–D31 data bus. NFD0_ON_D16 is to be set to 1.

26.5.4.3 16-bit SDRAM on EBI

Hardware Configuration



Software Configuration

The following configuration has to be performed:

- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A bit in the EBI Chip Select Assignment Register (CCFG_EBICSA) in the Bus Matrix.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

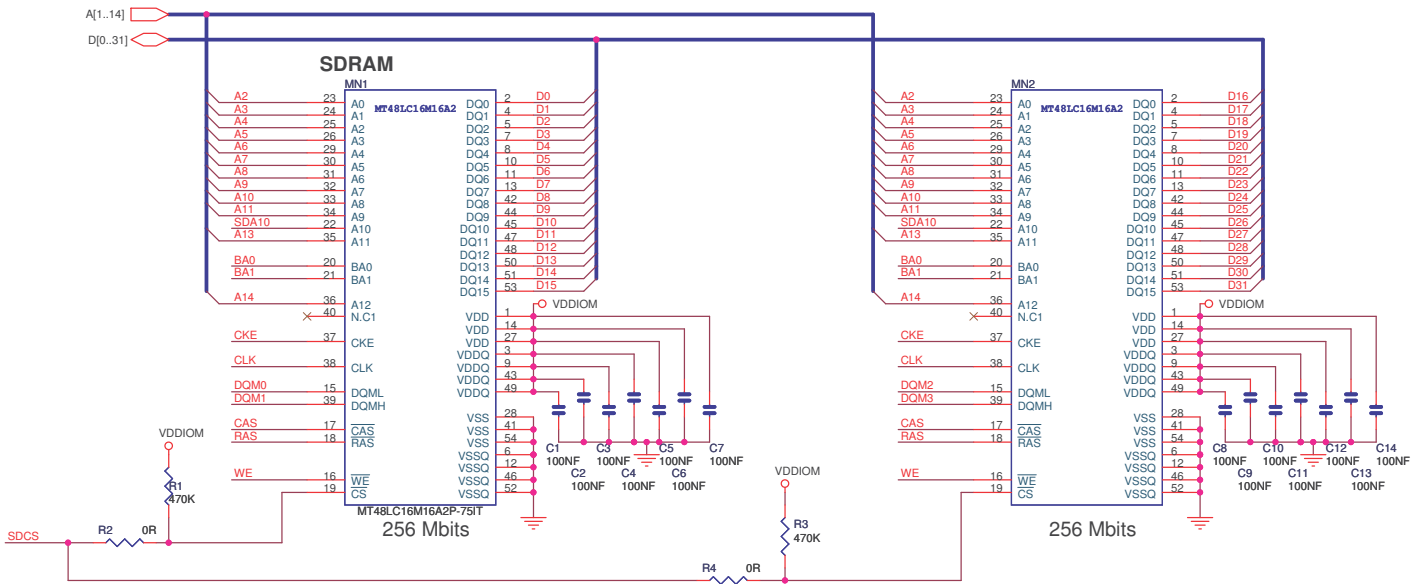
The Data Bus Width is to be programmed to 16 bits.

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

In this case VDDNF can be different from VDDIOM. NAND Flash device can be 3.3V or 1.8V and wired on D16–D31 data bus. NFD0_ON_D16 is to be set to 1.

26.5.4.4 2x16-bit SDRAM on EBI

Hardware Configuration



Software Configuration

The following configuration has to be performed:

- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A bit in the EBI Chip Select Assignment Register (CCFG_EBICSA) in the Bus Matrix.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

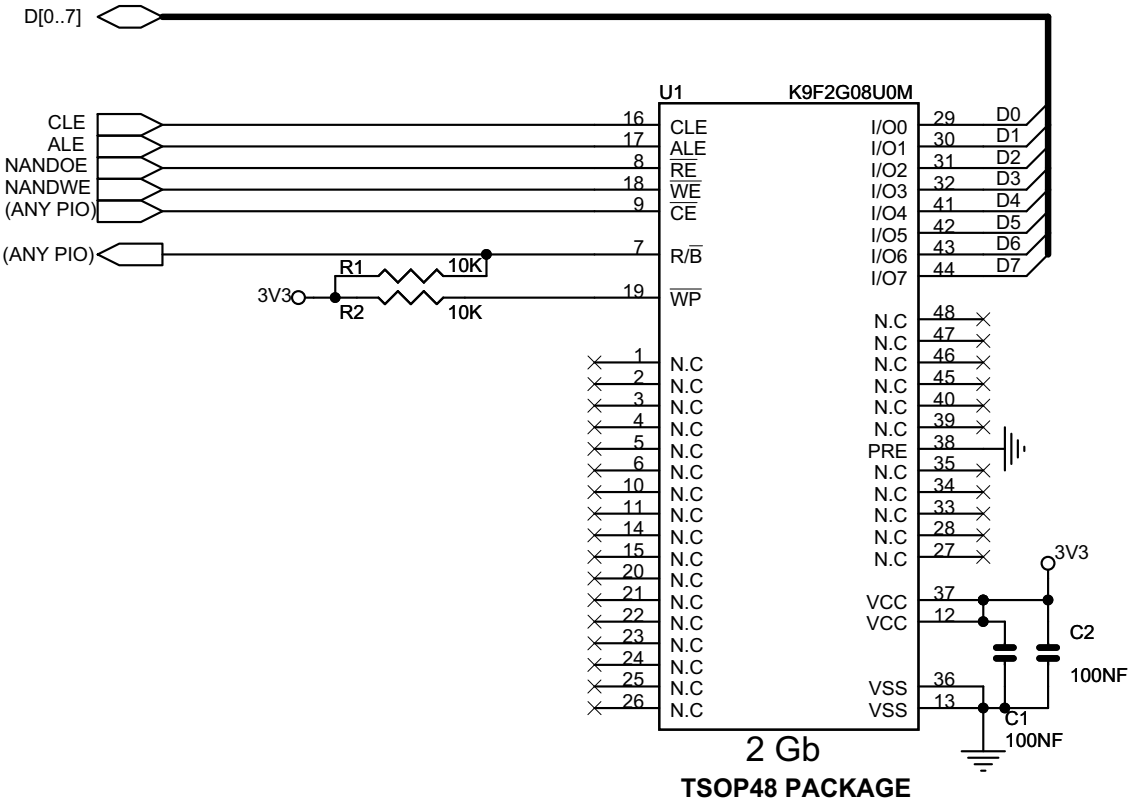
The Data Bus Width is to be programmed to 32 bits. The data lines D[16..31] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

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

In this case VDDNF must to be equal to VDDIOM. The NAND Flash device must be 3.3V and wired on D0–D15 data bus. NFD0_ON_D16 is to be set to 0.

26.5.4.5 8-bit NAND Flash with NFD0_ON_D16 = 0

Hardware Configuration



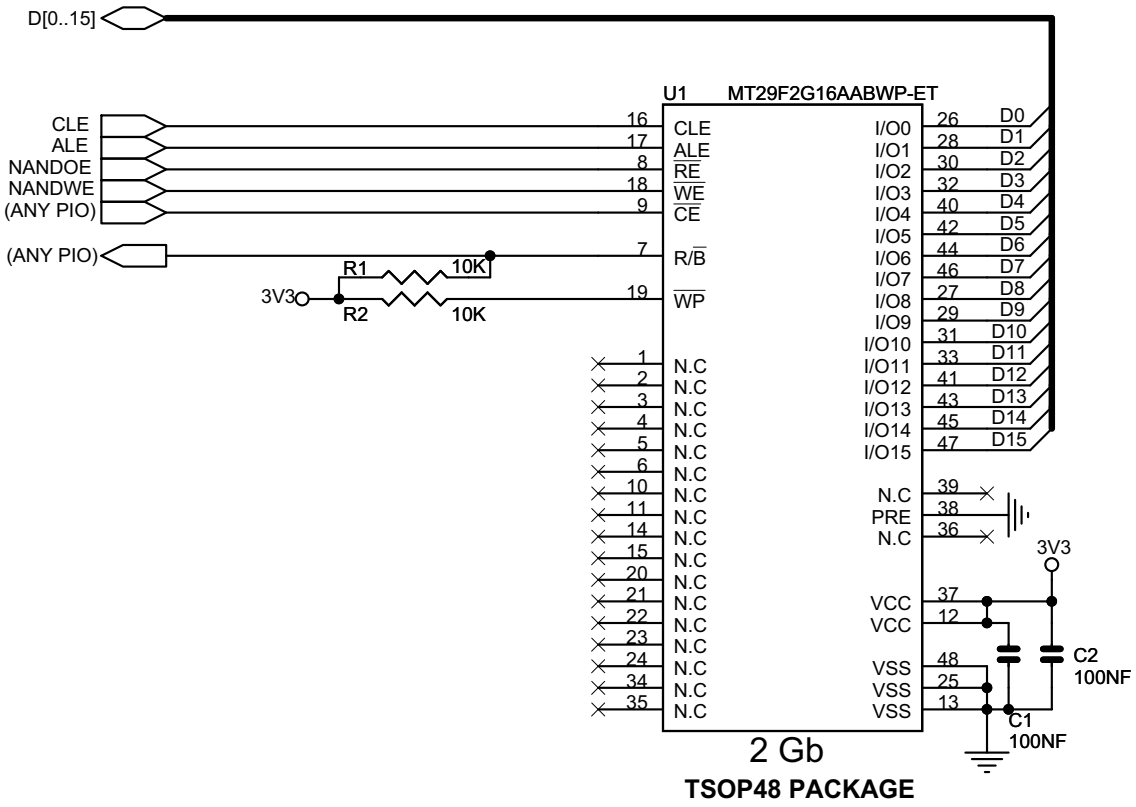
Software Configuration

The following configuration has to be performed:

- Set NFD0_ON_D16 = 0 in the EBI Chip Select Assignment Register located in the bus matrix memory space
- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register
- Reserve A21/A22 for ALE/CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bits A21 and A22 during accesses.
- 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.

26.5.4.6 16-bit NAND Flash with NFD0_ON_D16 = 0

Hardware Configuration

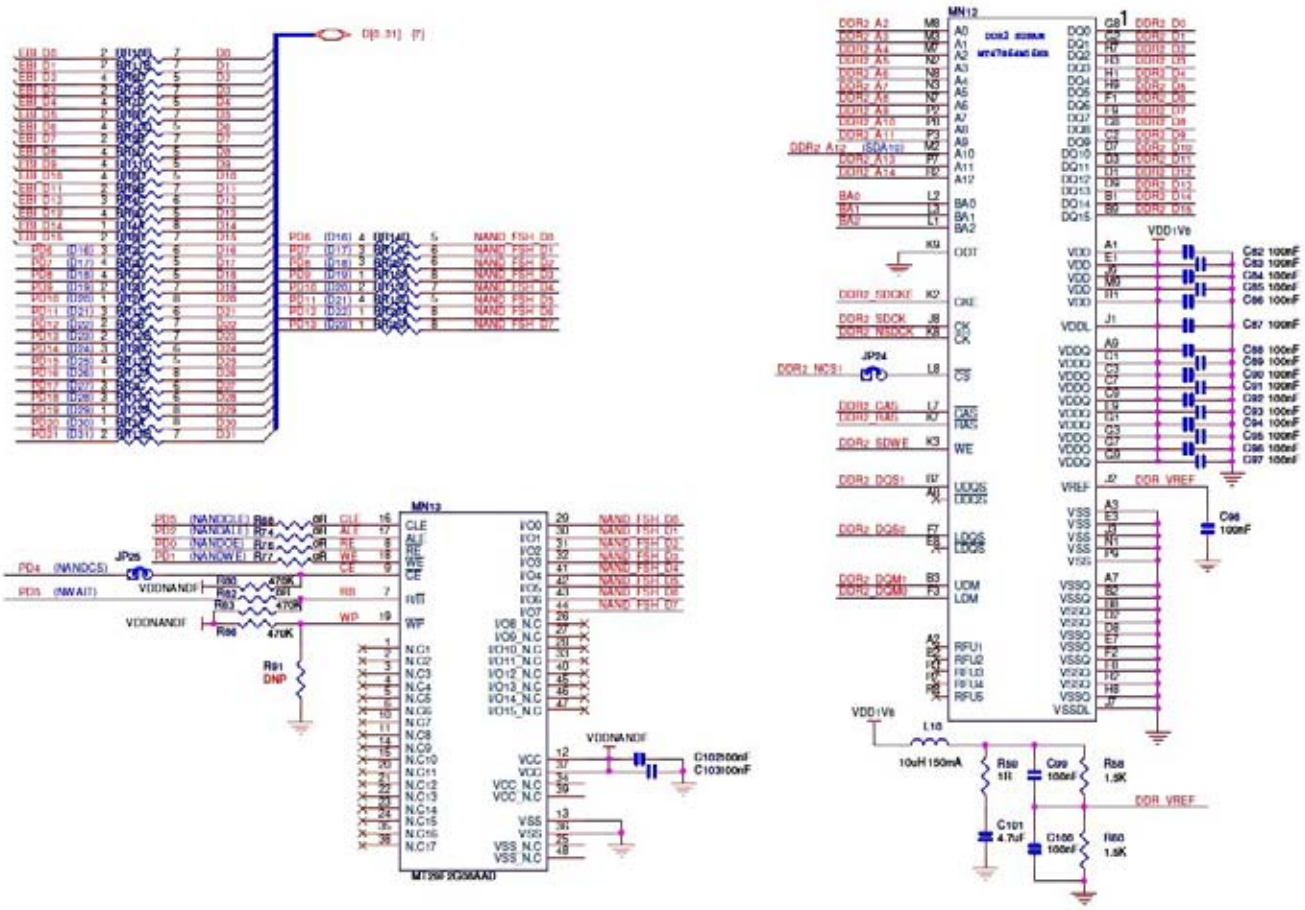


Software Configuration

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

26.5.4.7 8-bit NAND Flash with NFD0_ON_D16 = 1

Hardware Configuration



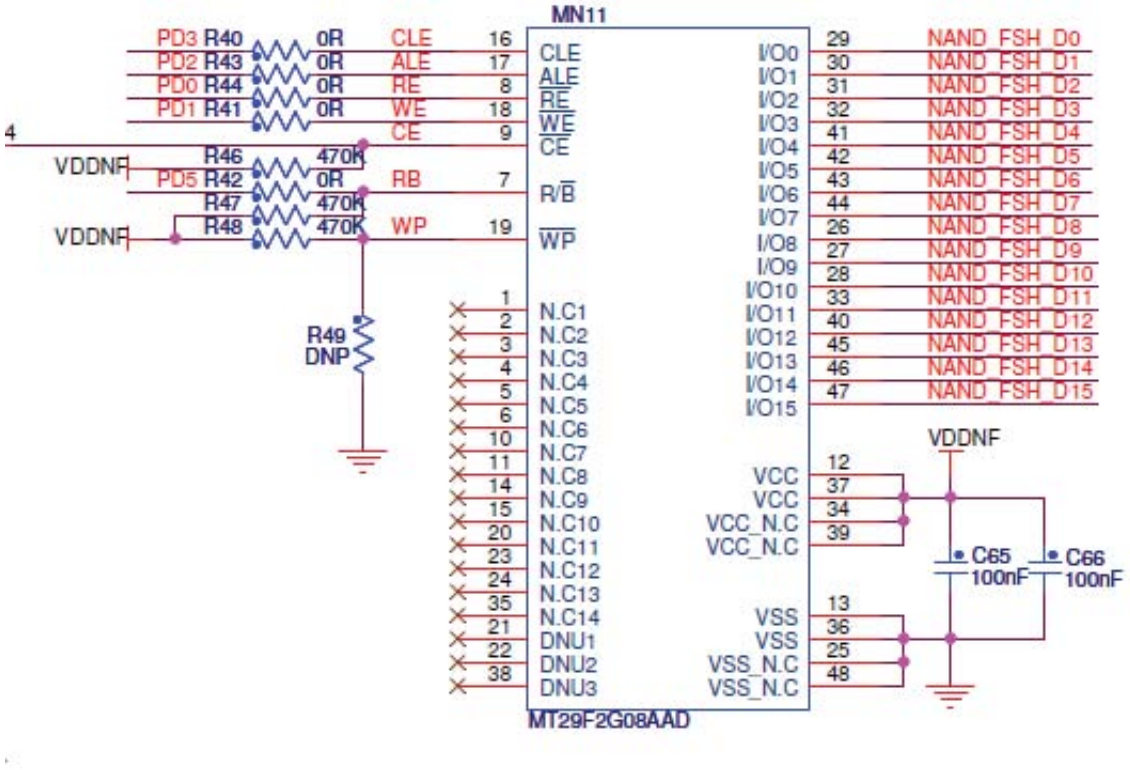
Software Configuration

The following configuration has to be performed:

- Set NFD0_ON_D16 = 1 in the EBI Chip Select Assignment Register in the Bus Matrix.
- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register
- 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.
- 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.

26.5.4.8 16-bit NAND Flash with NFD0_ON_D16 = 1

Hardware Configuration

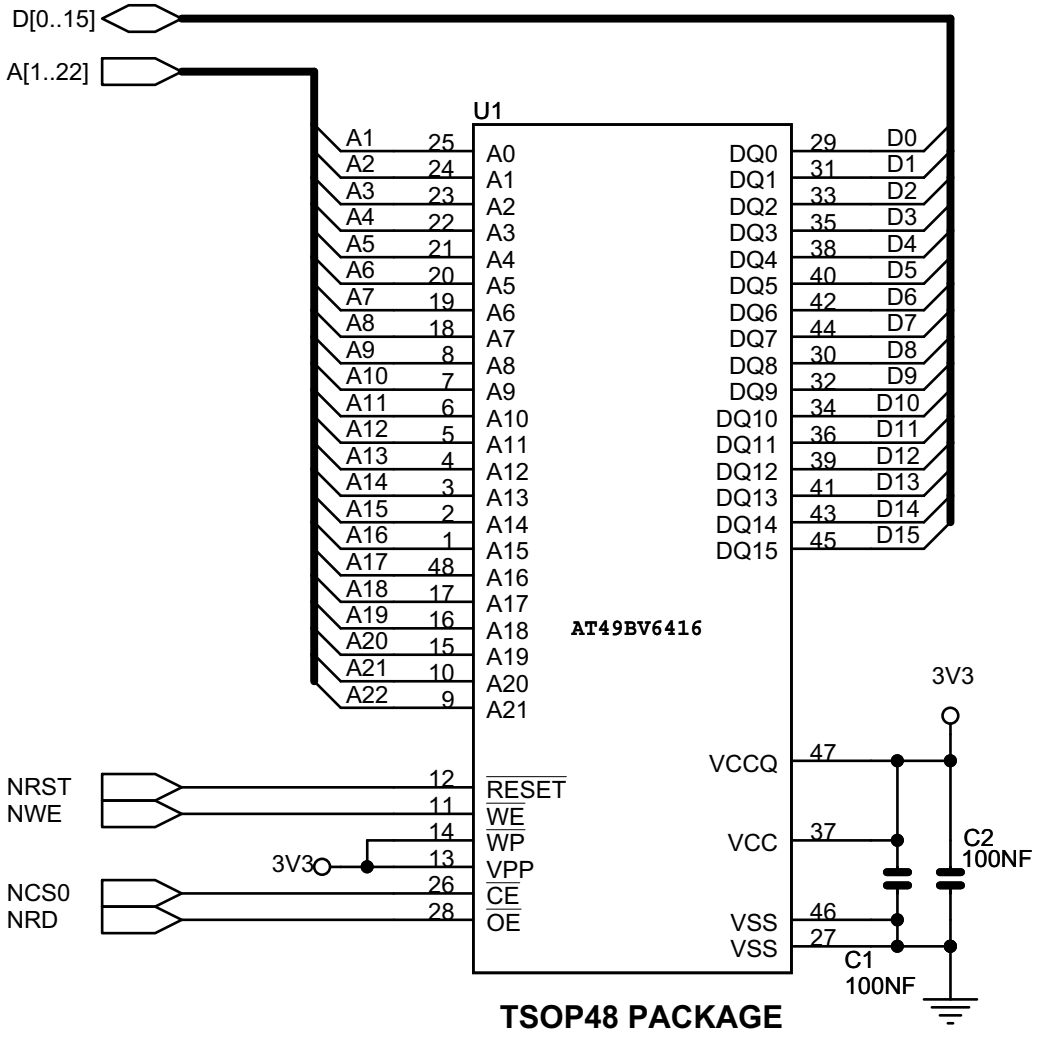


Software Configuration

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

26.5.4.9 NOR Flash on NCS0

Hardware Configuration



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.

27. Programmable Multibit ECC Controller (PMECC)

27.1 Description

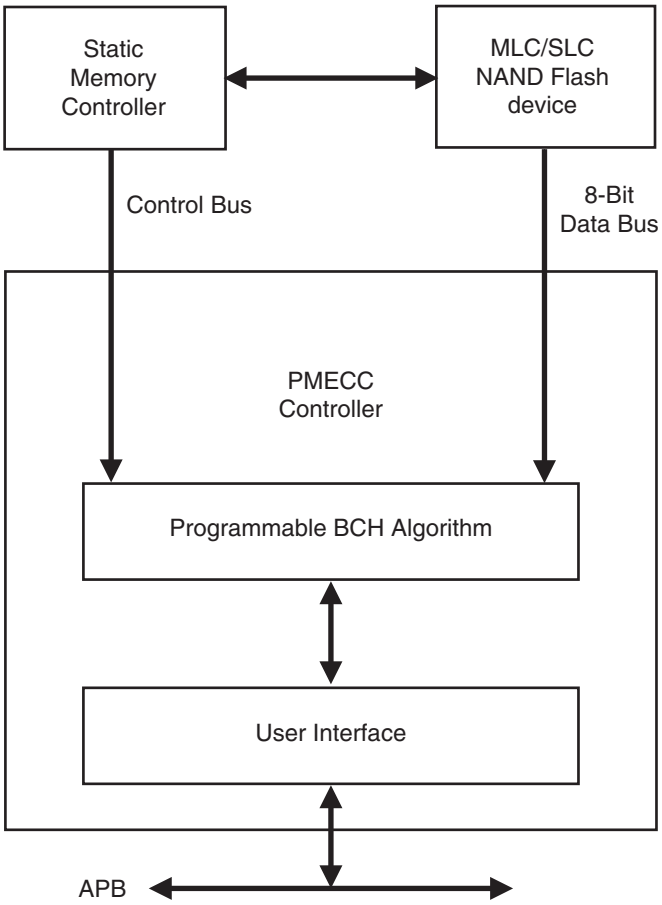
The Programmable Multibit ECC Controller (PMECC) is a programmable binary BCH (Bose, Chaudhuri and Hocquenghem) encoder/decoder. This controller can be used to generate redundancy information for both Single-Level Cell (SLC) and Multi-level Cell (MLC) NAND Flash devices. It supports redundancy for correction of 2, 4, 8, 12 or 24 bits of error per sector of data.

27.2 Embedded Characteristics

- 8-bit Nand Flash Data Bus Support
- Multibit Error Correcting Code.
- Algorithm based on binary shortened Bose, Chaudhuri and Hocquenghem (BCH) codes.
- Programmable Error Correcting Capability: 2, 4, 8, 12 and 24 bit of errors per sector.
- Programmable Sector Size: 512 bytes or 1024 bytes.
- Programmable Number of Sectors per page: 1, 2, 4 or 8 sectors of data per page.
- Programmable Spare Area Size.
- Supports Spare Area ECC Protection.
- Supports 8 Kbytes page size using 1024 bytes per sector and 4 kbytes page size using 512 bytes per sector.
- Configurable through APB interface
- Multibit Error Detection is Interrupt Driven.

27.3 Block Diagram

Figure 27-1. Block Diagram



27.4 Functional Description

The NAND Flash sector size is programmable and can be set to 512 bytes or 1024 bytes. The PMECC module generates redundancy at encoding time, when a NAND write page operation is performed. The redundancy is appended to the page and written in the spare area. This operation is performed by the processor. It moves the content of the PMECCx registers into the NAND Flash memory. The number of registers depends on the selected error correction capability, refer to Table 27-1 on page 346. This operation is executed for each sector. At decoding time, the PMECC module generates the remainder of the received codeword by minimal polynomials. When all polynomial remainders for a given sector are set to zero, no error occurred. When the polynomial remainders are other than zero, the codeword is corrupted and further processing is required.

The PMECC module generates an interrupt indicating that an error occurred. The processor must read the PMECCISR register. This register indicates which sector is corrupted.

To find the error location within a sector, the processor must execute the decoding steps as follows:

1. Syndrome computation
2. Find the error locator polynomials
3. Find the roots of the error locator polynomial

All decoding steps involve finite field computation. It means that a library of finite field arithmetic must be available to perform addition, multiplication and inversion. The finite field arithmetic operations can be performed through the use of a memory mapped lookup table, or direct software implementation. The software implementation presented is based on lookup tables. Two tables named gf_log and gf_antilog are used. If alpha is the primitive element of the field, then a power of alpha is in the field. Assume beta = alpha ^ index, then beta belongs to the field, and gf_log(beta) = gf_log(alpha ^ index) = index. The gf_antilog tables provide exponent inverse of the element, if beta = alpha ^ index, then gf_antilog(index) = beta.

The first step consists of the syndrome computation. The PMECC module computes the remainders and software must substitute the power of the primitive element.

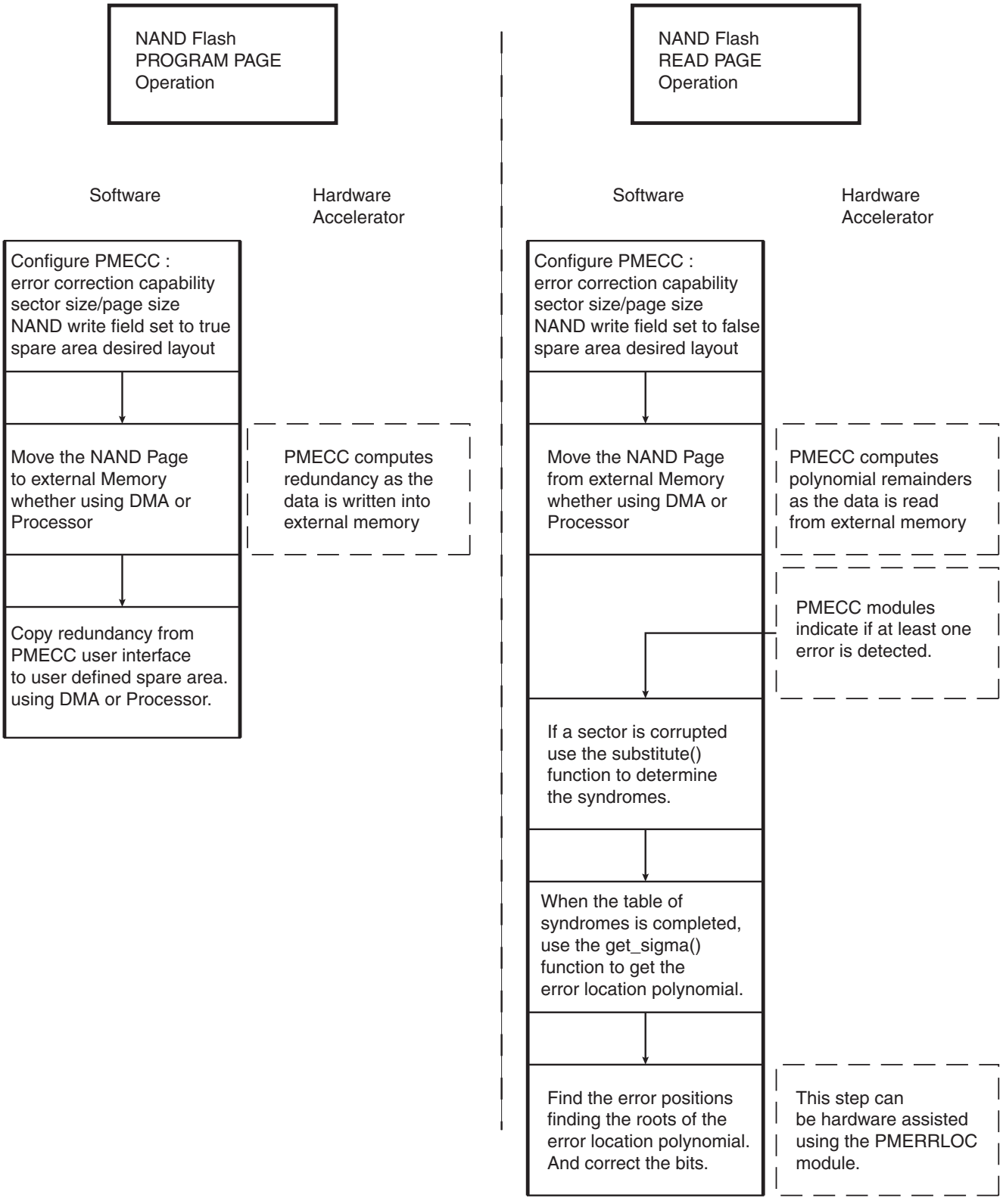
The procedure implementation is given in Section 27.5.1 “Remainder Substitution Procedure” on page 349.

The second step is the most software intensive. It is the Berlekamp’s iterative algorithm for finding the error-location polynomial.

The procedure implementation is given in Section 27.5.2 “Find the Error Location Polynomial Sigma(x)” on page 350.

The Last step is finding the root of the error location polynomial. This step can be very software intensive. Indeed, there is no straightforward method of finding the roots, except by evaluating each element of the field in the error location polynomial. However a hardware accelerator can be used to find the roots of the polynomial. The Programmable Multibit Error Correction Code Location (PMERRLOC) module provides this kind of hardware acceleration.

Figure 27-2. Software/Hardware Multibit Error Correction Dataflow



27.4.1 MLC/SLC Write Page Operation using PMECC

When an MLC write page operation is performed, the PMECC controller is configured with the NANDWR field of the PMECCFG register set to one. When the NAND spare area contains file system information and redundancy (PMECCx), the spare area is error protected, then the SPAREEN bit of the PMECCFG register is set to one. When the NAND spare area contains only redundancy information, the SPAREEN bit is set to zero.

When the write page operation is terminated, the user writes the redundancy in the NAND spare area. This operation can be done with DMA assistance.

Table 27-1. Relevant Redundancy Registers

BCH_ERR field	Sector size set to 512 bytes	Sector size set to 1024 bytes
0	PMECC_ECC0	PMECC_ECC0
1	PMECC_ECC0, PMECC_ECC1	PMECC_ECC0, PMECC_ECC1
2	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3
3	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6
4	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6, PMECC_ECC7, PMECC_ECC8, PMECC_ECC9	PMECC_ECC0, PMECC_ECC1, PMECC_ECC2, PMECC_ECC3, PMECC_ECC4, PMECC_ECC5, PMECC_ECC6, PMECC_ECC7, PMECC_ECC8, PMECC_ECC9, PMECC_ECC10

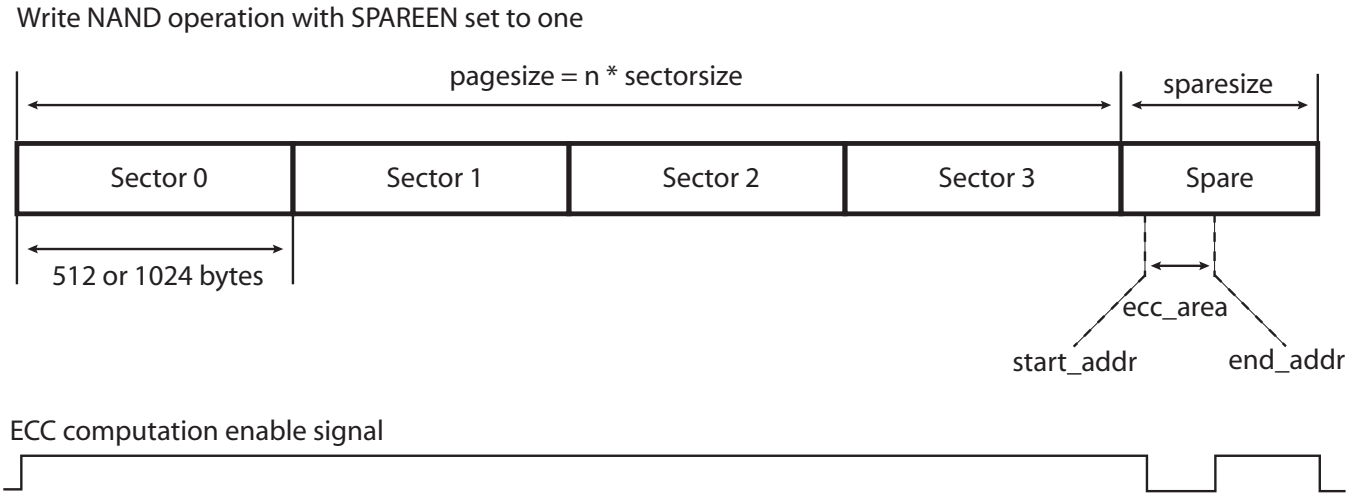
Table 27-2. Number of relevant ECC bytes per sector, copied from LSbyte to MSbyte

BCH_ERR field	Sector size set to 512 bytes	Sector size set to 1024 bytes
0	4 bytes	4 bytes
1	7 bytes	7 bytes
2	13 bytes	14 bytes
3	20 bytes	21 bytes
4	39 bytes	42 bytes

27.4.1.1 SLC/MLC Write Operation with Spare Enable Bit Set

When the SPAREEN field of the PMECC_CFG register is set to one, the spare area of the page is encoded with the stream of data of the last sector of the page. This mode is entered by writing one in the DATA field of the PMECC_CTRL register. When the encoding process is over, the redundancy is written to the spare area in user mode, USER field of the PMECC_CTRL must be set to one.

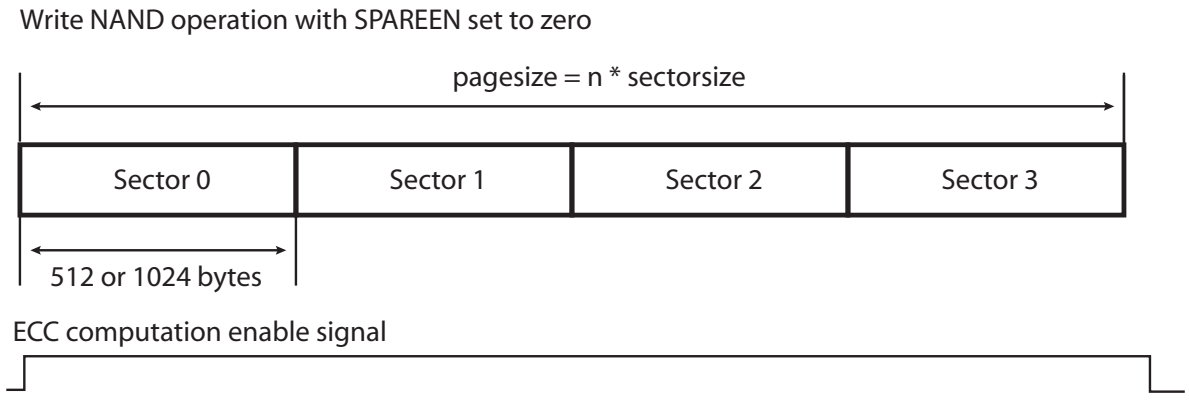
Figure 27-3. NAND Write Operation with Spare Encoding



27.4.1.2 MLC/SLC Write Operation with Spare Area Disabled

When the SPAREEN field of PMECC_CFG is set to zero the spare area is not encoded with the stream of data. This mode is entered by writing one to the DATA field of the PMECC_CTRL register.

Figure 27-4. NAND Write Operation



27.4.2 MLC/SLC Read Page Operation using PMECC

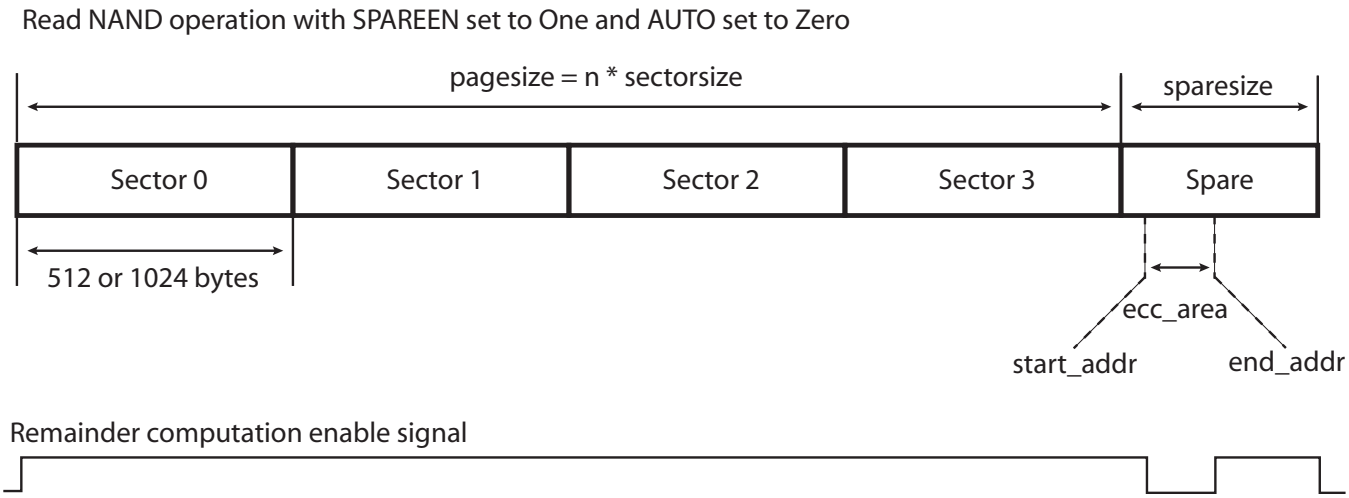
Table 27-3. Relevant Remainders Registers

BCH_ERR field	Sector size set to 512 bytes	Sector size set to 1024 bytes
0	PMECC_REM0	PMECC_REM0
1	PMECC_REM0, PMECC_REM1	PMECC_REM0, PMECC_REM1
2	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3,	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3
3	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7
4	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7, PMECC_REM8, PMECC_REM9, PMECC_REM10, PMECC_REM11	PMECC_REM0, PMECC_REM1, PMECC_REM2, PMECC_REM3, PMECC_REM4, PMECC_REM5, PMECC_REM6, PMECC_REM7, PMECC_REM8, PMECC_REM9, PMECC_REM10, PMECC_REM11

27.4.2.1 MLC/SLC Read Operation with Spare Decoding

When the spare area is protected, the spare area contains valid data. As the redundancy may be included in the middle of the information stream, the user programs the start address and the end address of the ECC area. The controller will automatically skip the ECC area. This mode is entered by writing one in the DATA field of the PMECC_CTRL register. When the page has been fully retrieved from NAND, the ECC area is read using the user mode by writing one to the USER field of the PMECC_CTRL register.

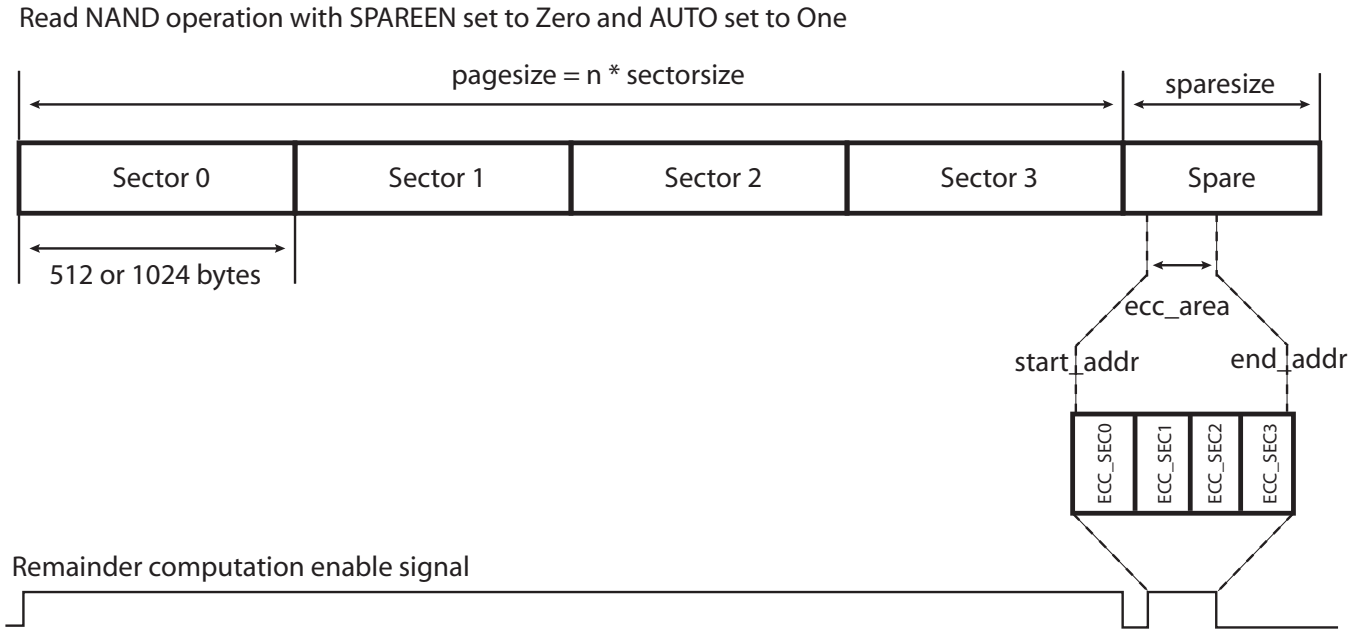
Figure 27-5. Read Operation with Spare Decoding



27.4.2.2 MLC/SLC Read Operation

If the spare area is not protected with the error correcting code, the redundancy area is retrieved directly. This mode is entered by writing one in the DATA field of the PMECC_CTRL register. When AUTO field is set to one the ECC is retrieved automatically, otherwise the ECC must be read using user mode.

Figure 27-6. Read Operation

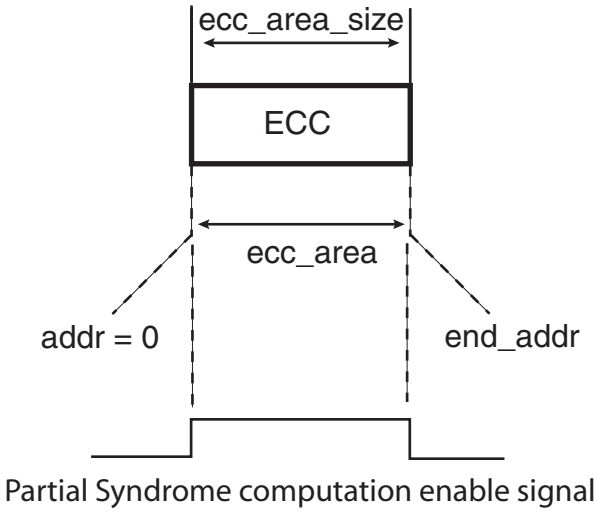


27.4.2.3 MLC/SLC User Read ECC Area

This mode allows a manual retrieve of the ECC.

This mode is entered writing one in the USER field of the PMECC_CTRL register.

Figure 27-7. User Read Mode



27.5 Software Implementation

27.5.1 Remainder Substitution Procedure

The substitute function evaluates the polynomial remainder, with different values of the field primitive elements. The finite field arithmetic addition operation is performed with the Exclusive or. The finite field arithmetic multiplication operation is performed through the gf_log, gf_antilog lookup tables.

The REM2NP1 and REMN2NP3 fields of the PMECC_REMx registers contain only odd remainders. Each bit indicates whether the coefficient of the polynomial remainder is set to zero or not.

NB_ERROR_MAX defines the maximum value of the error correcting capability.

NB_ERROR defines the error correcting capability selected at encoding/decoding time.

NB_FIELD_ELEMENTS defines the number of elements in the field.

si[] is a table that holds the current syndrome value, an element of that table belongs to the field. This is also a shared variable for the next step of the decoding operation.

oo[] is a table that contains the degree of the remainders.

```
int substitute()
{
int i;
int j;
for (i = 1; i < 2 * NB_ERROR_MAX; i++)
{
      si[i] = 0;
}
for (i = 1; i < 2*NB_ERROR; i++)
{
      for (j = 0; j < oo[i]; j++)
      {
            if (REM2NPX[i][j])
            {
                  si[i] = gf_antilog[(i * j)%NB_FIELD_ELEMENTS] ^ si[i];
            }
      }
}
return 0;
}
```

27.5.2 Find the Error Location Polynomial Sigma(x)

The sample code below gives a Berlekamp iterative procedure for finding the value of the error location polynomial.

The input of the procedure is the si[] table defined in the remainder substitution procedure.

The output of the procedure is the error location polynomial named smu (sigma mu). The polynomial coefficients belong to the field. The smu[NB_ERROR+1][] is a table that contains all these coefficients.

NB_ERROR_MAX defines the maximum value of the error correcting capability.

NB_ERROR defines the error correcting capability selected at encoding/decoding time.

NB_FIELD_ELEMENTS defines the number of elements in the field.

```
int get_sigma()
{
int i;
int j;
int k;
/* mu          */
int mu[NB_ERROR_MAX+2];
/* sigma ro   */
int sro[2*NB_ERROR_MAX+1];
/* discrepancy */
int dmu[NB_ERROR_MAX+2];
/* delta order   */
int delta[NB_ERROR_MAX+2];
/* index of largest delta */
int ro;
int largest;
```

```
int diff;
/*                    */
/*     First Row      */
/*                    */
/* Mu */
mu[0]  = -1; /* Actually -1/2 */
/* Sigma(x) set to 1 */
for (i = 0; i < (2*NB_ERROR_MAX+1); i++)
       smu[0][i] = 0;
smu[0][0] = 1;
/* discrepancy set to 1 */
dmu[0] = 1;
/* polynom order set to 0 */
lmu[0] = 0;
/* delta set to -1 */
delta[0]  = (mu[0] * 2 - lmu[0]) >> 1;
/*                     */
/*     Second Row      */
/*                     */
/* Mu */
mu[1]  = 0;
/* Sigma(x) set to 1 */
for (i = 0; i < (2*NB_ERROR_MAX+1); i++)
       smu[1][i] = 0;
smu[1][0] = 1;
/* discrepancy set to Syndrome 1 */
dmu[1] = si[1];
/* polynom order set to 0 */
lmu[1] = 0;
/* delta set to 0 */
delta[1]  = (mu[1] * 2 - lmu[1]) >> 1;
for (i=1; i <= NB_ERROR; i++)
{
       mu[i+1] = i << 1;
       /*************************************************/
       /*                                               */
       /*                                               */
       /*          Compute Sigma (Mu+1)                 */
       /*          And L(mu)                            */
       /* check if discrepancy is set to 0 */
       if (dmu[i] == 0)
       {
       /* copy polynom */
       for (j=0; j<2*NB_ERROR_MAX+1; j++)
       {
              smu[i+1][j] = smu[i][j];
       }
       /* copy previous polynom order to the next */
       lmu[i+1] = lmu[i];
       }
       else
       {
       ro      = 0;
       largest = -1;
       /* find largest delta with dmu != 0 */
```

```
        for (j=0; j<i; j++)
              {
              if (dmu[j])
                    {
                    if (delta[j] > largest)
                          {
                          largest = delta[j];
                          ro    = j;
                          }
                    }
              }
        /* initialize signal ro */
        for (k = 0; k < 2*NB_ERROR_MAX+1; k ++)
        {
              sro[k] = 0;
        }
        /* compute difference */
        diff = (mu[i] - mu[ro]);
        /* compute X ^ (2(mu-ro)) */
        for (k = 0; k < (2*NB_ERROR_MAX+1); k ++)
        {
              sro[k+diff] = smu[ro][k];
        }
        /* multiply by dmu * dmu[ro]^-1 */
        for (k = 0; k < 2*NB_ERROR_MAX+1; k ++)
        {
              /* dmu[ro] is not equal to zero by definition */
              /* check that operand are different from 0    */
              if (sro[k] && dmu[i])
              {
                    /* galois inverse */
                    sro[k] = gf_antilog[(gf_log[dmu[i]] + (NB_FIELD_ELEMENTS-
gf_log[dmu[ro]]) + gf_log[sro[k]]) % NB_FIELD_ELEMENTS];
              }
        }
        /* multiply by dmu * dmu[ro]^-1 */
        for (k = 0; k < 2*NB_ERROR_MAX+1; k++)
        {
              smu[i+1][k] = smu[i][k] ^ sro[k];
              if (smu[i+1][k])
              {
                    /* find the order of the polynom */
                    lmu[i+1] = k << 1;
              }
        }
}
/*                                                  */
/*                                                  */
/*      End Compute Sigma (Mu+1)                    */
/*      And L(mu)                                   */
/**************************************************/
/* In either case compute delta */
delta[i+1]  = (mu[i+1] * 2 - lmu[i+1]) >> 1;
/* In either case compute the discrepancy */
for (k = 0 ; k <= (lmu[i+1]>>1); k++)
```

```
{
        if (k == 0)
        dmu[i+1] = si[2*(i-1)+3];
        /* check if one operand of the multiplier is null, its index is -1 */
        else if (smu[i+1][k] && si[2*(i-1)+3-k])
        dmu[i+1] = gf_antilog[(gf_log[smu[i+1][k]] + gf_log[si[2*(i-1)+3-k]])%nn]
^ dmu[i+1];
}
}
return 0;
}
```

27.5.3 Find the Error Position

The output of the get_sigma() procedure is a polynomial stored in the smu[NB_ERROR+1][] table. The error position is the roots of that polynomial. The degree of this polynomial is very important information, as it gives the number of errors. The PMERRLOC module provides a hardware accelerator for this step.

27.6 Programmable Multibit ECC Controller (PMECC) User Interface

Table 27-4. Register Mapping

Offset	Register	Name	Access	Reset
0x00000000	PMECC Configuration Register	PMECC_CFG	Read-write	0x00000000
0x00000004	PMECC Spare Area Size Register	PMECC_SAREA	Read-write	0x00000000
0x00000008	PMECC Start Address Register	PMECC_SADDR	Read-write	0x00000000
0x0000000C	PMECC End Address Register	PMECC_EADDR	Read-write	0x00000000
0x00000010	PMECC Clock Control Register	PMECC_CLK	Read-write	0x00000000
0x00000014	PMECC Control Register	PMECC_CTRL	Write-only	0x00000000
0x00000018	PMECC Status Register	PMECC_SR	Read-only	0x00000000
0x0000001C	PMECC Interrupt Enable register	PMECC_IER	Write-only	0x00000000
0x00000020	PMECC Interrupt Disable Register	PMECC_IDR	Write-only	–
0x00000024	PMECC Interrupt Mask Register	PMECC_IMR	Read-only	0x00000000
0x00000028	PMECC Interrupt Status Register	PMECC_ISR	Read-only	0x00000000
0x0000002C	Reserved	–	–	–
0x040+sec_num*(0x40)+0x00	PMECC ECC 0 Register	PMECC_ECC0	Read-only	0x00000000
0x040+sec_num*(0x40)+0x04	PMECC ECC 1 Register	PMECC_ECC1	Read-only	0x00000000
0x040+sec_num*(0x40)+0x08	PMECC ECC 2 Register	PMECC_ECC2	Read-only	0x00000000
0x040+sec_num*(0x40)+0x0C	PMECC ECC 3 Register	PMECC_ECC3	Read-only	0x00000000
0x040+sec_num*(0x40)+0x10	PMECC ECC 4 Register	PMECC_ECC4	Read-only	0x00000000
0x040+sec_num*(0x40)+0x14	PMECC ECC 5 Register	PMECC_ECC5	Read-only	0x00000000
0x040+sec_num*(0x40)+0x18	PMECC ECC 6 Register	PMECC_ECC6	Read-only	0x00000000
0x040+sec_num*(0x40)+0x1C	PMECC ECC 7 Register	PMECC_ECC7	Read-only	0x00000000
0x040+sec_num*(0x40)+0x20	PMECC ECC 8 Register	PMECC_ECC8	Read-only	0x00000000
0x040+sec_num*(0x40)+0x24	PMECC ECC 9 Register	PMECC_ECC9	Read-only	0x00000000
0x040+sec_num*(0x40)+0x28	PMECC ECC 10 Register	PMECC_ECC10	Read-only	0x00000000
0x240+sec_num*(0x40)+0x00	PMECC REM 0 Register	PMECC_REM0	Read-only	0x00000000
0x240+sec_num*(0x40)+0x04	PMECC REM 1 Register	PMECC_REM1	Read-only	0x00000000
0x240+sec_num*(0x40)+0x08	PMECC REM 2 Register	PMECC_REM2	Read-only	0x00000000
0x240+sec_num*(0x40)+0x0C	PMECC REM 3 Register	PMECC_REM3	Read-only	0x00000000
0x240+sec_num*(0x40)+0x10	PMECC REM 4 Register	PMECC_REM4	Read-only	0x00000000
0x240+sec_num*(0x40)+0x14	PMECC REM 5 Register	PMECC_REM5	Read-only	0x00000000
0x240+sec_num*(0x40)+0x18	PMECC REM 6 Register	PMECC_REM6	Read-only	0x00000000
0x240+sec_num*(0x40)+0x1C	PMECC REM 7 Register	PMECC_REM7	Read-only	0x00000000
0x240+sec_num*(0x40)+0x20	PMECC REM 8 Register	PMECC_REM8	Read-only	0x00000000
0x240+sec_num*(0x40)+0x24	PMECC REM 9 Register	PMECC_REM9	Read-only	0x00000000

Table 27-4. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x240+sec_num*(0x40)+0x28	PMECC REM 10 Register	PMECC_REM10	Read-only	0x00000000
0x240+sec_num*(0x40)+0x2C	PMECC REM 11 Register	PMECC_REM11	Read-only	0x00000000
0x440 - 0x5FC	Reserved	–	–	–

27.6.1 PMECC Configuration Register

Name: PMECC_CFG

Address: 0xFFFFE000

Access: Read-write

Reset: 0x00000000

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

- **BCH_ERR: Error Correct Capability**

Value	Name	Description
0	BCH_ERR2	2 errors
1	BCH_ERR4	4 errors
2	BCH_ERR8	8 errors
3	BCH_ERR12	12 errors
4	BCH_ERR24	24 errors

- **SECTORSZ: Sector Size**

0: The ECC computation is based on a sector of 512 bytes.

1: The ECC computation is based on a sector of 1024 bytes.

- **PAGESIZE: Number of Sectors in the Page**

Value	Name	Description
0	PAGESIZE_1SEC	1 sector for main area (512 or 1024 bytes)
1	PAGESIZE_2SEC	2 sectors for main area (1024 or 2048 bytes)
2	PAGESIZE_4SEC	4 sectors for main area (2048 or 4096 bytes)
3	PAGESIZE_8SEC	8 errors for main area (4096 or 8192 bytes)

- **NANDWR: NAND Write Access**

:0: NAND read access

1: NAND write access

- **SPAREEN: Spare Enable**

– for NAND write access:

0: The spare area is skipped

1: The spare area is protected with the last sector of data.

– for NAND read access:

0: The spare area is skipped.

1: The spare area contains protected data or only redundancy information.

- **AUTO: Automatic Mode Enable**

This bit is only relevant in NAND Read Mode, when spare enable is activated.

0: Indicates that the spare area is not protected. In that case the ECC computation takes into account the ECC area located in the spare area. (within the start address and the end address).

1: Indicates that the spare is error protected. In this case, the ECC computation takes into account the whole spare area minus the ECC area in the ECC computation operation.

27.6.2 PMECC Spare Area Size Register

Name: PMECC_SAREA

Address: 0xFFFFE004

Access: Read-write

Reset: 0x00000000

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

- **SPARESIZE: Spare Area Size**

The spare area size is equal to (SPARESIZE+1) bytes.

27.6.3 PMECC Start Address Register

Name: PMECC_SADDR

Address: 0xFFFFE008

Access: Read-write

Reset: 0x00000000

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

- **STARTADDR: ECC Area Start Address (byte oriented address)**

This field indicates the first byte address of the ECC area. Location 0 matches the first byte of the spare area.

27.6.4 PMECC End Address Register

Name: PMECC_EADDR

Address: 0xFFFFE00C

Access: Read-write

Reset: 0x00000000

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

- **ENDADDR: ECC Area End Address (byte oriented address)**

This field indicates the last byte address of the ECC area.

27.6.5 PMECC Clock Control Register

Name: PMECC_CLK

Address: 0xFFFFE010

Access: Read-write

Reset: 0x00000000

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

- **CLKCTRL: Clock Control Register**

The PMECC Module data path Setup Time is set to CLKCTRL+1.

This field indicates the database setup times in number of clock cycles. At 133 MHz, this field must be programmed with 2, indicating that the setup time is 3 clock cycles.

27.6.6 PMECC Control Register

Name: PMECC_CTRL

Address: 0xFFFFE014

Access: Write-only

Reset: 0x00000000

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

- **RST: Reset the PMECC Module**

When set to one, this bit reset PMECC controller, configuration registers remain unaffected.

- **DATA: Start a Data Phase**

- **USER: Start a User Mode Phase**

- **ENABLE: PMECC Module Enable**

PMECC module must always be configured before being activated.

- **DISABLE: PMECC Module Disable**

PMECC module must always be configured after being deactivated.

27.6.7 PMECC Status Register

Name: PMECC_SR

Address: 0xFFFFE018

Access: Read-only

Reset: 0x00000000

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

- **BUSY: The Kernel of the PMECC is Busy**

- **ENABLE: PMECC Module Status**

0: The PMECC Module is disabled and can be configured.

1: The PMECC Module is enabled and the configuration registers cannot be written.

27.6.8 PMECC Interrupt Enable Register

Name: PMECC_IER

Address: 0xFFFFE01C

Access: Write-only

Reset: 0x00000000

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

- **ERRIE: Error Interrupt Enable**

27.6.9 PMECC Interrupt Disable Register

Name: PMECC_IDR

Address: 0xFFFFE020

Access: Write

Reset: 0x00000000

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

- **ERRID: Error Interrupt Disable**

27.6.10 PMECC Interrupt Mask Register

Name: PMECC_IMR

Address: 0xFFFFE024

Access: Read-only

Reset: 0x00000000

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

- **ERRIM: Error Interrupt Enable**

27.6.11 PMECC Interrupt Status Register

Name: PMECC_ISR

Address: 0xFFFFE028

Access: Read-only

Reset: 0x00000000

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

- **ERRIS: Error Interrupt Status Register**

When set to one, bit *i* of the PMECCISR register indicates that sector *i* is corrupted.

27.6.12 PMECC ECC x Register

Name: PMECC_ECCx [x=0..10] [sec_num=0..7]

Address: 0xFFFFE040 [0][0] .. 0xFFFFE068 [10][0]
0xFFFFE080 [0][1] .. 0xFFFFE0A8 [10][1]
0xFFFFE0C0 [0][2] .. 0xFFFFE0E8 [10][2]
0xFFFFE100 [0][3] .. 0xFFFFE128 [10][3]
0xFFFFE140 [0][4] .. 0xFFFFE168 [10][4]
0xFFFFE180 [0][5] .. 0xFFFFE1A8 [10][5]
0xFFFFE1C0 [0][6] .. 0xFFFFE1E8 [10][6]
0xFFFFE200 [0][7] .. 0xFFFFE228 [10][7]

Access: Read-only

Reset: 0x00000000

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

- **ECC: BCH Redundancy**

This register contains the remainder of the division of the codeword by the generator polynomial.

27.6.13 PMECC Remainder x Register

Name: PMECC_REMx [x=0..11] [sec_num=0..7]

Address: 0xFFFFE240 [0][0] .. 0xFFFFE26C [11][0]
0xFFFFE280 [0][1] .. 0xFFFFE2AC [11][1]
0xFFFFE2C0 [0][2] .. 0xFFFFE2EC [11][2]
0xFFFFE300 [0][3] .. 0xFFFFE32C [11][3]
0xFFFFE340 [0][4] .. 0xFFFFE36C [11][4]
0xFFFFE380 [0][5] .. 0xFFFFE3AC [11][5]
0xFFFFE3C0 [0][6] .. 0xFFFFE3EC [11][6]
0xFFFFE400 [0][7] .. 0xFFFFE42C [11][7]

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	REM2NP3					
23	22	21	20	19	18	17	16
REM2NP3							
15	14	13	12	11	10	9	8
–	–	REM2NP1					
7	6	5	4	3	2	1	0
REM2NP1							

- **REM2NP1: BCH Remainder 2 * N + 1**

When sector size is set to 512 bytes, bit REM2NP1[13] is not used and read as zero.

If bit *i* of the REM2NP1 field is set to one then the coefficient of the X ^ *i* is set to one, otherwise the coefficient is zero.

- **REM2NP3: BCH Remainder 2 * N + 3**

When sector size is set to 512 bytes, bit REM2NP3[29] is not used and read as zero.

If bit *i* of the REM2NP3 field is set to one then the coefficient of the X ^ *i* is set to one, otherwise the coefficient is zero.

28. Programmable Multibit ECC Error Location Controller (PMERRLOC)

28.1 Description

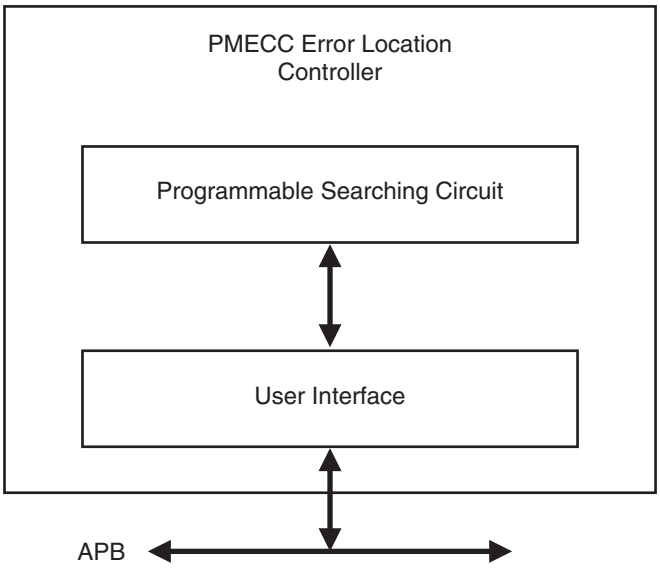
The PMECC Error Location Controller provides hardware acceleration for determining roots of polynomials over two finite fields: GF(2^13) and GF(2^14). It integrates 24 fully programmable coefficients. These coefficients belong to GF(2^13) or GF(2^14). The coefficient programmed in the PMERRLOC_SIGMAx register is the coefficient of degree x in the polynomial.

28.2 Embedded Characteristics

- Provides Hardware Acceleration for determining roots of polynomials defined over a finite field
- Programmable Finite Field GF(2^13) or GF(2^14)
- Finds Roots of Error Locator Polynomial
- Programmable Number of Roots

28.3 Block Diagram

Figure 28-1. Block Diagram



28.4 Functional Description

The PMERRLOC search operation is started as soon as a write access is detected in the ELEN register and can be disabled by writing to the ELDIS register. The ENINIT field of the ELEN register shall be initialized with the number of Galois field elements to test. The set of the roots can be limited to a valid range.

Table 28-1. ENINIT field value for a sector size of 512 bytes

Error Correcting Capability	ENINIT Value
2	**4122**
4	**4148**
8	**4200**
12	**4252**
24	**4408**

Table 28-2. ENINIT field value for a sector size of 1024 bytes

Error Correcting Capability	ENINIT Value
2	8220
4	8248
8	8304
12	8360
24	8528

When the PMEERRLOC engine is searching for roots the BUSY field of the ELSR remains asserted. An interrupt is asserted at the end of the computation, and the DONE bit of the ELSIR register is set. The ERR_CNT field of the ELISR indicates the number of errors. The error position can be read in the PMERRLOCx registers.

28.5 Programmable Multibit ECC Error Location Controller (PMERRLOC) User Interface

Table 28-3. Register Mapping

Offset	Register	Name	Access	Reset
0x000	Error Location Configuration Register	PMERRLOC_ELCFG	Read-write	0x00000000
0x004	Error Location Primitive Register	PMERRLOC_ELPRIM	Read-only	0x00000000
0x008	Error Location Enable Register	PMERRLOC_ELEN	Read-write	0x00000000
0x00C	Error Location Disable Register	PMERRLOC_ELDIS	Read-write	0x00000000
0x010	Error Location Status Register	PMERRLOC_ELSR	Read-write	0x00000000
0x014	Error Location Interrupt Enable register	PMERRLOC_ELIER	Read-only	0x00000000
0x018	Error Location Interrupt Disable Register	PMERRLOC_ELIDR	Read-only	0x00000000
0x01C	Error Location Interrupt Mask Register	PMERRLOC_ELIMR	Read-only	0x00000000
0x020	Error Location Interrupt Status Register	PMERRLOC_ELISR	Read-only	0x00000000
0x024	Reserved	–	–	–
0x028	PMECC SIGMA 0 Register	PMERRLOC_SIGMA0	Read-write	0x00000000
...	...	...	...	...
0x088	PMECC SIGMA 24 Register	PMERRLOC_SIGMA24	Read-write	0x00000000
0x08C	PMECC Error Location 0 Register	PMERRLOC_EL0	Read-only	0x00000000
...	...	...	...	...
0x0E4	PMECC Error Location 23 Register	PMERRLOC_EL23	Read-only	0x00000000
0xE8 - 0X1FC	Reserved	–	–	–

28.5.1 Error Location Configuration Register

Name: PMERRLOC_ELCFG

Address: 0xFFFFE600

Access: Read-write

Reset: 0x00000000

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

- **ERRNUM: Number of Errors**

- **SECTORSZ: Sector Size**

0: The ECC computation is based on a 512-byte sector.

1: The ECC computation is based on a 1024-byte sector.

28.5.2 Error Location Primitive Register

Name: PMERRLOC_ELPRIM

Address: 0xFFFFE604

Access: Read-only

Reset: 0x00000000

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

- **PRIMITIV: Primitive Polynomial**

28.5.3 Error Location Enable Register

Name: PMERRLOC_ELEN

Address: 0xFFFFE608

Access: Read-write

Reset: 0x00000000

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

- **ENINIT: Initial Number of Bits in the Codeword**

28.5.4 Error Location Disable Register

Name: PMERRLOC_ELDIS

Address: 0xFFFFE60C

Access: Read-write

Reset: 0x00000000

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

- **DIS: Disable Error Location Engine**

28.5.5 Error Location Status Register

Name: PMERRLOC_ELSR

Address: 0xFFFFE610

Access: Read-write

Reset: 0x00000000

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

- **BUSY: Error Location Engine Busy**

28.5.6 Error Location Interrupt Enable Register

Name: PMERRLOC_ELIER

Address: 0xFFFFE614

Access: Read-only

Reset: 0x00000000

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

- **DONE: Computation Terminated Interrupt Enable**

28.5.7 Error Location Interrupt Disable Register

Name: PMERRLOC_ELIDR

Address: 0xFFFFE618

Access: Read-only

Reset: 0x00000000

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

- **DONE: Computation Terminated Interrupt Disable**

28.5.8 Error Location Interrupt Mask Register

Name: PMERRLOC_ELIMR

Address: 0xFFFFE61C

Access: Read-only

Reset: 0x00000000

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

- **DONE: Computation Terminated Interrupt Mask**

28.5.9 Error Location Interrupt Status Register

Name: PMERRLOC_ELISR

Address: 0xFFFFE620

Access: Read-only

Reset: 0x00000000

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

- **DONE: Computation Terminated Interrupt Status**
- **ERR_CNT: Error Counter Value**

28.5.10 Error Location SIGMAx Register

Name: PMERRLOC_SIGMAx [x=0..24]

Address: 0xFFFFE628 [0] .. 0xFFFFE688 [24]

Access: Read-Write

Reset: 0x00000000

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

- **SIGMAx: Coefficient of Degree x in the SIGMA Polynomial.**

SIGMAx belongs to the finite field GF(2^13) when the sector size is set to 512 bytes.

SIGMAx belongs to the finite field GF(2^14) when the sector size is set to 1024 bytes.

28.5.11 PMECC Error Locationx Register

Name: PMERRLOC_ELx [x=0..23]

Address: 0xFFFFE68C

Access: Read-only

Reset: 0x00000000

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

- **ERRLOCN: Error Position within the Set {sector area, spare area}.**

ERRLOCN points to 0 when the first bit of the main area is corrupted.

If the sector size is set to 512 bytes, the ERRLOCN points to 4096 when the last bit of the sector area is corrupted.

If the sector size is set to 1024 bytes, the ERRLOCN points to 8192 when the last bit of the sector area is corrupted.

If the sector size is set to 512 bytes, the ERRLOCN points to 4097 when the first bit of the spare area is corrupted.

If the sector size is set to 1024 bytes, the ERRLOCN points to 8193 when the first bit of the spare area is corrupted.

29. Static Memory Controller (SMC)

29.1 Description

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 6 Chip Selects and a 26-bit address bus. The 32-bit data bus can be configured to interface with 8-, 16-, or 32-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

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

29.2 Embedded Characteristics

- 6 Chip Selects Available
- 64-Mbyte Address Space per Chip Select
- 8-, 16- or 32-bit Data Bus
- Word, Halfword, Byte Transfers
- Byte Write or Byte Select Lines
- Programmable Setup, Pulse And Hold Time for Read Signals per Chip Select
- Programmable Setup, Pulse And Hold Time for Write Signals per Chip Select
- Programmable Data Float Time per Chip Select
- Compliant with LCD Module
- External Wait Request
- Automatic Switch to Slow Clock Mode
- Asynchronous Read in Page Mode Supported: Page Size Ranges from 4 to 32 Bytes

29.3 I/O Lines Description

Table 29-1. 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

29.4 Multiplexed Signals

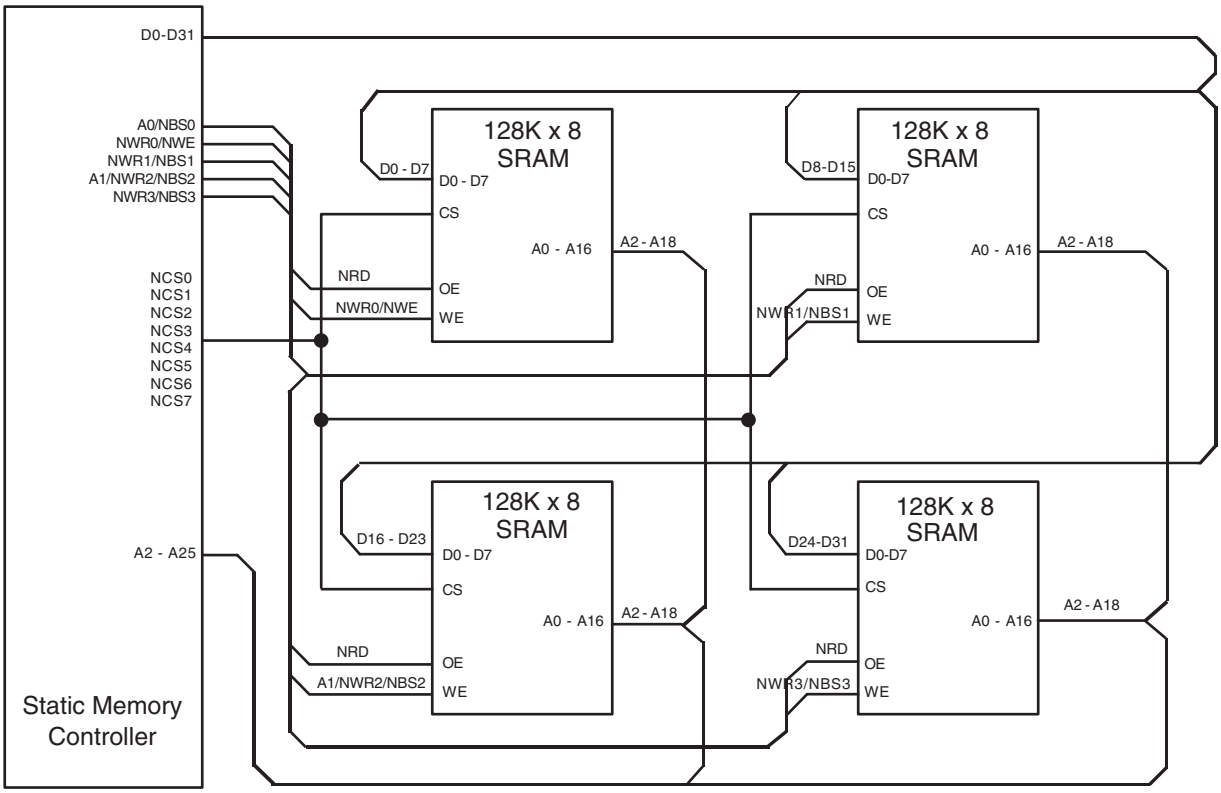
Table 29-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see “Byte Write or Byte Select Access” on page 387
A0	NBS0		8-bit or 16-/32-bit data bus, see “Data Bus Width” on page 387
NWR1	NBS1		Byte-write or byte-select access see “Byte Write or Byte Select Access” on page 387
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see “Data Bus Width” on page 387. Byte-write or byte-select access, see “Byte Write or Byte Select Access” on page 387
NWR3	NBS3		Byte-write or byte-select access see “Byte Write or Byte Select Access” on page 387

29.5 Application Example

29.5.1 Hardware Interface

Figure 29-1. SMC Connections to Static Memory Devices



29.6 Product Dependencies

29.6.1 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.

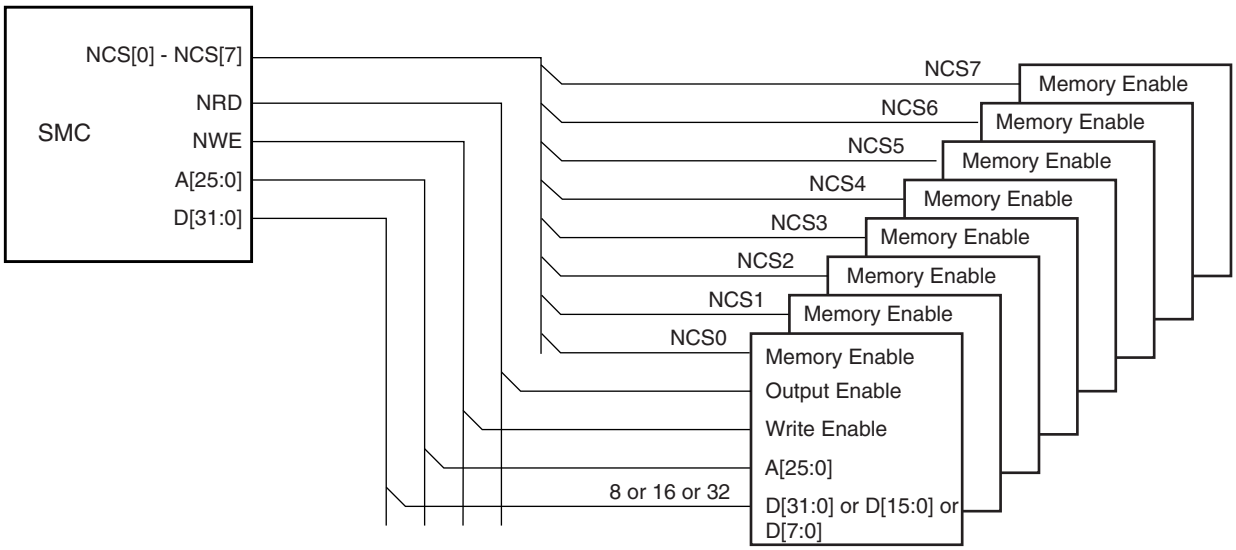
29.7 External Memory Mapping

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 29-2).

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 29-2. Memory Connections for Eight External Devices



29.8 Connection to External Devices

29.8.1 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 29-3 shows how to connect a 512K x 8-bit memory on NCS2. Figure 29-4 shows how to connect a 512K x 16-bit memory on NCS2. Figure 29-5 shows two 16-bit memories connected as a single 32-bit memory

29.8.2 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 29-3. Memory Connection for an 8-bit Data Bus

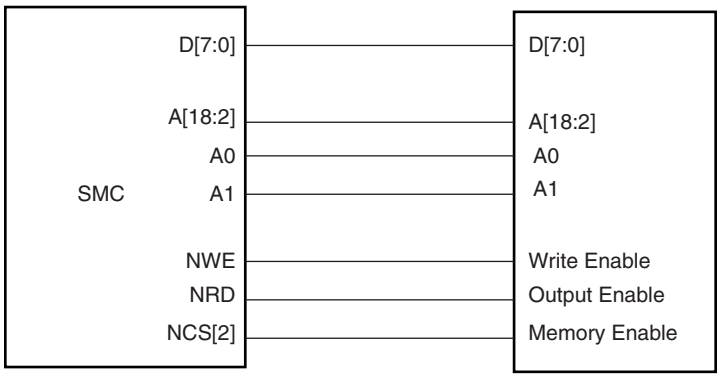


Figure 29-4. Memory Connection for a 16-bit Data Bus

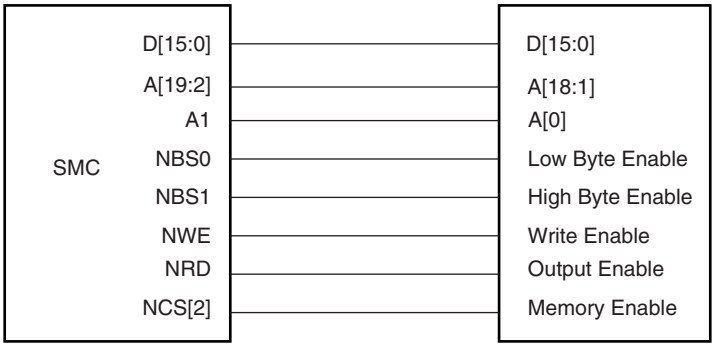
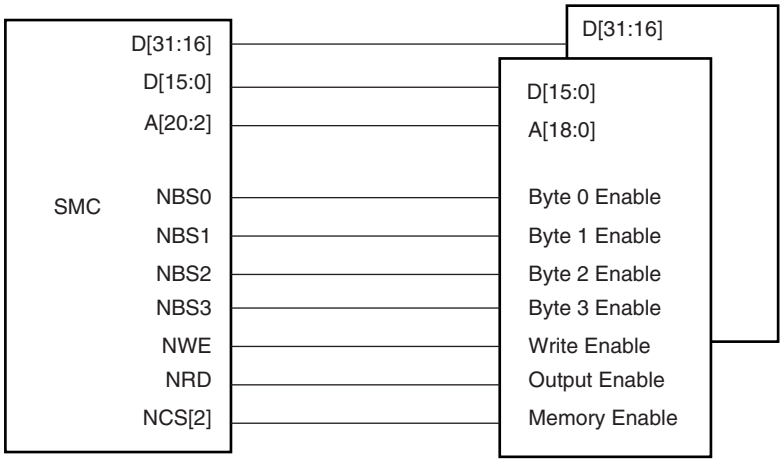


Figure 29-5. Memory Connection for a 32-bit Data Bus



29.8.2.1 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 29-6.

29.8.2.2 Byte Select Access

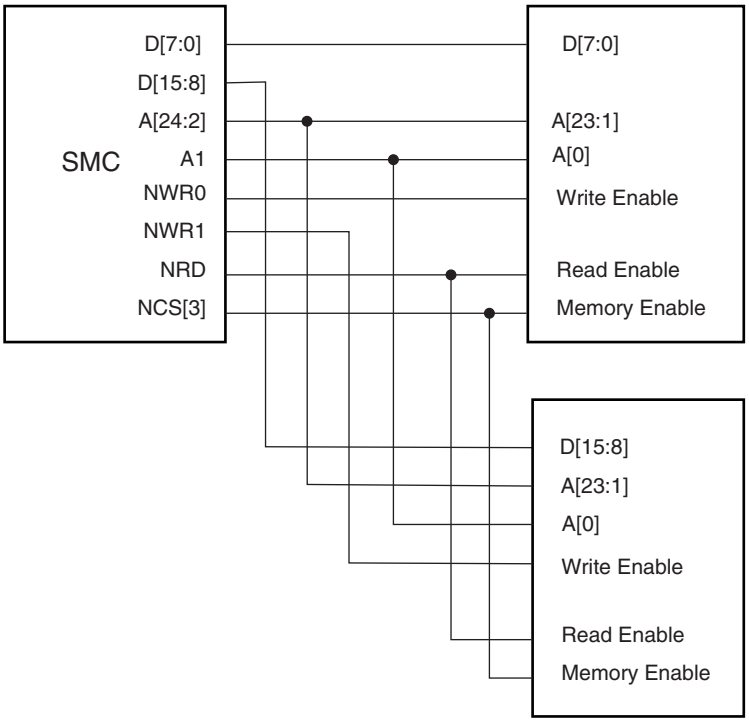
In this mode, read/write operations can be enabled/disabled at a byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

- 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 29-7 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 29-6. Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option



29.8.2.3 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 29-3 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 29-7. Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)

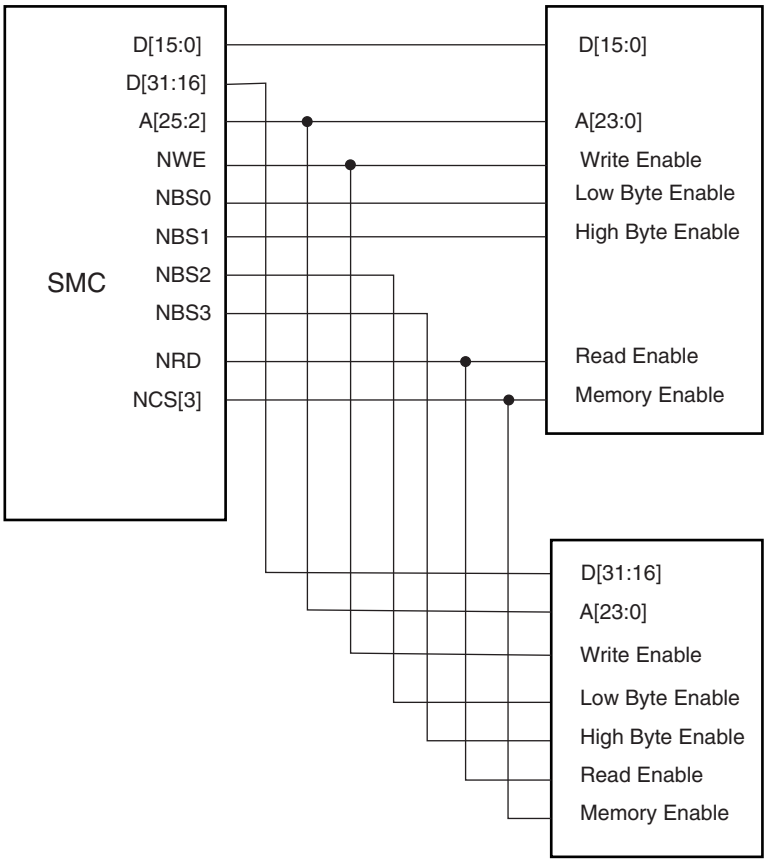


Table 29-3. SMC Multiplexed Signal Translation

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			

29.9 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS3) always have the same timing as the A address bus. NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR3) in byte write access type. NWR0 to NWR3 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..5] chip select lines.

29.9.1 Read Waveforms

The read cycle is shown on Figure 29-8.

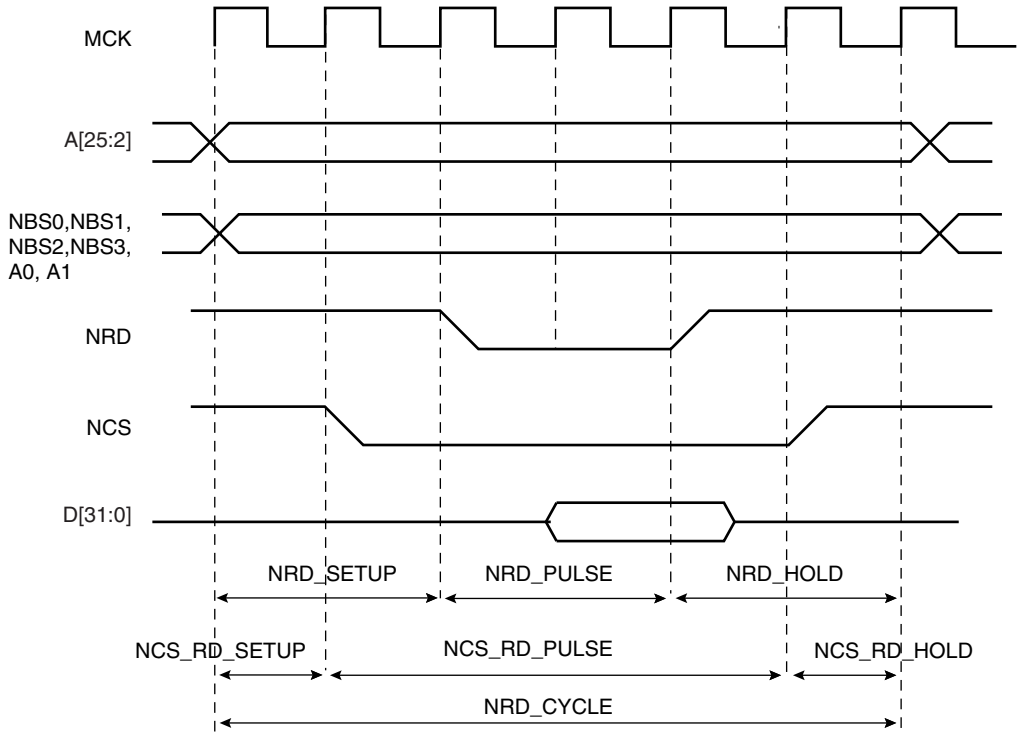
The read cycle starts with the address setting on the memory address bus, i.e.:

{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 29-8. Standard Read Cycle



29.9.1.1 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.

29.9.1.2 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.

29.9.1.3 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:

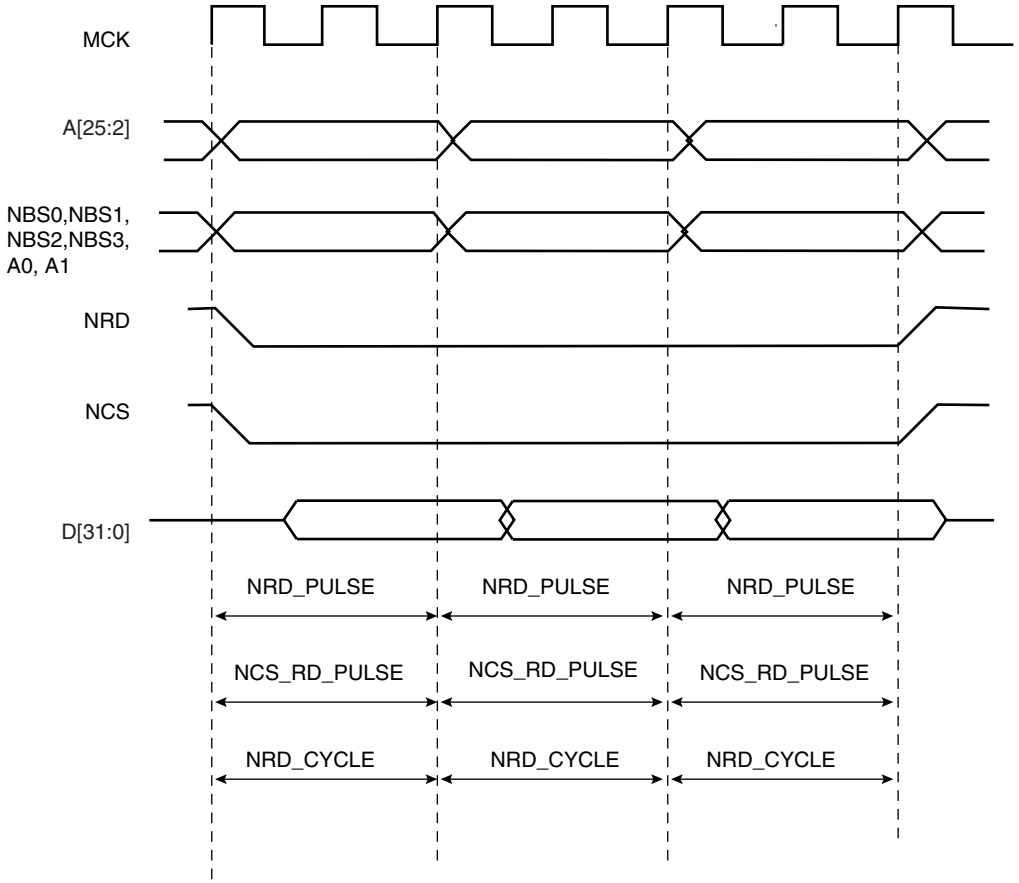
NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

29.9.1.4 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 29-9).

Figure 29-9. No Setup, No Hold On NRD and NCS Read Signals



29.9.1.5 Null Pulse

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

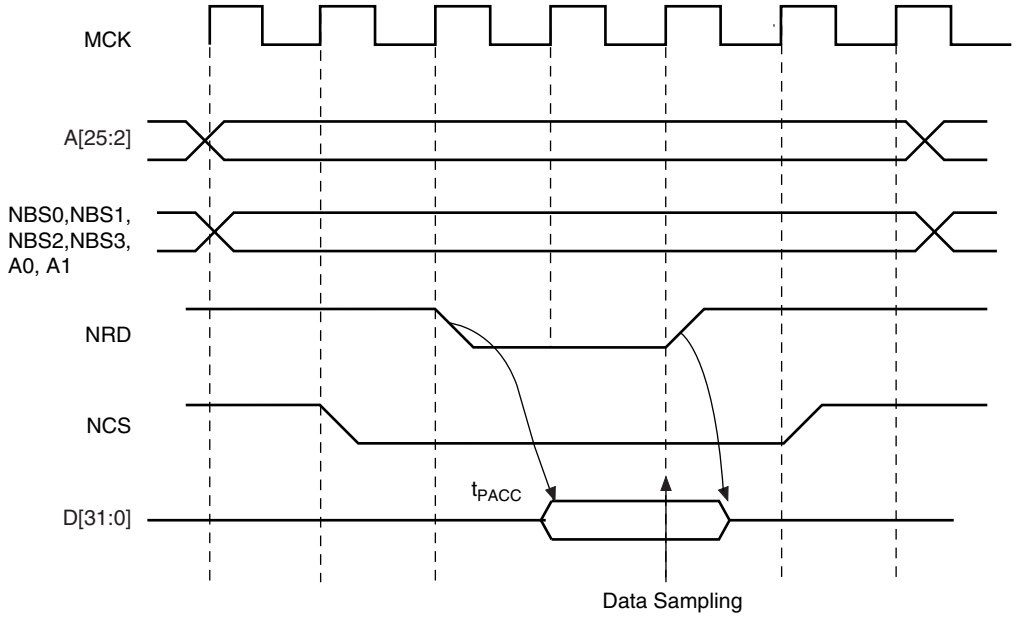
29.9.2 Read Mode

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.

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

Figure 29-10 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.

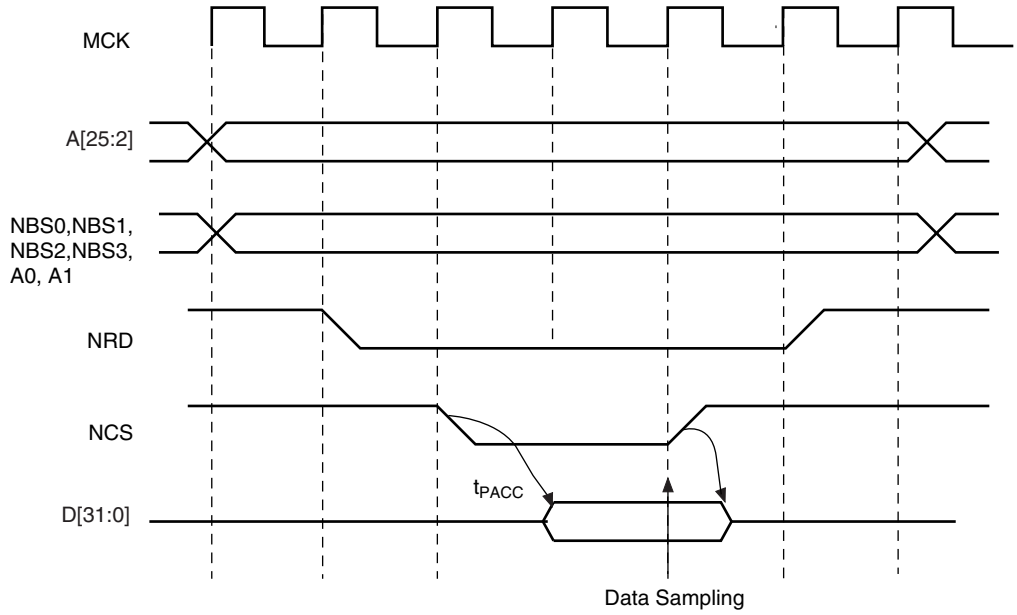
Figure 29-10.READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD



29.9.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 29-11 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 29-11.READ_MODE = 0: Data is sampled by SMC before the rising edge of NCS



29.9.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 29-12. The write cycle starts with the address setting on the memory address bus.

29.9.3.1 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.

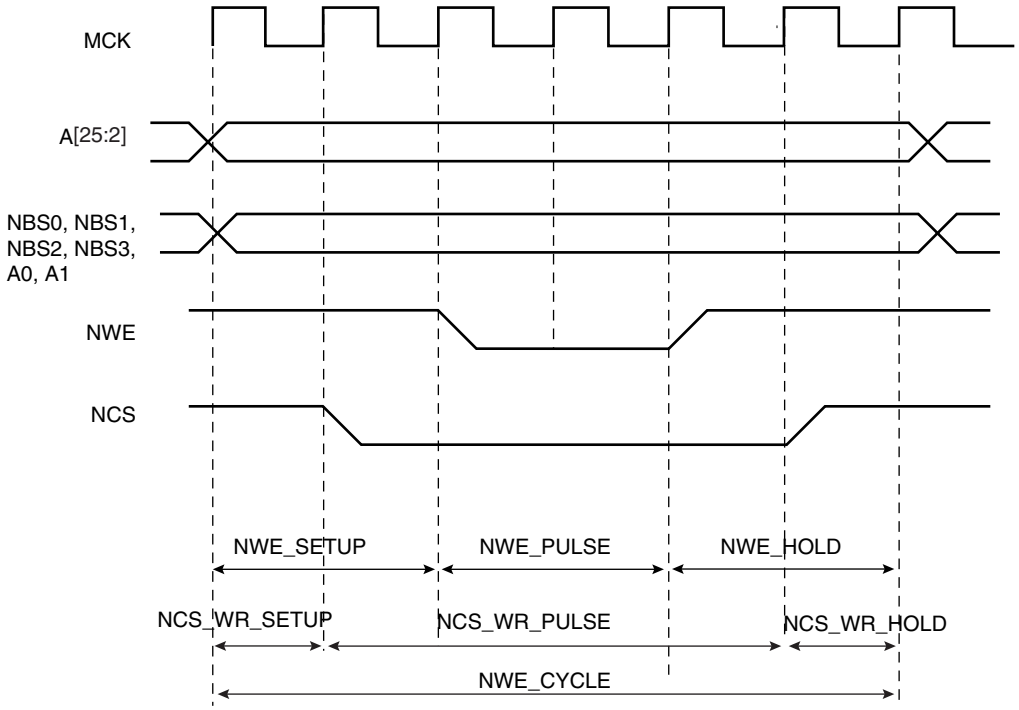
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

29.9.3.2 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 29-12.Write Cycle



29.9.3.3 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:

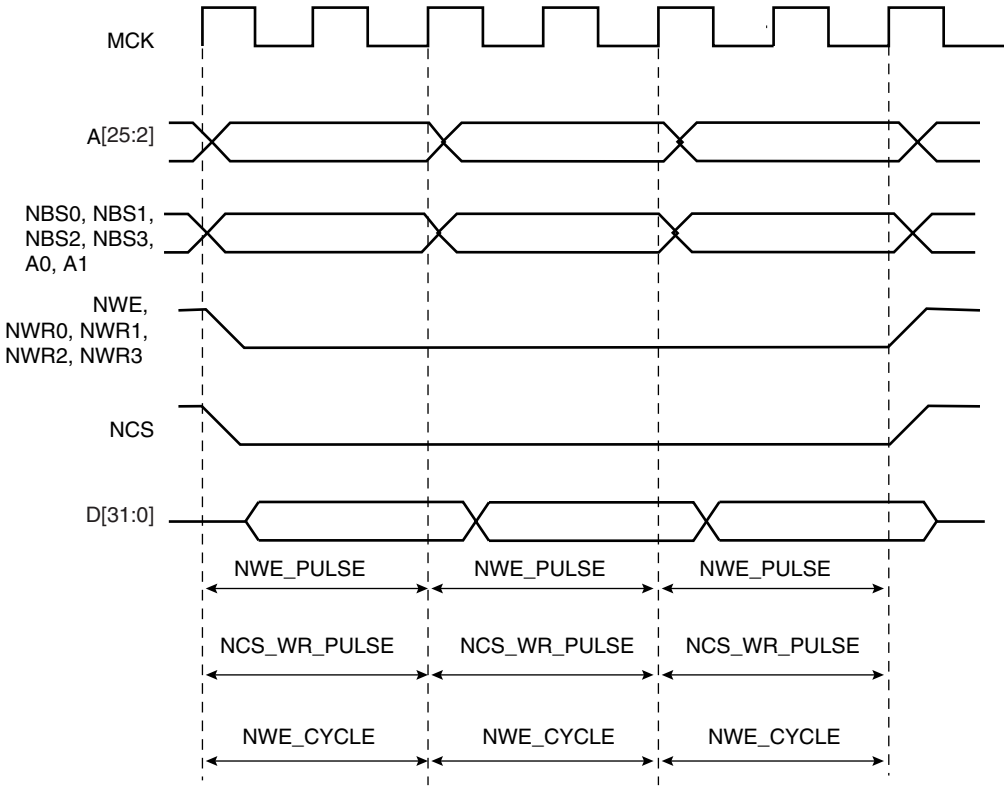
NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

29.9.3.4 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 29-13). 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 29-13.Null Setup and Hold Values of NCS and NWE in Write Cycle



29.9.3.5 Null Pulse

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

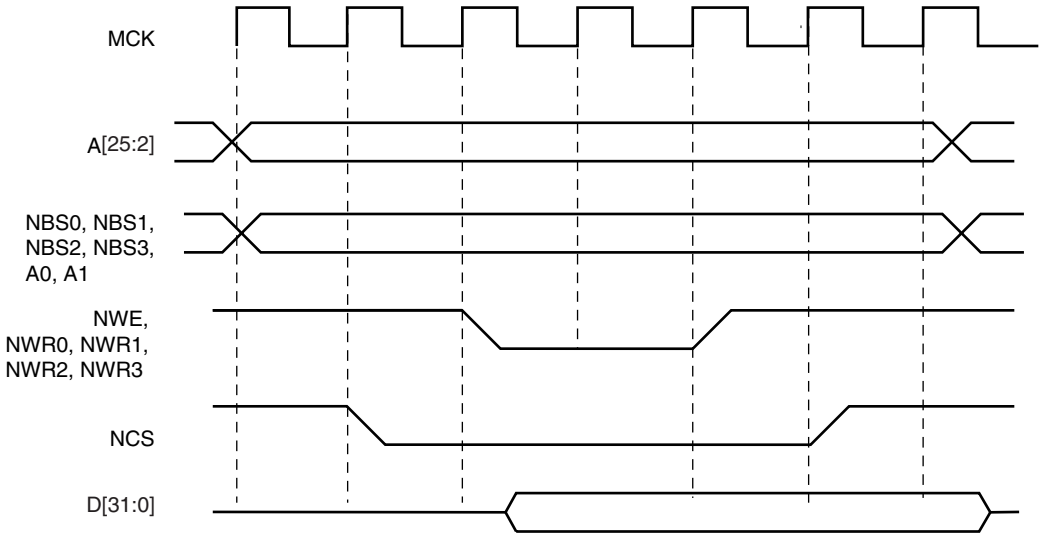
29.9.4 Write Mode

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

29.9.4.1 Write is Controlled by NWE (WRITE_MODE = 1)

Figure 29-14 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 switched to output mode after the NWE_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

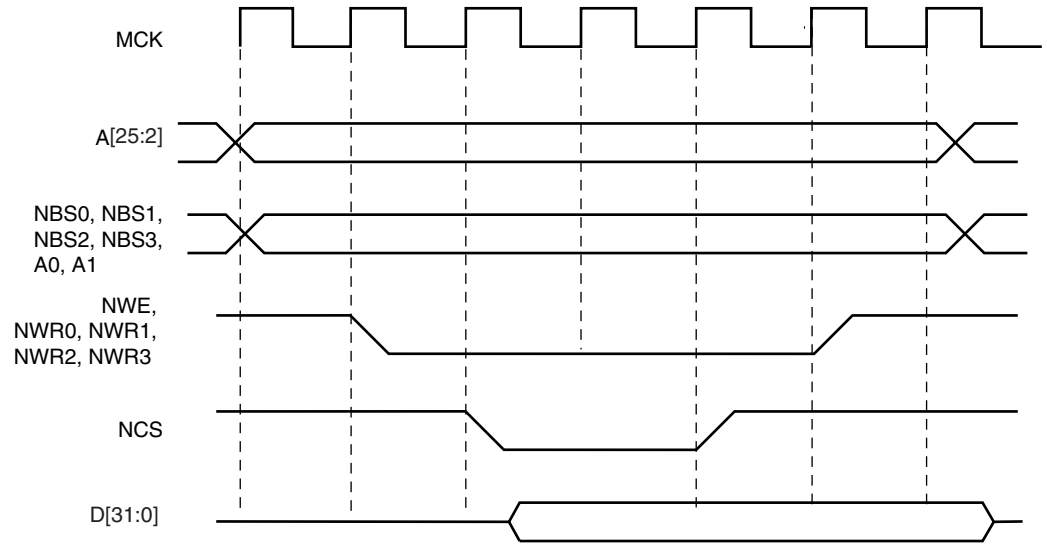
Figure 29-14.WRITE_MODE = 1. The write operation is controlled by NWE



29.9.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 29-15 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 switched to output mode after the NCS_WR_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 29-15.WRITE_MODE = 0. The write operation is controlled by NCS



29.9.5 Write Protected Registers

To prevent any single software error that may corrupt SMC behavior, the registers listed below can be write-protected by setting the WPEN bit in the SMC Write Protect Mode Register (SMC_WPMR).

If a write access in a write-protected register is detected, then the WPVS flag in the SMC Write Protect Status Register (SMC_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically reset after reading the SMC Write Protect Status Register (SMC_WPSR).

List of the write-protected registers:

- Section 29.16.1 ”SMC Setup Register”
- Section 29.16.2 ”SMC Pulse Register”
- Section 29.16.3 ”SMC Cycle Register”
- Section 29.16.4 ”SMC MODE Register”
- Section 29.16.5 ”SMC DELAY I/O Register”

29.9.6 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 29-4 shows how the timing parameters are coded and their permitted range.

Table 29-4. 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

29.9.7 Reset Values of Timing Parameters

Table 29-8, “Register Mapping,” on page 420 gives the default value of timing parameters at reset.

29.9.8 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 400.

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.

29.10 Automatic Wait States

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

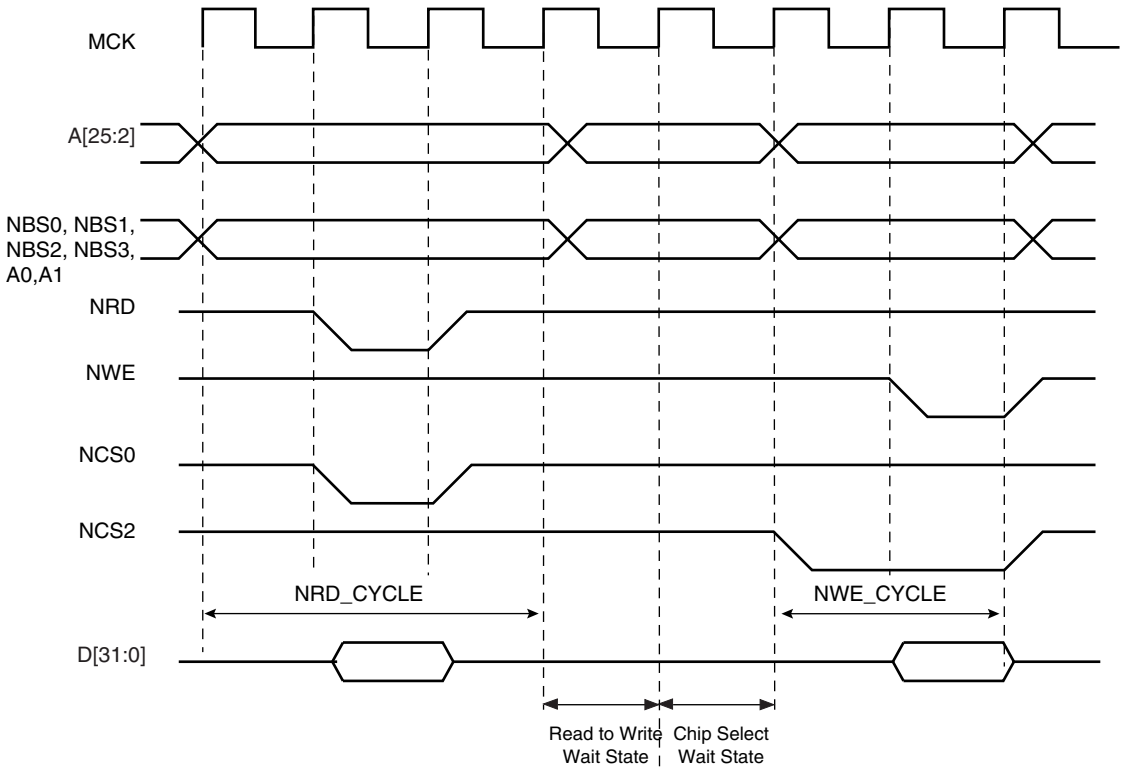
29.10.1 Chip Select Wait States

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

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

Figure 29-16 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

Figure 29-16.Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



29.10.2 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 29-17).
- 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 29-18). 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 29-19.

Figure 29-17.Early Read Wait State: Write with No Hold Followed by Read with No Setup

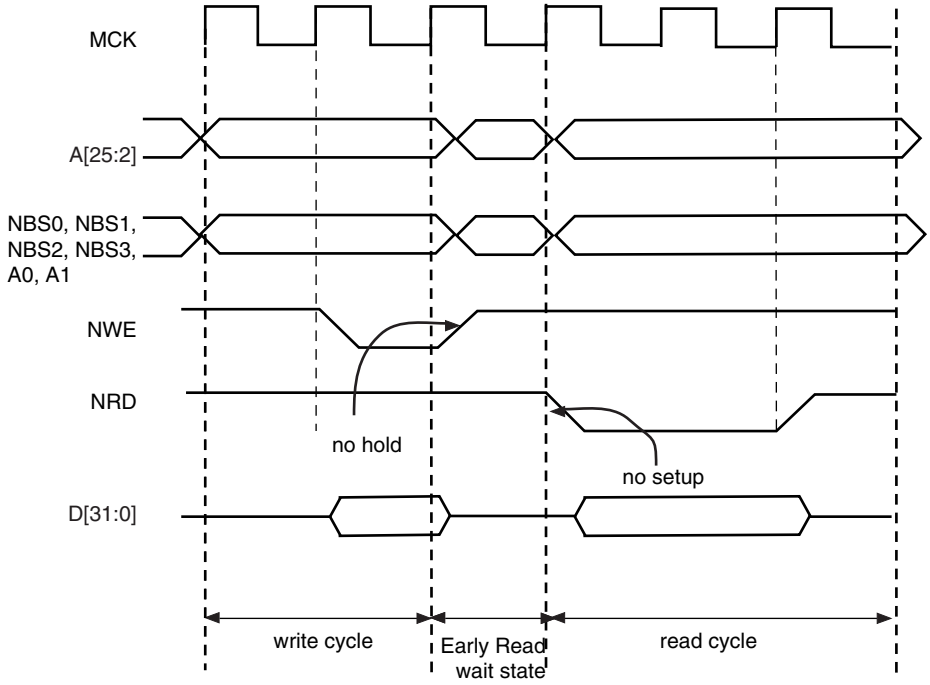


Figure 29-18.Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup

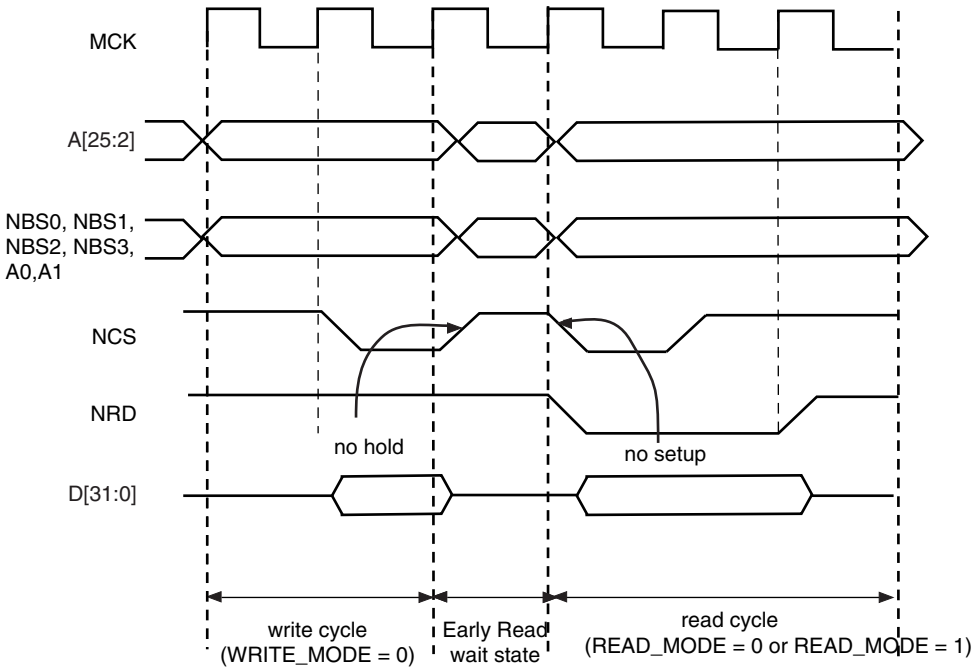
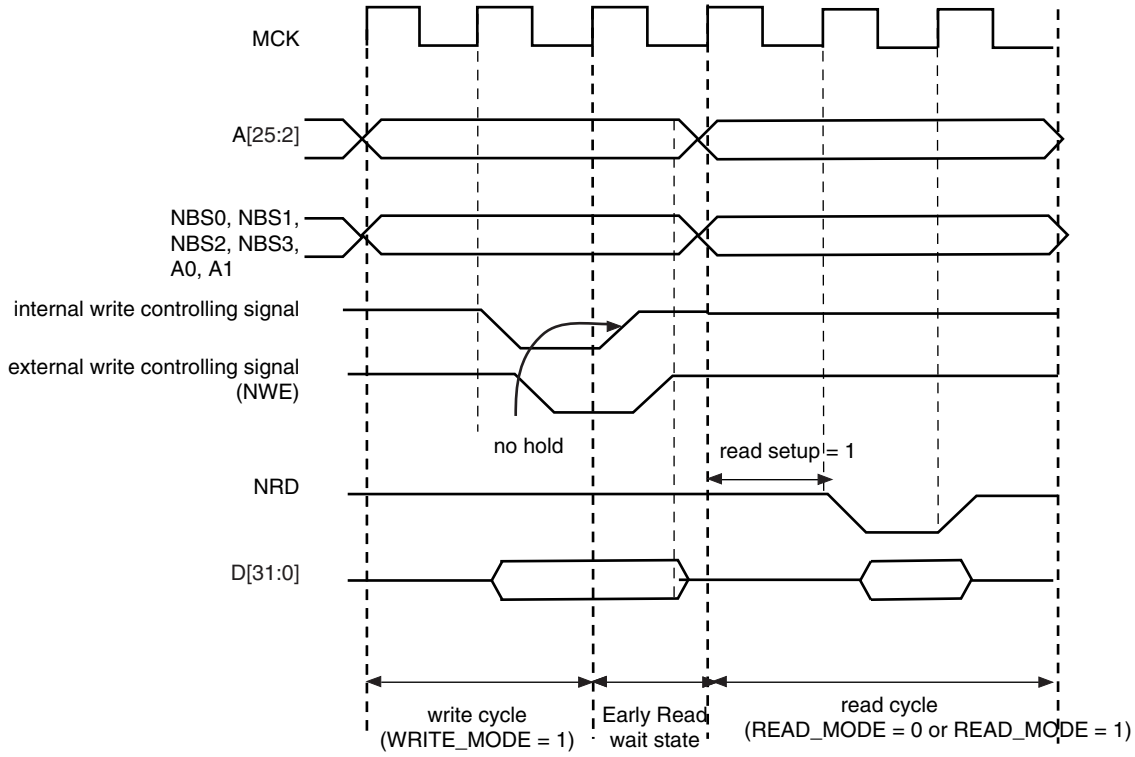


Figure 29-19.Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



29.10.3 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.

29.10.3.1 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.

29.10.3.2 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 413).

29.10.4 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 29-16 on page 399.

29.11 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.

29.11.1 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 29-20 illustrates the Data Float Period in NRD-controlled mode (READ_MODE =1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 29-21 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

Figure 29-20.TDF Period in NRD Controlled Read Access (TDF = 2)

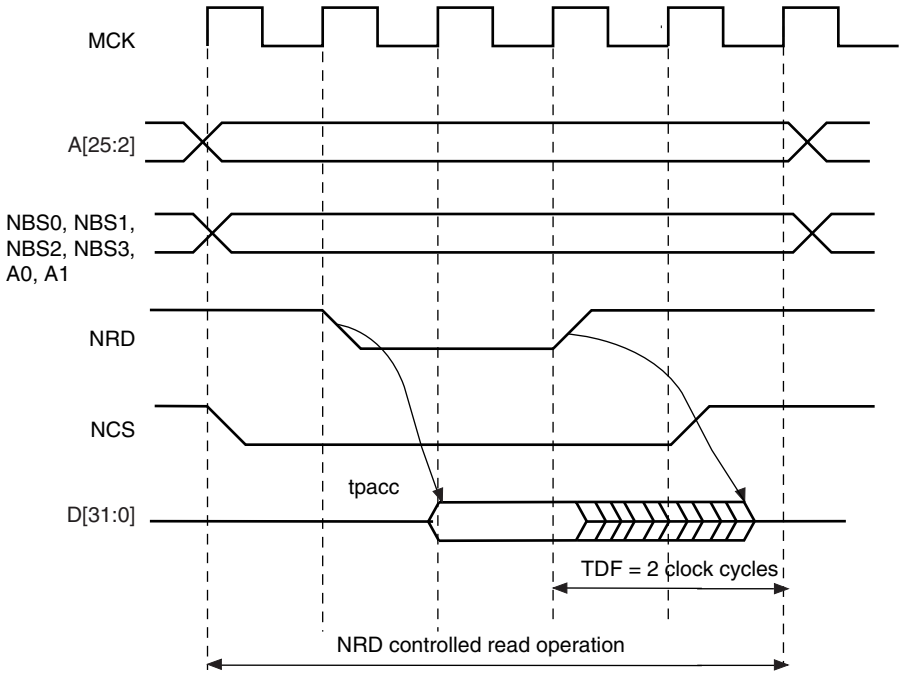
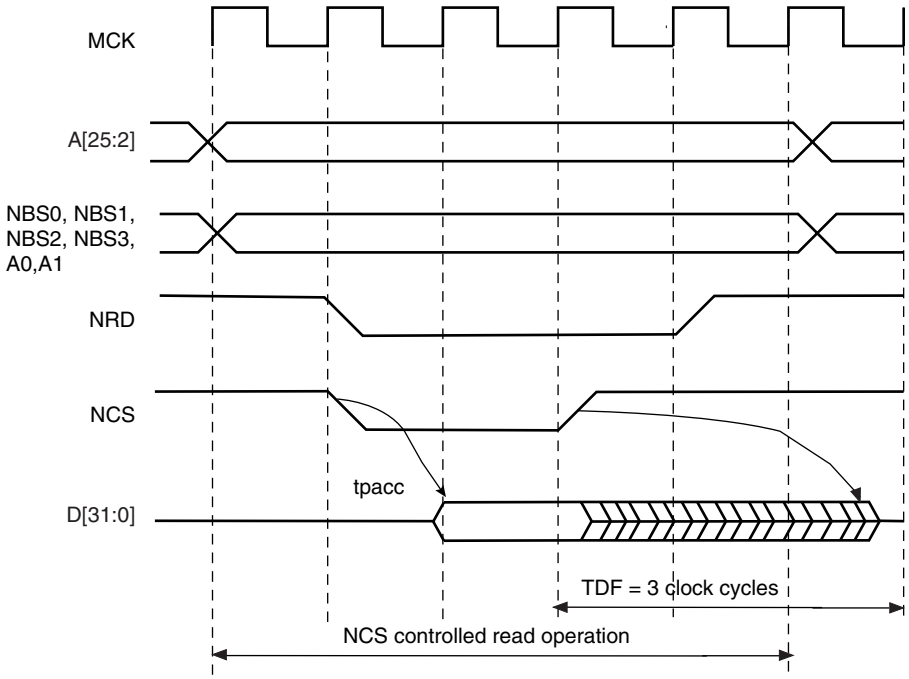


Figure 29-21.TDF Period in NCS Controlled Read Operation (TDF = 3)



29.11.2 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.

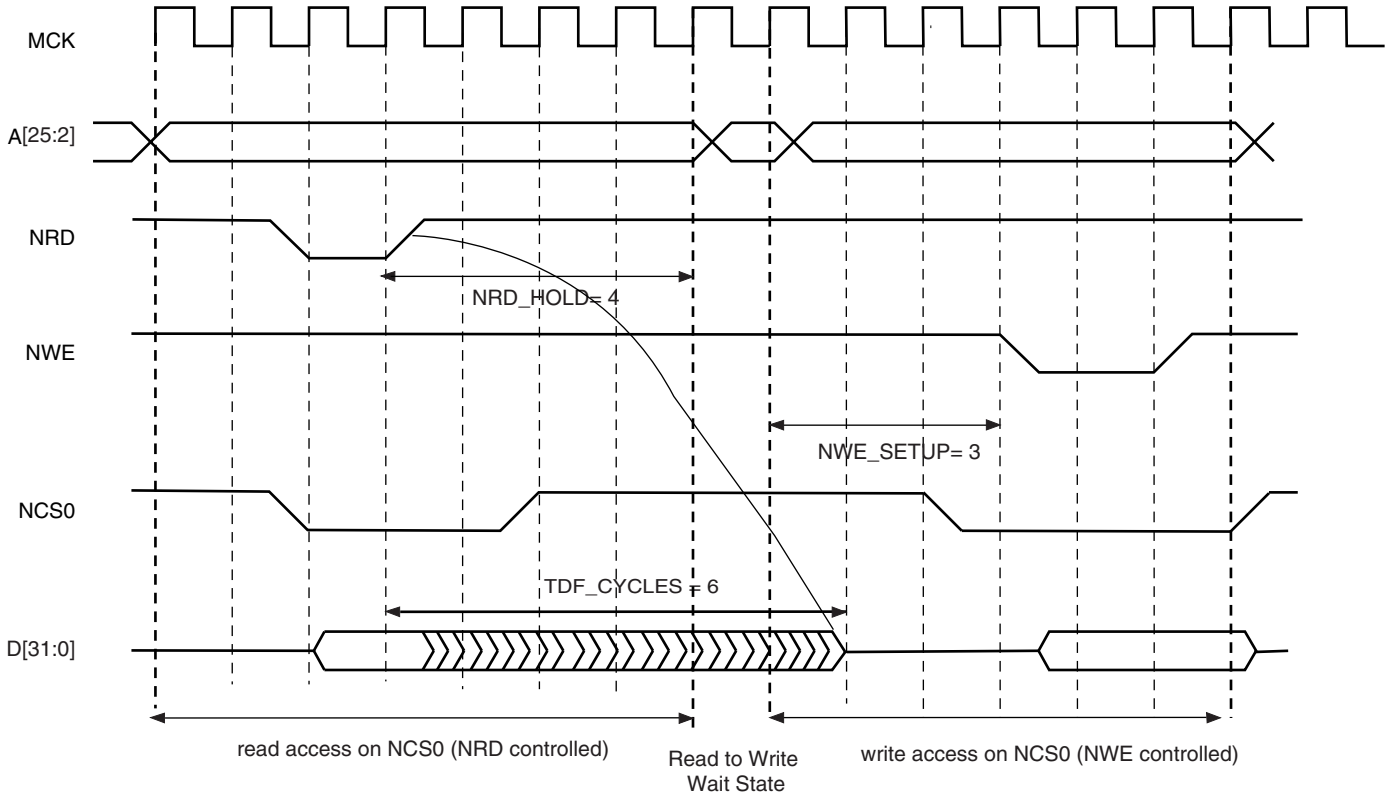
Figure 29-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

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 29-22.TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



29.11.3 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 29-23, Figure 29-24 and Figure 29-25 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 29-23.TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

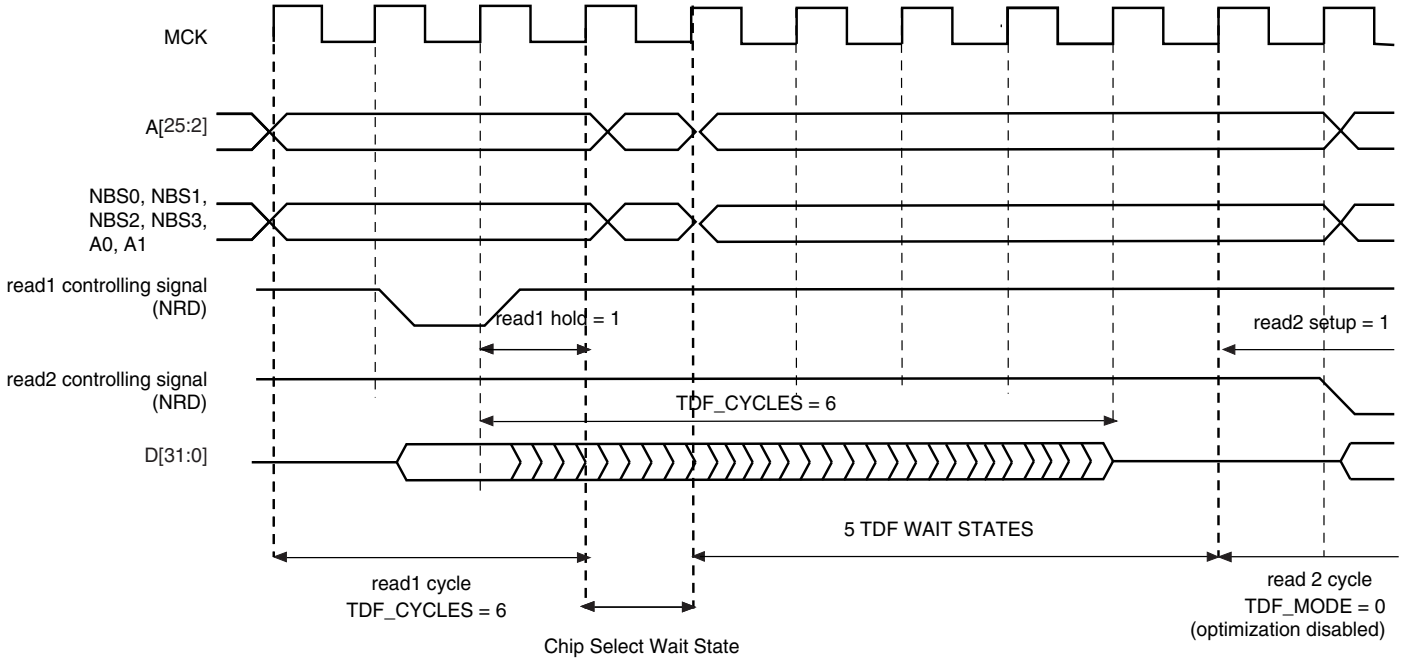


Figure 29-24. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

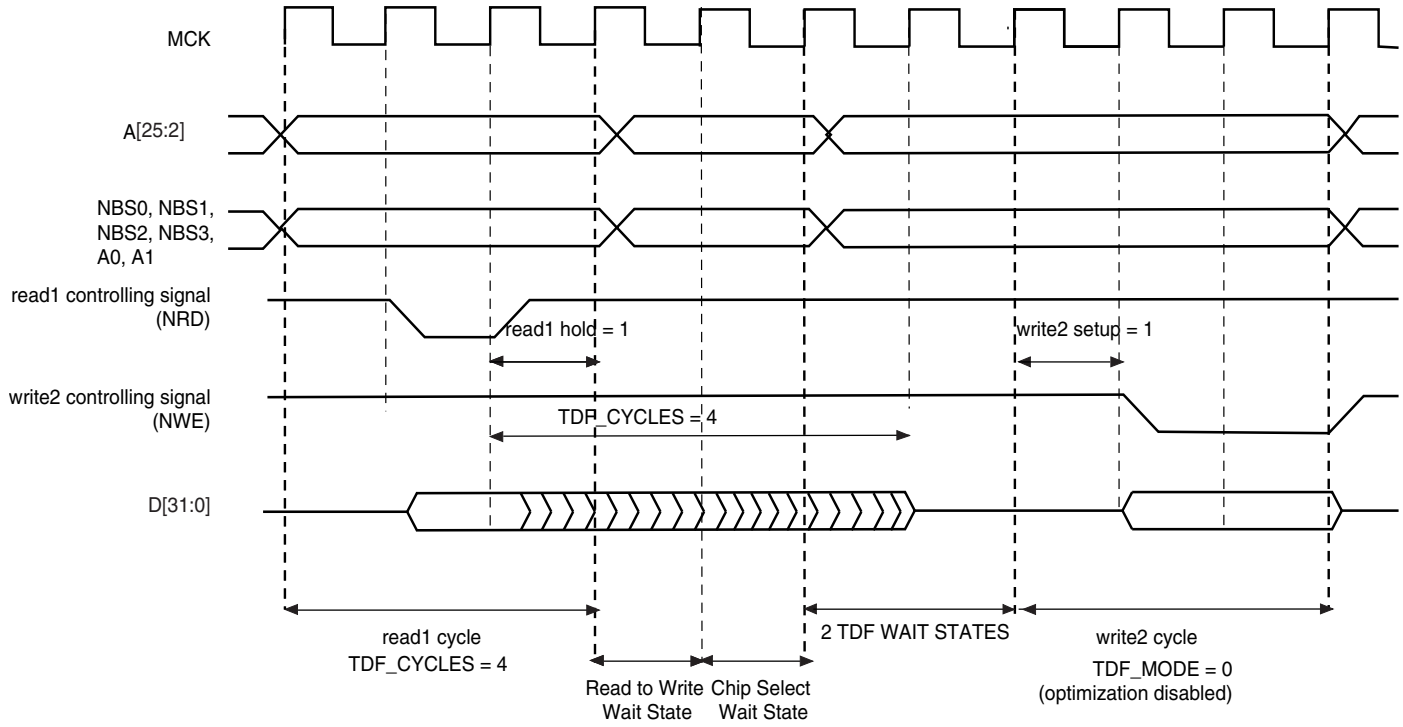
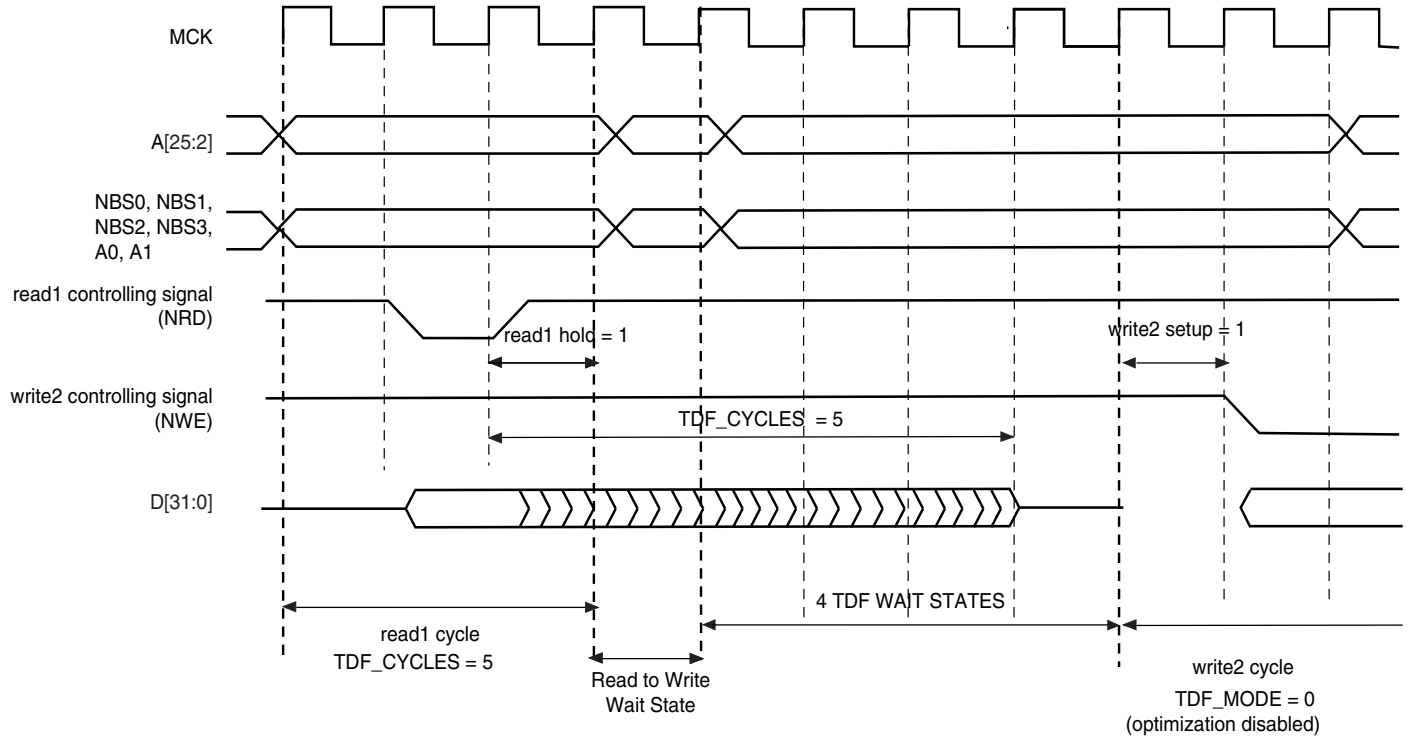


Figure 29-25.TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



29.12 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.

29.12.1 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 416), or in Slow Clock Mode (“Slow Clock Mode” on page 413).**

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.

29.12.2 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 29-26. 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 29-27.

Figure 29-26.Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

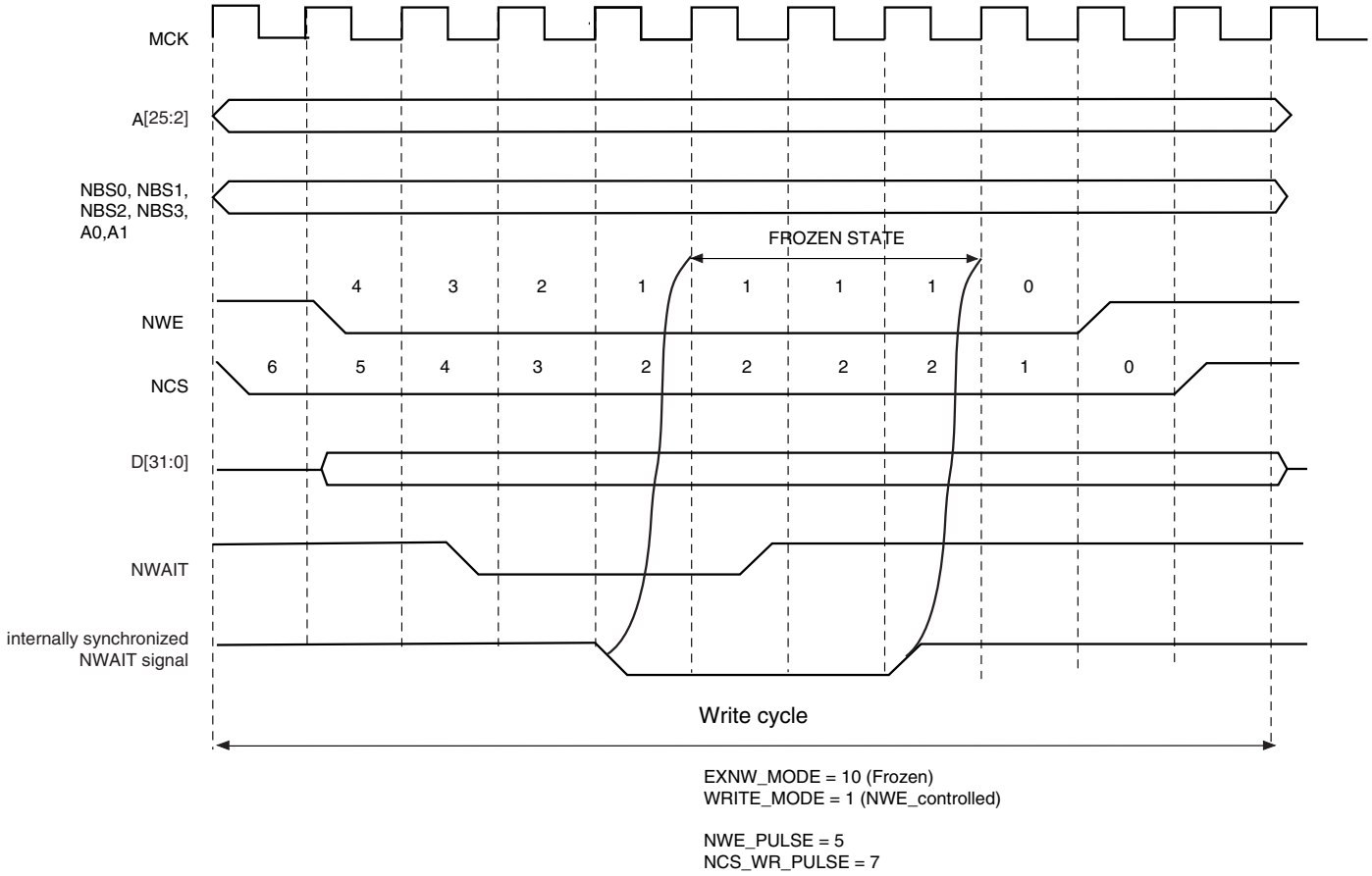
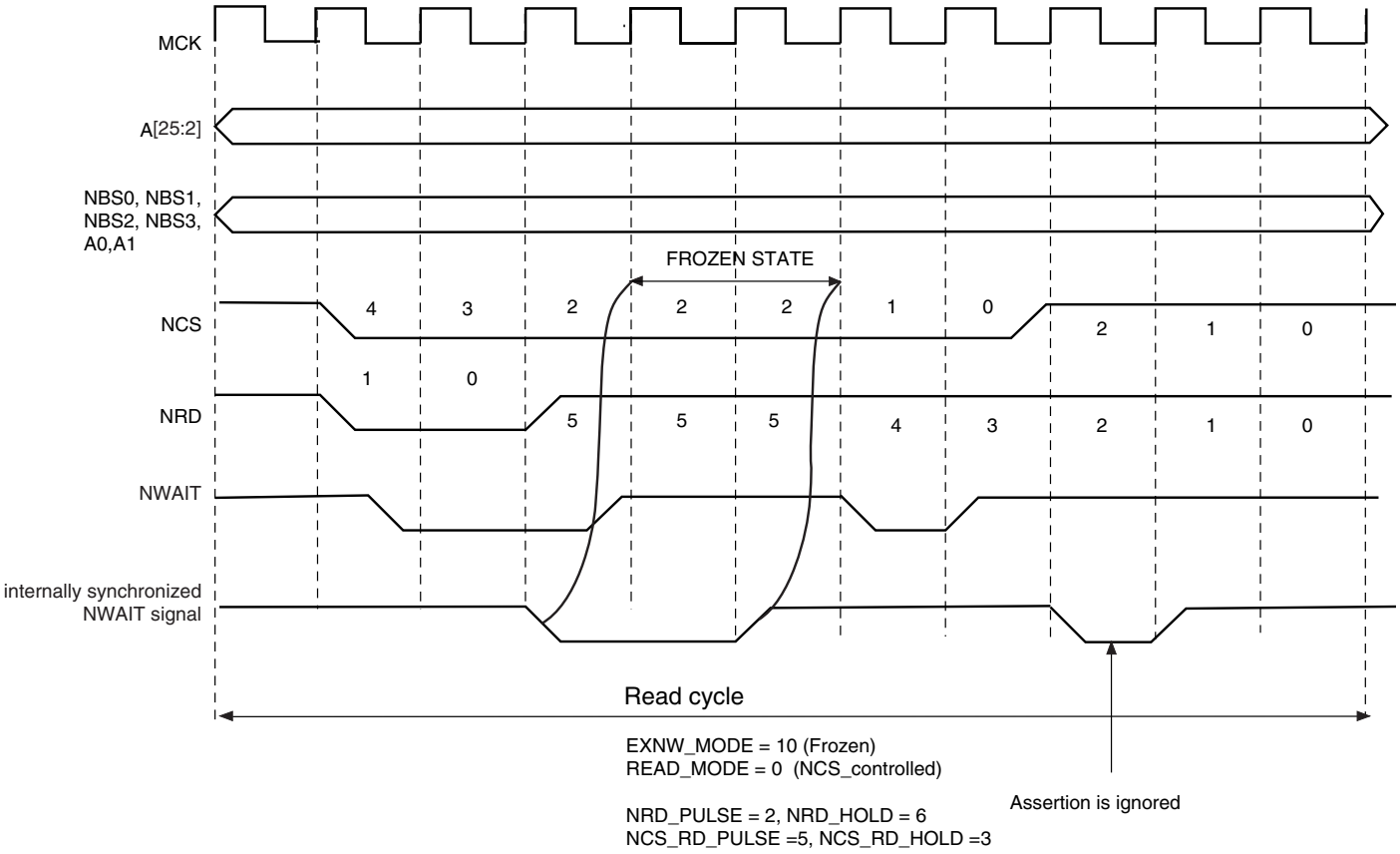


Figure 29-27.Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



29.12.3 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 29-28 and Figure 29-29. 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 29-29.

Figure 29-28.NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)

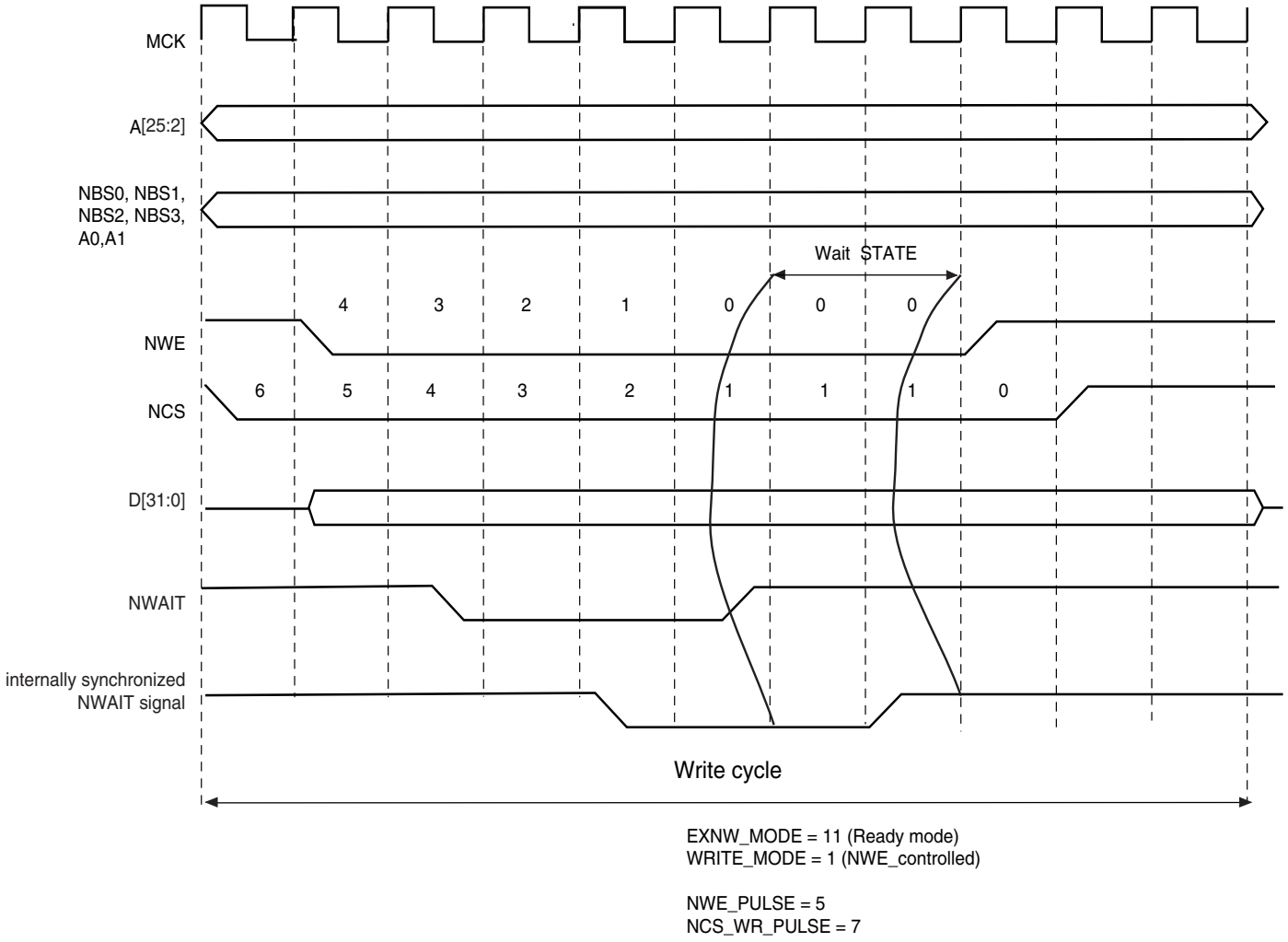
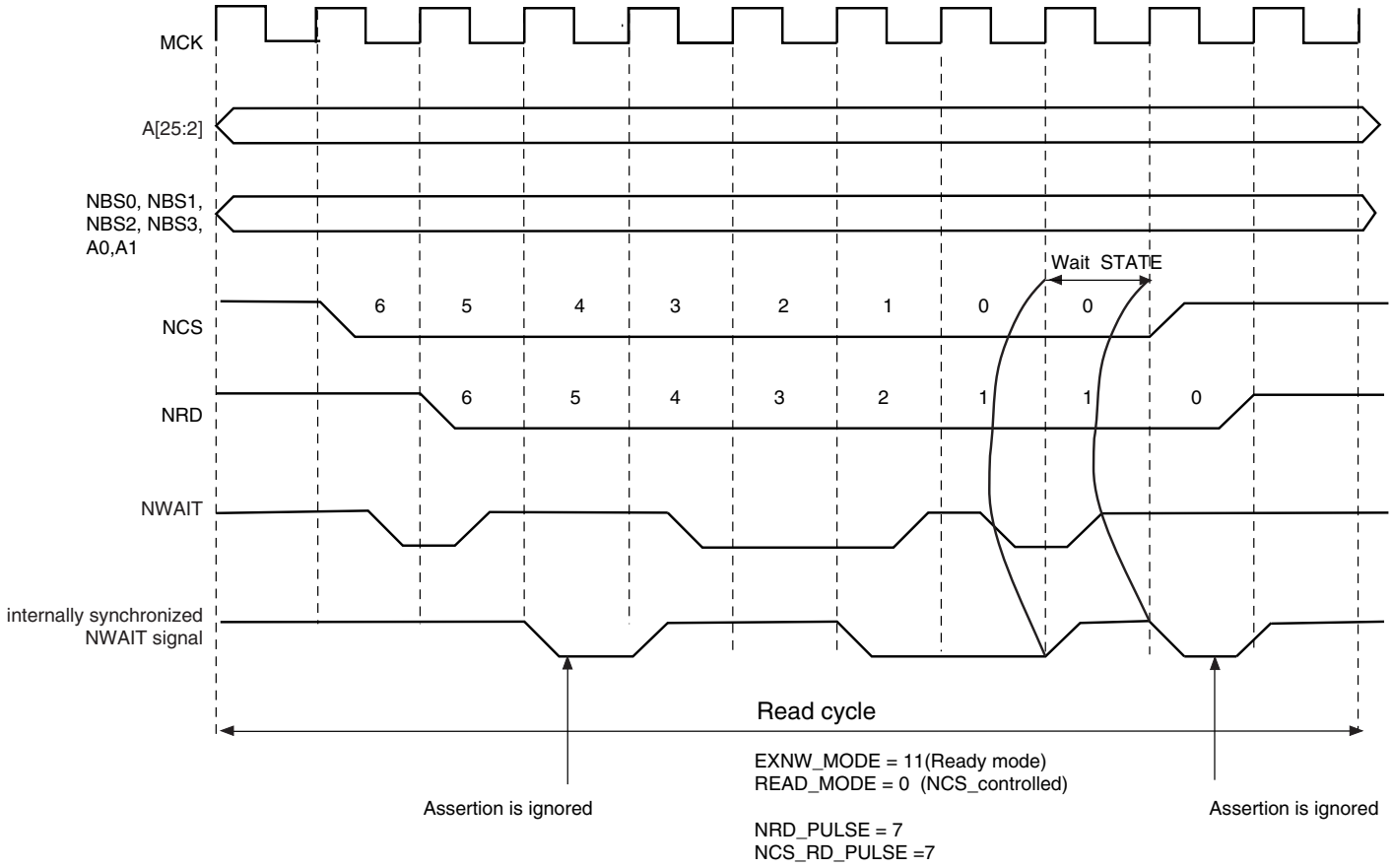


Figure 29-29.NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)



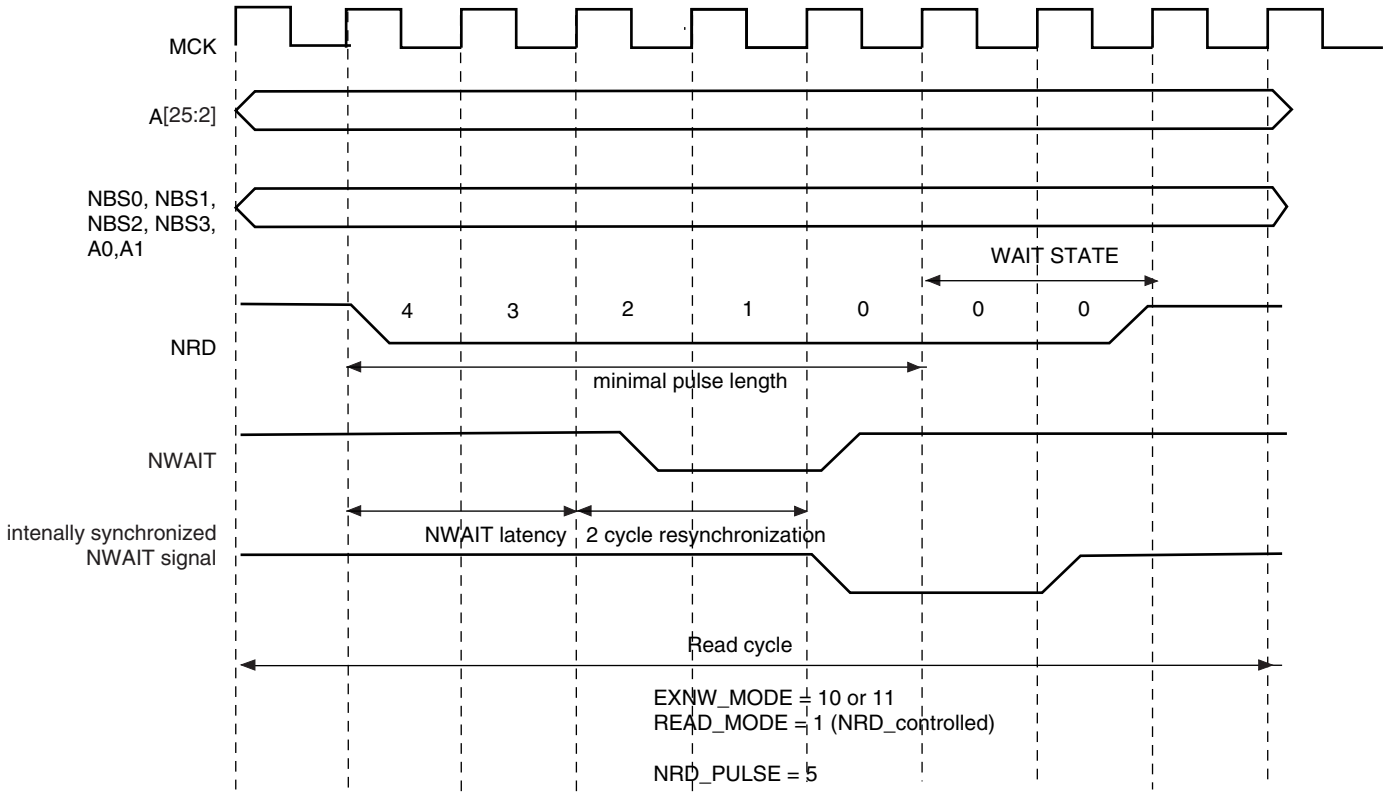
29.12.4 NWAIT Latency and Read/Write Timings

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 29-30.

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 29-30.NWAIT Latency



29.13 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 32 kHz 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.

29.13.1 Slow Clock Mode Waveforms

Figure 29-31 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 29-5 indicates the value of read and write parameters in slow clock mode.

Figure 29-31. Read/write Cycles in Slow Clock Mode

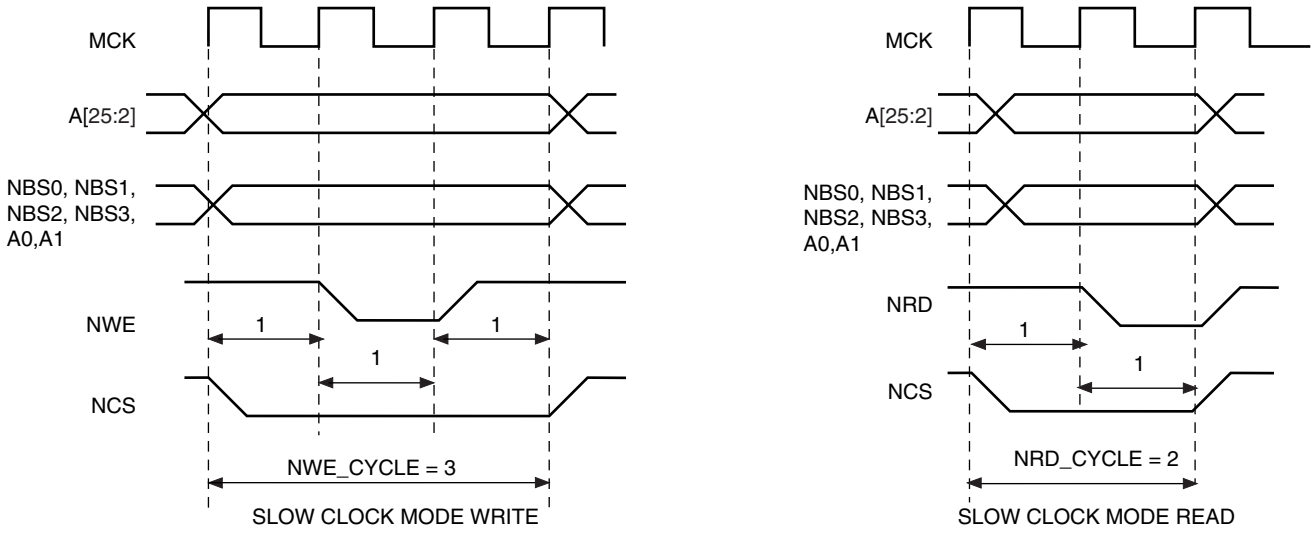


Table 29-5. Read and Write Timing Parameters in Slow Clock Mode

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

29.13.2 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 29-32 on page 414. The external device may not be fast enough to support such timings.

Figure 29-33 illustrates the recommended procedure to properly switch from one mode to the other.

Figure 29-32.Clock Rate Transition Occurs while the SMC is Performing a Write Operation

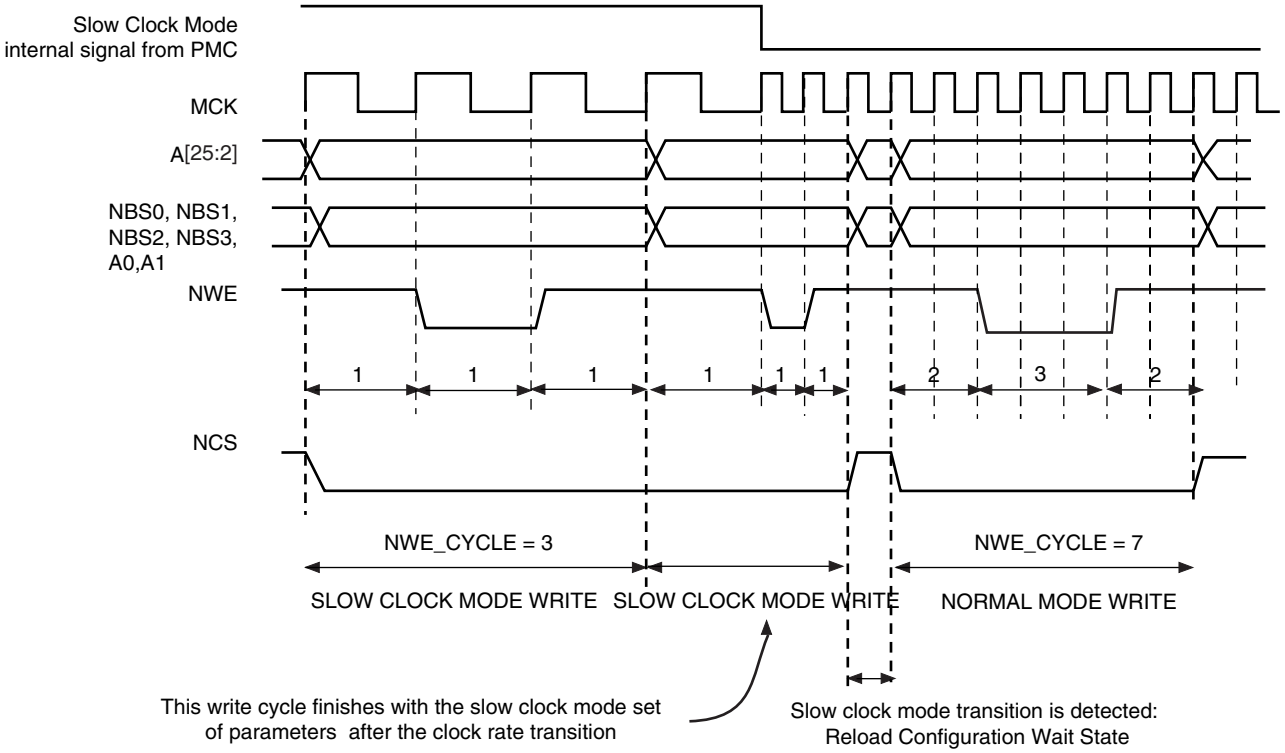
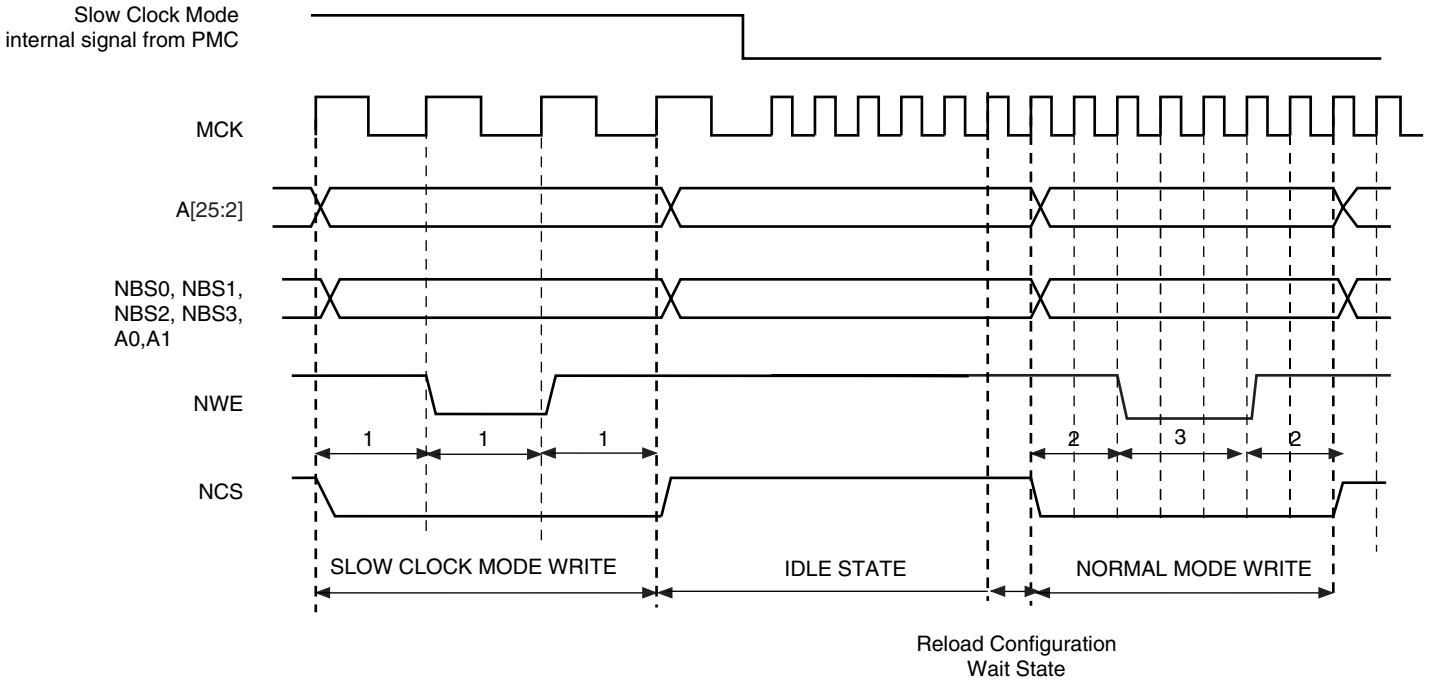


Figure 29-33.Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



29.14 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 29-6.

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 29-34. 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 29-6. 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]

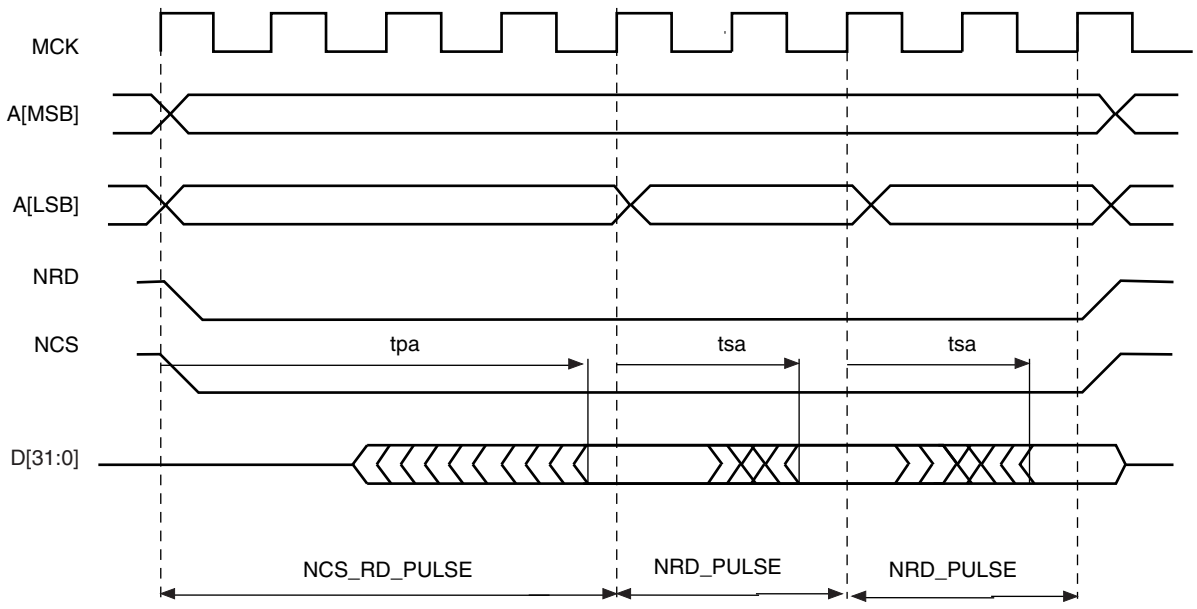
Notes: 1. A denotes the address bus of the memory device

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

29.14.1 Protocol and Timings in Page Mode

Figure 29-34 shows the NRD and NCS timings in page mode access.

Figure 29-34.Page Mode Read Protocol (Address MSB and LSB are defined in Table 29-6)



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 29-7:

Table 29-7. 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}.

29.14.2 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).

29.14.3 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.

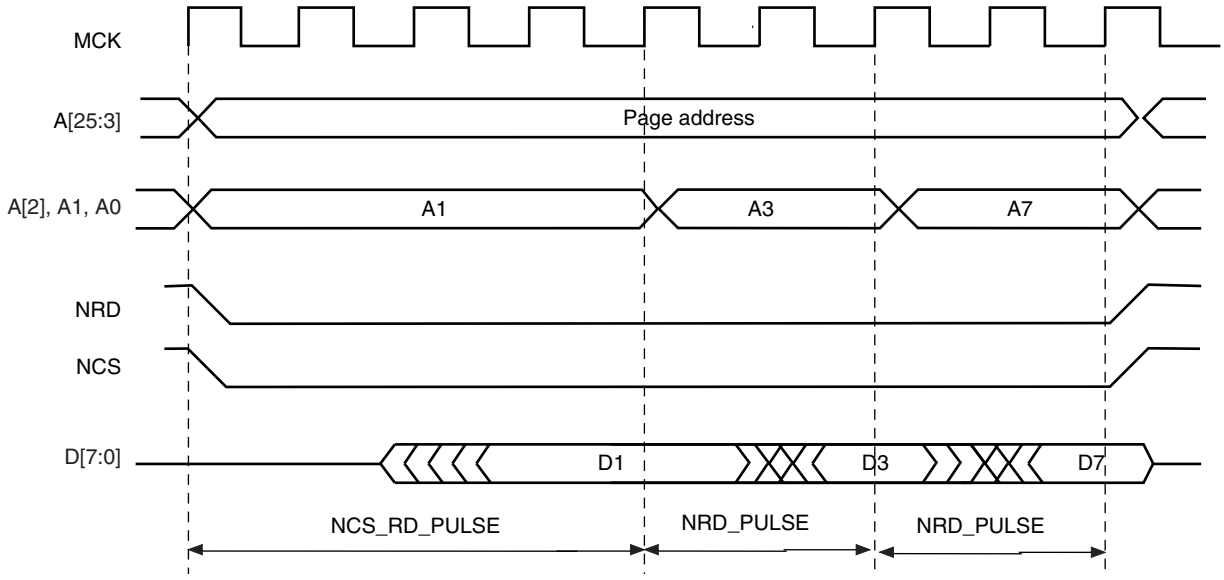
29.14.4 Sequential and Non-sequential Accesses

If the chip select and the MSB of addresses as defined in Table 29-6 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 29-35 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 29-35. Access to Non-sequential Data within the Same Page



29.15 Programmable IO Delays

The external bus interface consists of a data bus, an address bus and control signals. The simultaneous switching outputs on these busses may lead to a peak of current in the internal and external power supply lines.

In order to reduce the peak of current in such cases, additional propagation delays can be adjusted independently for pad buffers by means of configuration registers, SMC_DELAY1-8.

The additional programmable delays for each IO range from 0 to 4 ns (Worst Case PVT). The delay can differ between IOs supporting this feature. Delay can be modified per programming for each IO. The minimal additional delay that can be programmed on a PAD suppporting this feature is 1/16 of the maximum programmable delay.

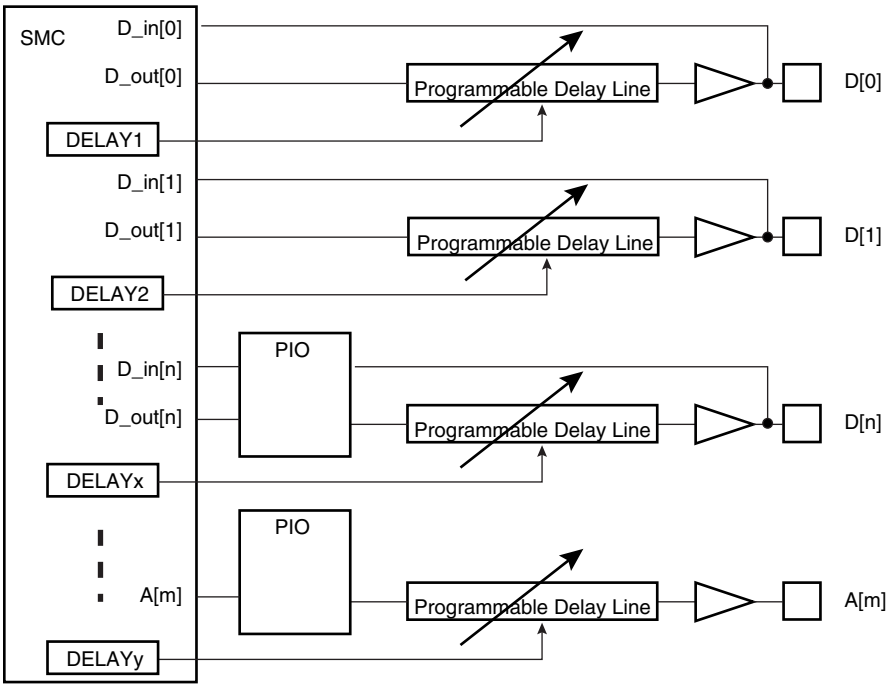
When programming 0x0 in fields “Delay1 to Delay 8”, no delay is added (reset value) and the propagation delay of the pad buffers is the inherent delay of the pad buffer. When programming 0xF in field “Delay1” the propagation delay of the corresponding pad is maximal.

SMC_DELAY1, SMC_DELAY2 allow to configure delay on D[15:0], SMC_DELAY1[3:0] corresponds to D[0] and SMC_DELAY2[3:0] corresponds to D[8].

SMC_DELAY3, SMC_DELAY4 allow to configure delay on D[31:16], SMC_DELAY3[3:0] corresponds to D[16] and SMC_DELAY4[3:0] corresponds to D[24]. In case of multiplexing through the PIO controller, refer to the alternate function of D[31:16].

SMC_DELAY5, 6, 7 and 8 allow to configure delay on A[25:0], SMC_DELAY5[3:0] corresponds to A[0]. In case of multiplexing through the PIO controller, refer to the alternate function of A[25:0].

Figure 29-36.Programmable IO Delays



29.16 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 29-8. For each chip select, a set of 4 registers is used to program the parameters of the external device connected on it. In Table 29-8, “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 29-8. 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
0xC0	SMC Delay on I/O	SMC_DELAY1	Read-write	0x00000000
0xC4	SMC Delay on I/O	SMC_DELAY2	Read-write	0x00000000
0xC8	SMC Delay on I/O	SMC_DELAY3	Read-write	0x00000000
0xCC	SMC Delay on I/O	SMC_DELAY4	Read-write	0x00000000
0xD0	SMC Delay on I/O	SMC_DELAY5	Read-write	0x00000000
0xD4	SMC Delay on I/O	SMC_DELAY6	Read-write	0x00000000
0xD8	SMC Delay on I/O	SMC_DELAY7	Read-write	0x00000000
0xDC	SMC Delay on I/O	SMC_DELAY8	Read-write	0x00000000
0xE4	SMC Write Protect Mode Register	SMC_WPMR	Read-write	0x00000000
0xE8	SMC Write Protect Status Register	SMC_WPSR	Read-only	0x00000000
0xEC-0xFC	Reserved	-	-	-

29.16.1 SMC Setup Register

Name: SMC_SETUP[0..5]

Address: 0xFFFFEA00 [0], 0xFFFFEA10 [1], 0xFFFFEA20 [2], 0xFFFFEA30 [3], 0xFFFFEA40 [4], 0xFFFFEA50 [5]

Access: Read-write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

- **NWE_SETUP: NWE Setup Length**

The NWE signal setup length is defined as:

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

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

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

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

- **NRD_SETUP: NRD Setup Length**

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

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

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

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

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

29.16.2 SMC Pulse Register

Name: SMC_PULSE[0..5]

Address: 0xFFFFEA04 [0], 0xFFFFEA14 [1], 0xFFFFEA24 [2], 0xFFFFEA34 [3], 0xFFFFEA44 [4], 0xFFFFEA54 [5]

Access: Read-write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

- **NWE_PULSE: NWE Pulse Length**

The NWE signal pulse length is defined as:

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

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

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

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

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

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

- **NRD_PULSE: NRD Pulse Length**

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

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

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

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

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

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

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

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

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

29.16.3 SMC Cycle Register

Name: SMC_CYCLE[0..5]

Address: 0xFFFFEA08 [0], 0xFFFFEA18 [1], 0xFFFFEA28 [2], 0xFFFFEA38 [3], 0xFFFFEA48 [4], 0xFFFFEA58 [5]

Access: Read-write

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

23	22	21	20	19	18	17	16
NRD_CYCLE							

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

7	6	5	4	3	2	1	0
NWE_CYCLE							

- **NWE_CYCLE: Total Write Cycle Length**

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

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

- **NRD_CYCLE: Total Read Cycle Length**

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

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

29.16.4 SMC MODE Register

Name: SMC_MODE[0..5]

Address: 0xFFFFEA0C [0], 0xFFFFEA1C [1], 0xFFFFEA2C [2], 0xFFFFEA3C [3], 0xFFFFEA4C [4], 0xFFFFEA5C [5]

Access: Read-write

31	30	29	28	27	26	25	24
–	–	PS		–	–	–	PMEN
23	22	21	20	19	18	17	16
–	–	–	TDF_MODE	TDF_CYCLES			
15	14	13	12	11	10	9	8
–	–	DBW		–	–	–	BAT
7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

- **READ_MODE:**

1: The read operation is controlled by the NRD signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NRD.

0: The read operation is controlled by the NCS signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NCS.

- **WRITE_MODE**

1: The write operation is controlled by the NWE signal.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NWE.

0: The write operation is controlled by the NCS signal.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NCS.

- **EXNW_MODE: NWAIT Mode**

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNW_MODE		NWAIT Mode
0	0	Disabled
0	1	Reserved
1	0	Frozen Mode
1	1	Ready Mode

- Disabled Mode: The NWAIT input signal is ignored on the corresponding Chip Select.
- Frozen Mode: If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.
- Ready Mode: The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

- **BAT: Byte Access Type**

This field is used only if DBW defines a 16- or 32-bit data bus.

- 1: Byte write access type:
  - Write operation is controlled using NCS, NWR0, NWR1, NWR2, NWR3.
  - Read operation is controlled using NCS and NRD.
- 0: Byte select access type:
  - Write operation is controlled using NCS, NWE, NBS0, NBS1, NBS2 and NBS3
  - Read operation is controlled using NCS, NRD, NBS0, NBS1, NBS2 and NBS3

- **DBW: Data Bus Width**

DBW		Data Bus Width
0	0	8-bit bus
0	1	16-bit bus
1	0	32-bit bus
1	1	Reserved

- **TDF_CYCLES: Data Float Time**

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

- **TDF_MODE: TDF Optimization**

1: TDF optimization is enabled.

- The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization is disabled.

- The number of TDF wait states is inserted before the next access begins.

- **PMEN: Page Mode Enabled**

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

0: Standard read is applied.

- **PS: Page Size**

If page mode is enabled, this field indicates the size of the page in bytes.

PS		Page Size
0	0	4-byte page
0	1	8-byte page
1	0	16-byte page
1	1	32-byte page

29.16.5 SMC DELAY I/O Register

Name: SMC_DELAY 1-8

Address: 0xFFFFEAC0 [1] .. 0xFFFFEADC [8]

Access: Read-write

Reset: See Table 29-8

31	30	29	28	27	26	25	24
Delay8				Delay7			
23	22	21	20	19	18	17	16
Delay6				Delay5			
15	14	13	12	11	10	9	8
Delay4				Delay3			
7	6	5	4	3	2	1	0
Delay2				Delay1			

- **Delay x:**

Gives the number of elements in the delay line.

29.16.6 SMC Write Protect Mode Register

Name: SMC_WPMR

Address: 0xFFFFEAE4

Access: Read-write

Reset: See Table 29-8

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x534D43 (“SMC” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x534D43 (“SMC” in ASCII).

Protects the registers listed below:

- Section 29.16.1 ”SMC Setup Register”
- Section 29.16.2 ”SMC Pulse Register”
- Section 29.16.3 ”SMC Cycle Register”
- Section 29.16.4 ”SMC MODE Register”
- Section 29.16.5 ”SMC DELAY I/O Register”

- **WPKEY: Write Protect KEY**

Should be written at value 0x534D43 (“SMC” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

29.16.7 SMC Write Protect Status Register

Name: SMC_WPSR

Address: 0xFFFFEAE8

Access: Read-only

Reset: See Table 29-8

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

- **WPVS: Write Protect Enable**

0 = No Write Protect Violation has occurred since the last read of the SMC_WPSR register.

1 = A Write Protect Violation occurred since the last read of the SMC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading SMC_WPSR automatically clears all fields.

30. DDR SDR SDRAM Controller (DDRSDRC)

30.1 Description

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

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

The DDRSDRC supports a read or write burst length of 8 locations which frees the command and address bus to anticipate the next command, thus reducing latency imposed by the SDRAM protocol and improving the SDRAM bandwidth. Moreover i**t keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.** The DDRSDRC supports a CAS latency of 2 or 3 and optimizes the read access depending on the frequency.

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

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

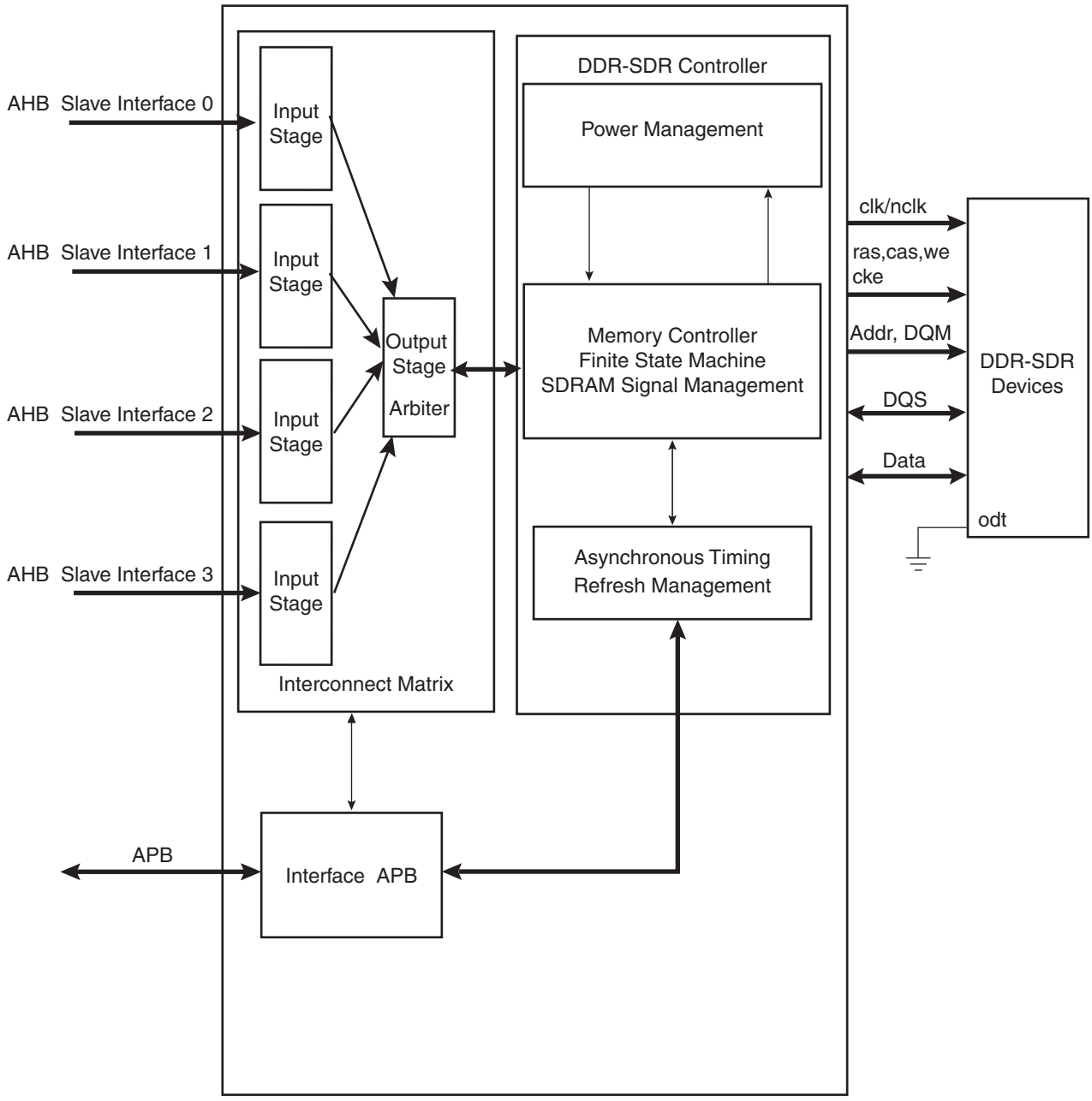
Note: The term “SDRAM device” regroups SDR-SDRAM, Low-power SDR-SDRAM, Low-power DDR1-SDRAM and DDR2-SDRAM devices.

30.2 Embedded Characteristics

- AMBA Compliant Interface, interfaces Directly to the ARM Advanced High performance Bus (AHB)
  - Four AHB Interfaces, Management of All Accesses Maximizes Memory Bandwidth and Minimizes Transaction Latency
  - AHB Transfer: Word, Half Word, Byte Access
- Supports DDR2-SDRAM, Low-power DDR1-SDRAM, SDR-SDRAM and Low-power SDR-SDRAM
- Numerous Configurations Supported
  - **2K, 4K, 8K, 16K Row Address Memory Parts**
  - **SDRAM with Four and Eight Internal Banks**
  - **SDR-SDRAM with 16- or 32-bit Data Path**
  - DDR-SDRAM with **16-bit Data Path**
  - One Chip Select for SDRAM Device (256 Mbyte Address Space)
- Programming Facilities
  - **Multibank Ping-pong Access (up to 4 banks or 8 banks opened at the same time = Reduces Average Latency of Transactions)**
  - **Timing Parameters Specified by Software**
  - **Automatic Refresh Operation, Refresh Rate is Programmable**
  - Automatic Update of DS, TCR and PASR Parameters (Low-power SDRAM Devices)
- **Energy-saving Capabilities**
  - **Self-refresh, Power-down, Active Power-down and Deep Power-down Modes Supported**
- **SDRAM Power-up Initialization by Software**
- **CAS Latency of 2, 3 Supported**
- Reset Function Supported (DDR2-SDRAM)
- ODT (On-die Termination) Not Supported
- **Auto Precharge Command Not Used**
- SDR-SDRAM with 16-bit Datapath and Eight Columns Not Supported
- DDR2-SDRAM with Eight **Internal Banks Supported**
- Linear and Interleaved Decoding Supported
- SDR-SDRAM or Low-power DDR1-SDRAM with 2 Internal Banks Not Supported
- Clock Frequency Change in Precharge Power-down Mode Not Supported
- OCD (Off-chip Driver) Mode Not Supported

30.3 DDRSDRC Module Diagram

Figure 30-1. DDRSDRC Module Diagram



DDRSDRC is partitioned in two blocks (see Figure 30-1):

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

30.4 Initialization Sequence

The addresses given are for example purposes only. The real address depends on implementation in the product.

30.4.1 SDR-SDRAM Initialization

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

1. Program the memory device type into the Memory Device Register (see Section 30.7.8 on page 468).
2. Program the features of the SDR-SDRAM device into the Timing Register (asynchronous timing (trc, tras, etc.)), and into the Configuration Register (number of columns, rows, banks, cas latency) (see Section 30.7.3 on page 459, Section 30.7.4 on page 462 and Section 30.7.5 on page 464).
3. For low-power SDRAM, temperature-compensated self refresh (TCSR), drive strength (DS) and partial array self refresh (PASR) must be set in the Low-power Register (see Section 30.7.7 on page 466).

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

4. A NOP command is issued to the SDR-SDRAM. Program NOP command into Mode Register, the application must set Mode to 1 in the Mode Register (See Section 30.7.1 on page 457). Perform a write access to any SDR-SDRAM address to acknowledge this command. Now the clock which drives SDR-SDRAM device is enabled.
5. An all banks precharge command is issued to the SDR-SDRAM. Program all banks precharge command into Mode Register, the application must set Mode to 2 in the Mode Register (See Section 30.7.1 on page 457). Perform a write access to any SDR-SDRAM address to acknowledge this command.
6. Eight auto-refresh (CBR) cycles are provided. Program the auto refresh command (CBR) into Mode Register, the application must set Mode to 4 in the Mode Register (see Section 30.7.1 on page 457).Performs a write access to any SDR-SDRAM location eight times to acknowledge these commands.
7. A Mode Register set (MRS) cycle is issued to program the parameters of the SDR-SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.

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

8. For low-power SDR-SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDR-SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is 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 0x20800000.
9. The application must go into Normal Mode, setting Mode to 0 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access at any location in the SDRAM to acknowledge this command.
10. Write the refresh rate into the count field in the DDRSDRC Refresh Timer register (see page 458). (Refresh rate = delay between refresh cycles). The SDR-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the refresh timer count register must to be set with (15.625*100 MHz) = 1562 i.e. 0x061A or (7.81*100 MHz) = 781 i.e. 0x030d

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

30.4.2 Low-power DDR1-SDRAM Initialization

The initialization sequence is generated by software. The low-power DDR1-SDRAM devices are initialized by the following sequence:

1. Program the memory device type into the Memory Device Register (see Section 30.7.8 on page 468).
2. Program the features of the low-power DDR1-SDRAM device into the Configuration Register: asynchronous timing (trc, tras, etc.), number of columns, rows, banks, cas latency. See Section 30.7.3 on page 459, Section 30.7.4 on page 462 and Section 30.7.5 on page 464.
3. Program temperature compensated self refresh (tcr), Partial array self refresh (pasr) and Drive strength (ds) into the Low-power Register. See Section 30.7.7 on page 466.

4. An NOP command will be issued to the low-power DDR1-SDRAM. Program NOP command into the Mode Register, the application must set Mode to 1 in the Mode Register (see Section 30.7.1 on page 457). Perform a write access to any DDR1-SDRAM address to acknowledge this command. Now clocks which drive DDR1-SDRAM device are enabled.

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

5. An all banks precharge command is issued to the low-power DDR1-SDRAM. Program all banks precharge command into the Mode Register, the application must set Mode to 2 in the Mode Register (See Section 30.7.1 on page 457). Perform a write access to any low-power DDR1-SDRAM address to acknowledge this command
6. Two auto-refresh (CBR) cycles are provided. Program the auto refresh command (CBR) into the Mode Register, the application must set Mode to 4 in the Mode Register (see Section 30.7.1 on page 457). Perform a write access to any low-power DDR1-SDRAM location twice to acknowledge these commands.
7. An Extended Mode Register set (EMRS) cycle is issued to program the low-power DDR1-SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 BA[0] is set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the low-power DDR1-SDRAM write access should be done at address 0x20800000.

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

8. A Mode Register set (MRS) cycle is issued to program the parameters of the low-power DDR1-SDRAM devices, in particular CAS latency, burst length. The application must set Mode to 3 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the low-power DDR1-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] bits are set to 0. For example, with a 16-bit 128 MB low-power DDR1-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000. The application must go into Normal Mode, setting Mode to 0 in the Mode Register (see Section 30.7.1 on page 457) and performing a write access at any location in the low-power DDR1-SDRAM to acknowledge this command.
9. Perform a write access to any low-power DDR1-SDRAM address.
10. Write the refresh rate into the count field in the DDRSDRC Refresh Timer register (see page 458). (Refresh rate = delay between refresh cycles). The low-power DDR1-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the refresh timer count register must to be set with (15.625*100 MHz) = 1562 i.e. 0x061A or (7.81*100 MHz) = 781 i.e. 0x030d
11. After initialization, the low-power DDR1-SDRAM device is fully functional.

30.4.3 DDR2-SDRAM Initialization

The initialization sequence is generated by software. The DDR2-SDRAM devices are initialized by the following sequence:

1. Program the memory device type into the Memory Device Register (see Section 30.7.8 on page 468).
2. Program the features of DDR2-SDRAM device into the Timing Register (asynchronous timing (trc, tras, etc.)), and into the Configuration Register (number of columns, rows, banks, cas latency and output drive strength) (see Section 30.7.3 on page 459, Section 30.7.4 on page 462 and Section 30.7.5 on page 464).
3. An NOP command is issued to the DDR2-SDRAM. Program the NOP command into the Mode Register, the application must set Mode to 1 in the Mode Register (see Section 30.7.1 on page 457). Perform a write access to any DDR2-SDRAM address to acknowledge this command. Now clocks which drive DDR2-SDRAM device are enabled.

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

4. An NOP command is issued to the DDR2-SDRAM. Program the NOP command into the Mode Register, the application must set Mode to 1 in the Mode Register (see Section 30.7.1 on page 457). Perform a write access to any DDR2-SDRAM address to acknowledge this command. Now CKE is driven high.
5. An all banks precharge command is issued to the DDR2-SDRAM. Program all banks precharge command into the Mode Register, the application must set Mode to 2 in the Mode Register (See Section 30.7.1 on page 457). Perform a write access to any DDR2-SDRAM address to acknowledge this command

6. An Extended Mode Register set (EMRS2) cycle is issued to chose between commercial or high temperature operations. The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is set to 0. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be done at the address 0x20800000.

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

7. An Extended Mode Register set (EMRS3) cycle is issued to set the Extended Mode Register to “0”. The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be done at the address 0x20C00000.
8. An Extended Mode Register set (EMRS1) cycle is issued to enable DLL. The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 0 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be done at the address 0x20400000.

An additional 200 cycles of clock are required for locking DLL

9. Program DLL field into the Configuration Register (see Section 30.7.3 on page 459) to high (Enable DLL reset).
10. A Mode Register set (MRS) cycle is issued to reset DLL. The application must set Mode to 3 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] bits are set to 0. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.
11. An all banks precharge command is issued to the DDR2-SDRAM. Program all banks precharge command into the Mode Register, the application must set Mode to 2 in the Mode Register (See Section 30.7.1 on page 457). Perform a write access to any DDR2-SDRAM address to acknowledge this command
12. Two auto-refresh (CBR) cycles are provided. Program the auto refresh command (CBR) into the Mode Register, the application must set Mode to 4 in the Mode Register (see Section 30.7.1 on page 457). Performs a write access to any DDR2-SDRAM location twice to acknowledge these commands.
13. Program DLL field into the Configuration Register (see Section 30.7.3 on page 459) to low (Disable DLL reset).
14. A Mode Register set (MRS) cycle is issued to program the parameters of the DDR2-SDRAM devices, in particular CAS latency, burst length and to disable DLL reset. The application must set Mode to 3 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000
15. Program OCD field into the Configuration Register (see Section 30.7.3 on page 459) to high (OCD calibration default).
16. An Extended Mode Register set (EMRS1) cycle is issued to OCD default value. The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 0 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be done at the address 0x20400000.
17. Program OCD field into the Configuration Register (see Section 30.7.3 on page 459) to low (OCD calibration mode exit).
18. An Extended Mode Register set (EMRS1) cycle is issued to enable OCD exit. The application must set Mode to 5 in the Mode Register (see Section 30.7.1 on page 457) and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 0 and BA[0] is set to 1. For example, with a 16-bit 128 MB DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be done at the address 0x20400000.

19. A mode Normal command is provided. Program the Normal mode into Mode Register (see Section 30.7.1 on page 457). Perform a write access to any DDR2-SDRAM address to acknowledge this command.
20. Perform a write access to any DDR2-SDRAM address.
21. Write the refresh rate into the count field in the Refresh Timer register (see page 458). (Refresh rate = delay between refresh cycles). The DDR2-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 133 MHz frequency, the refresh timer count register must to be set with (15.625*133 MHz) = 2079 i.e. 0x081f or (7.81*133 MHz) = 1039 i.e. 0x040f.

After initialization, the DDR2-SDRAM devices are fully functional.

30.5 Functional Description

30.5.1 SDRAM Controller Write Cycle

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

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

To initiate a single access, the DDRSDRC checks if the page access is already open. If row/bank addresses match with the previous row/bank addresses, the controller generates a write command. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t RP) commands and active/write (t RCD) command. As the burst length is fixed to 8, in the case of single access, it has to stop the burst, otherwise seven invalid values may be written. In the case of SDR-SDRAM devices, a Burst Stop command is generated to interrupt the write operation. In the case of DDR-SDRAM devices, Burst Stop command is not supported for the burst write operation. In order to then interrupt the write operation, Dm must be set to 1 to mask invalid data (see Figure 30-2 on page 436 and Figure 30-5 on page 437) and DQS must continue to toggle.

To initiate a burst access, the DDRSDRC uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write non-sequential access, then an automatic access break is inserted, the DDRSDRC generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (tRP) commands and active/write (tRCD) commands.

For a definition of timing parameters, refer to Section 30.7.4 “DDRSDRC Timing Parameter 0 Register” on page 462.

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

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

Figure 30-2. Single Write Access, Row Closed, Low-power DDR1-SDRAM Device

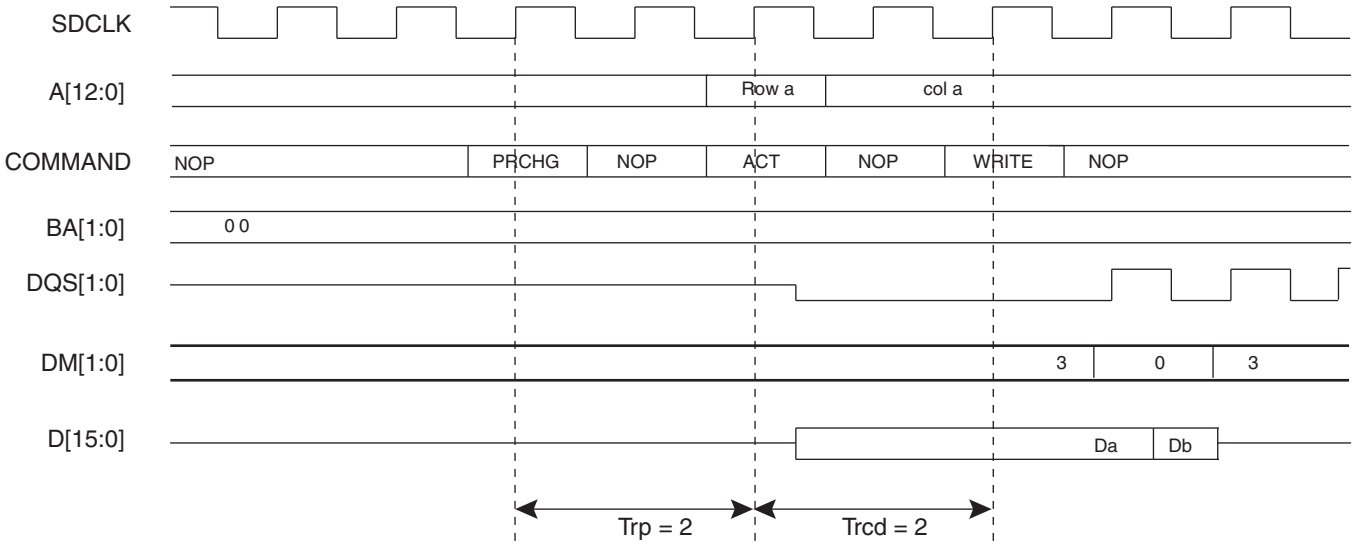


Figure 30-3. Single Write Access, Row Closed, DDR2-SDRAM Device

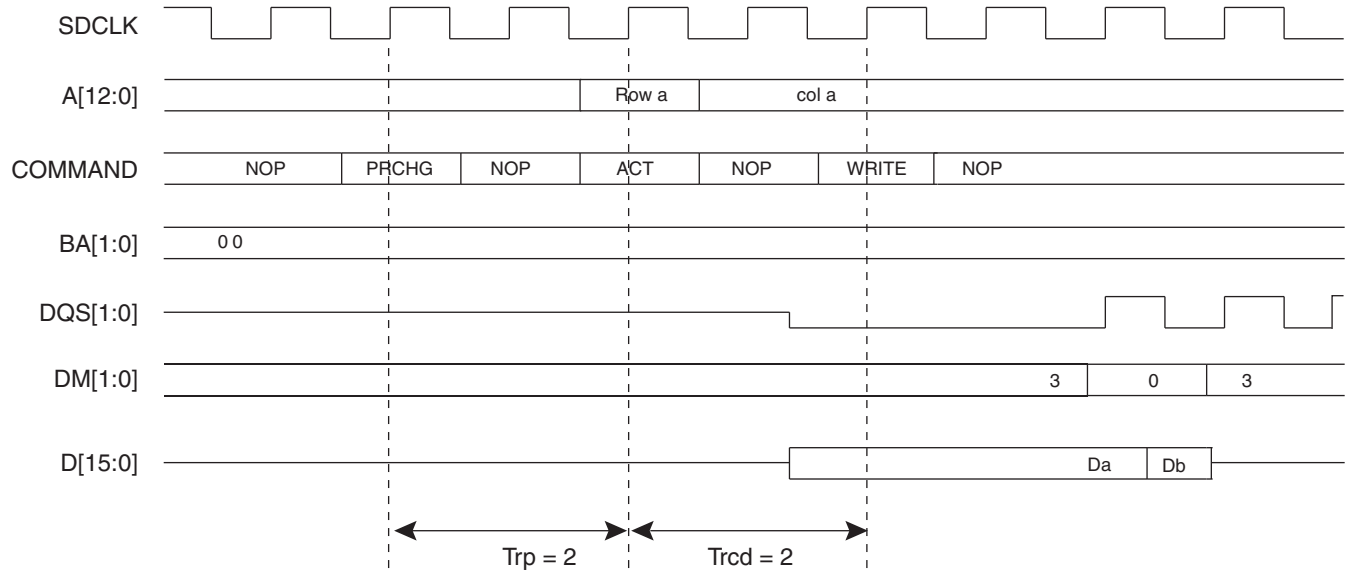


Figure 30-4. Single Write Access, Row Closed, SDR-SDRAM Device

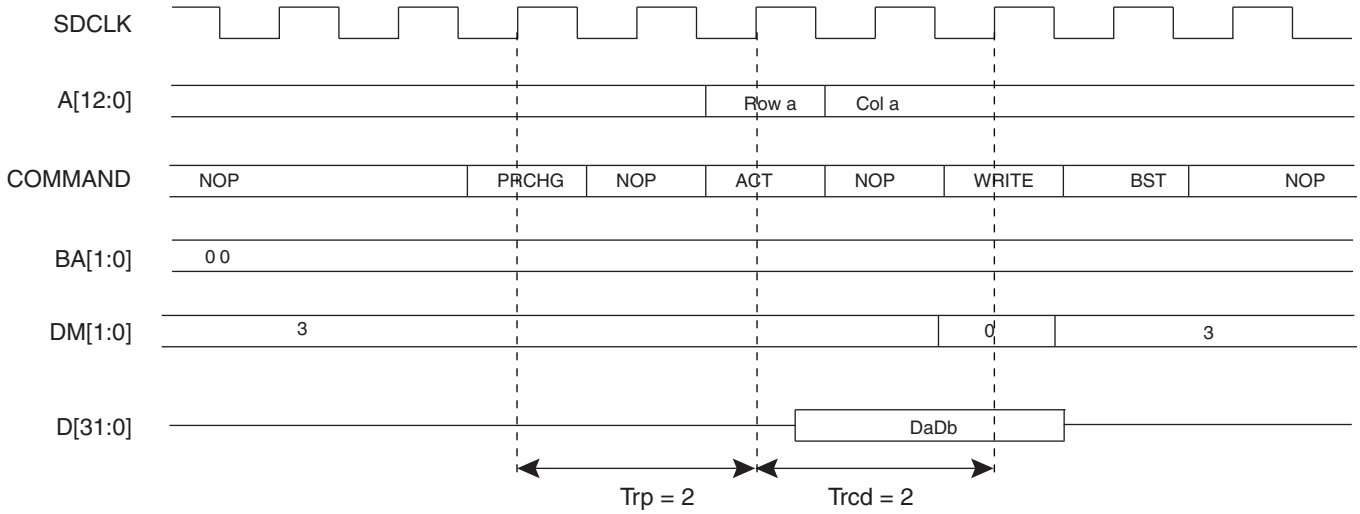


Figure 30-5. Burst Write Access, Row Closed, Low-power DDR1-SDRAM Device

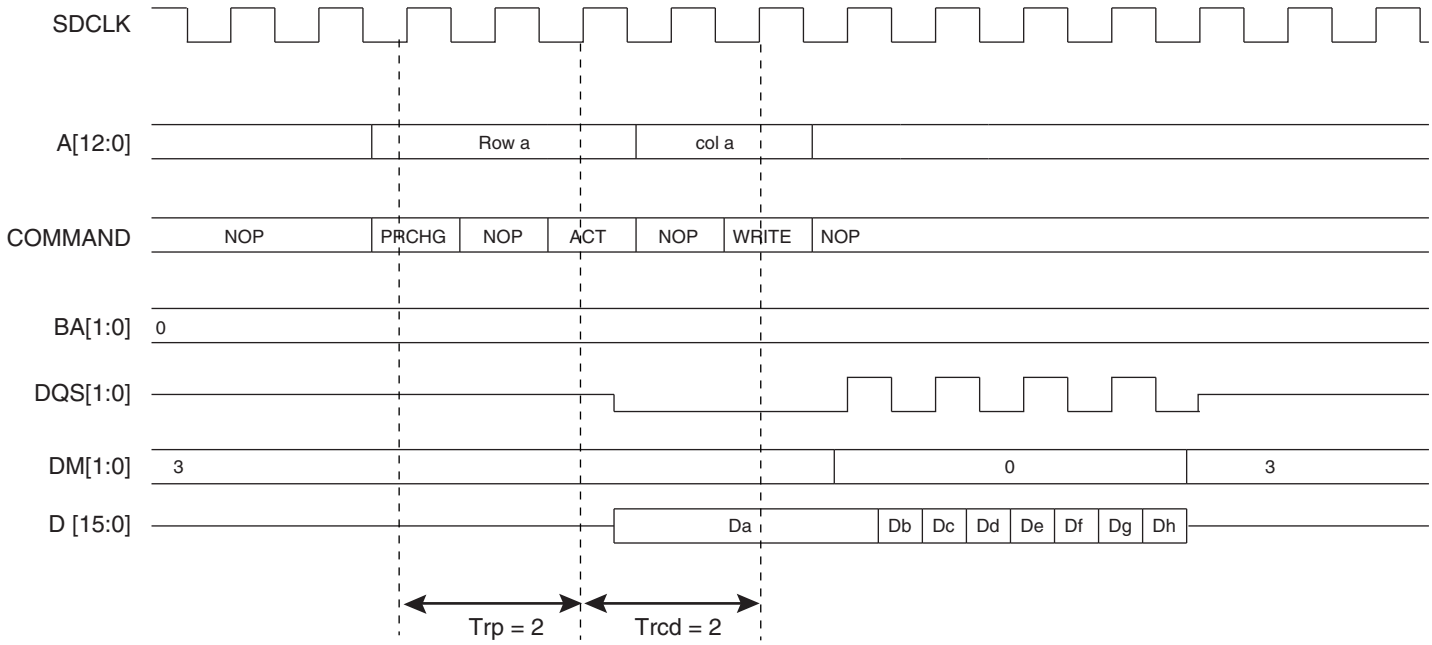


Figure 30-6. Burst Write Access, Row Closed, DDR2-SDRAM Device

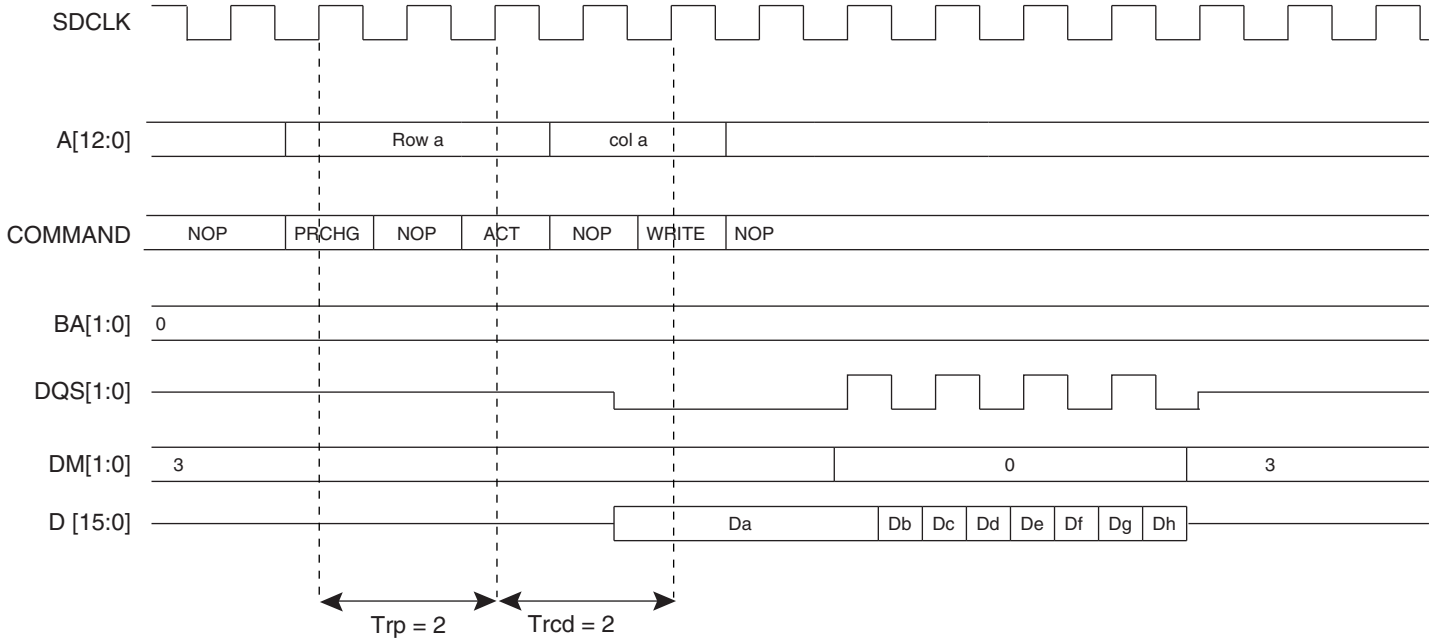
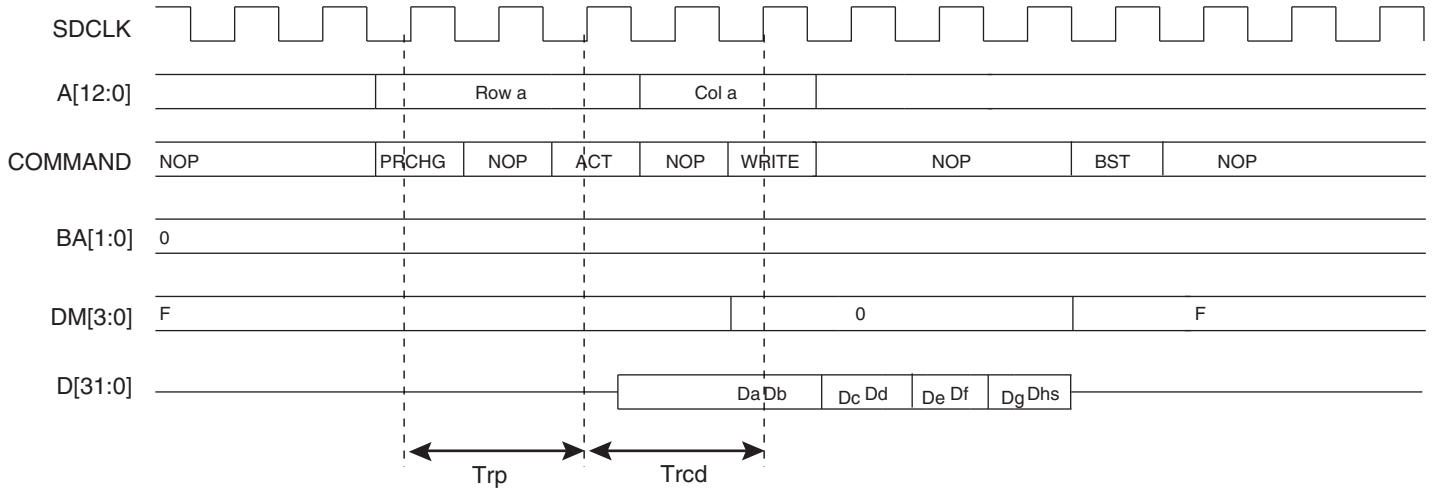
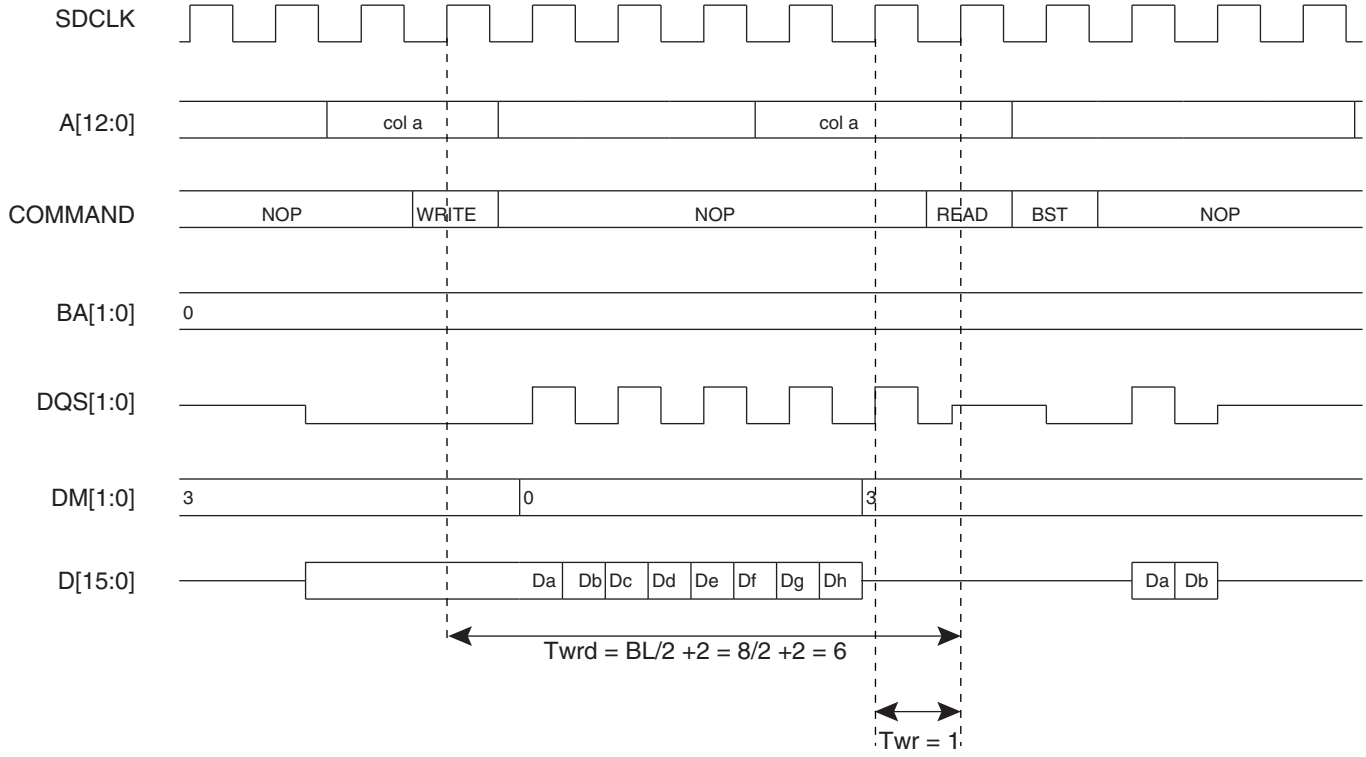


Figure 30-7. Burst Write Access, Row Closed, SDR-SDRAM Device



A write command can be followed by a read command. To avoid breaking the current write burst, Twtr/Twrd (bl/2 + 2 = 6 cycles) should be met. See Figure 30-8 on page 439.

Figure 30-8. Write Command Followed By a Read Command without Burst Write Interrupt, Low-power DDR1-SDRAM Device



In the case of a single write access, write operation should be interrupted by a read access but DM must be input 1 cycle prior to the read command to avoid writing invalid data. See Figure 30-9 on page 439.

Figure 30-9. Single Write Access Followed By A Read Access Low-power DDR1-SDRAM Devices

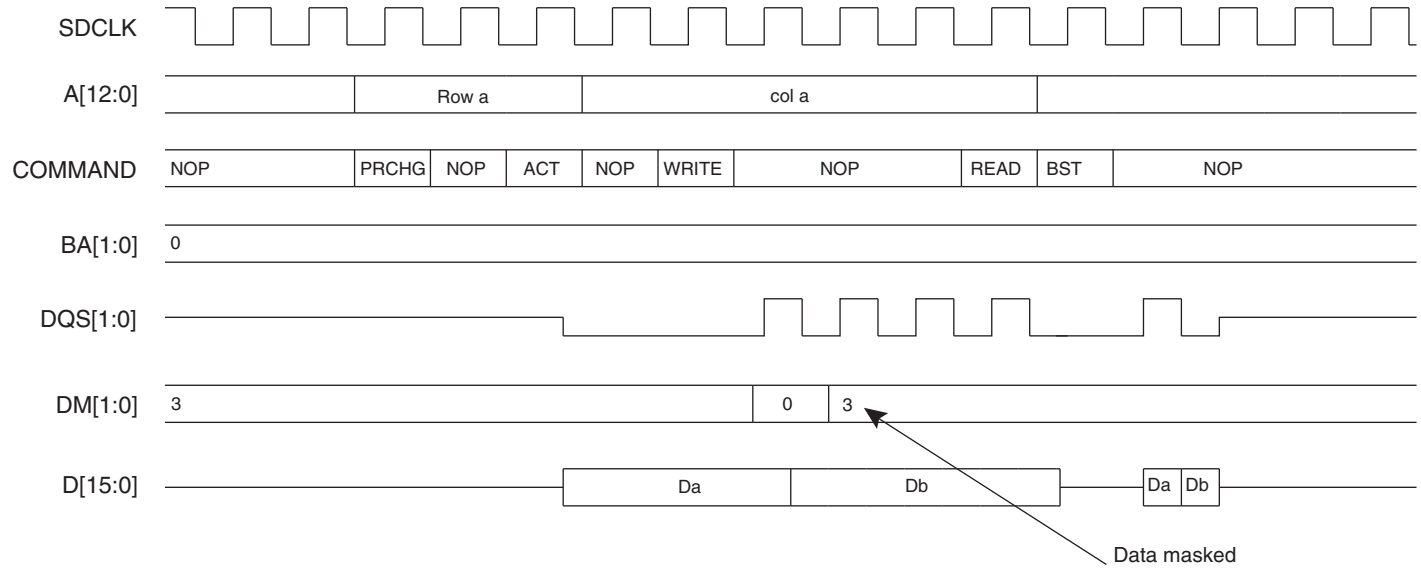
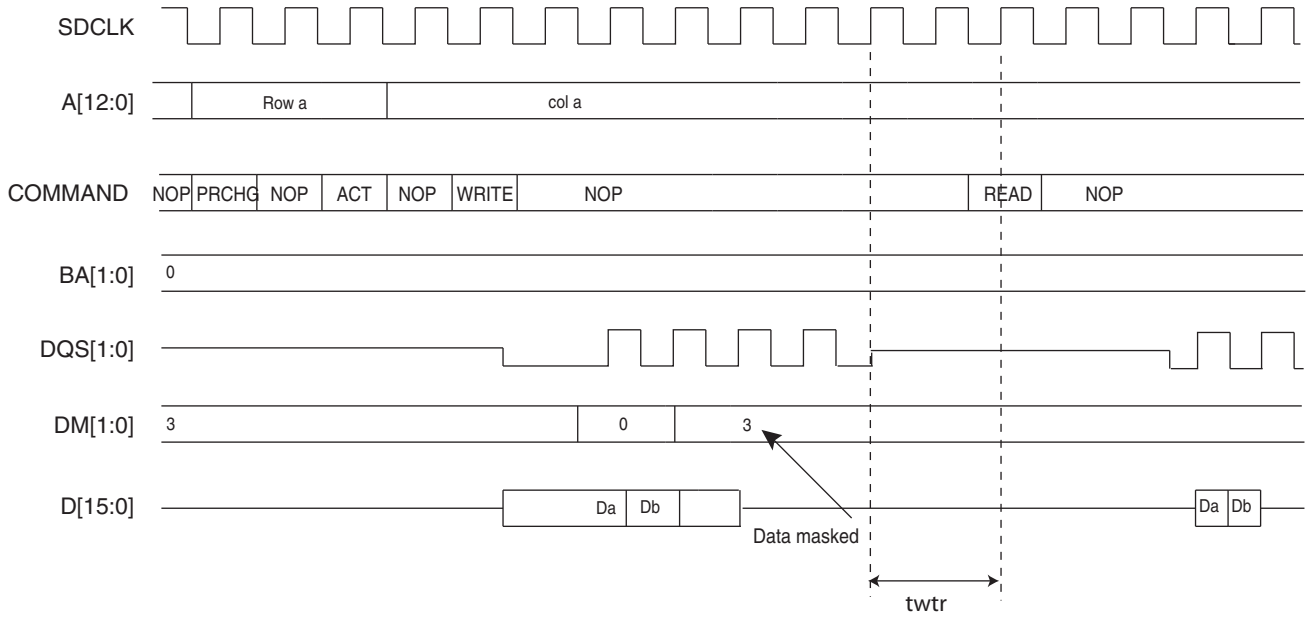


Figure 30-10.SINGLE Write Access Followed By A Read Access, DDR2 -SDRAM Device



30.5.2 SDRAM Controller Read Cycle

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

The SDRAM devices are programmed with a burst length equal to 8 which determines the length of a sequential data output by the read command that is set to 8. The latency from read command to data output is equal to 2 or 3. This value is programmed during the initialization phase (see Section 30.4.1 “SDR-SDRAM Initialization” on page 432).

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

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

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

For a definition of timing parameters, refer to Section 30.7.3 “DDRSDRC Configuration Register” on page 459.

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

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

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

Figure 30-11.Single Read Access, Row Close, Latency = 2,Low-power DDR1-SDRAM Device

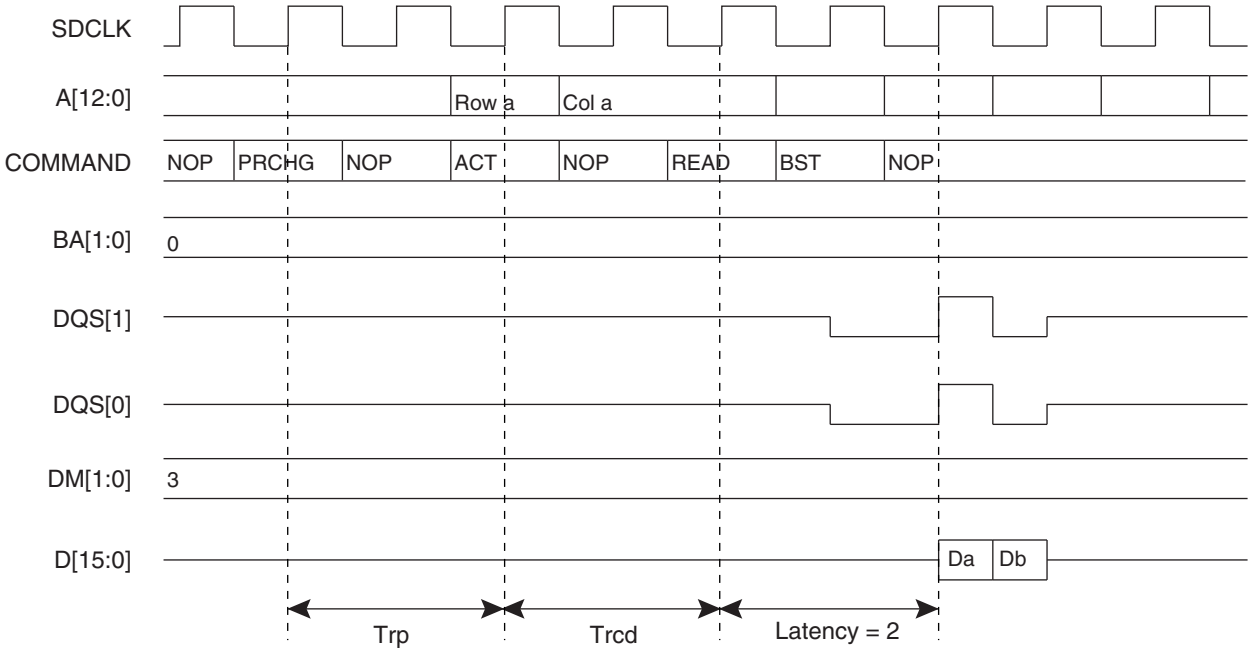


Figure 30-12.Single Read Access, Row Close, Latency = 3, DDR2-SDRAM Device

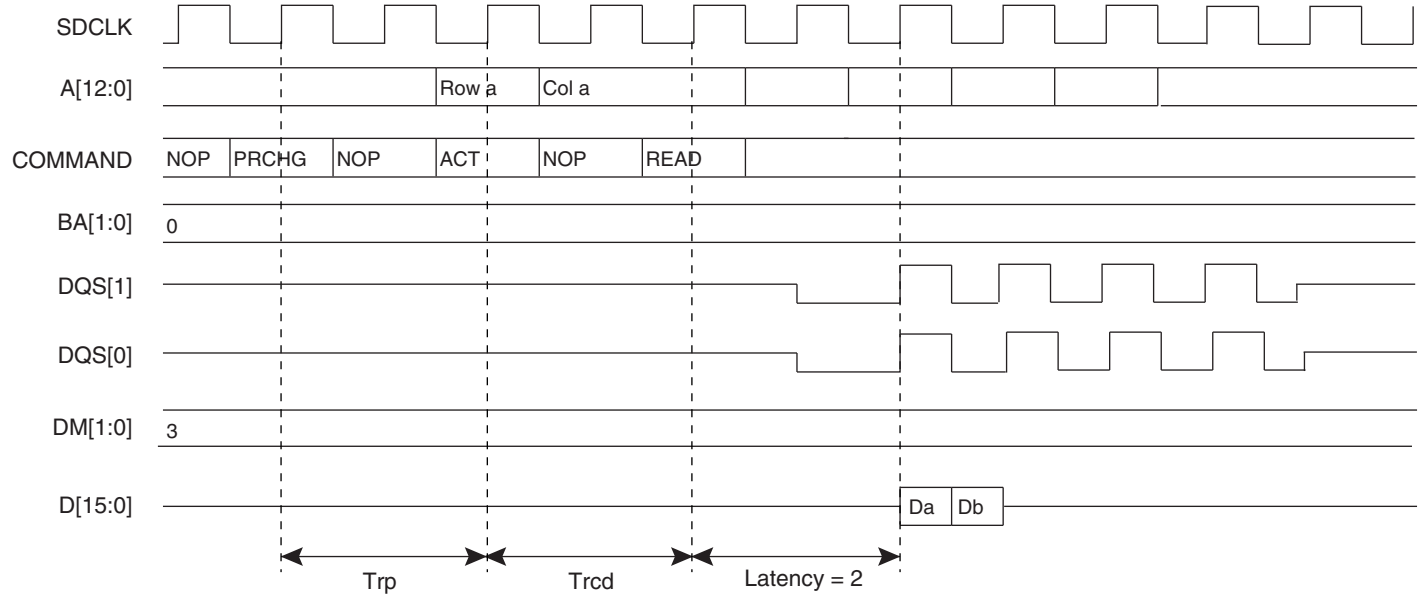


Figure 30-13.Single Read Access, Row Close, Latency = 2, SDR-SDRAM Device

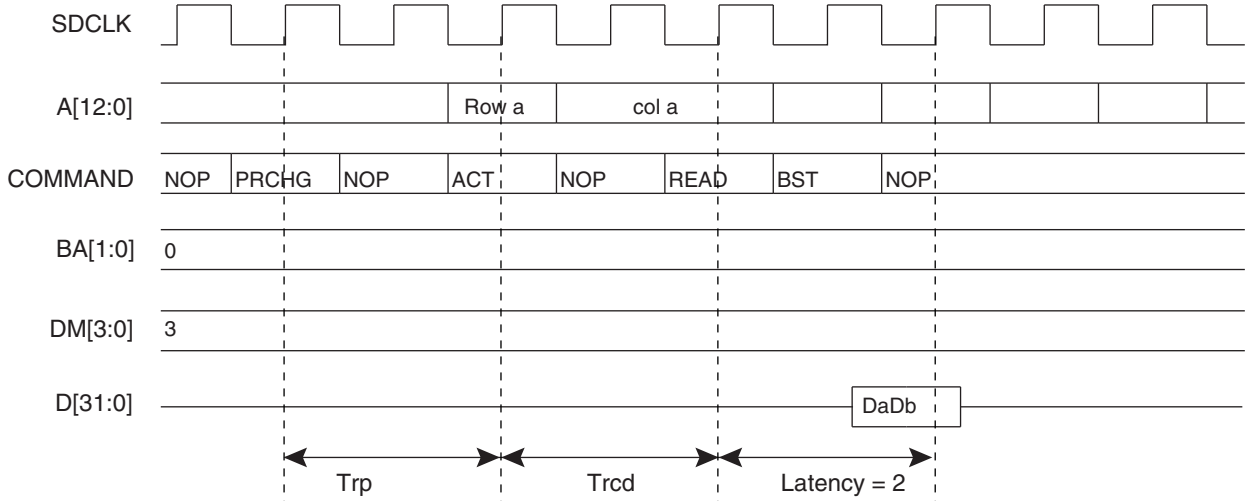


Figure 30-14.Burst Read Access, Latency = 2, Low-power DDR1-SDRAM Devices

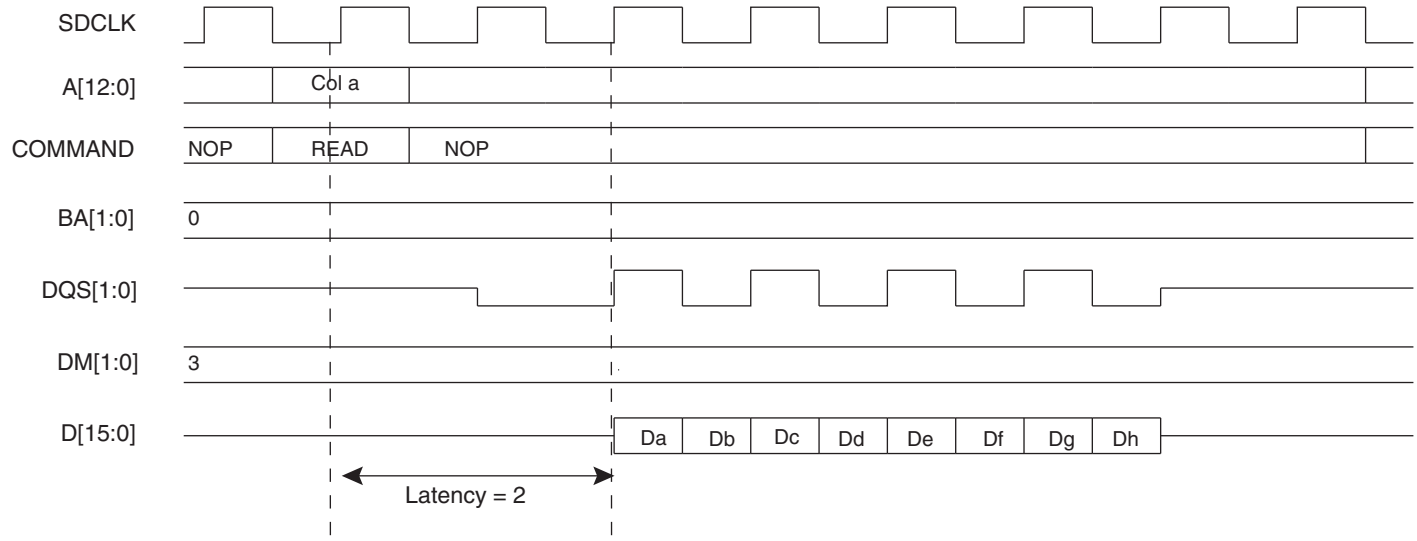


Figure 30-15.Burst Read Access, Latency = 3, DDR2-SDRAM Devices

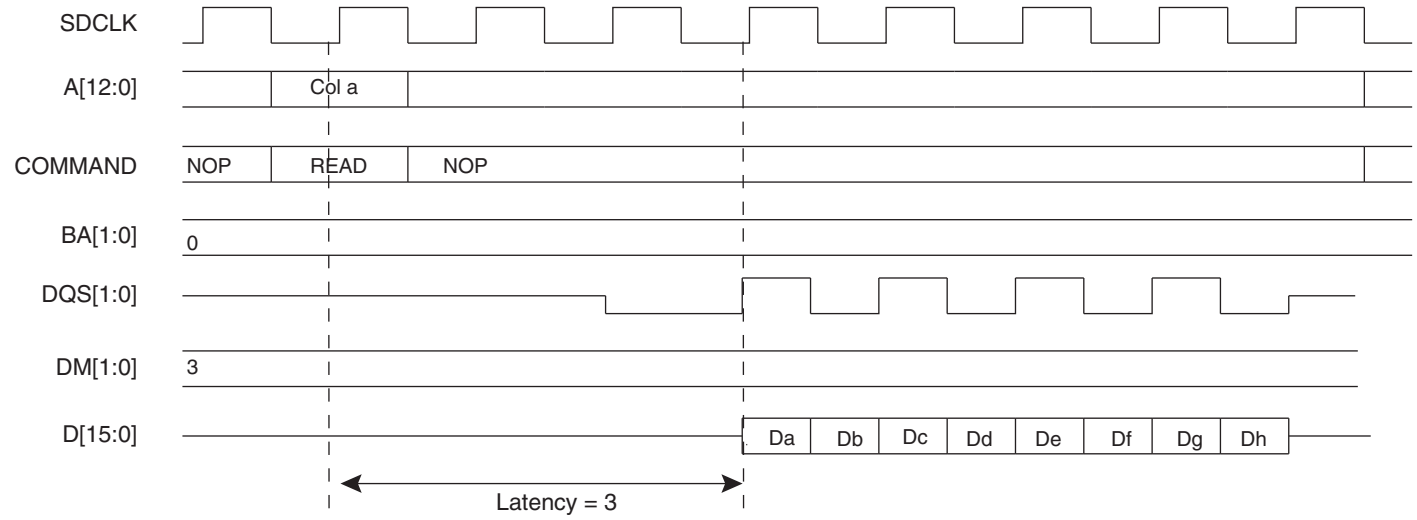
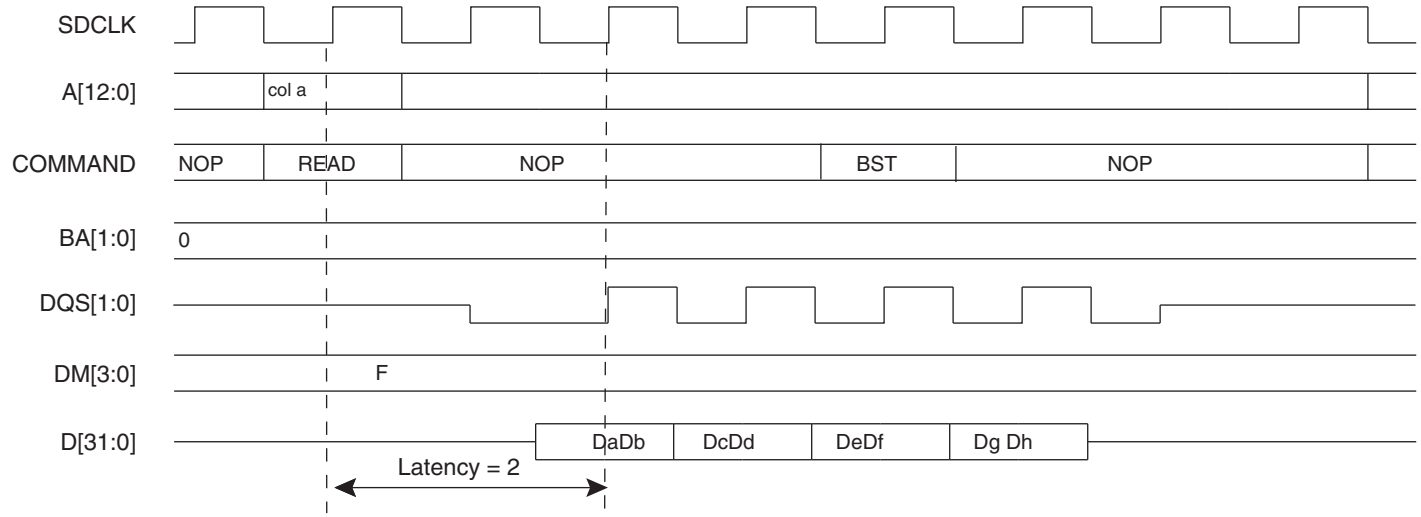


Figure 30-16.Burst Read Access, Latency = 2, SDR-SDRAM Devices



30.5.3 Refresh (Auto-refresh Command)

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

30.5.4 Power Management

30.5.4.1 Self Refresh Mode

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

Self refresh mode is used to reduce power consumption, i.e., when no access to the SDRAM device is possible. In this case, power consumption is very low. In self refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except CKE, which remains low. As soon as the SDRAM device is selected, the DDRSDRC provides a sequence of commands and exits self refresh mode.

The DDRSDRC re-enables self refresh mode as soon as the SDRAM device is not selected. It is possible to define when self refresh mode will be enabled by setting the register LPR (see Section 30.7.7 “DDRSDRC Low-power Register” on page 466), timeout command bit:

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

As soon as the SDRAM device is no longer selected, PRECHARGE ALL BANKS command is generated followed by a SELF-REFREFSH command. If, between these two commands an SDRAM access is detected, SELF-REFREFSH command will be replaced by an AUTO-REFRESH command. According to the application, more AUTO-REFRESH commands will be performed when the self refresh mode is enabled during the application.

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

After initialization, as soon as PASR/DS/TCSR fields are modified, the Extended Mode Register in the memory of the external device is accessed automatically and PASR/DS/TCSR bits are updated before entry into self refresh mode if DDRSDRC does not share an external bus with another controller or during a refresh command, and a pending read or write access, if DDRSDRC does share an external bus with another controller. This type of update is a function of the UPD_MR bit (see Section 30.7.7 “DDRSDRC Low-power Register” on page 466).

The low-power SDR-SDRAM must remain in self refresh mode for a minimum period of TRAS periods and may remain in self refresh mode for an indefinite period. (See Figure 30-17)

The low-power DDR1-SDRAM must remain in self refresh mode for a minimum of TRFC periods and may remain in self refresh mode for an indefinite period.

The DDR2-SDRAM must remain in self refresh mode for a minimum of TCKE periods and may remain in self refresh mode for an indefinite period.

Figure 30-17.Self Refresh Mode Entry, Timeout = 0

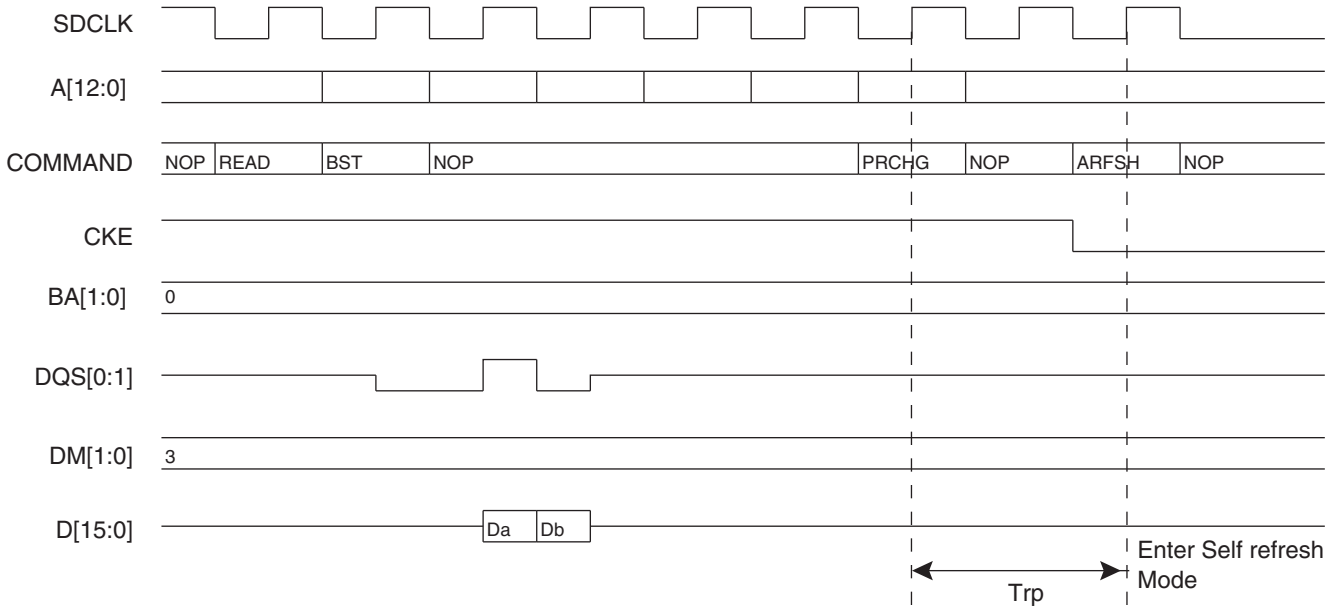


Figure 30-18.Self Refresh Mode Entry, Timeout = 1 or 2

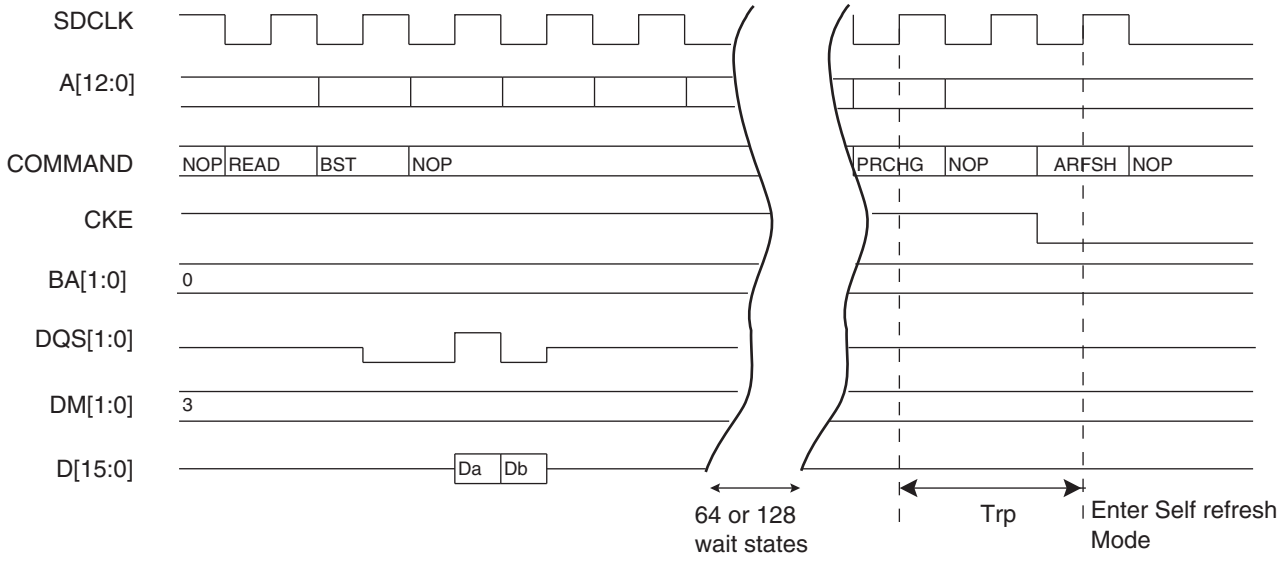


Figure 30-19.Self Refresh Mode Exit

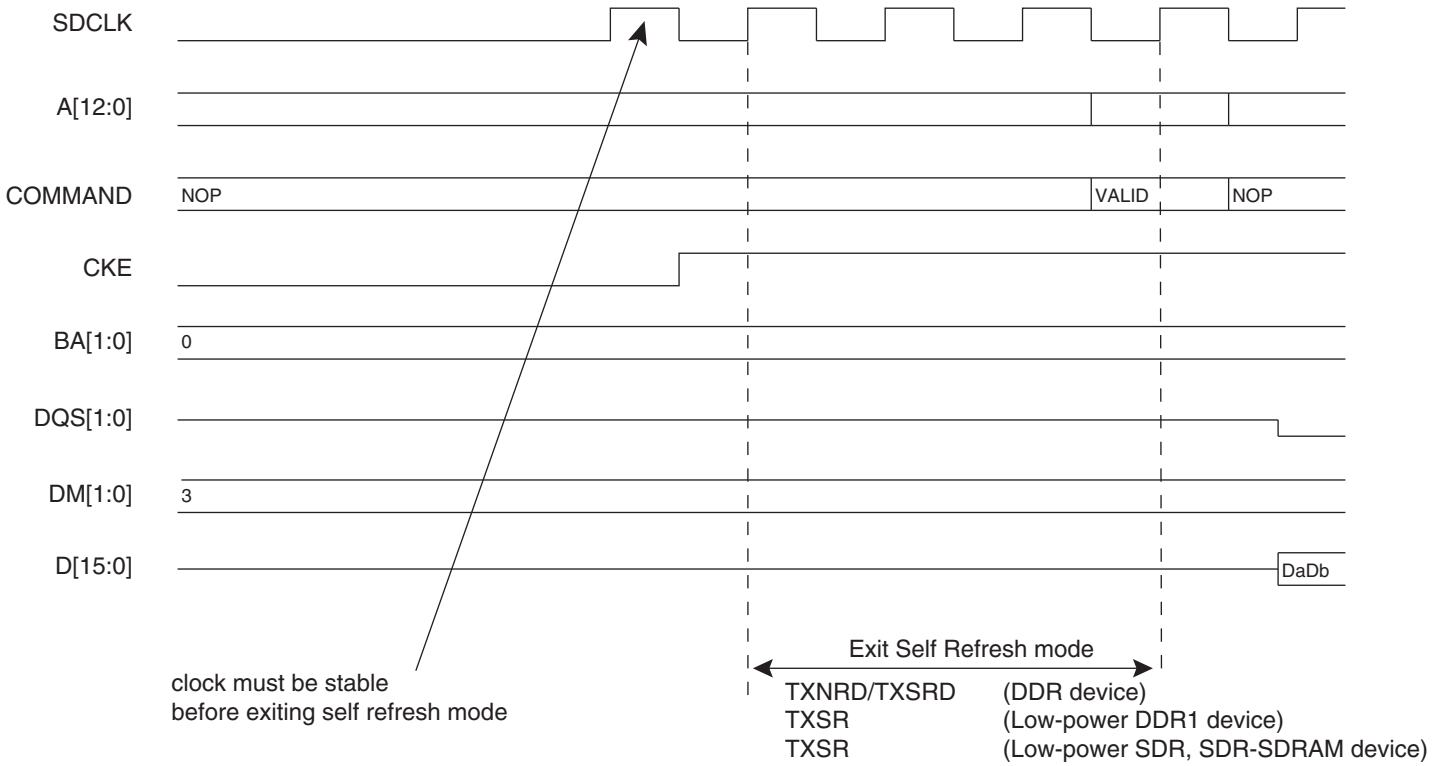


Figure 30-20.Self Refresh and Automatic Update

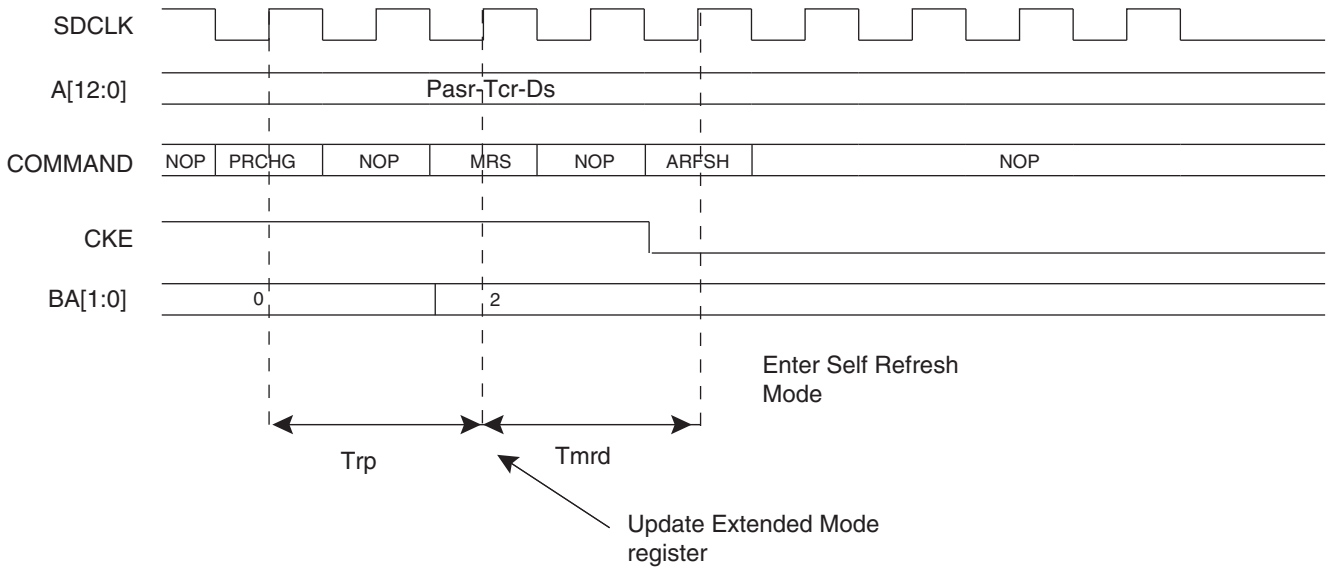
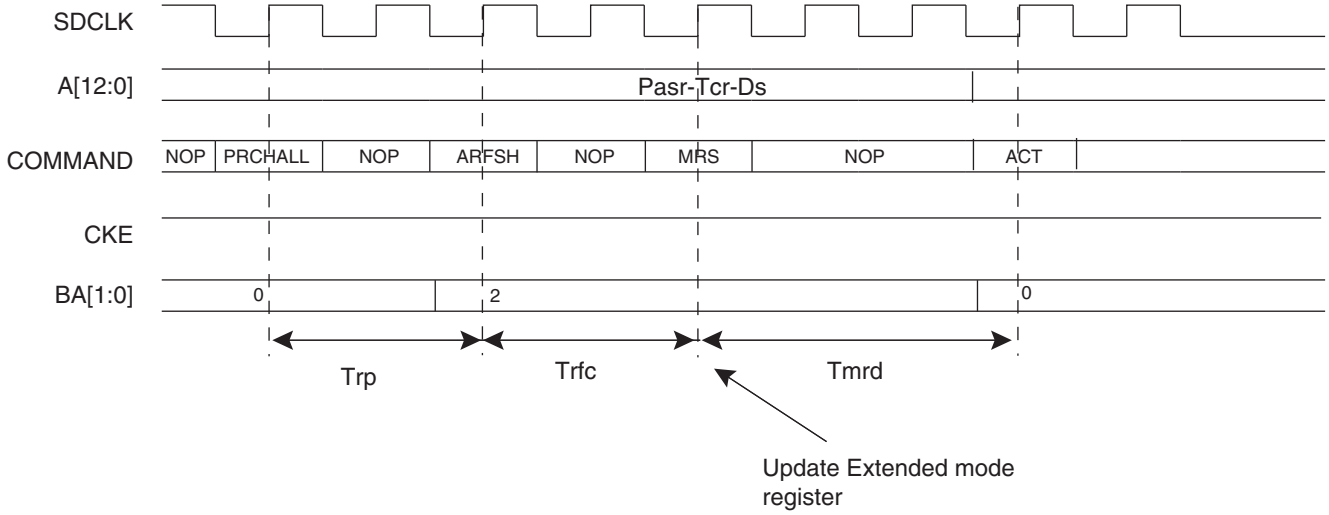


Figure 30-21.Automatic Update During AUTO-REFRESH Command and SDRAM Access



30.5.4.2 Power-down Mode

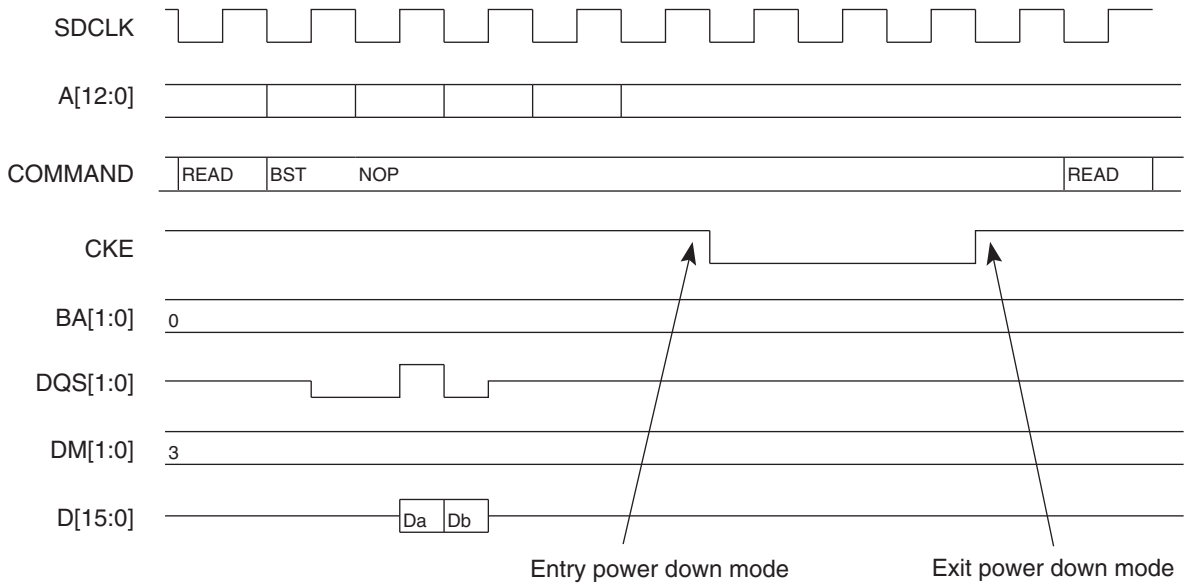
This mode is activated by setting the low-power command bits [LPCB] to ‘10’.

Power-down mode is used when no access to the SDRAM device is possible. In this mode, power consumption is greater than in self refresh mode. This state is similar to normal mode (No low-power mode/No self refresh mode), but the CKE pin is low and the input and output buffers are deactivated as soon the SDRAM device is no longer accessible. In contrast to self refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms). As no auto-refresh operations are performed in this mode, the DDRSDRC carries out the refresh operation. In order to exit low-power mode, a NOP command is required in the case of Low-power SDR-SDRAM and SDR-SDRAM devices. In the case of Low-power DDR1-SDRAM devices, the controller generates a NOP command during a delay of at least TXP. In addition, Low-power DDR1-SDRAM and DDR2-SDRAM must remain in power-down mode for a minimum period of TCKE periods.

The exit procedure is faster than in self refresh mode. See Figure 30-22 on page 448. The DDRSDRC returns to power-down mode as soon as the SDRAM device is not selected. It is possible to define when power-down mode is enabled by setting the register LPR, timeout command bit.

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

Figure 30-22.Power-down Entry/Exit, Timeout = 0

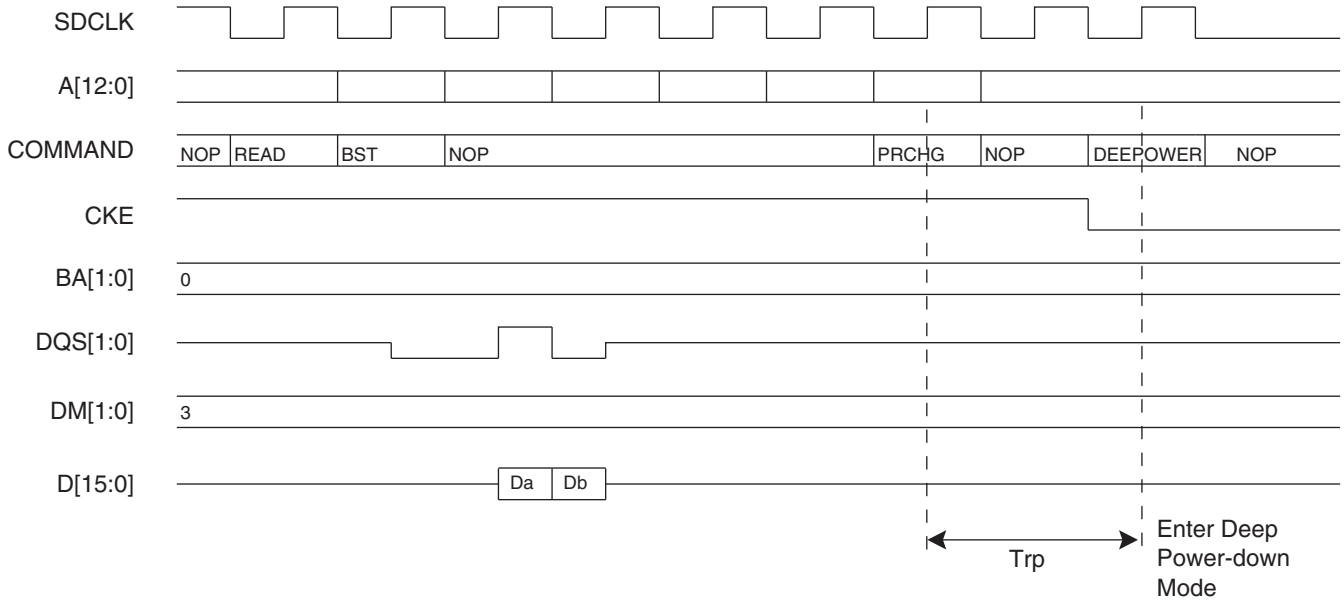


30.5.4.3 Deep Power-down Mode

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

This mode is activated by setting the low-power command bits [LPCB] to ‘11’. When this mode is enabled, the DDRSDRC leaves normal mode (mode == 000) and the controller is frozen. To exit deep power-down mode, the low-power bits (LPCB) must be set to “00”, an initialization sequence must be generated by software. See Section 30.4.2 “Low-power DDR1-SDRAM Initialization” on page 432.

Figure 30-23.Deep Power-down Mode Entry



30.5.4.4 Reset Mode

The reset mode is a feature of the DDR2-SDRAM. This mode is activated by setting the low-power command bits (LPCB) to 11 and the clock frozen command bit (CLK_FR) to 1.

When this mode is enabled, the DDRSDRC leaves normal mode (mode == 000) and the controller is frozen. Before enabling this mode, the end user must assume there is not an access in progress.

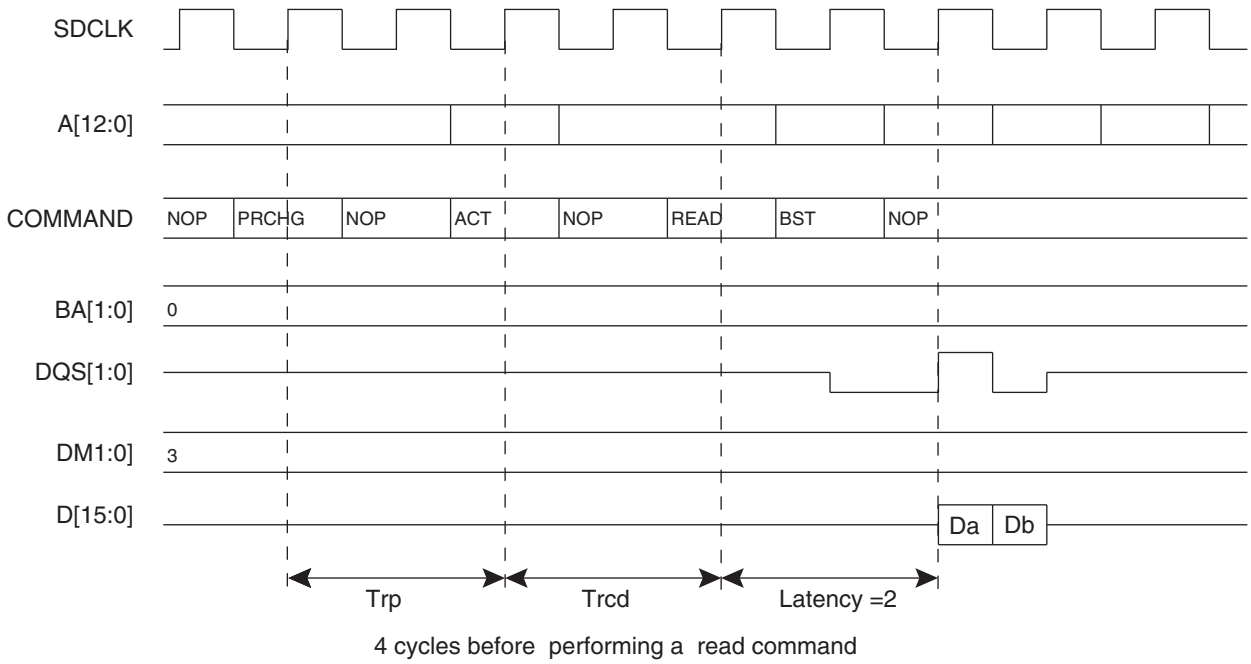
To exit reset mode, the low-power command bits (LPCB) must be set to “00”, clock frozen command bit (CLK_FR) set to 0 and an initialization sequence must be generated by software. See Section 30.4.3 “DDR2-SDRAM Initialization” on page 433.

30.5.5 Multi-port Functionality

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

This operation represents a significative loss. (see Figure 30-24).

Figure 30-24.Trp and Trcd Timings



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

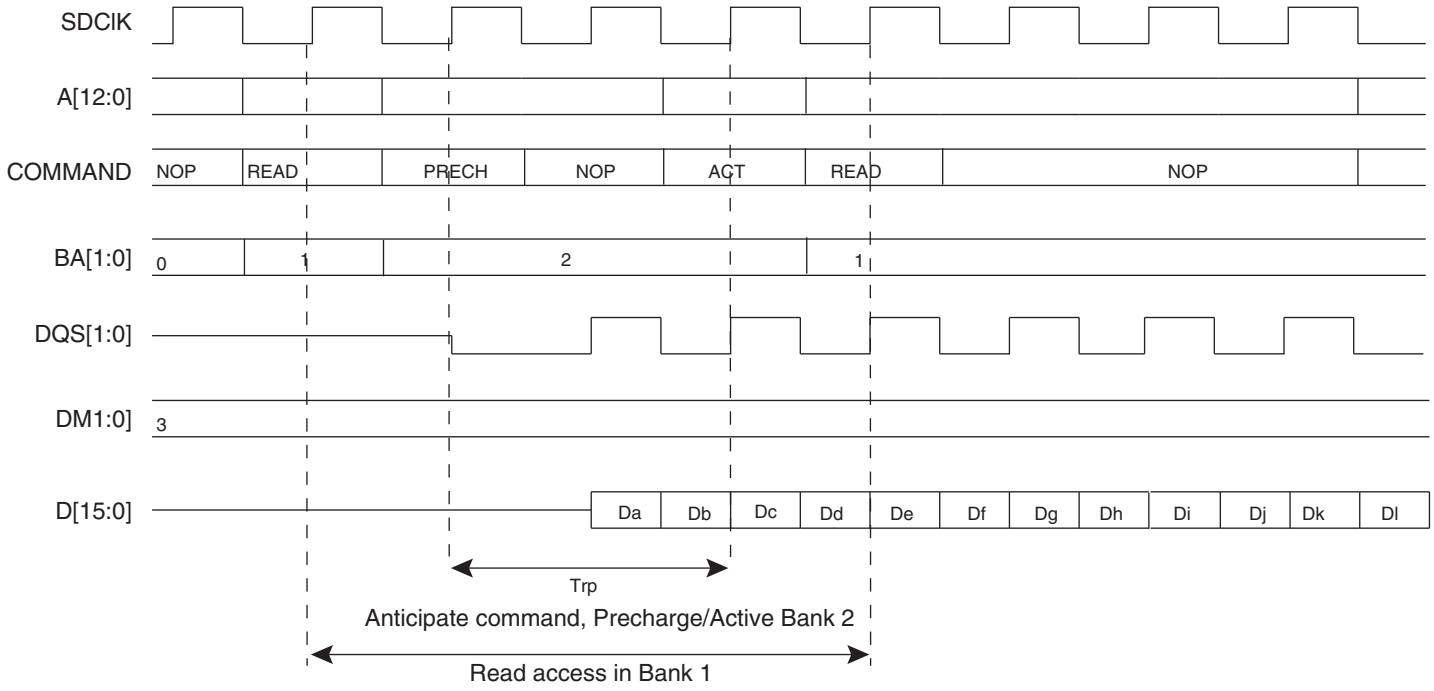
DDRSDRC is a multi-port controller since four masters can simultaneously reach the controller. This feature improves the bandwidth of the system because it can detect four requests on the AHB slave inputs and thus anticipate the commands that follow, PRECHARGE and ACTIVE commands in bank X during current access in bank Y. This allows Trp and Trcd timings to be masked (see Figure 30-25). In the best case, all accesses are done as if the banks and rows were already open. The best condition is met when the four masters work in different banks. In the case of four simultaneous read accesses, when the four banks and associated rows are open, the controller reads with a continuous flow and masks the cas latency for each different access. To allow a continuous flow, the read command must be set at 2 or 3 cycles (cas latency) before the end of current access. This requires that the scheme of arbitration changes since the round-robin arbitration cannot be respected. If the controller anticipates a read access, and thus before the end of current access a master with a high priority arises, then this master will not serviced.

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

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

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

Figure 30-25.Anticipate Precharge/Active Command in Bank 2 during Read Access in Bank 1



30.5.6 Write Protected Registers

To prevent any single software error that may corrupt DDRSDRC behavior, the registers listed below can be write-protected by setting the WPEN bit in the DDRSDRC Write Protect Mode Register (DDRSDRC_WPMR).

If a write access in a write-protected register is detected, then the WPVS flag in the DDRSDRC Write Protect Status Register (DDRSDRC_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically reset after reading the DDRSDRC Write Protect Status Register (DDRSDRC_WPSR).

Following is a list of the write protected registers:

- “DDRSDRC Mode Register” on page 457
- “DDRSDRC Refresh Timer Register” on page 458
- “DDRSDRC Configuration Register” on page 459
- “DDRSDRC Timing Parameter 0 Register” on page 462
- “DDRSDRC Timing Parameter 1 Register” on page 464
- “DDRSDRC Timing Parameter 2 Register” on page 465
- “DDRSDRC Memory Device Register” on page 468
- “DDRSDRC High Speed Register” on page 470

30.6 Software Interface/SDRAM Organization, Address Mapping

The SDRAM address space is organized into banks, rows and columns. The DDRSDRC maps different memory types depending on the values set in the DDRSDRC Configuration Register. See Section 30.7.3 “DDRSDRC Configuration Register” on page 459. The following figures illustrate the relation between CPU addresses and columns, rows and banks addresses for 16-bit memory data bus widths and 32-bit memory data bus widths.

The DDRSDRC supports address mapping in linear mode and interleaved mode.

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

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

The DDRSDRC makes the SDRAM devices access protocol transparent to the user. Table 30-1 to Table 30-15 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

30.6.1 SDRAM Address Mapping for 16-bit Memory Data Bus Width and Four Banks

Table 30-1. Linear Mapping for SDRAM Configuration, 2K Rows, 512/1024/2048/4096 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[10:0]											Column[8:0]									M0
				Bk[1:0]			Row[10:0]											Column[9:0]									M0
			Bk[1:0]				Row[10:0]											Column[10:0]									M0
		Bk[1:0]					Row[10:0]											Column[11:0]									M0

Table 30-2. Linear Mapping for SDRAM Configuration: 4K Rows, 512/1024/2048/4096 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]			Row[11:0]											Column[8:0]									M0
			Bk[1:0]				Row[11:0]											Column[9:0]									M0
		Bk[1:0]					Row[11:0]											Column[10:0]									M0
	Bk[1:0]						Row[11:0]											Column[11:0]									M0

Table 30-3. Linear Mapping for SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Bk[1:0]				Row[12:0]											Column[8:0]									M0
		Bk[1:0]					Row[12:0]											Column[9:0]									M0
	Bk[1:0]						Row[12:0]											Column[10:0]									M0
Bk[1:0]							Row[12:0]											Column[11:0]									M0

Table 30-4. Linear Mapping for SDRAM Configuration: 16K Rows, 512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Bk[1:0]		Row[13:0]														Column[8:0]									M0
	Bk[1:0]			Row[13:0]														Column[9:0]									M0
Bk[1:0]				Row[13:0]														Column[10:0]									M0

Table 30-5. Interleaved Mapping for SDRAM Configuration, 2K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Row[10:0]											Bk[1:0]		Column[8:0]									M0
				Row[10:0]											Bk[1:0]			Column[9:0]									M0
			Row[10:0]											Bk[1:0]				Column[10:0]									M0
		Row[10:0]											Bk[1:0]					Column[11:0]									M0

Table 30-6. Interleaved Mapping for SDRAM Configuration: 4K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Row[11:0]												Bk[1:0]		Column[8:0]									M0
			Row[11:0]												Bk[1:0]			Column[9:0]									M0
		Row[11:0]												Bk[1:0]				Column[10:0]									M0
	Row[11:0]												Bk[1:0]					Column[11:0]									M0

Table 30-7. Interleaved Mapping for SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Row[12:0]													Bk[1:0]		Column[8:0]									M0
		Row[12:0]													Bk[1:0]			Column[9:0]									M0
	Row[12:0]													Bk[1:0]				Column[10:0]									M0
Row[12:0]													Bk[1:0]					Column[11:0]									M0

Table 30-8. Interleaved Mapping for SDRAM Configuration: 16K Rows, 512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Row[13:0]														Bk[1:0]		Column[8:0]									M0
	Row[13:0]														Bk[1:0]		Column[9:0]										M0
Row[13:0]														Bk[1:0]		Column[10:0]											M0

30.6.2 SDRAM Address Mapping for 16-bit Memory Data Bus Width and Eight Banks

Table 30-9. Linear Mapping for SDRAM Configuration: 8K Rows, 1024 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[2:0]			Row[12:0]													Column[9:0]										M0

Table 30-10. Linear Mapping for SDRAM Configuration: 16K Rows, 1024 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[2:0]			Row[13:0]														Column[9:0]										M0

Table 30-11. Interleaved Mapping for SDRAM Configuration: 8K Rows, 1024 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
	Row[12:0]													Bk[2:0]			Column[9:0]										M0

Table 30-12. Interleaved Mapping for SDRAM Configuration: 16K Rows, 1024 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
Row[12:0]														Bk[2:0]			Column[9:0]										M0

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

Table 30-13. SDR-SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[10:0]											Column[7:0]								M[1:0]	

Table 30-13. SDR-SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]	Row[10:0]												Column[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 30-14. SDR-SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]	Row[11:0]													Column[7:0]								M[1:0]	
			Bk[1:0]	Row[11:0]													Column[8:0]									M[1:0]	
		Bk[1:0]	Row[11:0]													Column[9:0]										M[1:0]	
	Bk[1:0]	Row[11:0]													Column[10:0]											M[1:0]	

Table 30-15. SDR-SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
			Bk[1:0]	Row[12:0]														Column[7:0]								M[1:0]	
		Bk[1:0]	Row[12:0]														Column[8:0]									M[1:0]	
	Bk[1:0]	Row[12:0]														Column[9:0]										M[1:0]	
Bk[1:0]	Row[12:0]														Column[10:0]											M[1:0]	

Notes: 1. M[1:0] is the byte address inside a 32-bit word.
2. Bk[1] = BA1, Bk[0] = BA0

30.7 DDR SDR SDRAM Controller (DDRSDRC) User Interface

The User Interface is connected to the APB bus.

The DDRSDRC is programmed using the registers listed in Table 30-16

Table 30-16. Register Mapping

Offset	Register	Name	Access	Reset
0x00	DDRSDRC Mode Register	DDRSDRC_MR	Read-write	0x00000000
0x04	DDRSDRC Refresh Timer Register	DDRSDRC_RTR	Read-write	0x00000000
0x08	DDRSDRC Configuration Register	DDRSDRC_CR	Read-write	0x7024
0x0C	DDRSDRC Timing Parameter 0 Register	DDRSDRC_TPR0	Read-write	0x20227225
0x10	DDRSDRC Timing Parameter 1 Register	DDRSDRC_TPR1	Read-write	0x3c80808
0x14	DDRSDRC Timing Parameter 2 Register	DDRSDRC_TPR2	Read-write	0x2062
0x18	Reserved	–	–	–
0x1C	DDRSDRC Low-power Register	DDRSDRC_LPR	Read-write	0x10000
0x20	DDRSDRC Memory Device Register	DDRSDRC_MD	Read-write	0x10
0x24	DDRSDRC DLL Information Register	DDRSDRC_DLL	Read-only	0x00000001
0x2C	DDRSDRC High Speed Register	DDRSDRC_HS	Read-write	0x0
0x54-0x58	Reserved	-	-	-
0x60-0xE0	Reserved	–	–	–
0xE4	DDRSDRC Write Protect Mode Register	DDRSDRC_WPMR	Read-write	0x00000000
0xE8	DDRSDRC Write Protect Status Register	DDRSDRC_WPSR	Read-only	0x00000000

30.7.1 DDRSDRC Mode Register

Name: DDRSDRC_MR

Address: 0xFFFFE800

Access: Read-write

Reset: See Table 30-16

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

This register can only be written if the bit WPEN is cleared in "DDRSDRC Write Protect Mode Register" on page 471.

- **MODE: DDRSDRC Command Mode**

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

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

30.7.2 DDRSDRC Refresh Timer Register

Name: DDRSDRC_RTR

Address: 0xFFFFE804

Access: Read-write

Reset: See Table 30-16

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							

This register can only be written if the bit WPEN is cleared in “DDRSDRC Write Protect Mode Register” on page 471.

- **COUNT: DDRSDRC Refresh Timer Count**

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

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

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

30.7.3 DDRSDRC Configuration Register

Name: DDRSDRC_CR

Address: 0xFFFFE808

Access: Read-write

Reset: See Table 30-16

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DECOD	–	NB	–	ACTBST	–	EBISHARE
15	14	13	12	11	10	9	8
–	OCD			–	–	DIS_DLL	DIC/DS
7	6	5	4	3	2	1	0
DLL	CAS			NR		NC	

This register can only be written if the bit WPEN is cleared in "DDRSDRC Write Protect Mode Register" on page 471.

- **NC: Number of Column Bits**

The reset value is 9 column bits.

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

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

- **NR: Number of Row Bits**

The reset value is 12 row bits.

NR	Row bits
00	11
01	12
10	13
11	14

- **CAS: CAS Latency**

The reset value is 2 cycles.

CAS	DDR2 CAS Latency	SDR CAS Latency
000	Reserved	Reserved
001	Reserved	Reserved
010	Reserved	2

CAS	DDR2 CAS Latency	SDR CAS Latency
011	3	3
100	Reserved	Reserved
101	Reserved	Reserved
110	Reserved	Reserved
111	Reserved	Reserved

- **DLL: Reset DLL**

Reset value is 0.

This field defines the value of Reset DLL.

0 = Disable DLL reset.

1 = Enable DLL reset.

This value is used during the power-up sequence.

Note: Note: This field is found only in DDR2-SDRAM devices.

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

Reset value is 0.

This field defines the output drive strength.

0 = Normal driver strength.

1 = Weak driver strength.

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

Note: Note: This field is found only in DDR2-SDRAM devices.

- **DIS_DLL: Disable DLL**

Reset value is 0.

0 = Enable DLL

1 = Disable DLL

Note: Note: This field is found only in DDR2-SDRAM devices.

- **OCD: Off-chip Driver**

Reset value is 7.

Notes: 1. OCD is NOT supported by the controller, but these values MUST be programmed during the initialization sequence.
2. This field is found only in DDR2-SDRAM devices.

OCD	Use
000	OCD calibration mode exit, maintain setting
111	OCD calibration default

- **EBISHARE: External Bus Interface is Shared**

The DDR controller embedded in the EBI is used at the same time as another memory controller (SMC,..)

Reset value is 0.

0 = Only the DDR controller function is used.

1 = The DDR controller shares the EBI with another memory controller (SMC, NAND,..)

- **ACTBST: ACTIVE Bank X to Burst Stop Read Access Bank Y**

Reset value is 0.

0 = After an ACTIVE command in Bank X, BURST STOP command can be issued to another bank to stop current read access.

1 = After an ACTIVE command in Bank X, BURST STOP command cannot be issued to another bank to stop current read access.

This field is unique to SDR-SDRAM, Low-power SDR-SDRAM and Low-power DDR1-SDRAM devices.

- **NB: Number of Banks**

The reset value is four banks.

NB	Number of banks
0	4
1	8

Note: Only DDR-SDRAM 2 devices support eight internal banks.

- **DECOD: Type of Decoding**

The reset value is 0: sequential decoding.

0 = Sequential Decoding.

1 = Interleaved Decoding.

30.7.4 DDRSDRC Timing Parameter 0 Register

Name: DDRSDRC_TPR0

Address: 0xFFFFE80C

Access: Read-write

Reset: See Table 30-16

31	30	29	28	27	26	25	24
TMRD				REDUCE_WRRD	TWTR		
23	22	21	20	19	18	17	16
TRRD				TRP			
15	14	13	12	11	10	9	8
TRC				TWR			
7	6	5	4	3	2	1	0
TRCD				TRAS			

This register can only be written if the bit WPEN is cleared in "DDRSDRC Write Protect Mode Register" on page 471.

- **TRAS: Active to Precharge Delay**

Reset Value is 5 cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 0 and 15.

- **TRCD: Row to Column Delay**

Reset Value is 2 cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 0 and 15.

- **TWR: Write Recovery Delay**

Reset value is 2 cycles.

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

- **TRC: Row Cycle Delay**

Reset value is 7 cycles.

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

- **TRP: Row Precharge Delay**

Reset Value is 2 cycles.

This field defines the delay between a Precharge Command and another command in number of cycles. Number of cycles is between 0 and 15.

- **TRRD: Active bankA to Active bankB**

Reset value is 2 cycles.

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

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

Reset value is 0.

This field is unique to Low-power DDR1-SDRAM devices and DDR2-SDRAM devices.

This field defines the internal write to read command Time in number of cycles. Number of cycles is between 1 and 7.

- **REDUCE_WRRD: Reduce Write to Read Delay**

Reset value is 0.

This field reduces the delay between write to read access for low-power DDR-SDRAM devices with a latency equal to 2. To use this feature, TWTR field must be equal to 0. Important to note is that some devices do not support this feature.

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

Reset Value is 2 cycles.

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

30.7.5 DDRSDRC Timing Parameter 1 Register

Name: DDRSDRC_TPR1

Address: 0xFFFFE810

Access: Read-write

Reset: See Table 30-16

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

This register can only be written if the bit WPEN is cleared in "DDRSDRC Write Protect Mode Register" on page 471.

- **TRFC: Row Cycle Delay**

Reset Value is 8 cycles.

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

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

Reset Value is 8 cycles.

This field defines the delay between cke set high and a non Read Command in number of cycles. Number of cycles is between 0 and 255. This field is used for SDR-SDRAM and DDR-SDRAM devices. In the case of SDR-SDRAM devices and Low-power DDR1-SDRAM, this field is equivalent to TXSR timing.

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

Reset Value is 200 cycles.

This field defines the delay between cke set high and a Read Command in number of cycles. Number of cycles is between 0 and 255 cycles.This field is unique to DDR-SDRAM devices. In the case of a Low-power DDR1-SDRAM, this field must be written to 0.

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

Reset Value is 3 cycles.

This field defines the delay between cke set high and a Valid Command in number of cycles. Number of cycles is between 0 and 15 cycles. This field is unique to Low-power DDR1-SDRAM devices and DDR2-SDRAM devices.

30.7.6 DDRSDRC Timing Parameter 2 Register

Name: DDRSDRC_TPR2

Address: 0xFFFFE814

Access: Read-write

Reset: See Table 30-16

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

This register can only be written if the WPEN bit is cleared in "DDRSDRC Write Protect Mode Register" on page 471.

- **TXARD: Exit Active Power Down Delay to Read Command in Mode "Fast Exit".**

The Reset Value is 2 cycles.

This field defines the delay between cke set high and a Read Command in number of cycles. Number of cycles is between 0 and 15.

Note: This field is found only in DDR2-SDRAM devices.

- **TXARDS: Exit Active Power Down Delay to Read Command in Mode "Slow Exit".**

The Reset Value is 6 cycles.

This field defines the delay between cke set high and a Read Command in number of cycles. Number of cycles is between 0 and 15.

Note: This field is found only in DDR2-SDRAM devices.

- **TRPA: Row Precharge All Delay**

The Reset Value is 0 cycle.

This field defines the delay between a Precharge ALL banks Command and another command in number of cycles. Number of cycles is between 0 and 15.

Note: This field is found only in DDR2-SDRAM devices.

- **TRTP: Read to Precharge**

The Reset Value is 2 cycles.

This field defines the delay between Read Command and a Precharge command in number of cycle.

Number of cycles is between 0 and 7.

- **TFAW: Four Active window**

The Reset Value is 4 cycles.

DDR2 devices with 8-banks (1Gb or larger) have an additional requirement: t_{FAW}. This requires that no more than four ACTIVATE commands may be issued in any given t_{FAW} (MIN) period.

Number of cycles is between 0 and 15.

Note: This field is found only in DDR-SDRAM 2 devices with eight internal banks

30.7.7 DDRSDRC Low-power Register

Name: DDRSDRC_LPR

Address: 0xFFFFE81C

Access: Read-write

Reset: See Table 30-16

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

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

Reset value is “00”.

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

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

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

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

- **CLK_FR: Clock Frozen Command Bit**

Reset value is “0”.

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

1 = Clock(s) is/are frozen.

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

- **PASR: Partial Array Self Refresh**

Reset value is “0”.

This field is unique to Low-power SDRAM. It is used to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self refresh mode.

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

After the initialization sequence, as soon as PASR field is modified, Extended Mode Register in the external device memory is accessed automatically and PASR bits are updated. In function of the UPD_MR bit, update is done before entering in self refresh mode or during a refresh command and a pending read or write access.

- **DS: Drive Strength**

Reset value is “0”.

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

After the initialization sequence, as soon as DS field is modified, Extended Mode Register is accessed automatically and DS bits are updated. In function of UPD_MR bit, update is done before entering in self refresh mode or during a refresh command and a pending read or write access.

- **TIMEOUT: Low Power Mode**

Reset value is “00”.

This field defines when low-power mode is enabled.

00	The SDRAM controller activates the SDRAM low-power mode immediately after the end of the last transfer.
01	The SDRAM controller activates the SDRAM low-power mode 64 clock cycles after the end of the last transfer.
10	The SDRAM controller activates the SDRAM low-power mode 128 clock cycles after the end of the last transfer.
11	Reserved

- **APDE: Active Power Down Exit Time**

Reset value is “1”.

This mode is unique to DDR2-SDRAM devices. This mode allows to determine the active power-down mode, which determines performance versus power saving**.**

0 = Fast Exit

1 = Slow Exit

After the initialization sequence, as soon as APDE field is modified Extended Mode Register, located in the memory of the external device, is accessed automatically and APDE bits are updated. In function of the UPD_MR bit, update is done before entering in self refresh mode or during a refresh command and a pending read or write access

- **UPD_MR: Update Load Mode Register and Extended Mode Register**

Reset value is “0”.

This bit is used to enable or disable automatic update of the Load Mode Register and Extended Mode Register. This update is function of DDRSDRC integration in a system. DDRSDRC can either share or not share an external bus with another controller.

00	Update is disabled.
01	DDRSDRC shares external bus. Automatic update is done during a refresh command and a pending read or write access in SDRAM device.
10	DDRSDRC does not share external bus. Automatic update is done before entering in self refresh mode.
11	Reserved

30.7.8 DDRSDRC Memory Device Register

Name: DDRSDRC_MD

Address: 0xFFFFE820

Access: Read-write

Reset: See Table 30-16

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

This register can only be written if the bit WPEN is cleared in "DDRSDRC Write Protect Mode Register" on page 471.

- **MD: Memory Device**

Indicates the type of memory used.

Reset value is for SDR-SDRAM device.

000 = SDR-SDRAM

001 = Low-power SDR-SDRAM

010 = Reserved

011 = Low-power DDR1-SDRAM

110 = DDR2-SDRAM

- **DBW: Data Bus Width**

Reset value is 16 bits.

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

1 = Data bus width is 16 bits.

30.7.9 DDRSDRC DLL Register

Name: DDRSDRC_DLL

Address: 0xFFFFE824

Access: Read-only

Reset: See Table 30-16

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

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

- **MDINC: DLL Master Delay Increment**

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

1 = The DLL is incrementing the Master delay counter.

- **MDDEC: DLL Master Delay Decrement**

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

1 = The DLL is decrementing the Master delay counter.

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

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

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

- **MDVAL: DLL Master Delay Value**

Value of the Master delay counter.

30.7.10 DDRSDRC High Speed Register

Name: DDRSDRC_HS

Address: 0xFFFFE82C

Access: Read-write

Reset: See Table 30-16

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

This register can only be written if the bit WPEN is cleared in “DDRSDRC Write Protect Mode Register” on page 471.

- **DIS_ANTICIP_READ: Anticip Read Access**

0 = anticip read access is enabled.

1 = anticip read access is disabled (default).

DIS_ANTICIP_READ allows DDR2 read access optimization with multi-port.

As this feature is based on the “bank open policy”, the software must map different buffers in different DDR2 banks to take advantage of that feature.

30.7.11 DDRSDRC Write Protect Mode Register

Name: DDRSDRC_WPMR

Address: 0xFFFFE8E4

Access: Read-write

Reset See Table 30-16

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x444452 (“DDR” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x444452 (“DDR” in ASCII).

Protects the registers:

- “DDRSDRC Mode Register” on page 457
- “DDRSDRC Refresh Timer Register” on page 458
- “DDRSDRC Configuration Register” on page 459
- “DDRSDRC Timing Parameter 0 Register” on page 462
- “DDRSDRC Timing Parameter 1 Register” on page 464
- “DDRSDRC Timing Parameter 2 Register” on page 465
- “DDRSDRC Memory Device Register” on page 468
- “DDRSDRC High Speed Register” on page 470

- **WPKEY: Write Protect KEY**

Should be written at value 0x444452 (“DDR” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

30.7.12 DDRSDRC Write Protect Status Register

Name: DDRSDRC_WPSR

Address: 0xFFFFE8E8

Access: Read-only

Reset: See Table 30-16

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the DDRSDRC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the DDRSDRC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading DDRSDRC_WPSR automatically clears all fields.

31. DMA Controller (DMAC)

31.1 Description

The DMA Controller (DMAC) is an AHB-central DMA controller core that transfers data from a source peripheral to a destination peripheral over one or more AMBA buses. One channel is required for each source/destination pair. In the most basic configuration, the DMAC has one master interface and one channel. The master interface reads the data from a source and writes it to a destination. Two AMBA transfers are required for each DMAC data transfer. This is also known as a dual-access transfer.

The DMAC is programmed via the APB interface.

The DMAC embeds 8 channels.

31.2 Embedded Characteristics

- 2 AHB-Lite Master Interfaces
- DMA Module Supports the Following Transfer Schemes: Peripheral-to-Memory, Memory-to-Peripheral, Peripheral-to-Peripheral and Memory-to-Memory
- Source and Destination Operate independently on BYTE (8-bit), HALF-WORD (16-bit) and WORD (32-bit)
- Supports Hardware and Software Initiated Transfers
- Supports Multiple Buffer Chaining Operations
- Supports Incrementing/decrementing/fixed Addressing Mode Independently for Source and Destination
- Supports Programmable Address Increment/decrement on User-defined Boundary Condition to Enable Picture-in-Picture Mode
- Programmable Arbitration Policy, Modified Round Robin and Fixed Priority are Available
- Supports Specified Length and Unspecified Length AMBA AHB Burst Access to Maximize Data Bandwidth
- AMBA APB Interface Used to Program the DMA Controller
- 8 DMA Channels
- 12 External Request Lines
- Embedded FIFO
- Channel Locking and Bus Locking Capability

31.2.1 DMA Controller 0

- Two Masters
- Embeds 8 channels
- 64-byte FIFO for channel 0, 16-byte FIFO for Channel 1 to 7
- Features:
  - Linked List support with Status Write Back operation at End of Transfer
  - Word, HalfWord, Byte transfer support.
  - Memory to memory transfer
  - Peripheral to memory
  - Memory to peripheral

The DMA controller can handle the transfer between peripherals and memory and so receives the triggers from the peripherals below. The hardware interface numbers are also given in Table 31-1.

Table 31-1. DMA Channel Definition

Instance name	T/R	DMA Channel HW Interface Number
HSMCI0	RX/TX	0
SPI0	TX	1
SPI0	RX	2
USART0	TX	3
USART0	RX	4
USART1	TX	5
USART1	RX	6
TWI0	TX	7
TWI0	RX	8
TWI2	TX	9
TWI2	RX	10
UART0	TX	11
UART0	RX	12
SSC	TX	13
SSC	RX	14

31.2.2 DMA Controller 1

- Two Masters
- Embeds 8 channels
- 16-byte FIFO per Channel
- Features:
  - Linked List support with Status Write Back operation at End of Transfer
  - Word, HalfWord, Byte transfer support.
  - Peripheral to memory
  - Memory to peripheral

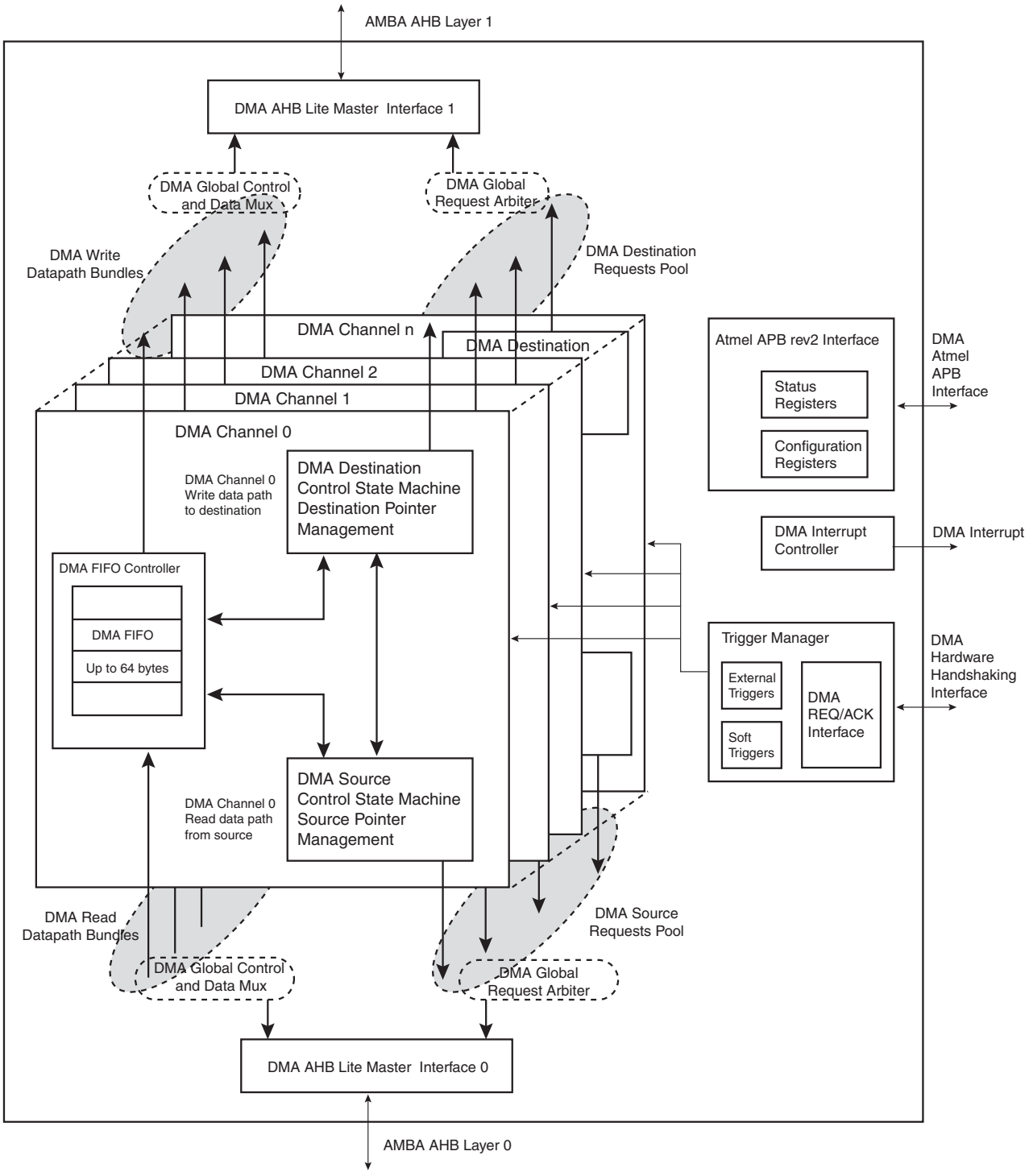
The DMA controller can handle the transfer between peripherals and memory and so receives the triggers from the peripherals below. The hardware interface numbers are also given in Table 31-2.

Table 31-2. DMA Channel Definition

Instance name	T/R	DMA Channel HW Interface Number
HSMCI1	RX/TX	0
SPI1	TX	1
SPI1	RX	2
SMD	TX	3
SMD	RX	4
TWI1	TX	5
TWI1	RX	6
ADC	RX	7
DBGU	TX	8
DBGU	RX	9
UART1	TX	10
UART1	RX	11
USART2	TX	12
USART2	RX	13
USART3	TX	14
USART3	RX	15

31.3 Block Diagram

Figure 31-1. DMA Controller (DMAC) Block Diagram



31.4 Functional Description

31.4.1 Basic Definitions

Source peripheral: Device on an AMBA layer from where the DMAC reads data, which is then stored in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel.

Destination peripheral: Device to which the DMAC writes the stored data from the FIFO (previously read from the source peripheral).

Memory: Source or destination that is always “ready” for a DMAC transfer and does not require a handshaking interface to interact with the DMAC.

Programmable Arbitration Policy: Modified Round Robin and Fixed Priority are available by means of the ARB_CFG bit in the Global Configuration Register (DMAC_GCFG). The fixed priority is linked to the channel number. The highest DMAC channel number has the highest priority.

Channel: Read/write datapath between a source peripheral on one configured AMBA layer and a destination peripheral on the same or different AMBA layer that occurs through the channel FIFO. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.

Master interface: DMAC is a master on the AHB bus reading data from the source and writing it to the destination over the AHB bus.

Slave interface: The APB interface over which the DMAC is programmed. The slave interface in practice could be on the same layer as any of the master interfaces or on a separate layer.

Handshaking interface: A set of signal registers that conform to a protocol and *handshake* between the DMAC and source or destination peripheral to control the transfer of a single or chunk transfer between them. This interface is used to request, acknowledge, and control a DMAC transaction. A channel can receive a request through one of two types of handshaking interface: hardware or software.

Hardware handshaking interface: Uses hardware signals to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral.

Software handshaking interface: Uses software registers to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral. No special DMAC handshaking signals are needed on the I/O of the peripheral. This mode is useful for interfacing an existing peripheral to the DMAC without modifying it.

Flow controller: The device (either the DMAC or source/destination peripheral) that determines the length of and terminates a DMAC buffer transfer. If the length of a buffer is known before enabling the channel, then the DMAC should be programmed as the flow controller. If the length of a buffer is not known prior to enabling the channel, the source or destination peripheral needs to terminate a buffer transfer. In this mode, the peripheral is the flow controller.

Transfer hierarchy: Figure 31-2 on page 478 illustrates the hierarchy between DMAC transfers, buffer transfers, chunk or single, and AMBA transfers (single or burst) for non-memory peripherals. Figure 31-3 on page 478 shows the transfer hierarchy for memory.

Figure 31-2. DMAC Transfer Hierarchy for Non-Memory Peripheral

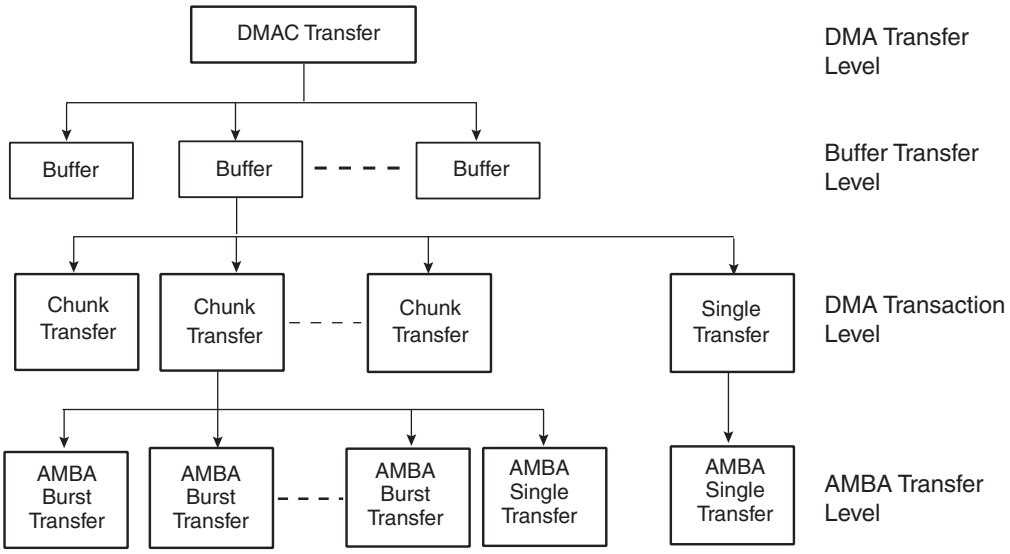
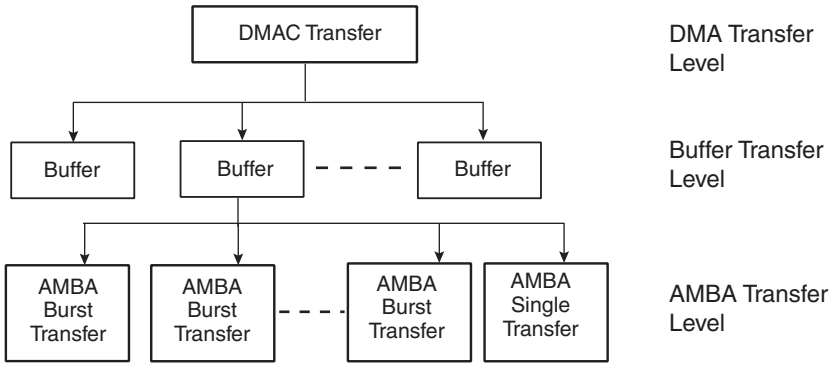


Figure 31-3. DMAC Transfer Hierarchy for Memory



Buffer: A buffer of DMAC data. The amount of data (length) is determined by the flow controller. For transfers between the DMAC and memory, a buffer is broken directly into a sequence of AMBA bursts and AMBA single transfers.

For transfers between the DMAC and a non-memory peripheral, a buffer is broken into a sequence of DMAC transactions (single and chunks). These are in turn broken into a sequence of AMBA transfers.

Transaction: A basic unit of a DMAC transfer as determined by either the hardware or software handshaking interface. A transaction is only relevant for transfers between the DMAC and a source or destination peripheral if the source or destination peripheral is a non-memory device. There are two types of transactions: single transfer and chunk transfer.

- **Single transfer:** The length of a single transaction is always 1 and is converted to a single AMBA access.
- **Chunk transfer:** The length of a chunk is programmed into the DMAC. The chunk is then converted into a sequence of AHB access.DMAC executes each AMBA burst transfer by performing incremental bursts that are no longer than 16 beats.

DMAC transfer: Software controls the number of buffers in a DMAC transfer. Once the DMAC transfer has completed, then hardware within the DMAC disables the channel and can generate an interrupt to signal the completion of the DMAC transfer. You can then re-program the channel for a new DMAC transfer.

Single-buffer DMAC transfer: Consists of a single buffer.

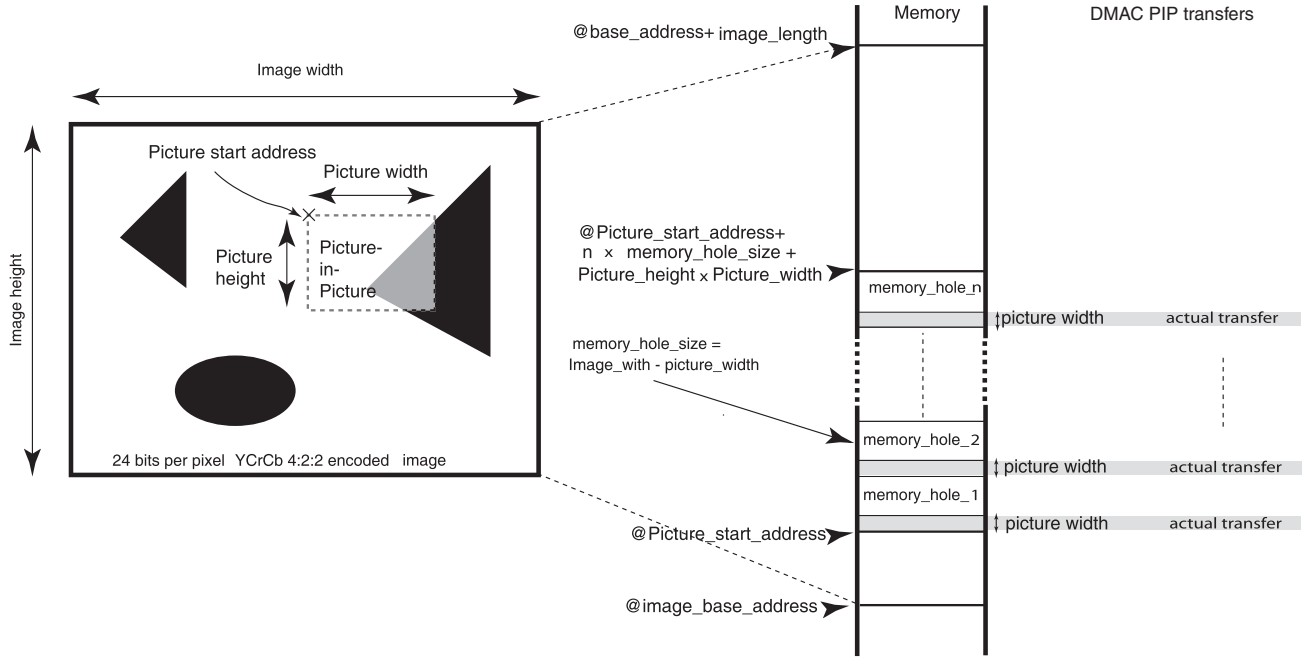
Multi-buffer DMAC transfer: A DMAC transfer may consist of multiple DMAC buffers. Multi-buffer DMAC transfers are supported through buffer chaining (linked list pointers), auto-reloading of channel registers, and contiguous buffers. The source and destination can independently select which method to use.

- **Linked lists (buffer chaining) –** A descriptor pointer (DSCR) points to the location in system memory where the next linked list item (LLI) exists. The LLI is a set of registers that describe the next buffer (buffer descriptor) and a descriptor pointer register. The DMAC fetches the LLI at the beginning of every buffer when buffer chaining is enabled.
- **Replay –** The DMAC automatically reloads the channel registers at the end of each buffers to the value when the channel was first enabled.
- **Contiguous buffers –** Where the address of the next buffer is selected to be a continuation from the end of the previous buffer.

Picture-in-Picture Mode: DMAC contains a Picture-in-Picture mode support. When this mode is enabled, addresses are automatically incremented by a programmable value when the DMAC channel transfer count reaches a user defined boundary.

Figure 31-4 on page 479 illustrates a memory mapped image 4:2:2 encoded located at image_base_address in memory. A user defined start address is defined at Picture_start_address. The incremented value is set to memory_hole_size = image_width - picture_width, and the boundary is set to picture_width.

Figure 31-4. Picture-In-Picture Mode Support



Channel locking: Software can program a channel to keep the AHB master interface by locking the arbitration for the master bus interface for the duration of a DMAC transfer, buffer, or chunk.

Bus locking: Software can program a channel to maintain control of the AMBA bus by asserting hmastlock for the duration of a DMAC transfer, buffer, or transaction (single or chunk). Channel locking is asserted for the duration of bus locking at a minimum.

31.4.2 Memory Peripherals

Figure 31-3 on page 478 shows the DMAC transfer hierarchy of the DMAC for a memory peripheral. There is no handshaking interface with the DMAC, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request. The alternative to not having a transaction-level handshaking interface is to allow the DMAC to attempt AMBA transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these AMBA transfers, it inserts wait states onto the bus until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking

interface, the peripheral can signal to the DMAC that it is ready to transmit/receive data, and then the DMAC can access the peripheral without the peripheral inserting wait states onto the bus.

31.4.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or chunk transfers. The operation of the handshaking interface is different and depends on whether the peripheral or the DMAC is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMAC that it is ready to transfer/accept data over the AMBA bus. A non-memory peripheral can request a DMAC transfer through the DMAC using one of two handshaking interfaces:

- Hardware handshaking
- Software handshaking

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

31.4.3.1 Software Handshaking

When the slave peripheral requires the DMAC to perform a DMAC transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller.

The interrupt service routine then uses the software registers to initiate and control a DMAC transaction. These software registers are used to implement the software handshaking interface.

The SRC_H2SEL/DST_H2SEL bit in the DMAC_CFGx channel configuration register must be set to zero to enable software handshaking.

When the peripheral is not the flow controller, then the last transaction register DMAC_LAST is not used, and the values in these registers are ignored.

Chunk Transactions

Writing a 1 to the DMAC_CREQ[2x] register starts a source chunk transaction request, where x is the channel number. Writing a 1 to the DMAC_CREQ[2x+1] register starts a destination chunk transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_CREQ[2*x*] or DMAC_CREQ[2x+1].

Single Transactions

Writing a 1 to the DMAC_SREQ[2x] register starts a source single transaction request, where x is the channel number. Writing a 1 to the DMAC_SREQ[2x+1] register starts a destination single transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_SREQ[x] or DMAC_SREQ[2x+1].

The software can poll the relevant channel bit in the DMAC_CREQ[2x]/DMAC_CREQ[2x+1] and DMAC_SREQ[x]/DMAC_SREQ[2x+1] registers. When both are 0, then either the requested chunk or single transaction has completed.

31.4.4 DMAC Transfer Types

A DMAC transfer may consist of single or multi-buffer transfers. On successive buffers of a multi-buffer transfer, the DMAC_SADDRx/DMAC_DADDRx registers in the DMAC are reprogrammed using either of the following methods:

- Buffer chaining using linked lists
- Replay mode
- Contiguous address between buffers

On successive buffers of a multi-buffer transfer, the DMAC_CTRLAx and DMAC_CTRLBx registers in the DMAC are re-programmed using either of the following methods:

- Buffer chaining using linked lists

- Replay mode

When buffer chaining using linked lists is the multi-buffer method of choice, and on successive buffers, the DMAC_DSCRx register in the DMAC is re-programmed using the following method:

- Buffer chaining using linked lists

A buffer descriptor (LLI) consists of following registers, DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx, DMAC_CTRLBx.These registers, along with the DMAC_CFGx register, are used by the DMAC to set up and describe the buffer transfer.

31.4.4.1 Multi-buffer Transfers

Buffer Chaining Using Linked Lists

In this case, the DMAC re-programs the channel registers prior to the start of each buffer by fetching the buffer descriptor for that buffer from system memory. This is known as an LLI update.

DMAC buffer chaining is supported by using a Descriptor Pointer register (DMAC_DSCRx) that stores the address in memory of the next buffer descriptor. Each buffer descriptor contains the corresponding buffer descriptor (DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx DMAC_CTRLBx).

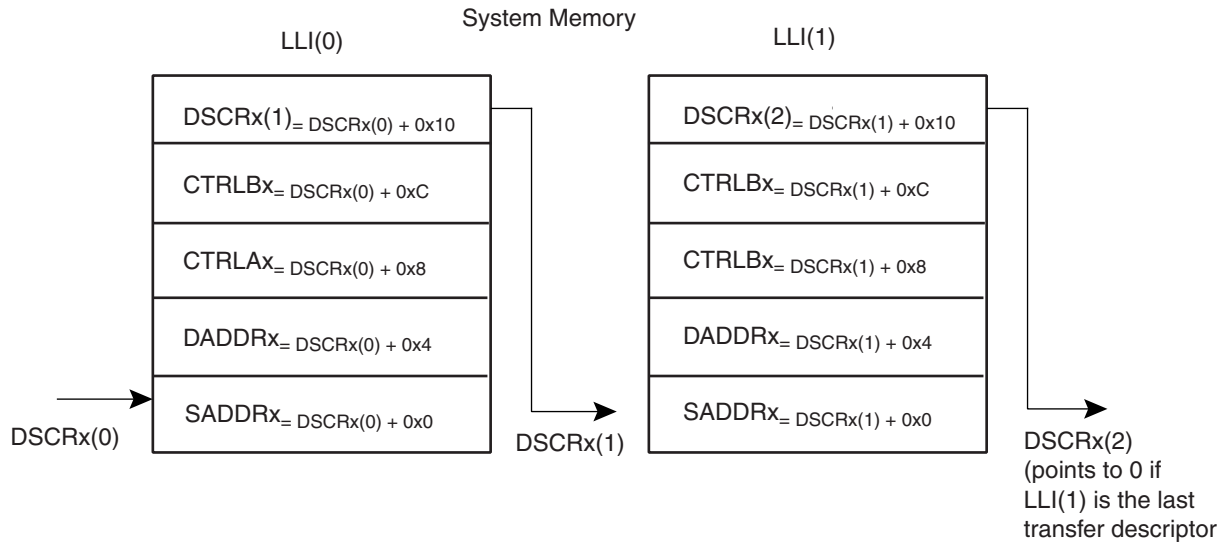
To set up buffer chaining, a sequence of linked lists must be programmed in memory.

The DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx registers are fetched from system memory on an LLI update. The updated content of the DMAC_CTRLAx register is written back to memory on buffer completion. Figure 31-5 on page 481 shows how to use chained linked lists in memory to define multi-buffer transfers using buffer chaining.

The Linked List multi-buffer transfer is initiated by programming DMAC_DSCRx with DSCRx(0) (LLI(0) base address) different from zero. Other fields and registers are ignored and overwritten when the descriptor is retrieved from memory.

The last transfer descriptor must be written to memory with its next descriptor address set to 0.

Figure 31-5. Multi Buffer Transfer Using Linked List



31.4.4.2 Programming DMAC for Multiple Buffer Transfers

Table 31-3. Multiple Buffers Transfer Management Table

Transfer Type	AUTO	SRC_REP	DST_REP	SRC_DSCR	DST_DSCR	BTSIZE	DSCR	SADDR	DADDR	Other Fields
1) Single Buffer or Last buffer of a multiple buffer transfer	0	–	–	–	–	USR	0	USR	USR	USR
2) Multi Buffer transfer with contiguous DADDR	0	–	0	0	1	LLI	USR	LLI	CONT	LLI
3) Multi Buffer transfer with contiguous SADDR	0	0	–	1	0	LLI	USR	CONT	LLI	LLI
4) Multi Buffer transfer with LLI support	0	–	–	0	0	LLI	USR	LLI	LLI	LLI
5) Multi Buffer transfer with DADDR reloaded	0	–	1	0	1	LLI	USR	LLI	REP	LLI
6) Multi Buffer transfer with SADDR reloaded	0	1	–	1	0	LLI	USR	REP	LLI	LLI
7) Multi Buffer transfer with BTSIZE reloaded and contiguous DADDR	1	–	0	0	1	REP	USR	LLI	CONT	LLI
8) Multi Buffer transfer with BTSIZE reloaded and contiguous SADDR	1	0	–	1	0	REP	USR	CONT	LLI	LLI
9) Automatic mode channel is stalling BTsize is reloaded	1	0	0	1	1	REP	USR	CONT	CONT	REP
10) Automatic mode BTSIZE, SADDR and DADDR reloaded	1	1	1	1	1	REP	USR	REP	REP	REP
11) Automatic mode BTSIZE, SADDR reloaded and DADDR contiguous	1	1	0	1	1	REP	USR	REP	CONT	REP

Notes:
1. USR means that the register field is manually programmed by the user.
2. CONT means that address are contiguous.
3. REP means that the register field is updated with its previous value. If the transfer is the first one, then the user must manually program the value.
4. Channel stalled is true if the relevant BTC interrupt is not masked.
5. LLI means that the register field is updated with the content of the linked list item.

Replay Mode of Channel Registers

During automatic replay mode, the channel registers are reloaded with their initial values at the completion of each buffer and the new values used for the new buffer. Depending on the row number in Table 31-3 on page 482, some or all of the DMAC_SADDRx, DMAC_DADDRx, DMAC_CTRLAx and DMAC_CTRLBx channel registers are reloaded from their initial value at the start of a buffer transfer.

Contiguous Address Between Buffers

In this case, the address between successive buffers is selected to be a continuation from the end of the previous buffer. Enabling the source or destination address to be contiguous between buffers is a function of DMAC_CTRLAx.SRC_DSCR, DMAC_CFGx.DST_REP, DMAC_CFGx.SRC_REP and DMAC_CTRLAx.DST_DSCR registers.

Suspension of Transfers Between Buffers

At the end of every buffer transfer, an end of buffer interrupt is asserted if:

- The channel buffer interrupt is unmasked, DMAC_EBCIMR.BTCx = '1', where x is the channel number.

Note: The Buffer Transfer Completed Interrupt is generated at the completion of the buffer transfer to the destination.

At the end of a chain of multiple buffers, an end of linked list interrupt is asserted if:

- The channel end of the Chained Buffer Transfer Completed Interrupt is unmasked, DMAC_EBCIMR.CBTCx = '1', when n is the channel number.

31.4.4.3 Ending Multi-buffer Transfers

All multi-buffer transfers must end as shown in Row 1 of Table 31-3 on page 482. At the end of every buffer transfer, the DMAC samples the row number, and if the DMAC is in Row 1 state, then the previous buffer transferred was the last buffer and the DMAC transfer is terminated.

For rows 9, 10 and 11 of Table 31-3 on page 482, (DMAC_DSCRx = 0 and DMAC_CTRLBx.AUTO is set), multi-buffer DMAC transfers continue until the automatic mode is disabled by writing a '1' in DMAC_CTRLBx.AUTO bit. This bit should be programmed to zero in the end of buffer interrupt service routine that services the next-to-last buffer transfer. This puts the DMAC into Row 1 state.

For rows 2, 3, 4, 5, and 6 (DMAC_CRTLBx.AUTO cleared), the user must set up the last buffer descriptor in memory so that LLI.DMAC_DSCRx is set to 0.

31.4.5 Programming a Channel

Four registers, the DMAC_DSCRx, the DMAC_CTRLAx, the DMAC_CTRLBx and DMAC_CFGx, need to be programmed to set up whether single or multi-buffer transfers take place, and which type of multi-buffer transfer is used. The different transfer types are shown in Table 31-3 on page 482.

The "BTSIZE, SADDR and DADDR" columns indicate where the values of DMAC_SARx, DMAC_DARx, DMAC_CTLx, and DMAC_LLPx are obtained for the next buffer transfer when multi-buffer DMAC transfers are enabled.

31.4.5.1 Programming Examples

Single-buffer Transfer (Row 1)

1. Read the Channel Handler Status Register DMAC_CHSR.ENAx Field to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register, DMAC_EBCISR.
3. Program the following channel registers:

1. Write the starting source address in the DMAC_SADDRx register for channel x.
2. Write the starting destination address in the DMAC_DADDRx register for channel x.
3. Write the next descriptor address in the DMA_DSCRx register for channel x with 0x0..
4. Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 1 as shown in Table 31-3 on page 482. Program the DMAC_CTRLBx register with both AUTO fields set to 0.
5. Write the control information for the DMAC transfer in the DMAC_CTRLAx and DMAC_CTRLBx registers for channel x. For example, in the register, you can program the following:

- i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
- ii. Set up the transfer characteristics, such as:
  - Transfer width for the source in the SRC_WIDTH field.
  - Transfer width for the destination in the DST_WIDTH field.
  - Source AHB Master interface layer in the SIF field where source resides.
  - Destination AHB Master Interface layer in the DIF field where destination resides.
  - Incrementing/decrementing or fixed address for source in SRC_INC field.
  - Incrementing/decrementing or fixed address for destination in DST_INC field.

6. Write the channel configuration information into the DMAC_CFGx register for channel x.

- i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a ‘1’ activates the hardware handshaking interface to handle source/destination requests. Writing a ‘0’ activates the software handshaking interface to handle source/destination requests.
- ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

7. If source Picture-in-Picture mode is enabled (DMAC_CTRLBx.SRC_PIP is enabled), program the DMAC_SPIPx register for channel x.
8. If destination Picture-in-Picture mode is enabled (DMAC_CTRLBx.DST_PIP is enabled), program the DMAC_DPIPx register for channel x.

4. After the DMAC selected channel has been programmed, enable the channel by writing a ‘1’ to the DMAC_CHER.ENAx bit, where x is the channel number. Make sure that bit 0 of DMAC_EN.ENABLE register is enabled.
5. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
6. Once the transfer completes, the hardware sets the interrupts and disables the channel. At this time, you can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the Channel Handler Status Register (DMAC_CHSR.ENAx) bit until it is cleared by hardware, to detect when the transfer is complete.

Multi-buffer Transfer with Linked List for Source and Linked List for Destination (Row 4)

1. Read the Channel Handler Status register to choose a free (disabled) channel.
2. Set up the chain of Linked List Items (otherwise known as buffer descriptors) in memory. Write the control information in the LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers location of the buffer descriptor for each LLI in memory (see Figure 31-6 on page 486) for channel x. For example, in the register, you can program the following:
   1. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
   2. Set up the transfer characteristics, such as:

- i. Transfer width for the source in the SRC_WIDTH field.
- ii. Transfer width for the destination in the DST_WIDTH field.

- iii. Source AHB master interface layer in the SIF field where source resides.
- iv. Destination AHB master interface layer in the DIF field where destination resides.
- v. Incrementing/decrementing or fixed address for source in SRC_INCR field.
- vi. Incrementing/decrementing or fixed address for destination DST_INCR field.

3. Write the channel configuration information into the DMAC_CFGx register for channel x.
    1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.
    2. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.
4. Make sure that the LLI.DMAC_CTRLBx register locations of all LLI entries in memory (except the last) are set as shown in Row 4 of Table 31-3 on page 482. The LLI.DMAC_CTRLBx register of the last Linked List Item must be set as described in Row 1 of Table 31-3. Figure 31-5 on page 481 shows a Linked List example with two list items.
5. Make sure that the LLI.DMAC_DSCRx register locations of all LLI entries in memory (except the last) are non-zero and point to the base address of the next Linked List Item.
6. Make sure that the LLI.DMAC_SADDRx/LLI.DMAC_DADDRx register locations of all LLI entries in memory point to the start source/destination buffer address preceding that LLI fetch.
7. Make sure that the LLI.DMAC_CTRLAx.DONE field of the LLI.DMAC_CTRLAx register locations of all LLI entries in memory are cleared.
8. If source Picture-in-Picture mode is enabled (DMAC_CTRLBx.SRC_PIP is enabled), program the DMAC_SPIPx register for channel x.
9. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DST_PIP is enabled), program the DMAC_DPIPx register for channel x.
10. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the status register: DMAC_EBCISR.
11. Program the DMAC_CTRLBx, DMAC_CFGx registers according to Row 4 as shown in Table 31-3 on page 482.
12. Program the DMAC_DSCRx register with DMAC_DSCRx(0), the pointer to the first Linked List item.
13. Finally, enable the channel by writing a '1' to the DMAC_CHER.ENAx bit, where x is the channel number. The transfer is performed.
14. The DMAC fetches the first LLI from the location pointed to by DMAC_DSCRx(0).

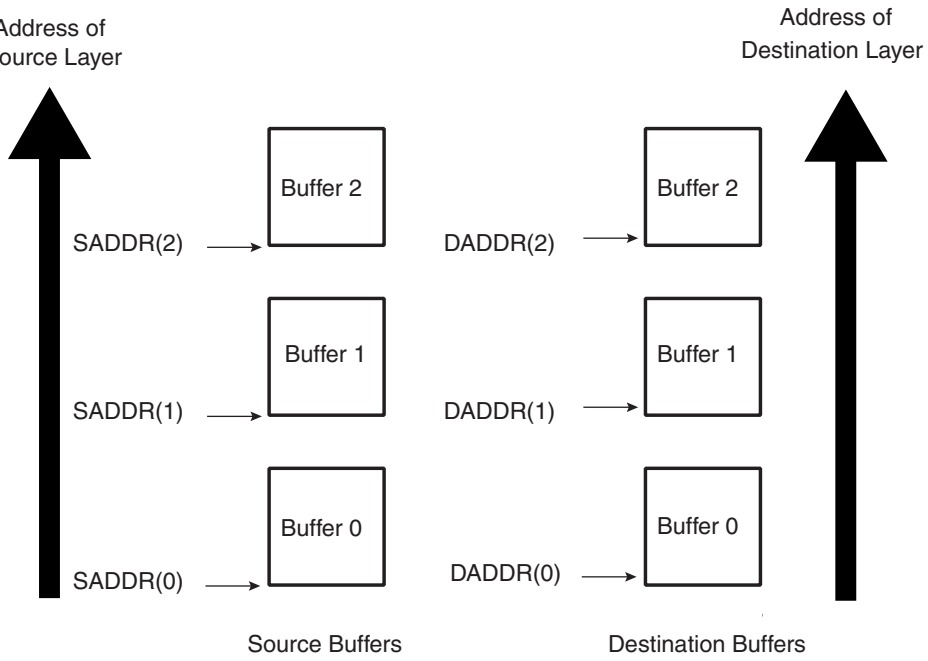
Note: The LLI.DMAC_SADDRx, LLI. DMAC_DADDRx, LLI.DMAC_DSCRx, LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The DMAC automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLBx and DMAC_CTRLAx channel registers from the DMAC_DSCRx(0).

15. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
16. Once the buffer of data is transferred, the DMAC_CTRLAx register is written out to system memory at the same location and on the same layer (DMAC_DSCRx.DSCR_IF) where it was originally fetched, that is, the location of the DMAC_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAX.DONE bits have been updated by DMAC hardware. Additionally, the DMAC_CTRLAx.DONE bit is asserted when the buffer transfer has completed.

Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLAx.DONE bit was cleared at the start of the transfer.

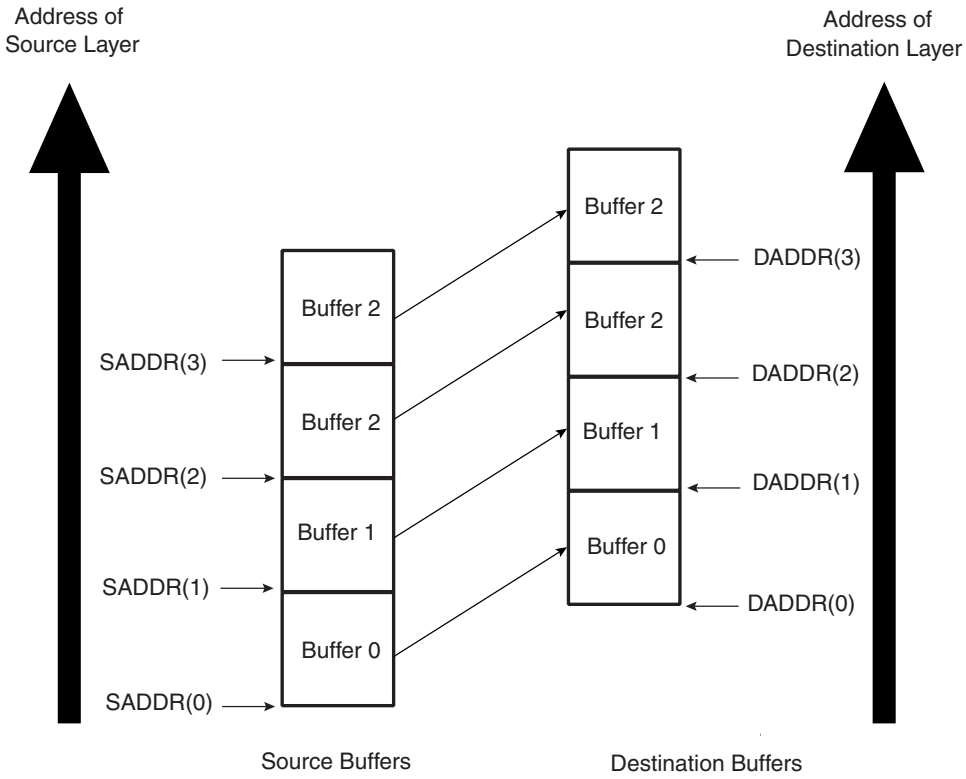
17. The DMAC does not wait for the buffer interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current DMAC_DSCRx register and automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx channel registers. The DMAC transfer continues until the DMAC determines that the DMAC_CTRLBx and DMAC_DSCRx registers at the end of a buffer transfer match described in Row 1 of Table 31-3 on page 482. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer. The DMAC transfer might look like that shown in Figure 31-6 on page 486.

Figure 31-6. Multi-buffer with Linked List Address for Source and Destination



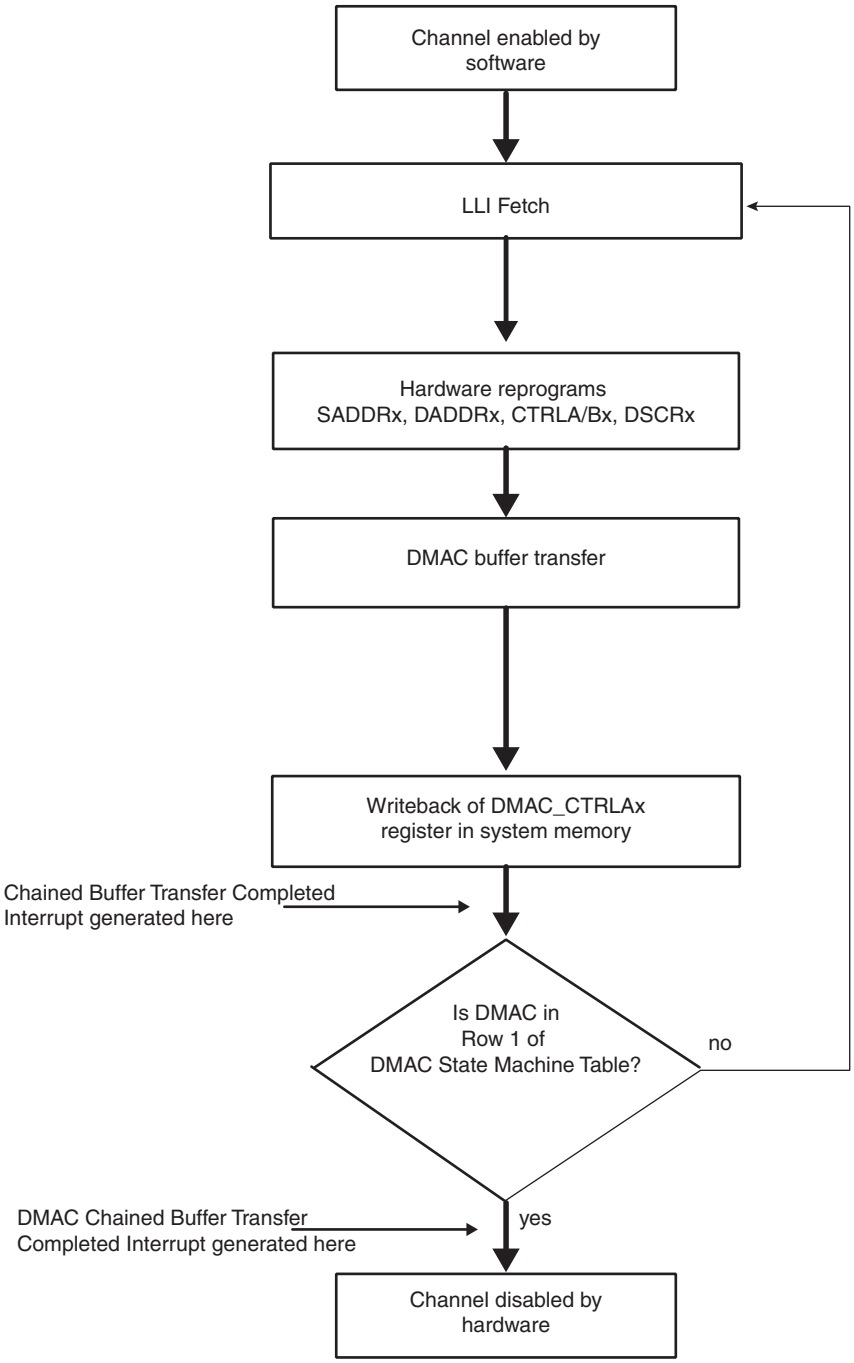
If the user needs to execute a DMAC transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum buffer size DMAC_CTRLAx.BTSIZE, then this can be achieved using the type of multi-buffer transfer as shown in Figure 31-7 on page 487.

Figure 31-7. Multi-buffer with Linked Address for Source and Destination Buffers are Contiguous



The DMAC transfer flow is shown in Figure 31-8 on page 488.

Figure 31-8. DMAC Transfer Flow for Source and Destination Linked List Address



Multi-buffer Transfer with Source Address Auto-reloaded and Destination Address Auto-reloaded (Row 10)

1. Read the Channel Handler Status register to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register. Program the following channel registers:
    1. Write the starting source address in the DMAC_SADDRx register for channel x.
    2. Write the starting destination address in the DMAC_DADDRx register for channel x.
    3. Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 10 as shown in Table 31-3 on page 482. Program the DMAC_DSCRx register with ‘0’.
    4. Write the control information for the DMAC transfer in the DMAC_CTRLAx and DMAC_CTRLBx register for channel x. For example, in the register, you can program the following:
    - i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
    - ii. Set up the transfer characteristics, such as:
        - Transfer width for the source in the SRC_WIDTH field.
        - Transfer width for the destination in the DST_WIDTH field.
        - Source AHB master interface layer in the SIF field where source resides.
        - Destination AHB master interface layer in the DIF field where destination resides.
        - Incrementing/decrementing or fixed address for source in SRC_INCR field.
        - Incrementing/decrementing or fixed address for destination in DST_INCR field.
    5. If source Picture-in-Picture mode is enabled (DMAC_CTRLBx.SPIP is enabled), program the DMAC_SPIPx register for channel x.
    6. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP), program the DMAC_DPIPx register for channel x.
    7. Write the channel configuration information into the DMAC_CFGx register for channel x. Ensure that the reload bits, DMAC_CFGx.SRC_REP, DMAC_CFGx.DST_REP and DMAC_CTRLBx.AUTO are enabled.
    - i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_h2SEL bits, respectively. Writing a ‘1’ activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a ‘0’ activates the software handshaking interface to handle source/destination requests.
    - ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.
3. After the DMAC selected channel has been programmed, enable the channel by writing a ‘1’ to the DMAC_CHER.ENAx bit where the channel number is. Make sure that bit 0 of the DMAC_EN register is enabled.
4. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges on completion of each chunk/single transaction and carries out the buffer transfer.
5. When the buffer transfer has completed, the DMAC reloads the DMAC_SADDRx, DMAC_DADDRx and DMAC_CTRLAx registers. The hardware sets the Buffer Transfer Completed Interrupt. The DMAC then samples the row number as shown in Table 31-3 on page 482. If the DMAC is in Row 1, then the DMAC transfer has completed. The hardware sets the Chained Buffer Transfer Completed Interrupt and disables the channel. So you can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the Channel Enable in the Channel Status Register (DMAC_CHSR.ENAx) until it is disabled, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
6. The DMAC transfer proceeds as follows:

1. If the Buffer Transfer Completed Interrupt is unmasked (DMAC_EBCIMR.BTCx = ‘1’, where x is the channel number), the hardware sets the Buffer Transfer Completed Interrupt when the buffer transfer has completed. It then stalls until the STALx bit of DMAC_CHSR register is cleared by software, writing ‘1’ to DMAC_CHER.KEEPx bit, where x is the channel number. If the next buffer is to be the last buffer in the DMAC transfer, then the buffer complete ISR (interrupt service routine) should clear the automatic mode bit in the DMAC_CTRLBx.AUTO bit. This puts the DMAC into Row 1 as shown in Table 31-3 on page 482. If the next buffer is not the last buffer in the DMAC transfer, then the reload bits should remain enabled to keep the DMAC in Row 4.
2. If the Buffer Transfer Completed Interrupt is masked (DMAC_EBCIMR.BTCx = ‘0’, where x is the channel number), the hardware does not stall until it detects a write to the Buffer Transfer Completed Interrupt Enable register DMAC_EBCIER register, but starts the next buffer transfer immediately. In this case, the software must clear the automatic mode bit in the DMAC_CTRLB to put the DMAC into ROW 1 of Table 31-3 on page 482 before the last buffer of the DMAC transfer has completed. The transfer is similar to that shown in Figure 31-9 on page 490. The DMAC transfer flow is shown in Figure 31-10 on page 491.

Figure 31-9. Multi-buffer DMAC Transfer with Source and Destination Address Auto-reloaded

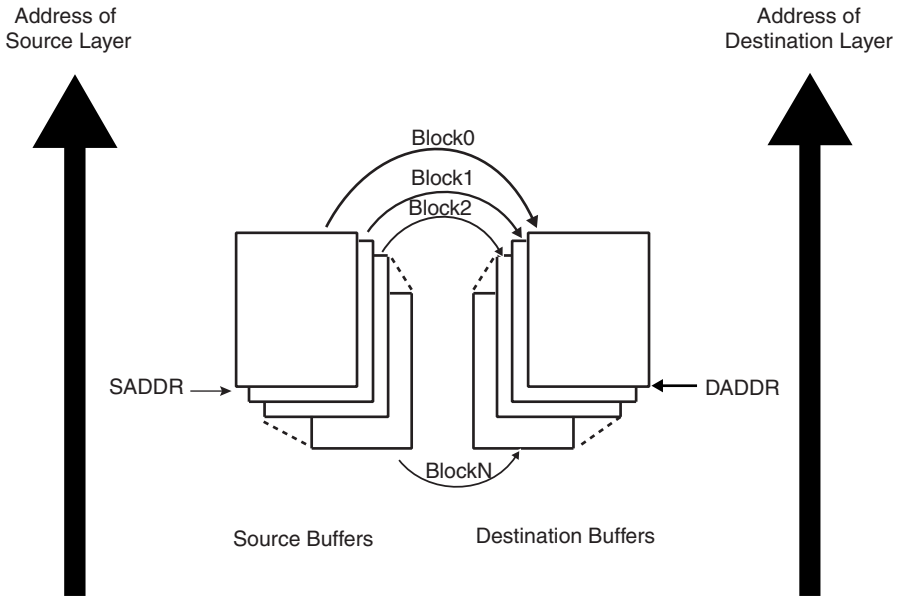
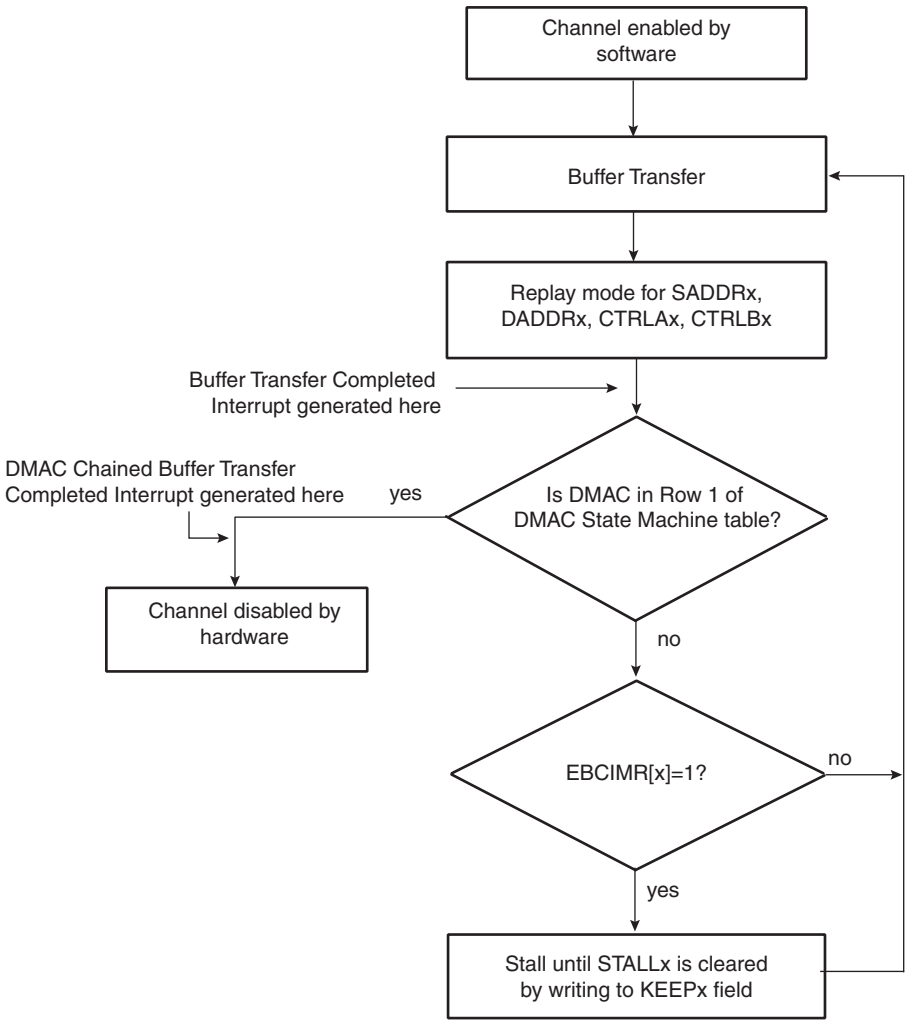


Figure 31-10.DMAC Transfer Flow for Source and Destination Address Auto-reloaded



Multi-buffer Transfer with Source Address Auto-reloaded and Linked List Destination Address (Row 6)

1. Read the Channel Handler Status register to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as buffer descriptors) in memory. Write the control information in the LLI.DMAC_CTRLAx and DMAC_CTRLBx registers location of the buffer descriptor for each LLI in memory for channel x. For example, in the register, you can program the following:
    1. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control peripheral by programming the FC of the DMAC_CTRLBx register.
    2. Set up the transfer characteristics, such as:
    - i. Transfer width for the source in the SRC_WIDTH field.
    - ii. Transfer width for the destination in the DST_WIDTH field.
    - iii. Source AHB master interface layer in the SIF field where source resides.
    - iv. Destination AHB master interface layer in the DIF field where destination resides.
    - v. Incrementing/decrementing or fixed address for source in SRC_INCR field.
    - vi. Incrementing/decrementing or fixed address for destination DST_INCR field.
3. Write the starting source address in the DMAC_SADDRx register for channel x.

Note: The values in the LLI.DMAC_SADDRx register locations of each of the Linked List Items (LLIs) set up 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.

1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface source/destination requests.
2. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

5. Make sure that the LLI.DMAC_CTRLBx register locations of all LLIs in memory (except the last one) are set as shown in Row 6 of Table 31-3 on page 482 while the LLI.DMAC_CTRLBx register of the last Linked List item must be set as described in Row 1 of Table 31-3. Figure 31-5 on page 481 shows a Linked List example with two list items.
6. Make sure that the LLI.DMAC_DSCRx register locations of all LLIs in memory (except the last one) are non-zero and point to the next Linked List Item.
7. Make sure that the LLI.DMAC_DADDRx register locations of all LLIs in memory point to the start destination buffer address proceeding that LLI fetch.
8. Make sure that the LLI.DMAC_CTLx.DONE field of the LLI.DMAC_CTRLA register locations of all LLIs in memory is cleared.
9. If source Picture-in-Picture is enabled (DMAC_CTRLBx.SPIP is enabled), program the DMAC_SPIPx register for channel x.
10. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP is enabled), program the DMAC_DPIPx register for channel x.
11. Clear any pending interrupts on the channel from the previous DMAC transfer by reading to the DMAC_EBCISR register.
12. Program the DMAC_CTLx and DMAC_CFGx registers according to Row 6 as shown in Table 31-3 on page 482.
13. Program the DMAC_DSCRx register with DMAC_DSCRx(0), the pointer to the first Linked List item.
14. Finally, enable the channel by writing a '1' to the DMAC_CHER.ENAx bit, where x is the channel number. The transfer is performed. Make sure that bit 0 of the DMAC_EN register is enabled.
15. The DMAC fetches the first LLI from the location pointed to by DMAC_DSCRx(0).

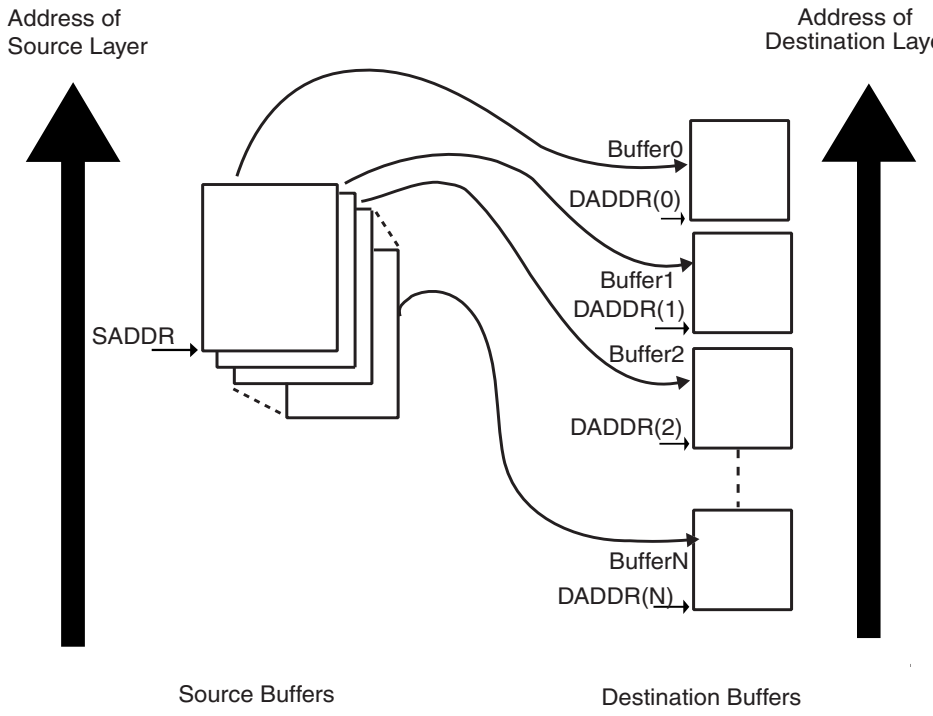
Note: The LLI.DMAC_SADDRx, LLI.DMAC_DADDRx, LLI. DMAC_LLPx LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The LLI.DMAC_SADDRx register, although fetched, is not used.

16. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
17. The DMAC_CTRLAx register is written out to the system memory. The DMAC_CTRLAx register is written out to the same location on the same layer (DMAC_DSCRx.DSCR_IF) where it was originally fetched, that is the location of the DMAC_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out, because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAx.DONE fields have been updated by hardware within the DMAC. The LLI.DMAC_CTRLAx.DONE bit is asserted to indicate buffer completion. Therefore, the software can poll the LLI.DMAC_CTRLAx.DONE field of the DMAC_CTRLAx register in the LLi to ascertain when a buffer transfer has completed.

Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the polled LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLA.DONE bit was cleared at the start of the transfer.

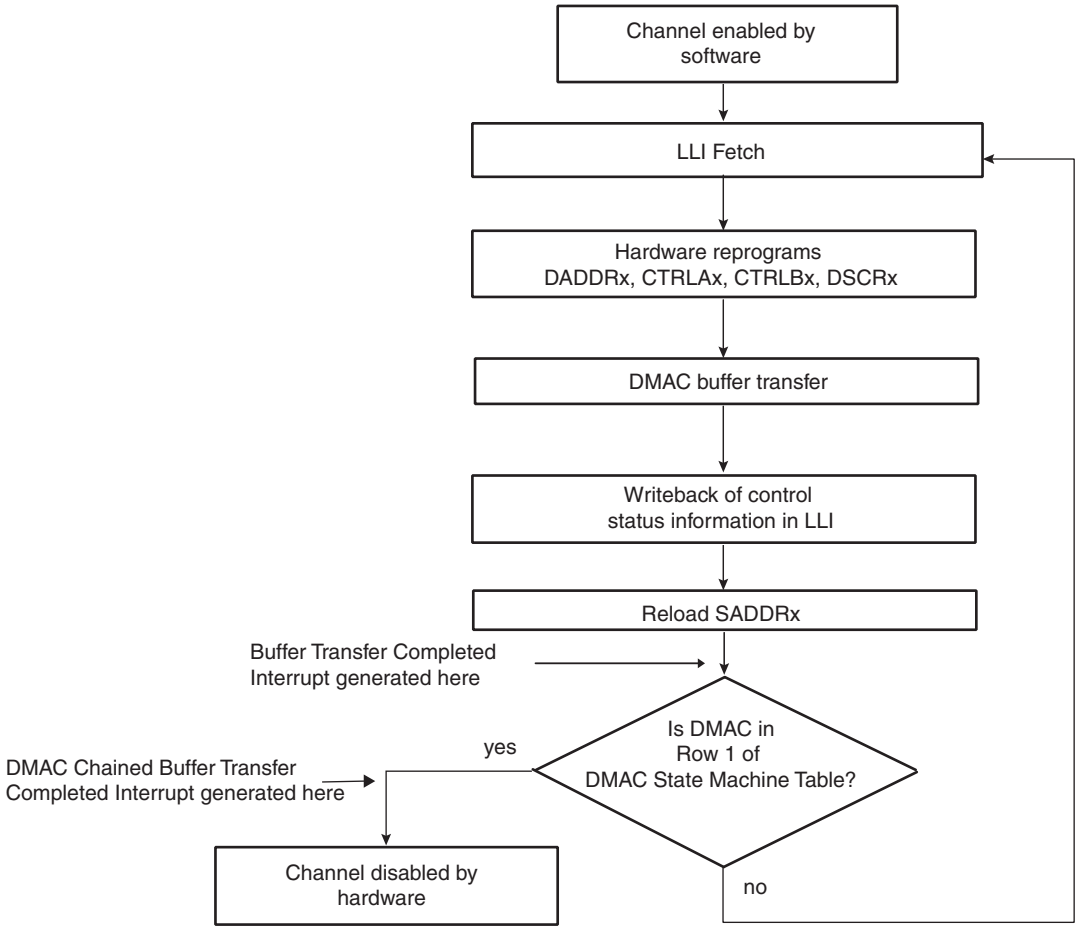
18. The DMAC reloads the DMAC_SADDRx register from the initial value. The hardware sets the Buffer Transfer Completed Interrupt. The DMAC samples the row number as shown in Table 31-3 on page 482. If the DMAC is in Row 1, then the DMAC transfer has completed. The hardware sets the Chained Buffer Transfer Completed Interrupt and disables the channel. You can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the Channel Enable. (DMAC_CHSR.ENAx) bit until it is cleared by hardware, to detect when the transfer is complete. If the DMAC is not in Row 1 as shown in Table 31-3 on page 482, the following step is performed.
19. The DMAC fetches the next LLI from the memory location pointed to by the current DMAC_DSCRx register, and automatically reprograms the DMAC_DADDRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx channel registers. Note that the DMAC_SADDRx is not re-programmed as the reloaded value is used for the next DMAC buffer transfer. If the next buffer is the last buffer of the DMAC transfer, then the DMAC_CTRLBx and DMAC_DSCRx registers just fetched from the LLI should match Row 1 of Table 31-3 on page 482. The DMAC transfer might look like that shown in Figure 31-11 on page 493.

Figure 31-11.Multi-buffer DMAC Transfer with Source Address Auto-reloaded and Linked List Destination Address



The DMAC Transfer flow is shown in Figure 31-12 on page 494.

Figure 31-12.DMAC Transfer Flow for Replay Mode at Source and Linked List Destination Address



Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address (Row 11)

1. Read the Channel Handler Status register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading to the Interrupt Status Register.
3. Program the following channel registers:
    1. Write the starting source address in the DMAC_SADDRx register for channel x.
    2. Write the starting destination address in the DMAC_DADDRx register for channel x.
    3. Program DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx according to Row 11 as shown in Table 31-3 on page 482. Program the DMAC_DSCRx register with ‘0’. DMAC_CTRLBx.AUTO field is set to ‘1’ to enable automatic mode support.
    4. Write the control information for the DMAC transfer in the DMAC_CTRLBx and DMAC_CTRLAx register for channel x. For example, in this register, you can program the following:

- i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
- ii. Set up the transfer characteristics, such as:
    - Transfer width for the source in the SRC_WIDTH field.
    - Transfer width for the destination in the DST_WIDTH field.
    - Source AHB master interface layer in the SIF field where source resides.
    - Destination AHB master interface master layer in the DIF field where destination resides.

- Incrementing/decrementing or fixed address for source in SRC_INCR field.
- Incrementing/decrementing or fixed address for destination in DST_INCR field.

5. If source Picture-in-Picture is enabled (DMAC_CTRLBx.SPIP is enabled), program the DMAC_SPIPx register for channel x.
6. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP), program the DMAC_DPIPx register for channel x.
7. Write the channel configuration information into the DMAC_CFGx register for channel x.

- i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.
- ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

4. After the DMAC channel has been programmed, enable the channel by writing a '1' to the DMAC_CHER.ENAx bit, where x is the channel number. Make sure that bit 0 of the DMAC_EN.ENABLE register is enabled.
5. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
6. When the buffer transfer has completed, the DMAC reloads the DMAC_SADDRx register. The DMAC_DADDRx register remains unchanged. The hardware sets the Buffer Transfer Completed Interrupt. The DMAC then samples the row number as shown in Table 31-3 on page 482. If the DMAC is in Row 1, then the DMAC transfer has completed. The hardware sets the Chained Buffer Transfer Completed Interrupt and disables the channel. So you can either respond to the Buffer Transfer Completed Interrupt or Chained Buffer Transfer Completed Interrupt, or poll for the enable (ENAx) field in the Channel Status Register (DMAC_CHSR.ENAx bit) until it is cleared by hardware, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
7. The DMAC transfer proceeds as follows:
    1. If the Buffer Transfer Completed Interrupt is unmasked (DMAC_EBCIMR.BTCx = '1', where x is the channel number), the hardware sets the Buffer Transfer Completed Interrupt when the buffer transfer has completed. It then stalls until STALx bit of DMAC_CHSR is cleared by writing in the KEEPx field of DMAC_CHER register, where x is the channel number. If the next buffer is to be the last buffer in the DMAC transfer, then the buffer complete ISR (interrupt service routine) should clear the automatic mode bit, DMAC_CTRLBx.AUTO. This puts the DMAC into Row 1 as shown in Table 31-3 on page 482. If the next buffer is not the last buffer in the DMAC transfer, then the automatic transfer mode bit should remain enabled to keep the DMAC in Row 11 as shown in Table 31-3 on page 482.
    2. If the Buffer Transfer Completed Interrupt is masked (DMAC_EBCIMR.BTCx = '0', where x is the channel number), the hardware does not stall until it detects a write to the Buffer Transfer Completed Interrupt Enable register, but starts the next buffer transfer immediately. In this case, the software must clear the automatic mode bit, DMAC_CTRLBx.AUTO, to put the device into ROW 1 of Table 31-3 on page 482 before the last buffer of the DMAC transfer has completed.

The transfer is similar to that shown in Figure 31-13 on page 496.

The DMAC Transfer flow is shown in Figure 31-14 on page 497.

Figure 31-13.Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address

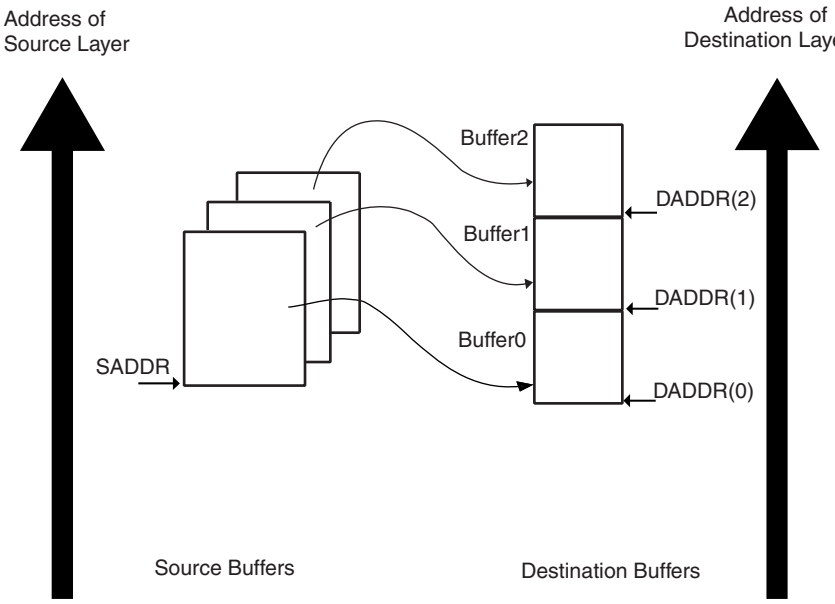
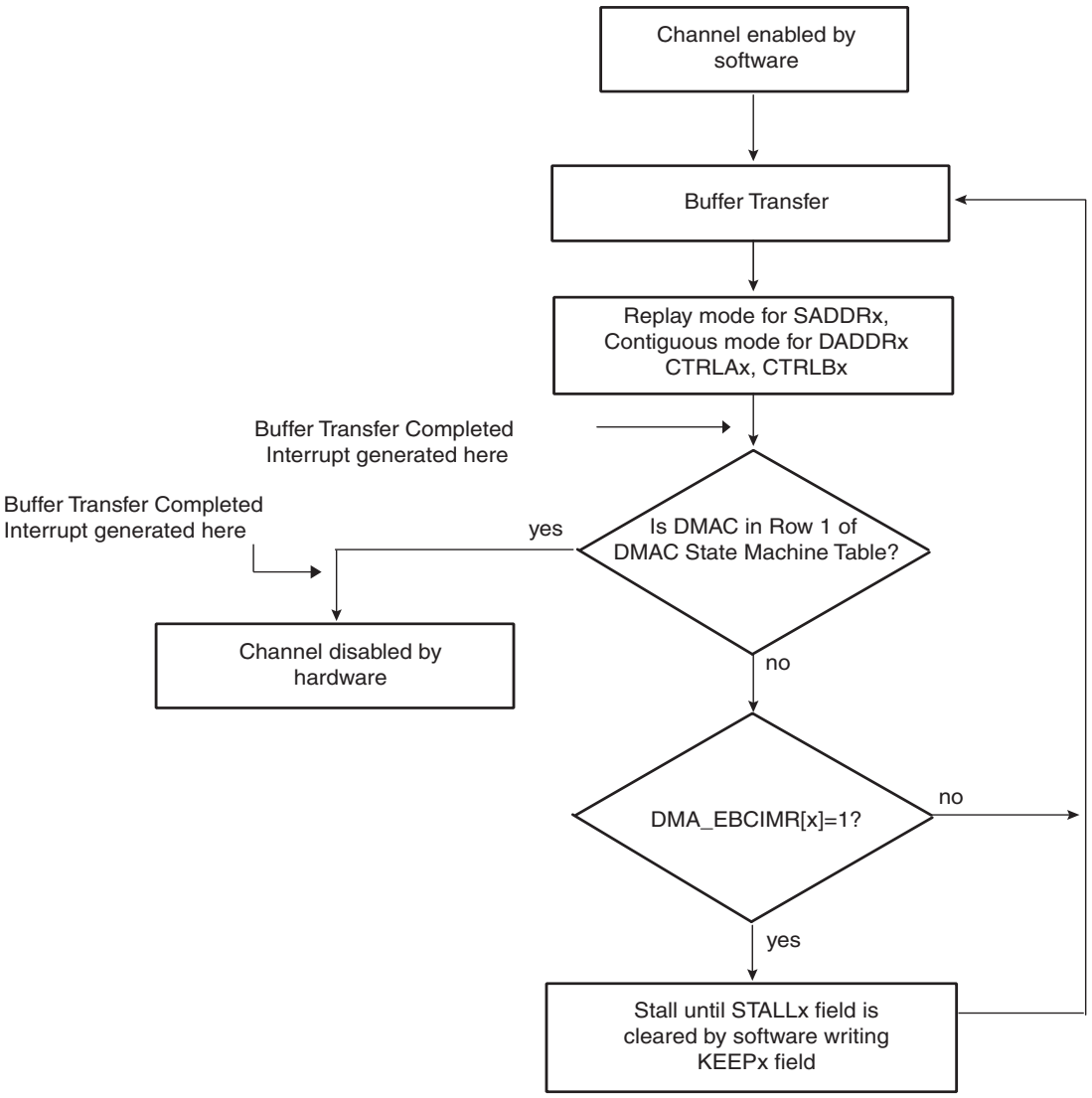


Figure 31-14.DMAC Transfer Replay Mode is Enabled for the Source and Contiguous Destination Address



Multi-buffer DMAC Transfer with Linked List for Source and Contiguous Destination Address (Row 2)

1. Read the Channel Handler Status register to choose a free (disabled) channel.
2. Set up the linked list in memory. Write the control information in the LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx register location of the buffer descriptor for each LLI in memory for channel x. For example, in the register, you can program the following:
    1. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
    2. Set up the transfer characteristics, such as:
    - i. Transfer width for the source in the SRC_WIDTH field.
    - ii. Transfer width for the destination in the DST_WIDTH field.
    - iii. Source AHB master interface layer in the SIF field where source resides.
    - iv. Destination AHB master interface layer in the DIF field where destination resides.
    - v. Incrementing/decrementing or fixed address for source in SRC_INCR field.
    - vi. Incrementing/decrementing or fixed address for destination DST_INCR field.
3. Write the starting destination address in the DMAC_DADDRx register for channel x.

Note: The values in the LLI.DMAC_DADDRx register location of each Linked List Item (LLI) in memory, although fetched during an LLI fetch, are not used.

4. Write the channel configuration information into the DMAC_CFGx register for channel x.
    1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.
    2. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripherals. This requires programming the SRC_PER and DST_PER bits, respectively.
5. Make sure that all LLI.DMAC_CTRLBx register locations of the LLI (except the last) are set as shown in Row 2 of Table 31-3 on page 482, while the LLI.DMAC_CTRLBx register of the last Linked List item must be set as described in Row 1 of Table 31-3. Figure 31-5 on page 481 shows a Linked List example with two list items.
6. Make sure that the LLI.DMAC_DSCRx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Make sure that the LLI.DMAC_SADDRx register locations of all LLIs in memory point to the start source buffer address proceeding that LLI fetch.
8. Make sure that the LLI.DMAC_CTRLAx.DONE field of the LLI.DMAC_CTRLAx register locations of all LLIs in memory is cleared.
9. If source Picture-in-Picture is enabled (DMAC_CTRLBx.SPIP is enabled), program the DMAC_SPIPx register for channel x.
10. If destination Picture-in-Picture is enabled (DMAC_CTRLBx.DPIP is enabled), program the DMAC_DPIPx register for channel x.
11. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register.
12. Program the DMAC_CTRLAx, DMAC_CTRLBx and DMAC_CFGx registers according to Row 2 as shown in Table 31-3 on page 482
13. Program the DMAC_DSCRx register with DMAC_DSCRx(0), the pointer to the first Linked List item.
14. Finally, enable the channel by writing a '1' to the DMAC_CHER.ENAx bit. The transfer is performed. Make sure that bit 0 of the DMAC_EN register is enabled.
15. The DMAC fetches the first LLI from the location pointed to by DMAC_DSCRx(0).

Note: The LLI.DMAC_SADDRx, LLI.DMAC_DADDRx, LLI.DMAC_DSCRx and LLI.DMAC_CTRLA/Bx registers are fetched. The LLI.DMAC_DADDRx register location of the LLI, although fetched, is not used. The DMAC_DADDRx register in the DMAC remains unchanged.

16. Source and destination requests single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
17. Once the buffer of data is transferred, the DMAC_CTRLAx register is written out to the system memory at the same location and on the same layer (DMAC_DSCRx.DSCR_IF) where it was originally fetched, that is, the location of the DMAC_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAX.DONE fields have been updated by DMAC hardware. Additionally, the DMAC_CTRLAx.DONE bit is asserted when the buffer transfer has completed.

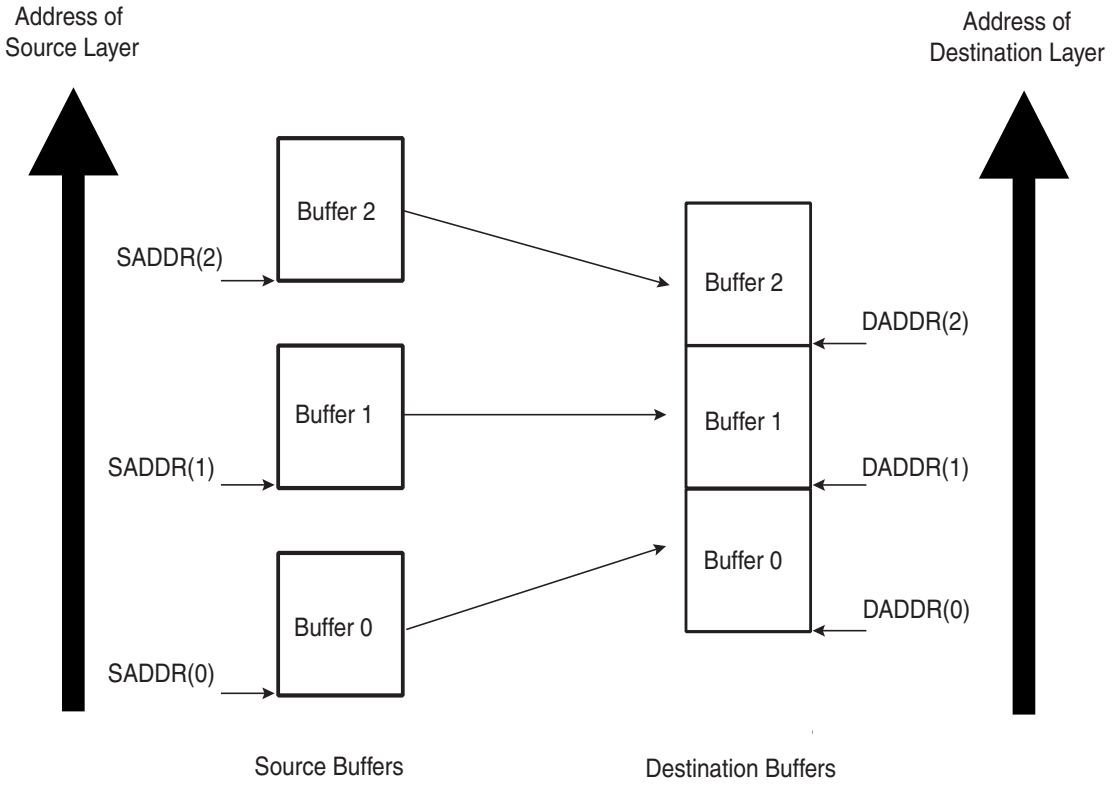
Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLAx.DONE bit was cleared at the start of the transfer.

18. The DMAC does not wait for the buffer interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by the current DMAC_DSCRx register, then automatically reprograms the

DMAC_SADDRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx channel registers. The DMAC_DADDRx register is left unchanged. The DMAC transfer continues until the DMAC samples the DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx registers at the end of a buffer transfer match that described in Row 1 of Table 31-3 on page 482. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer.

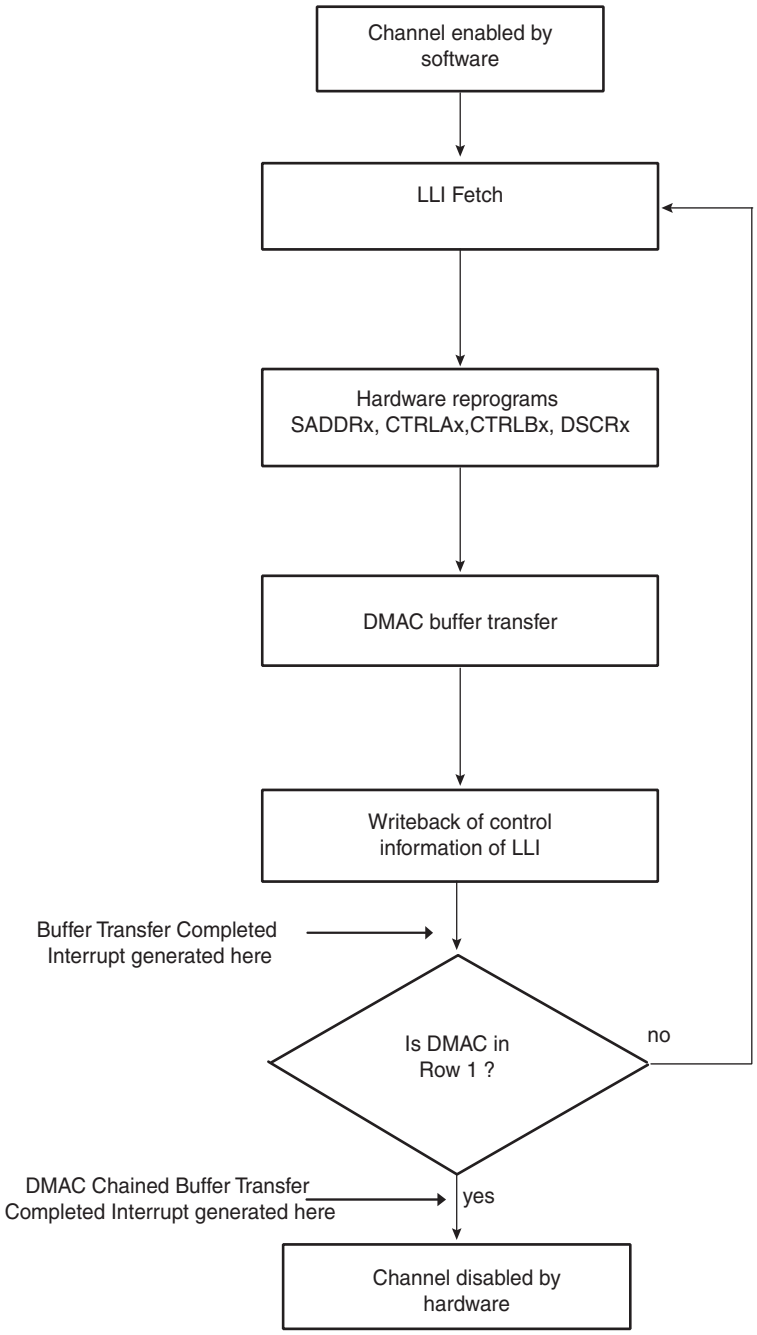
The DMAC transfer might look like that shown in Figure 31-15 on page 499. Note that the destination address is decrementing.

Figure 31-15.DMAC Transfer with Linked List Source Address and Contiguous Destination Address



The DMAC transfer flow is shown in Figure 31-16 on page 500.

Figure 31-16.DMAC Transfer Flow for Linked List Source Address and Contiguous Destination Address



31.4.6 Disabling a Channel Prior to Transfer Completion

Under normal operation, the software enables a channel by writing a ‘1’ to the Channel Handler Enable Register, DMAC_CHER.ENAx, and the hardware disables a channel on transfer completion by clearing the DMAC_CHSR.ENAx register bit.

The recommended way for software to disable a channel without losing data is to use the SUSPx bit in conjunction with the EMPTx bit in the Channel Handler Status Register.

1. If the software wishes to disable a channel n prior to the DMAC transfer completion, then it can set the DMAC_CHER.SUSPx bit to tell the DMAC to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
2. The software can now poll the DMAC_CHSR.EMPTx bit until it indicates that the channel n FIFO is empty, where n is the channel number.
3. The DMAC_CHER.ENAx bit can then be cleared by software once the channel n FIFO is empty, where n is the channel number.

When DMAC_CTRLAx.SRC_WIDTH is less than DMAC_CTRLAx.DST_WIDTH and the DMAC_CHSRx.SUSPx bit is high, the DMAC_CHSRx.EMPTx is asserted once the contents of the FIFO does not permit a single word of DMAC_CTRLAx.DST_WIDTH to be formed. However, there may still be data in the channel FIFO but not enough to form a single transfer of DMAC_CTLx.DST_WIDTH width. In this configuration, once the channel is disabled, the remaining data in the channel FIFO are not transferred to the destination peripheral. It is permitted to remove the channel from the suspension state by writing a ‘1’ to the DMAC_CHER.RESx field register. The DMAC transfer completes in the normal manner. n defines the channel number.

Note: If a channel is disabled by software, an active single or chunk transaction is not guaranteed to receive an acknowledgement.

31.4.6.1 Abnormal Transfer Termination

A DMAC transfer may be terminated abruptly by software by clearing the channel enable bit, DMAC_CHDR.ENAx, where x is the channel number. This does not mean that the channel is disabled immediately after the DMAC_CHSR.ENAx bit is cleared over the APB interface. Consider this as a request to disable the channel. The DMAC_CHSR.ENAx must be polled and then it must be confirmed that the channel is disabled by reading back 0.

The software may terminate all channels abruptly by clearing the global enable bit in the DMAC Configuration Register (DMAC_EN.ENABLE bit). Again, this does not mean that all channels are disabled immediately after the DMAC_EN.ENABLE is cleared over the APB slave interface. Consider this as a request to disable all channels. The DMAC_CHSR.ENABLE must be polled and then it must be confirmed that all channels are disabled by reading back ‘0’.

Note: If the channel enable bit is cleared while there is data in the channel FIFO, this data is not sent to the destination peripheral and is not present when the channel is re-enabled. For read sensitive source peripherals, such as a source FIFO, this data is therefore lost. When the source is not a read sensitive device (i.e., memory), disabling a channel without waiting for the channel FIFO to empty may be acceptable as the data is available from the source peripheral upon request and is not lost.

Note: If a channel is disabled by software, an active single or chunk transaction is not guaranteed to receive an acknowledgement.

31.5 DMAC Software Requirements

- There must not be any write operation to Channel registers in an active channel after the channel enable is made HIGH. If any channel parameters must be reprogrammed, this can only be done after disabling the DMAC channel.
- When the destination peripheral has been defined as the flow controller, source single transfer requests are not serviced until the destination peripheral has asserted its Last Transfer Flag.
- When the source peripheral has been defined as the flow controller, destination single transfer requests are not serviced until the source peripheral has asserted its Last Transfer Flag.
- When the destination peripheral has been defined as the flow controller, if the destination width is smaller than the source width, then a data loss may occur, and the loss is equal to the Source Single Transfer size in bytes-destination Single Transfer size in bytes.
- When a Memory to Peripheral transfer occurs, if the destination peripheral has been defined as the flow controller, then a prefetch operation is performed. It means that data is extracted from the memory before any request from the peripheral is generated.
- You must program the DMAC_SADDRx and DMAC_DADDRx channel registers with a byte, half-word and word aligned address depending on the source width and destination width.
- After the software disables a channel by writing into the channel disable register, it must re-enable the channel only after it has polled a 0 in the corresponding channel enable status register. This is because the current AHB Burst must terminate properly.
- If you program the BTSIZE field in the DMAC_CTRLA as zero, and the DMAC has been defined as the flow controller, then the channel is automatically disabled.
- When hardware handshaking interface protocol is fully implemented, a peripheral is expected to deassert any sreq or breq signals on receiving the ack signal irrespective of the request the ack was asserted in response to.
- Multiple Transfers involving the same peripheral must not be programmed and enabled on different channels, unless this peripheral integrates several hardware handshaking interfaces.
- When a Peripheral has been defined as the flow controller, the targeted DMAC Channel must be enabled before the Peripheral. If you do not ensure this and the First DMAC request is also the last transfer, the DMAC Channel might miss a Last Transfer Flag.
- When the AUTO Field is set to TRUE, then the BTSIZE Field is automatically reloaded from its previous value. BTSIZE must be initialized to a non zero value if the first transfer is initiated with the AUTO field set to TRUE, even if LLI mode is enabled, because the LLI fetch operation will not update this field.

31.6 Write Protection Registers

To prevent any single software error that may corrupt the DMAC behavior, the DMAC address space can be write-protected by setting the WPEN bit in the "DMAC Write Protect Mode Register" (DMAC_WPMR).

If a write access to anywhere in the DMAC address space is detected, then the WPVS flag in the DMAC Write Protect Status Register (MCI_WPSR) is set, and the WPVSRC field indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the DMAC Write Protect Mode Register (DMAC_WPMR) with the appropriate access key, WPKEY.

The protected registers are:

- "DMAC Global Configuration Register" on page 505
- "DMAC Enable Register" on page 506
- "DMAC Channel x [x = 0..7] Source Address Register" on page 517
- "DMAC Channel x [x = 0..7] Destination Address Register" on page 518
- "DMAC Channel x [x = 0..7] Descriptor Address Register" on page 519
- "DMAC Channel x [x = 0..7] Control A Register" on page 520
- "DMAC Channel x [x = 0..7] Control B Register" on page 522
- "DMAC Channel x [x = 0..7] Configuration Register" on page 524
- "DMAC Channel x [x = 0..7] Source Picture-in-Picture Configuration Register" on page 526
- "DMAC Channel x [x = 0..7] Destination Picture-in-Picture Configuration Register" on page 527

31.7 DMA Controller (DMAC) User Interface

Table 31-4. Register Mapping

0x000	DMAC Global Configuration Register	DMAC_GCFG	Read-write	0x10
0x004	DMAC Enable Register	DMAC_EN	Read-write	0x0
0x008	DMAC Software Single Request Register	DMAC_SREQ	Read-write	0x0
0x00C	DMAC Software Chunk Transfer Request Register	DMAC_CREQ	Read-write	0x0
0x010	DMAC Software Last Transfer Flag Register	DMAC_LAST	Read-write	0x0
0x014	Reserved			
0x018	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer Transfer Completed Interrupt Enable register.	DMAC_EBCIER	Write-only	–
0x01C	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer Transfer Completed Interrupt Disable register.	DMAC_EBCIDR	Write-only	–
0x020	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer transfer completed Mask Register.	DMAC_EBCIMR	Read-only	0x0
0x024	DMAC Error, Chained Buffer Transfer Completed Interrupt and Buffer transfer completed Status Register.	DMAC_EBCISR	Read-only	0x0
0x028	DMAC Channel Handler Enable Register	DMAC_CHER	Write-only	–
0x02C	DMAC Channel Handler Disable Register	DMAC_CHDR	Write-only	–
0x030	DMAC Channel Handler Status Register	DMAC_CHSR	Read-only	0x00FF0000
0x034	Reserved	–	–	–
0x038	Reserved	–	–	–
0x03C+ch_num*(0x28)+(0x0)	DMAC Channel Source Address Register	DMAC_SADDR	Read-write	0x0
0x03C+ch_num*(0x28)+(0x4)	DMAC Channel Destination Address Register	DMAC_DADDR	Read-write	0x0
0x03C+ch_num*(0x28)+(0x8)	DMAC Channel Descriptor Address Register	DMAC_DSCR	Read-write	0x0
0x03C+ch_num*(0x28)+(0xC)	DMAC Channel Control A Register	DMAC_CTRLA	Read-write	0x0
0x03C+ch_num*(0x28)+(0x10)	DMAC Channel Control B Register	DMAC_CTRLB	Read-write	0x0
0x03C+ch_num*(0x28)+(0x14)	DMAC Channel Configuration Register	DMAC_CFG	Read-write	0x01000000
0x03C+ch_num*(0x28)+(0x18)	DMAC Channel Source Picture-in-Picture Configuration Register	DMAC_SPIP	Read-write	0x0
0x03C+ch_num*(0x28)+(0x1C)	DMAC Channel Destination Picture-in-Picture Configuration Register	DMAC_DPIP	Read-write	0x0
0x03C+ch_num*(0x28)+(0x20)	Reserved	–	–	–
0x03C+ch_num*(0x28)+(0x24)	Reserved	–	–	–
0x1E4	DMAC Write Protect Mode Register	DMAC_WPMR	Read-write	0x0
0x1E8	DMAC Write Protect Status Register	DMAC_WPSR	Read-only	0x0
0x01EC- 0x1FC	Reserved	–	–	–

31.7.1 DMAC Global Configuration Register

Name: DMAC_GCFG

Address: 0xFFFFEC00 (0), 0xFFFFEE00 (1)

Access: Read-write

Reset: 0x00000010

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

Note: Bit fields 0, 1, 2, 3, have a default value of 0. This should not be changed.

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” .

- **ARB_CFG: Arbiter Configuration**

0 (FIXED): Fixed priority arbiter.

1 (ROUND_ROBIN): Modified round robin arbiter.

31.7.2 DMAC Enable Register

Name: DMAC_EN

Address: 0xFFFFEC04 (0), 0xFFFFEE04 (1)

Access: Read-write

Reset: 0x00000000

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

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” .

- **ENABLE: General Enable of DMA**

0: DMA Controller is disabled.

1: DMA Controller is enabled.

31.7.3 DMAC Software Single Request Register

Name: DMAC_SREQ

Address: 0xFFFFEC08 (0), 0xFFFFEE08 (1)

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DSREQ7	SSREQ7	DSREQ6	SSREQ6	DSREQ5	SSREQ5	DSREQ4	SSREQ4
7	6	5	4	3	2	1	0
DSREQ3	SSREQ3	DSREQ2	SSREQ2	DSREQ1	SSREQ1	DSREQ0	SSREQ0

- **DSREQx: Destination Request**

Request a destination single transfer on channel i.

- **SSREQx: Source Request**

Request a source single transfer on channel i.

31.7.4 DMAC Software Chunk Transfer Request Register

Name: DMAC_CREQ

Address: 0xFFFFEC0C (0), 0xFFFFEE0C (1)

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DCREQ7	SCREQ7	DCREQ6	SCREQ6	DCREQ5	SCREQ5	DCREQ4	SCREQ4
7	6	5	4	3	2	1	0
DCREQ3	SCREQ3	DCREQ2	SCREQ2	DCREQ1	SCREQ1	DCREQ0	SCREQ0

- **DCREQx: Destination Chunk Request**

Request a destination chunk transfer on channel i.

- **SCREQx: Source Chunk Request**

Request a source chunk transfer on channel i.

31.7.5 DMAC Software Last Transfer Flag Register

Name: DMAC_LAST

Address: 0xFFFFEC10 (0), 0xFFFFEE10 (1)

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DLAST7	SLAST7	DLAST6	SLAST6	DLAST5	SLAST5	DLAST4	SLAST4
7	6	5	4	3	2	1	0
DLAST3	SLAST3	DLAST2	SLAST2	DLAST1	SLAST1	DLAST0	SLAST0

- **DLASTx: Destination Last**

Writing one to DLASTx prior to writing one to DSREQx or DCREQx indicates that this destination request is the last transfer of the buffer.

- **SLASTx: Source Last**

Writing one to SLASTx prior to writing one to SSREQx or SCREQx indicates that this source request is the last transfer of the buffer.

31.7.6 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Enable Register

Name: DMAC_EBCIER

Address: 0xFFFFEC18 (0), 0xFFFFEE18 (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

- **BTCx: Buffer Transfer Completed [7:0]**

Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the BTC field to enable the interrupt for channel i.

- **CBTCx: Chained Buffer Transfer Completed [7:0]**

Chained Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the CBTC field to enable the interrupt for channel i.

- **ERRx: Access Error [7:0]**

Access Error Interrupt Enable Register. Set the relevant bit in the ERR field to enable the interrupt for channel i.

31.7.7 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Disable Register

Name: DMAC_EBCIDR

Address: 0xFFFFEC1C (0), 0xFFFFEE1C (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

- **BTCx: Buffer Transfer Completed [7:0]**

Buffer transfer completed Disable Interrupt Register. When set, a bit of the BTC field disables the interrupt from the relevant DMAC channel.

- **CBTCx: Chained Buffer Transfer Completed [7:0]**

Chained Buffer transfer completed Disable Register. When set, a bit of the CBTC field disables the interrupt from the relevant DMAC channel.

- **ERRx: Access Error [7:0]**

Access Error Interrupt Disable Register. When set, a bit of the ERR field disables the interrupt from the relevant DMAC channel.

31.7.8 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Mask Register

Name: DMAC_EBCIMR

Address: 0xFFFFEC20 (0), 0xFFFFEE20 (1)

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

- **BTCx: Buffer Transfer Completed [7:0]**

0: Buffer Transfer Completed Interrupt is disabled for channel i.

1: Buffer Transfer Completed Interrupt is enabled for channel i.

- **CBTCx: Chained Buffer Transfer Completed [7:0]**

0: Chained Buffer Transfer interrupt is disabled for channel i.

1: Chained Buffer Transfer interrupt is enabled for channel i.

- **ERRx: Access Error [7:0]**

0: Transfer Error Interrupt is disabled for channel i.

1: Transfer Error Interrupt is enabled for channel i.

31.7.9 DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register

Name: DMAC_EBCISR

Address: 0xFFFFEC24 (0), 0xFFFFEE24 (1)

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
CBTC7	CBTC6	CBTC5	CBTC4	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
BTC7	BTC6	BTC5	BTC4	BTC3	BTC2	BTC1	BTC0

- **BTCx: Buffer Transfer Completed [7:0]**

When BTC[*i*] is set, Channel *i* buffer transfer has terminated.

- **CBTCx: Chained Buffer Transfer Completed [7:0]**

When CBTC[*i*] is set, Channel *i* Chained buffer has terminated. LLI Fetch operation is disabled.

- **ERRx: Access Error [7:0]**

When ERR[*i*] is set, Channel *i* has detected an AHB Read or Write Error Access.

31.7.10 DMAC Channel Handler Enable Register

Name: DMAC_CHER

Address: 0xFFFFEC28 (0), 0xFFFFEE28 (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
KEEP7	KEEP6	KEEP5	KEEP4	KEEP3	KEEP2	KEEP1	KEEP0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SUSP7	SUSP6	SUSP5	SUSP4	SUSP3	SUSP2	SUSP1	SUSP0
7	6	5	4	3	2	1	0
ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	ENA0

- **ENAx: Enable [7:0]**

When set, a bit of the ENA field enables the relevant channel.

- **SUSPx: Suspend [7:0]**

When set, a bit of the SUSP field freezes the relevant channel and its current context.

- **KEEPx: Keep on [7:0]**

When set, a bit of the KEEP field resumes the current channel from an automatic stall state.

31.7.11 DMAC Channel Handler Disable Register

Name: DMAC_CHDR

Address: 0xFFFFEC2C (0), 0xFFFFEE2C (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RES7	RES6	RES5	RES4	RES3	RES2	RES1	RES0
7	6	5	4	3	2	1	0
DIS7	DIS6	DIS5	DIS4	DIS3	DIS2	DIS1	DIS0

- **DISx: Disable [7:0]**

Write one to this field to disable the relevant DMAC Channel. The content of the FIFO is lost and the current AHB access is terminated. Software must poll DIS[7:0] field in the DMAC_CHSR register to be sure that the channel is disabled.

- **RESx: Resume [7:0]**

Write one to this field to resume the channel transfer restoring its context.

31.7.12 DMAC Channel Handler Status Register

Name: DMAC_CHSR

Address: 0xFFFFEC30 (0), 0xFFFFEE30 (1)

Access: Read-only

Reset: 0x00FF0000

31	30	29	28	27	26	25	24
STAL7	STAL6	STAL5	STAL4	STAL3	STAL2	STAL1	STAL0
23	22	21	20	19	18	17	16
EMPT7	EMPT6	EMPT5	EMPT4	EMPT3	EMPT2	EMPT1	EMPT0
15	14	13	12	11	10	9	8
SUSP7	SUSP6	SUSP5	SUSP4	SUSP3	SUSP2	SUSP1	SUSP0
7	6	5	4	3	2	1	0
ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	ENA0

- **ENAx: Enable [7:0]**

A one in any position of this field indicates that the relevant channel is enabled.

- **SUSPx: Suspend [7:0]**

A one in any position of this field indicates that the channel transfer is suspended.

- **EMPTx: Empty [7:0]**

A one in any position of this field indicates that the relevant channel is empty.

- **STALx: Stalled [7:0]**

A one in any position of this field indicates that the relevant channel is stalling.

31.7.13 DMAC Channel x [x = 0..7] Source Address Register

Name: DMAC_SADDRx [x = 0..7]

Address: 0xFFFFEC3C (0)[0], 0xFFFFEC64 (0)[1], 0xFFFFEC8C (0)[2], 0xFFFFECB4 (0)[3], 0xFFFFECDC (0)[4], 0xFFFFED04 (0)[5], 0xFFFFED2C (0)[6], 0xFFFFED54 (0)[7], 0xFFFFEE3C (1)[0], 0xFFFFEE64 (1)[1], 0xFFFFEE8C (1)[2], 0xFFFFEEB4 (1)[3], 0xFFFFEEDC (1)[4], 0xFFFFEF04 (1)[5], 0xFFFFEF2C (1)[6], 0xFFFFEF54 (1)[7]

Access: Read-write

Reset: 0x00000000

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

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” .

- **SADDR: Channel x Source Address**

This register must be aligned with the source transfer width.

31.7.14 DMAC Channel x [x = 0..7] Destination Address Register

Name: DMAC_DADDRx [x = 0..7]

Address: 0xFFFFEC40 (0)[0], 0xFFFFEC68 (0)[1], 0xFFFFEC90 (0)[2], 0xFFFFECB8 (0)[3], 0xFFFFECE0 (0)[4], 0xFFFFED08 (0)[5], 0xFFFFED30 (0)[6], 0xFFFFED58 (0)[7], 0xFFFFEE40 (1)[0], 0xFFFFEE68 (1)[1], 0xFFFFEE90 (1)[2], 0xFFFFEEB8 (1)[3], 0xFFFFEEE0 (1)[4], 0xFFFFEF08 (1)[5], 0xFFFFEF30 (1)[6], 0xFFFFEF58 (1)[7]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
DADDR							

23	22	21	20	19	18	17	16
DADDR							

15	14	13	12	11	10	9	8
DADDR							

7	6	5	4	3	2	1	0
DADDR							

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” .

- **DADDR: Channel x Destination Address**

This register must be aligned with the destination transfer width.

31.7.15 DMAC Channel x [x = 0..7] Descriptor Address Register

Name: DMAC_DSCRx [x = 0..7]

Address: 0xFFFFEC44 (0)[0], 0xFFFFEC6C (0)[1], 0xFFFFEC94 (0)[2], 0xFFFFECBC (0)[3], 0xFFFFECE4 (0)[4], 0xFFFFED0C (0)[5], 0xFFFFED34 (0)[6], 0xFFFFED5C (0)[7], 0xFFFFEE44 (1)[0], 0xFFFFEE6C (1)[1], 0xFFFFEE94 (1)[2], 0xFFFFEEBC (1)[3], 0xFFFFEEE4 (1)[4], 0xFFFFEF0C (1)[5], 0xFFFFEF34 (1)[6], 0xFFFFEF5C (1)[7]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
DSCR							

23	22	21	20	19	18	17	16
DSCR							

15	14	13	12	11	10	9	8
DSCR							

7	6	5	4	3	2	1	0
DSCR						DSCR_IF	

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” .

- **DSCR_IF: Descriptor Interface Selection**

Value	Name	Description
00	AHB_IF0	The buffer transfer descriptor is fetched via AHB-Lite Interface 0
01	AHB_IF1	The buffer transfer descriptor is fetched via AHB-Lite Interface 1

- **DSCR: Buffer Transfer Descriptor Address**

This address is word aligned.

31.7.16 DMAC Channel x [x = 0..7] Control A Register

Name: DMAC_CTRLAx [x = 0..7]

Address: 0xFFFFEC48 (0)[0], 0xFFFFEC70 (0)[1], 0xFFFFEC98 (0)[2], 0xFFFFECC0 (0)[3], 0xFFFFECE8 (0)[4], 0xFFFFED10 (0)[5], 0xFFFFED38 (0)[6], 0xFFFFED60 (0)[7], 0xFFFFEE48 (1)[0], 0xFFFFEE70 (1)[1], 0xFFFFEE98 (1)[2], 0xFFFFEEC0 (1)[3], 0xFFFFEEE8 (1)[4], 0xFFFFEF10 (1)[5], 0xFFFFEF38 (1)[6], 0xFFFFEF60 (1)[7]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
DONE	–	DST_WIDTH		–	–	SRC_WIDTH	

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

15	14	13	12	11	10	9	8
BTSIZE							

7	6	5	4	3	2	1	0
BTSIZE							

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” on page 528

- **BTSIZE: Buffer Transfer Size**

The transfer size relates to the number of transfers to be performed, that is, for writes it refers to the number of source width transfers to perform when DMAC is flow controller. For Reads, BTSIZE refers to the number of transfers completed on the Source Interface. When this field is set to 0, the DMAC module is automatically disabled when the relevant channel is enabled.

- **SCSIZE: Source Chunk Transfer Size**

Value	Name	Description
000	CHK_1	1 data transferred
001	CHK_4	4 data transferred
010	CHK_8	8 data transferred
011	CHK_16	16 data transferred

- **DCSIZE: Destination Chunk Transfer Size**

Value	Name	Description
000	CHK_1	1 data transferred
001	CHK_4	4 data transferred
010	CHK_8	8 data transferred
011	CHK_16	16 data transferred

- **SRC_WIDTH: Transfer Width for the Source**

Value	Name	Description
00	BYTE	the transfer size is set to 8-bit width
01	HALF_WORD	the transfer size is set to 16-bit width
1X	WORD	the transfer size is set to 32-bit width

- **DST_WIDTH: Transfer Width for the Destination**

Value	Name	Description
00	BYTE	the transfer size is set to 8-bit width
01	HALF_WORD	the transfer size is set to 16-bit width
1X	WORD	the transfer size is set to 32-bit width

- **DONE: Current Descriptor Stop Command and Transfer Completed Memory Indicator**

0: The transfer is performed.

1: If SOD field of DMAC_CFG register is set to true, then the DMAC is automatically disabled when an LLI updates the content of this register.

The DONE field is written back to memory at the end of the current descriptor transfer.

31.7.17 DMAC Channel x [x = 0..7] Control B Register

Name: DMAC_CTRLBx [x = 0..7]

Address: 0xFFFFEC4C (0)[0], 0xFFFFEC74 (0)[1], 0xFFFFEC9C (0)[2], 0xFFFFECC4 (0)[3], 0xFFFFECEC (0)[4], 0xFFFFED14 (0)[5], 0xFFFFED3C (0)[6], 0xFFFFED64 (0)[7], 0xFFFFEE4C (1)[0], 0xFFFFEE74 (1)[1], 0xFFFFEE9C (1)[2], 0xFFFFEEC4 (1)[3], 0xFFFFEEEC (1)[4], 0xFFFFEF14 (1)[5], 0xFFFFEF3C (1)[6], 0xFFFFEF64 (1)[7]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
AUTO	IEN	DST_INCR		–	–	SRC_INCR	
23	22	21	20	19	18	17	16
FC			DST_DSCR	–	–	–	SRC_DSCR
15	14	13	12	11	10	9	8
–	–		DST_PIP	–	–	–	SRC_PIP
7	6	5	4	3	2	1	0
–	–	DIF		–	–	SIF	

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” .

- **SIF: Source Interface Selection Field**

Value	Name	Description
00	AHB_IF0	The source transfer is done via AHB-Lite Interface 0
01	AHB_IF1	The source transfer is done via AHB-Lite Interface 1

- **DIF: Destination Interface Selection Field**

Value	Name	Description
00	AHB_IF0	The destination transfer is done via AHB-Lite Interface 0
01	AHB_IF1	The destination transfer is done via AHB-Lite Interface 1

- **SRC_PIP: Source Picture-in-Picture Mode**

0 (DISABLE): Picture-in-Picture mode is disabled. The source data area is contiguous.

1 (ENABLE): Picture-in-Picture mode is enabled. When the source PIP counter reaches the programmable boundary, the address is automatically incremented by a user defined amount.

- **DST_PIP: Destination Picture-in-Picture Mode**

0 (DISABLE): Picture-in-Picture mode is disabled. The Destination data area is contiguous.

1 (ENABLE): Picture-in-Picture mode is enabled. When the Destination PIP counter reaches the programmable boundary the address is automatically incremented by a user-defined amount.

- **SRC_DSCR: Source Address Descriptor**

0 (FETCH_FROM_MEM): Source address is updated when the descriptor is fetched from the memory.

1 (FETCH_DISABLE): Buffer Descriptor Fetch operation is disabled for the source.

- **DST_DSCR: Destination Address Descriptor**

0 (FETCH_FROM_MEM): Destination address is updated when the descriptor is fetched from the memory.

1 (FETCH_DISABLE): Buffer Descriptor Fetch operation is disabled for the destination.

- **FC: Flow Control**

This field defines which device controls the size of the buffer transfer, also referred to as the Flow Controller.

Value	Name	Description
000	MEM2MEM_DMA_FC	Memory-to-Memory Transfer DMAC is flow controller
001	MEM2PER_DMA_FC	Memory-to-Peripheral Transfer DMAC is flow controller
010	PER2MEM_DMA_FC	Peripheral-to-Memory Transfer DMAC is flow controller
011	PER2PER_DMA_FC	Peripheral-to-Peripheral Transfer DMAC is flow controller

- **SRC_INCR: Incrementing, Decrementing or Fixed Address for the Source**

Value	Name	Description
00	INCREMENTING	The source address is incremented
01	DECREMENTING	The source address is decremented
10	FIXED	The source address remains unchanged

- **DST_INCR: Incrementing, Decrementing or Fixed Address for the Destination**

Value	Name	Description
00	INCREMENTING	The destination address is incremented
01	DECREMENTING	The destination address is decremented
10	FIXED	The destination address remains unchanged

- **IEN: Interrupt Enable Not**

0: When the buffer transfer is completed, the BTCx flag is set in the EBCISR status register. This bit is active low.

1: When the buffer transfer is completed, the BTCx flag is not set.

If this bit is cleared, when the buffer transfer is completed, the BTCx flag is set in the EBCISR status register.

- **AUTO: Automatic Multiple Buffer Transfer**

0 (DISABLE): Automatic multiple buffer transfer is disabled.

1 (ENABLE): Automatic multiple buffer transfer is enabled. This bit enables replay mode or contiguous mode when several buffers are transferred.

31.7.18 DMAC Channel x [x = 0..7] Configuration Register

Name: DMAC_CFGx [x = 0..7]

Address: 0xFFFFEC50 (0)[0], 0xFFFFEC78 (0)[1], 0xFFFFECA0 (0)[2], 0xFFFFECC8 (0)[3], 0xFFFFECF0 (0)[4], 0xFFFFED18 (0)[5], 0xFFFFED40 (0)[6], 0xFFFFED68 (0)[7], 0xFFFFEE50 (1)[0], 0xFFFFEE78 (1)[1], 0xFFFFEEA0 (1)[2], 0xFFFFEEC8 (1)[3], 0xFFFFEEF0 (1)[4], 0xFFFFEF18 (1)[5], 0xFFFFEF40 (1)[6], 0xFFFFEF68 (1)[7]

Access: Read-write

Reset: 0x0100000000

31	30	29	28	27	26	25	24
–	–	FIFOCFG		–	AHB_PROT		
23	22	21	20	19	18	17	16
–	LOCK_IF_L	LOCK_B	LOCK_IF	–	–	–	SOD
15	14	13	12	11	10	9	8
–	–	DST_H2SEL	DST_REP	–	–	SRC_H2SEL	SRC_REP
7	6	5	4	3	2	1	0
DST_PER				SRC_PER			

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” on page 528

- **SRC_PER: Source with Peripheral identifier**

Channel x Source Request is associated with peripheral identifier coded SRC_PER handshaking interface.

- **DST_PER: Destination with Peripheral identifier**

Channel x Destination Request is associated with peripheral identifier coded DST_PER handshaking interface.

- **SRC_REP: Source Reloaded from Previous**

0 (CONTIGUOUS_ADDR): When automatic mode is activated, source address is contiguous between two buffers.

1 (RELOAD_ADDR): When automatic mode is activated, the source address and the control register are reloaded from previous transfer.

- **SRC_H2SEL: Software or Hardware Selection for the Source**

0 (SW): Software handshaking interface is used to trigger a transfer request.

1 (HW): Hardware handshaking interface is used to trigger a transfer request.

- **DST_REP: Destination Reloaded from Previous**

0 (CONTIGUOUS_ADDR): When automatic mode is activated, destination address is contiguous between two buffers.

1 (RELOAD_ADDR): When automatic mode is activated, the destination and the control register are reloaded from the previous transfer.

- **DST_H2SEL: Software or Hardware Selection for the Destination**

0 (SW): Software handshaking interface is used to trigger a transfer request.

1 (HW): Hardware handshaking interface is used to trigger a transfer request.

- **SOD: Stop On Done**

0 (DISABLE): STOP ON DONE disabled, the descriptor fetch operation ignores DONE Field of CTRLA register.

1 (ENABLE): STOP ON DONE activated, the DMAC module is automatically disabled if DONE FIELD is set to 1.

- **LOCK_IF: Interface Lock**

0 (DISABLE): Interface Lock capability is disabled

1 (ENABLE): Interface Lock capability is enabled

- **LOCK_B: Bus Lock**

0 (DISABLE): AHB Bus Locking capability is disabled.

1(ENABLE): AHB Bus Locking capability is enabled.

- **LOCK_IF_L: Master Interface Arbiter Lock**

0 (CHUNK): The Master Interface Arbiter is locked by the channel x for a chunk transfer.

1 (BUFFER): The Master Interface Arbiter is locked by the channel x for a buffer transfer.

- **AHB_PROT: AHB Protection**

AHB_PROT field provides additional information about a bus access and is primarily used to implement some level of protection.

HPROT[3]	HPROT[2]	HPROT[1]	HPROT[0]	Description
			1	Data access
		AHB_PROT[0]		0: User Access 1: Privileged Access
	AHB_PROT[1]			0: Not Bufferable 1: Bufferable
AHB_PROT[2]				0: Not cacheable 1: Cacheable

- **FIFOCFG: FIFO Configuration**

Value	Name	Description
00	ALAP_CFG	The largest defined length AHB burst is performed on the destination AHB interface.
01	HALF_CFG	When half FIFO size is available/filled, a source/destination request is serviced.
10	ASAP_CFG	When there is enough space/data available to perform a single AHB access, then the request is serviced.

31.7.19 DMAC Channel x [x = 0..7] Source Picture-in-Picture Configuration Register

Name: DMAC_SPIPx [x = 0..7]

Address: 0xFFFFEC54 (0)[0], 0xFFFFEC7C (0)[1], 0xFFFFECA4 (0)[2], 0xFFFFECCC (0)[3], 0xFFFFECF4 (0)[4], 0xFFFFED1C (0)[5], 0xFFFFED44 (0)[6], 0xFFFFED6C (0)[7], 0xFFFFEE54 (1)[0], 0xFFFFEE7C (1)[1], 0xFFFFEEA4 (1)[2], 0xFFFFEECC (1)[3], 0xFFFFEEF4 (1)[4], 0xFFFFEF1C (1)[5], 0xFFFFEF44 (1)[6], 0xFFFFEF6C (1)[7]

Access: Read-write

Reset: 0x00000000

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

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” on page 528

- **SPIP_HOLE: Source Picture-in-Picture Hole**

This field indicates the value to add to the address when the programmable boundary has been reached.

- **SPIP_BOUNDARY: Source Picture-in-Picture Boundary**

This field indicates the number of source transfers to perform before the automatic address increment operation.

31.7.20 DMAC Channel x [x = 0..7] Destination Picture-in-Picture Configuration Register

Name: DMAC_DPIPx [x = 0..7]

Address: 0xFFFFEC58 (0)[0], 0xFFFFEC80 (0)[1], 0xFFFFECA8 (0)[2], 0xFFFFECD0 (0)[3], 0xFFFFECF8 (0)[4], 0xFFFFED20 (0)[5], 0xFFFFED48 (0)[6], 0xFFFFED70 (0)[7], 0xFFFFEE58 (1)[0], 0xFFFFEE80 (1)[1], 0xFFFFEEA8 (1)[2], 0xFFFFEED0 (1)[3], 0xFFFFEEF8 (1)[4], 0xFFFFEF20 (1)[5], 0xFFFFEF48 (1)[6], 0xFFFFEF70 (1)[7]

Access: Read-write

Reset: 0x00000000

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

This register can only be written if the WPEN bit is cleared in “DMAC Write Protect Mode Register” on page 528

- **DPIP_HOLE: Destination Picture-in-Picture Hole**

This field indicates the value to add to the address when the programmable boundary has been reached.

- **DPIP_BOUNDARY: Destination Picture-in-Picture Boundary**

This field indicates the number of source transfers to perform before the automatic address increment operation.

31.7.21 DMAC Write Protect Mode Register

Name: DMAC_WPMR

Address: 0xFFFFEDE4 (0), 0xFFFFEFE4 (1)

Access: Read-write

Reset: See Table 31-4

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x444D41 (“DMA” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x444D41 (“DMA” in ASCII).

Protects the registers:

- “DMAC Global Configuration Register” on page 505
- “DMAC Enable Register” on page 506
- “DMAC Channel x [x = 0..7] Source Address Register” on page 517
- “DMAC Channel x [x = 0..7] Destination Address Register” on page 518
- “DMAC Channel x [x = 0..7] Descriptor Address Register” on page 519
- “DMAC Channel x [x = 0..7] Control A Register” on page 520
- “DMAC Channel x [x = 0..7] Control B Register” on page 522
- “DMAC Channel x [x = 0..7] Configuration Register” on page 524
- “DMAC Channel x [x = 0..7] Source Picture-in-Picture Configuration Register” on page 526
- “DMAC Channel x [x = 0..7] Destination Picture-in-Picture Configuration Register” on page 527

- **WPKEY: Write Protect KEY**

Should be written at value 0x444D41 (“DMA” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

31.7.22 DMAC Write Protect Status Register

Name: DMAC_WPSR

Address: 0xFFFFEDE8 (0), 0xFFFFEFE8 (1)

Access: Read-only

Reset: See Table 31-4

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the DMAC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the DMAC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading DMAC_WPSR automatically clears all fields.

32. USB High Speed Device Port (UDPHS)

32.1 Description

The USB High Speed Device Port (UDPHS) is compliant with the Universal Serial Bus (USB), rev 2.0 High Speed device specification.

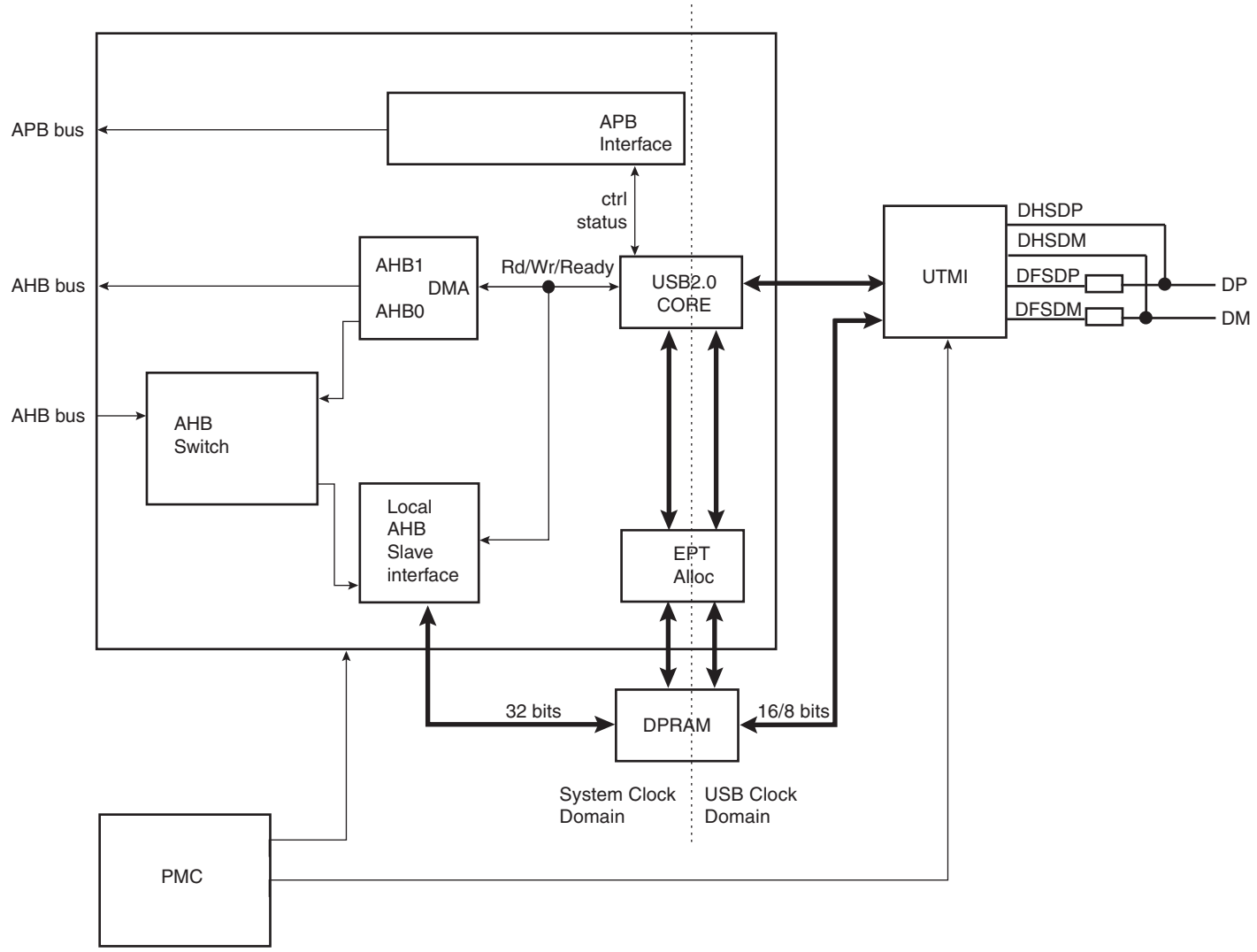
Each endpoint can be configured in one of several USB transfer types. It can be associated with one, two or three banks of a Dual-port RAM used to store the current data payload. If two or three banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints.

32.2 Embedded Characteristics

- 1 Device High Speed
- 1 UTMI transceiver shared between Host and Device
- USB v2.0 High Speed Compliant, 480 Mbits Per Second
- 7 Endpoints up to 1024 bytes
- Embedded Dual-port RAM for Endpoints
- Suspend/Resume Logic (Command of UTMI)
- Up to Three Memory Banks for Endpoints (Not for Control Endpoint)
- 4 Kbytes of DPRAM

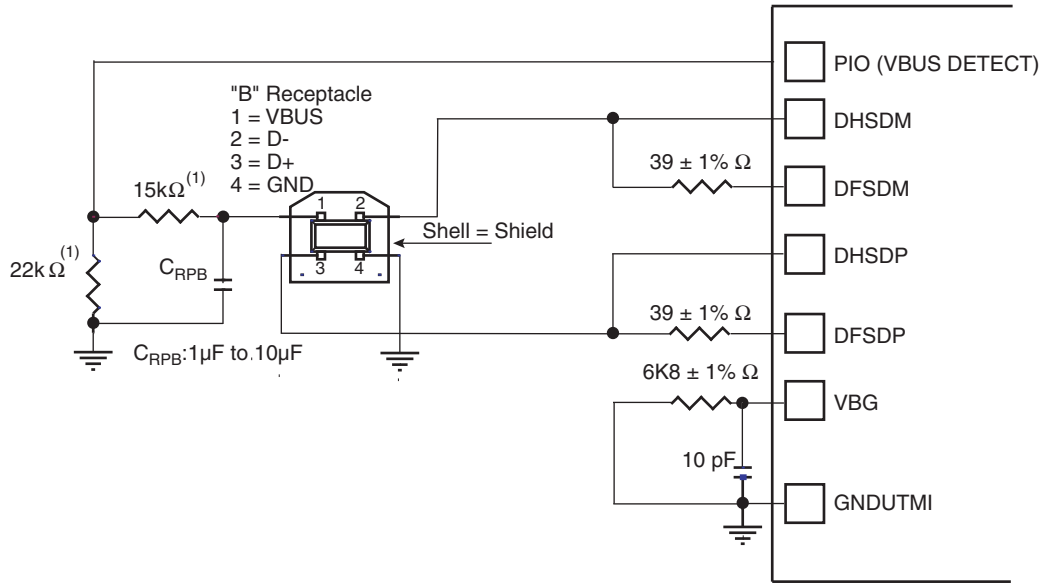
32.3 Block Diagram

Figure 32-1. Block Diagram



32.4 Typical Connection

Figure 32-2. Board Schematic



Note: The values shown on the 22 kΩ and 15 kΩ resistors are only valid with 3V3 supplied PIOs. Both 39 Ω resistors need to be placed as close to the device pins as possible.

32.5 Product Dependencies

32.5.1 Power Management

The UDPHS is not continuously clocked.

For using the UDPHS, the programmer must first enable the UDPHS Clock in the Power Management Controller (PMC_PCER register). Then enable the PLL (PMC_UCKR register).

However, if the application does not require UDPHS operations, the UDPHS clock can be stopped when not needed and restarted later.

32.5.2 Interrupt

The UDPHS interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the UDPHS

Table 32-1. Peripheral IDs

Instance	ID
UDPHS	23

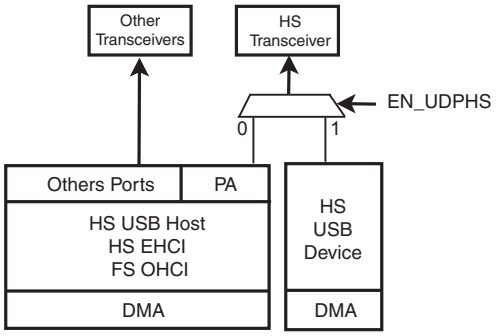
interrupt requires the Interrupt Controller to be programmed first.

32.6 Functional Description

32.6.1 UTMI Transceivers Sharing

The High Speed USB Host Port A is shared with the High Speed USB Device port and connected to the second UTMI transceiver. The selection between Host Port A and USB Device is controlled by the UDPHS enable bit (EN_UDPHS) located in the UDPHS_CTRL control register.

Figure 32-3. USB Selection



32.6.2 USB V2.0 High Speed Device Port Introduction

The USB V2.0 High Speed Device Port provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with a USB Device through a set of communication flows.

32.6.3 USB V2.0 High Speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

A device provides several logical communication pipes with the host. To each logical pipe is associated an endpoint. Transfer through a pipe belongs to one of the four transfer types:

- Control Transfers: Used to configure a device at attach time and can be used for other device-specific purposes, including control of other pipes on the device.
- Bulk Data Transfers: Generated or consumed in relatively large burst quantities and have wide dynamic latitude in transmission constraints.
- Interrupt Data Transfers: Used for timely but reliable delivery of data, for example, characters or coordinates with human-perceptible echo or feedback response characteristics.
- Isochronous Data Transfers: Occupy a prenegotiated amount of USB bandwidth with a prenegotiated delivery latency. (Also called streaming real time transfers.)

As indicated below, transfers are sequential events carried out on the USB bus.

Endpoints must be configured according to the transfer type they handle.

Table 32-2. USB Communication Flow

Transfer	Direction	Bandwidth	Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	8-1024	Yes	No
Interrupt	Unidirectional	Not guaranteed	8-1024	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8-512	Yes	Yes

32.6.4 USB Transfer Event Definitions

A transfer is composed of one or several transactions;

Table 32-3. USB Transfer Events

CONTROL (bidirectional)	Control Transfers (1)	• Setup transaction → Data IN transactions ∅Status OUT transaction • Setup transaction → Data OUT transactions ∅Status IN transaction • Setup transaction → Status IN transaction
IN (device toward host)	Bulk IN Transfer	• Data IN transaction → Data IN transaction
	Interrupt IN Transfer	• Data IN transaction → Data IN transaction
	Isochronous IN Transfer (2)	• Data IN transaction → Data IN transaction
OUT (host toward device)	Bulk OUT Transfer	• Data OUT transaction → Data OUT transaction
	Interrupt OUT Transfer	• Data OUT transaction → Data OUT transaction
	Isochronous OUT Transfer (2)	• Data OUT transaction→ Data OUT transaction

Notes: 1. Control transfer must use endpoints with one bank and can be aborted using a stall handshake.
2. Isochronous transfers must use endpoints configured with two or three banks.

An endpoint handles all transactions related to the type of transfer for which it has been configured.

Table 32-4. UDPHS Endpoint Description

Endpoint #	Mnemonic	Nb Bank	DMA	High Band Width	Max. Endpoint Size	Endpoint Type
0	EPT_0	1	N	N	64	Control
1	EPT_1	2	Y	Y	1024	Ctrl/Bulk/Iso(32.3)/Interrupt
2	EPT_2	2	Y	Y	1024	Ctrl/Bulk/Iso(32.3)/Interrupt
3	EPT_3	3	Y	N	1024	Ctrl/Bulk/Iso(32.3)/Interrupt
4	EPT_4	3	Y	N	1024	Ctrl/Bulk/Iso(32.3)/Interrupt
5	EPT_5	3	Y	Y	1024	Ctrl/Bulk/Iso(32.3)/Interrupt
6	EPT_6	3	Y	Y	1024	Ctrl/Bulk/Iso(32.3)/Interrupt

Note: 1. In Isochronous Mode (Iso), it is preferable that High Band Width capability is available.

The size of internal DPRAM is 4 Kbytes.

Suspend and resume are automatically detected by the UDPHS device, which notifies the processor by raising an interrupt.

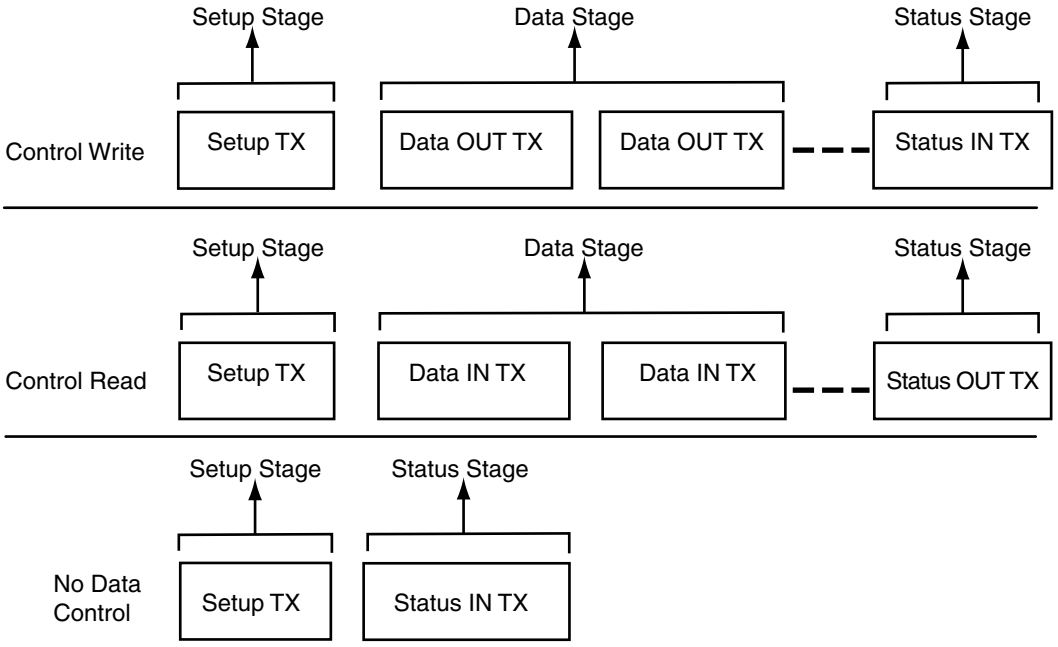
32.6.5 USB V2.0 High Speed BUS Transactions

Each transfer results in one or more transactions over the USB bus.

There are five kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction
4. Status IN Transaction
5. Status OUT Transaction

Figure 32-4. Control Read and Write Sequences



A status IN or OUT transaction is identical to a data IN or OUT transaction.

32.6.6 Endpoint Configuration

The endpoint 0 is always a control endpoint, it must be programmed and active in order to be enabled when the End Of Reset interrupt occurs.

To configure the endpoints:

- Fill the configuration register (UDPHS_EPTCFG) with the endpoint size, direction (IN or OUT), type (CTRL, Bulk, IT, ISO) and the number of banks.
- Fill the number of transactions (NB_TRANS) for isochronous endpoints.

Note: For control endpoints the direction has no effect.

- Verify that the EPT_MAPD flag is set. This flag is set if the endpoint size and the number of banks are correct compared to the FIFO maximum capacity and the maximum number of allowed banks.
- Configure control flags of the endpoint and enable it in UDPHS_EPTCTLENBx according to “UDPHS Endpoint Control Disable Register (Isochronous Endpoint)” on page 574.

Control endpoints can generate interrupts and use only 1 bank.

All endpoints (except endpoint 0) can be configured either as Bulk, Interrupt or Isochronous. See Table 32-4. UDPHS Endpoint Description.

The maximum packet size they can accept corresponds to the maximum endpoint size.

Note: The endpoint size of 1024 is reserved for isochronous endpoints.

The size of the DPRAM is 4 Kbytes. The DPR is shared by all active endpoints. The memory size required by the active endpoints must not exceed the size of the DPRAM.

SIZE_DPRAM = SIZE _EPT0

+ NB_BANK_EPT1 x SIZE_EPT1

+ NB_BANK_EPT2 x SIZE_EPT2

+ NB_BANK_EPT3 x SIZE_EPT3

+ NB_BANK_EPT4 x SIZE_EPT4

+ NB_BANK_EPT5 x SIZE_EPT5

+ NB_BANK_EPT6 x SIZE_EPT6

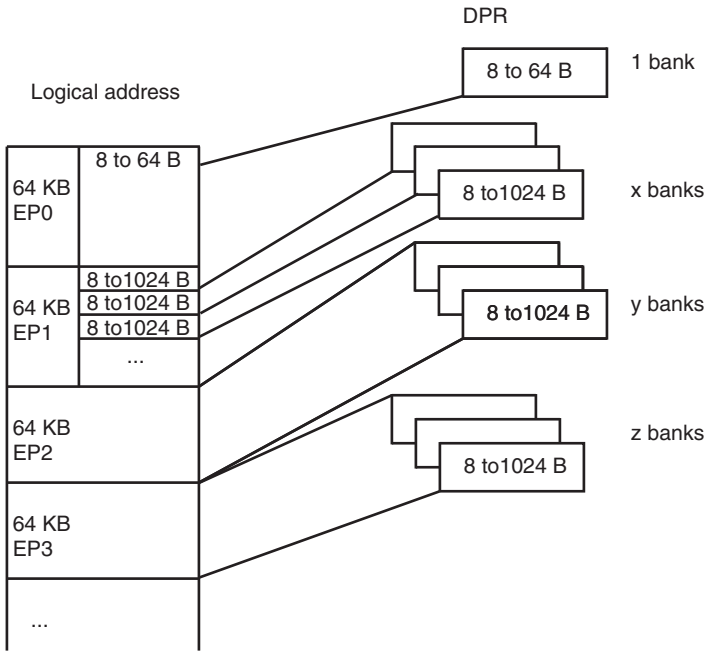
+... (refer to 32.7.8 UDPHS Endpoint Configuration Register)

If a user tries to configure endpoints with a size the sum of which is greater than the DPRAM, then the EPT_MAPD is not set.

The application has access to the physical block of DPR reserved for the endpoint through a 64 Kbyte logical address space.

The physical block of DPR allocated for the endpoint is remapped all along the 64 Kbyte logical address space. The application can write a 64 Kbyte buffer linearly.

Figure 32-5. Logical Address Space for DPR Access



Configuration examples of UDPHS_EPTCTLx (UDPHS Endpoint Control Disable Register (Isochronous Endpoint)) for Bulk IN endpoint type follow below.

- With DMA
  - AUTO_VALID: Automatically validate the packet and switch to the next bank.
  - EPT_ENABL: Enable endpoint.
- Without DMA:
  - TXRDY: An interrupt is generated after each transmission.
  - EPT_ENABL: Enable endpoint.

Configuration examples of Bulk OUT endpoint type follow below.

- With DMA
  - AUTO_VALID: Automatically validate the packet and switch to the next bank.
  - EPT_ENABL: Enable endpoint.
- Without DMA
  - RXRDY_TXKL: An interrupt is sent after a new packet has been stored in the endpoint FIFO.
  - EPT_ENABL: Enable endpoint.

32.6.7 DPRAM Management

Endpoints can only be allocated in ascending order, from the endpoint 0 to the last endpoint to be allocated. The user shall therefore configure them in the same order.

The allocation of an endpoint x starts when the Number of Banks field in the UDPHS Endpoint Configuration Register (UDPHS_EPTCFGx.BK_NUMBER) is different from zero. Then, the hardware allocates a memory area in the DPRAM and inserts it between the x-1 and x+1 endpoints. The x+1 endpoint memory window slides up and its data is lost. Note that the following endpoint memory windows (from x+2) do not slide.

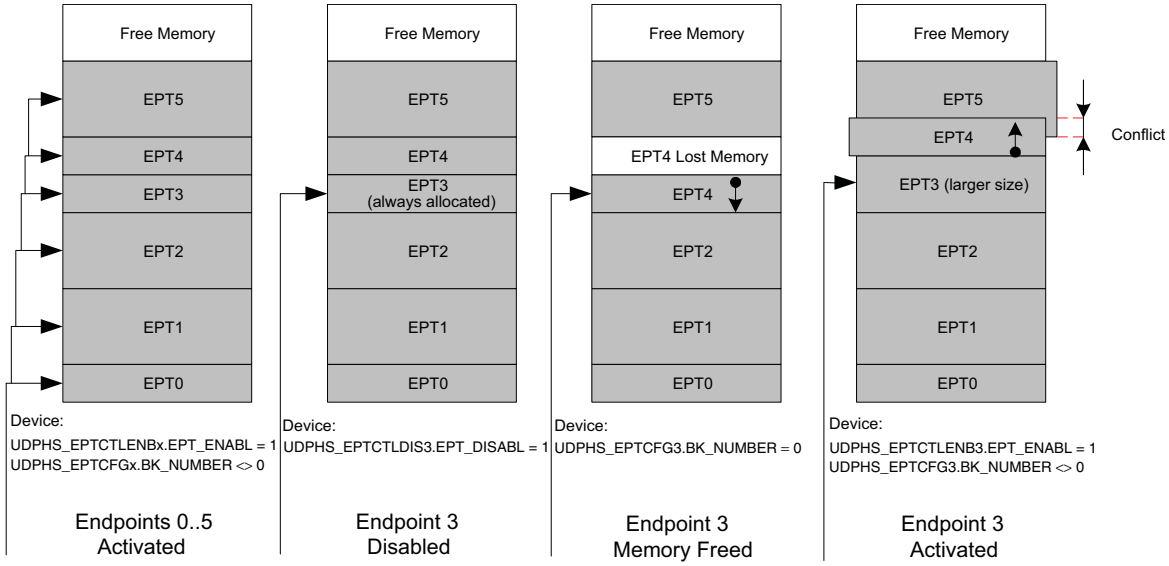
Disabling an endpoint, by writing a one to the Endpoint Disable bit in the UDPHS Endpoint Control Disable Register (UDPHS_EPTCTLDISx.EPT_DISABL), does not reset its configuration:

- The Endpoint Banks (UDPHS_EPTCFGx.BK_NUMBER),
- The Endpoint Size (UDPHS_EPTCFGx.EPT_SIZE),
- The Endpoint Direction (UDPHS_EPTCFGx.EPT_DIR), and
- The Endpoint Type (UDPHS_EPTCFGx.EPT_TYPE).

To free its memory, the user shall write a zero to the UDPHS_EPTCFGx.BK_NUMBER field. The x+1 endpoint memory window then slides down and its data is lost. Note that the following endpoint memory windows (from x+2) do not slide.

Figure 32-6 on page 538 illustrates the allocation and reorganization of the DPRAM in a typical example.

Figure 32-6. Allocation and Reorganization of the DPRAM



1. The endpoints 0 to 5 are enabled, configured and allocated in ascending order.
   Each endpoint then owns a memory area in the DPRAM.
2. The endpoint 3 is disabled, but its memory is kept allocated by the controller.
3. In order to free its memory, its UDPHS_EPTCFGx.BK_NUMBER field is written to zero. The endpoint 4 memory window slides down, but the endpoint 5 does not move.
4. If the user chooses to reconfigure the endpoint 3 with a larger size, the controller allocates a memory area after the endpoint 2 memory area and automatically slides up the endpoint 4 memory window. The endpoint 5 does not move and a memory conflict appears as the memory windows of the endpoints 4 and 5 overlap. The data of these endpoints is potentially lost.

Notes:
1. There is no way the data of the endpoint 0 can be lost (except if it is de-allocated) as the memory allocation and de-allocation may affect only higher endpoints.
2. Deactivating then reactivating the same endpoint with the same configuration only modifies temporarily the controller DPRAM pointer and size for this endpoint. Nothing changes in the DPRAM, higher endpoints seem not to have been moved and their data is preserved as far as nothing has been written or received into them while changing the allocation state of the first endpoint.
3. When the user writes a value different from zero to the UDPHS_EPTCFGx.BK_NUMBER field, the Endpoint Mapped bit (UDPHS_EPTCFGx.EPT_MAPD) is set only if the configured size and number of banks are correct as compared to the endpoint maximal allowed values and to the maximal FIFO size (i.e. the DPRAM size). The UDPHS_EPTCFGx.EPT_MAPD value does not consider memory allocation conflicts.

32.6.8 Transfer With DMA

USB packets of any length may be transferred when required by the UDPHS Device. These transfers always feature sequential addressing.

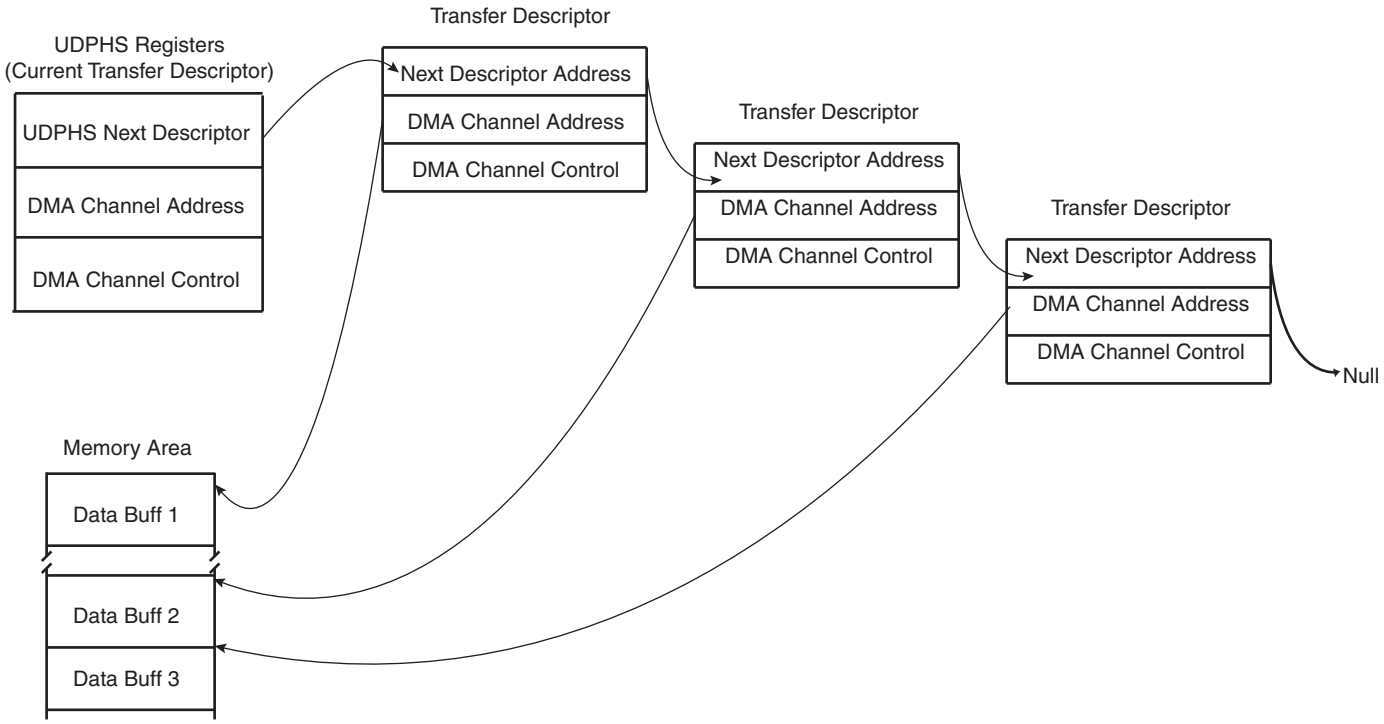
Packet data AHB bursts may be locked on a DMA buffer basis for drastic overall AHB bus bandwidth performance boost with paged memories. These clock-cycle consuming memory row (or bank) changes will then likely not occur, or occur only once instead of dozens times, during a single big USB packet DMA transfer in case another AHB master addresses the memory. This means up to 128-word single-cycle unbroken AHB bursts for Bulk endpoints and 256-word single-cycle unbroken bursts for isochronous endpoints. This maximum burst length is then controlled by the lowest programmed USB endpoint size (EPT_SIZE field in the UDPHS_EPTCFGx register) and DMA Size (BUFF_LENGTH field in the UDPHS_DMACONTROLx register).

The USB 2.0 device average throughput may be up to nearly 60 MBytes. Its internal slave average access latency decreases as burst length increases due to the 0 wait-state side effect of unchanged endpoints. If at least 0 wait-state word burst capability is also provided by the external DMA AHB bus slaves, each of both DMA AHB busses need less than 50% bandwidth allocation for full USB 2.0 bandwidth usage at 30 MHz, and less than 25% at 60 MHz.

The UDPHS DMA Channel Transfer Descriptor is described in “UDPHS DMA Channel Transfer Descriptor” on page 593.

Note: In case of debug, be careful to address the DMA to an SRAM address even if a remap is done.

Figure 32-7. Example of DMA Chained List



32.6.9 Transfer Without DMA

Important. If the DMA is not to be used, it is necessary that it be disabled because otherwise it can be enabled by previous versions of software **without warning**. If this should occur, the DMA can process data before an interrupt without knowledge of the user.

The recommended means to disable DMA is as follows:

```
// Reset IP UDPHS
    AT91C_BASE_UDPHS->UDPHS_CTRL &= ~AT91C_UDPHS_EN_UDPHS;
    AT91C_BASE_UDPHS->UDPHS_CTRL |= AT91C_UDPHS_EN_UDPHS;
// With OR without DMA !!!
```

```
    for( i=1; i<=((AT91C_BASE_UDPHS->UDPHS_IPFEATURES &
AT91C_UDPHS_DMA_CHANNEL_NBR)>>4); i++ ) {
// RESET endpoint canal DMA:
        // DMA stop channel command
        AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMACONTROL = 0;  // STOP
command
// Disable endpoint
        AT91C_BASE_UDPHS->UDPHS_EPT[i].UDPHS_EPTCTLDIS |= 0XFFFFFFFF;
// Reset endpoint config
        AT91C_BASE_UDPHS->UDPHS_EPT[i].UDPHS_EPTCTLCFG = 0;
// Reset DMA channel (Buff count and Control field)
        AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMACONTROL = 0x02;  // NON
STOP command
// Reset DMA channel 0 (STOP)
        AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMACONTROL = 0;  // STOP
command
// Clear DMA channel status (read the register for clear it)
        AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMASTATUS =
AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMASTATUS;
}
```

32.6.10 Handling Transactions with USB V2.0 Device Peripheral

32.6.10.1 Setup Transaction

The setup packet is valid in the DPR while RX_SETUP is set. Once RX_SETUP is cleared by the application, the UDPHS accepts the next packets sent over the device endpoint.

When a valid setup packet is accepted by the UDPHS:

- The UDPHS device automatically acknowledges the setup packet (sends an ACK response)
- Payload data is written in the endpoint
- Sets the RX_SETUP interrupt
- The BYTE_COUNT field in the UDPHS_EPTSTAx register is updated

An endpoint interrupt is generated while RX_SETUP in the UDPHS_EPTSTAx register is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect RX_SETUP polling UDPHS_EPTSTAx or catching an interrupt, read the setup packet in the FIFO, then clear the RX_SETUP bit in the UDPHS_EPTCLRSTA register to acknowledge the setup stage.

If STALL_SNT was set to 1, then this bit is automatically reset when a setup token is detected by the device. Then, the device still accepts the setup stage. (See Section 32.6.10.15 “STALL” on page 549).

32.6.10.2 NYET

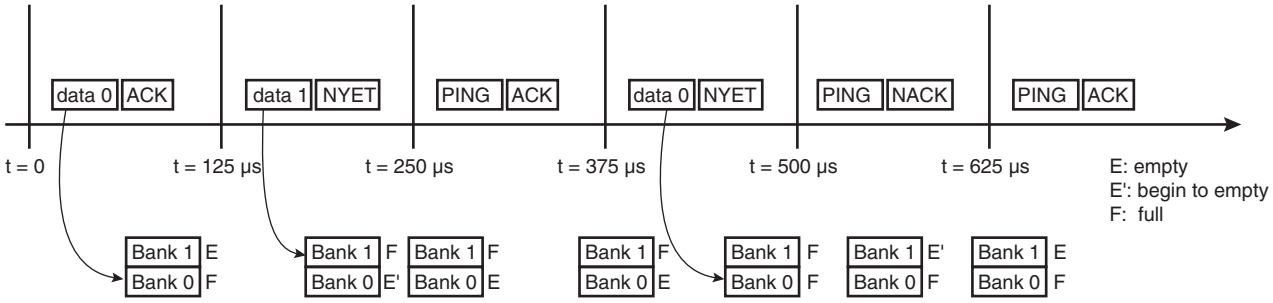
NYET is a High Speed only handshake. It is returned by a High Speed endpoint as part of the PING protocol.

High Speed devices must support an improved NAK mechanism for Bulk OUT and control endpoints (except setup stage). This mechanism allows the device to tell the host whether it has sufficient endpoint space for the next OUT transfer (see USB 2.0 spec 8.5.1 NAK Limiting via Ping Flow Control).

The NYET/ACK response to a High Speed Bulk OUT transfer and the PING response are automatically handled by hardware in the UDPHS_EPTCTLx register (except when the user wants to force a NAK response by using the NYET_DIS bit).

If the endpoint responds instead to the OUT/DATA transaction with an NYET handshake, this means that the endpoint accepted the data but does not have room for another data payload. The host controller must return to using a PING token until the endpoint indicates it has space available.

Figure 32-8. NYET Example with Two Endpoint Banks



32.6.10.3 Data IN

32.6.10.4 Bulk IN or Interrupt IN

Data IN packets are sent by the device during the data or the status stage of a control transfer or during an (interrupt/bulk/isochronous) IN transfer. Data buffers are sent packet by packet under the control of the application or under the control of the DMA channel.

There are three ways for an application to transfer a buffer in several packets over the USB:

- packet by packet (see 32.6.10.5 below)
- 64 Kbytes (see 32.6.10.5 below)
- DMA (see 32.6.10.6 below)

32.6.10.5 Bulk IN or Interrupt IN: Sending a Packet Under Application Control (Device to Host)

The application can write one or several banks.

A simple algorithm can be used by the application to send packets regardless of the number of banks associated to the endpoint.

Algorithm Description for Each Packet:

- The application waits for TXRDY flag to be cleared in the UDPHS_EPTSTAx register before it can perform a write access to the DPR.
- The application writes one USB packet of data in the DPR through the 64 Kbyte endpoint logical memory window.
- The application sets TXRDY flag in the UDPHS_EPTSETSTAx register.

The application is notified that it is possible to write a new packet to the DPR by the TXRDY interrupt. This interrupt can be enabled or masked by setting the TXRDY bit in the UDPHS_EPTCTLENB/UDPHS_EPTCTLDIS register.

Algorithm Description to Fill Several Packets:

Using the previous algorithm, the application is interrupted for each packet. It is possible to reduce the application overhead by writing linearly several banks at the same time. The AUTO_VALID bit in the UDPHS_EPTCTLx must be set by writing the AUTO_VALID bit in the UDPHS_EPTCTLENBx register.

The auto-valid-bank mechanism allows the transfer of data (IN and OUT) without the intervention of the CPU. This means that bank validation (set TXRDY or clear the RXRDY_TXKL bit) is done by hardware.

- The application checks the BUSY_BANK_STA field in the UDPHS_EPTSTAx register. The application must wait that at least one bank is free.
- The application writes a number of bytes inferior to the number of free DPR banks for the endpoint. Each time the application writes the last byte of a bank, the TXRDY signal is automatically set by the UDPHS.
- If the last packet is incomplete (i.e., the last byte of the bank has not been written) the application must set the TXRDY bit in the UDPHS_EPTSETSTAx register.

The application is notified that all banks are free, so that it is possible to write another burst of packets by the BUSY_BANK interrupt. This interrupt can be enabled or masked by setting the BUSY_BANK flag in the UDPHS_EPTCTLENB and UDPHS_EPTCTLDIS registers.

This algorithm must not be used for isochronous transfer. In this case, the ping-pong mechanism does not operate.

A Zero Length Packet can be sent by setting just the TXRDY flag in the UDPHS_EPTSETSTAx register.

32.6.10.6 Bulk IN or Interrupt IN: Sending a Buffer Using DMA (Device to Host)

The UDPHS integrates a DMA host controller. This DMA controller can be used to transfer a buffer from the memory to the DPR or from the DPR to the processor memory under the UDPHS control. The DMA can be used for all transfer types except control transfer.

Example DMA configuration:

1. Program UDPHS_DMAADDRESS x with the address of the buffer that should be transferred.
2. Enable the interrupt of the DMA in UDPHS_IEN
3. Program UDPHS_ DMACONTROLx:
   - Size of buffer to send: size of the buffer to be sent to the host.
   - END_B_EN: The endpoint can validate the packet (according to the values programmed in the AUTO_VALID and SHRT_PCKT fields of UDPHS_EPTCTLx.) (See “UDPHS Endpoint Control Disable Register (Isochronous Endpoint)” on page 574 and Figure 32-13. Autovalid with DMA)
   - END_BUFFIT: generate an interrupt when the BUFF_COUNT in UDPHS_DMASTATUSx reaches 0.
   - CHANN_ENB: Run and stop at end of buffer

The auto-valid-bank mechanism allows the transfer of data (IN & OUT) without the intervention of the CPU. This means that bank validation (set TXRDY or clear the RXRDY_TXKL bit) is done by hardware.

A transfer descriptor can be used. Instead of programming the register directly, a descriptor should be programmed and the address of this descriptor is then given to UDPHS_DMANXTDSC to be processed after setting the LDNXT_DSC field (Load Next Descriptor Now) in UDPHS_DMACONTROLx register.

The structure that defines this transfer descriptor must be aligned.

Each buffer to be transferred must be described by a DMA Transfer descriptor (see “UDPHS DMA Channel Transfer Descriptor” on page 593). Transfer descriptors are chained. Before executing transfer of the buffer, the UDPHS may fetch a new transfer descriptor from the memory address pointed by the UDPHS_DMANXTDSCx register. Once the transfer is complete, the transfer status is updated in the UDPHS_DMASTATUSx register.

To chain a new transfer descriptor with the current DMA transfer, the DMA channel must be stopped. To do so, INTDIS_DMA and TXRDY may be set in the UDPHS_EPTCTLENBx register. It is also possible for the application to wait for the completion of all transfers. In this case the LDNXT_DSC field in the last transfer descriptor UDPHS_DMACONTROLx register must be set to 0 and CHANN_ENB set to 1.

Then the application can chain a new transfer descriptor.

The INTDIS_DMA can be used to stop the current DMA transfer if an enabled interrupt is triggered. This can be used to stop DMA transfers in case of errors.

The application can be notified at the end of any buffer transfer (ENB_BUFFIT bit in the UDPHS_DMACONTROLx register).

Figure 32-9. Data IN Transfer for Endpoint with One Bank

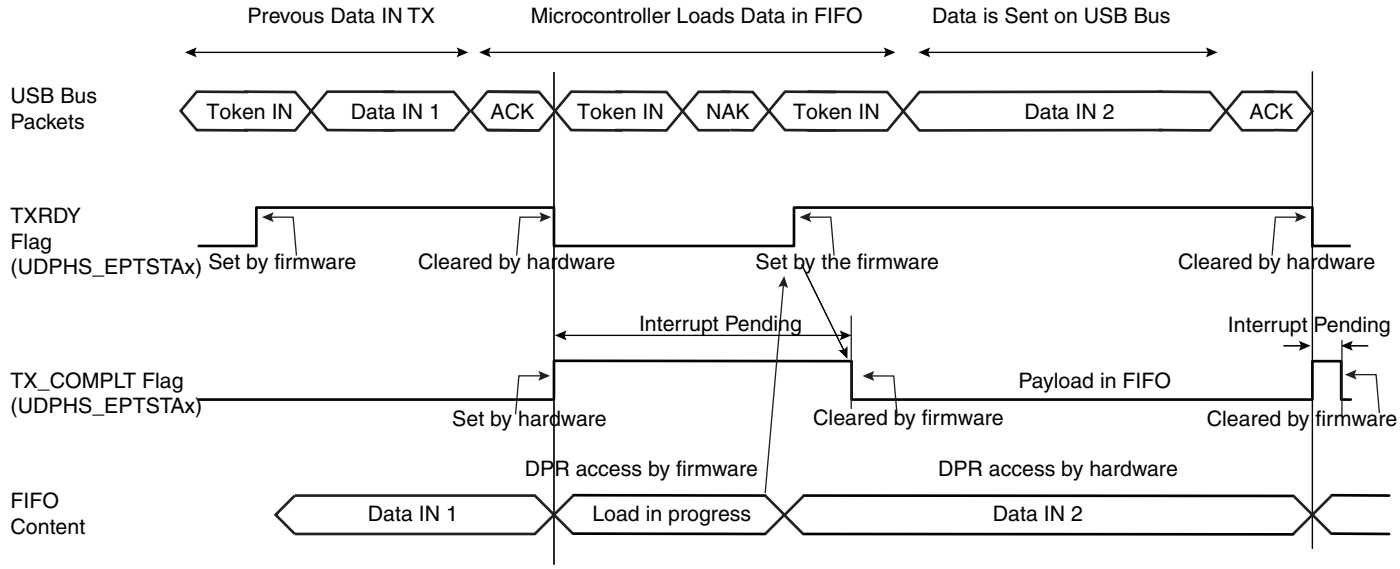


Figure 32-10.Data IN Transfer for Endpoint with Two Banks

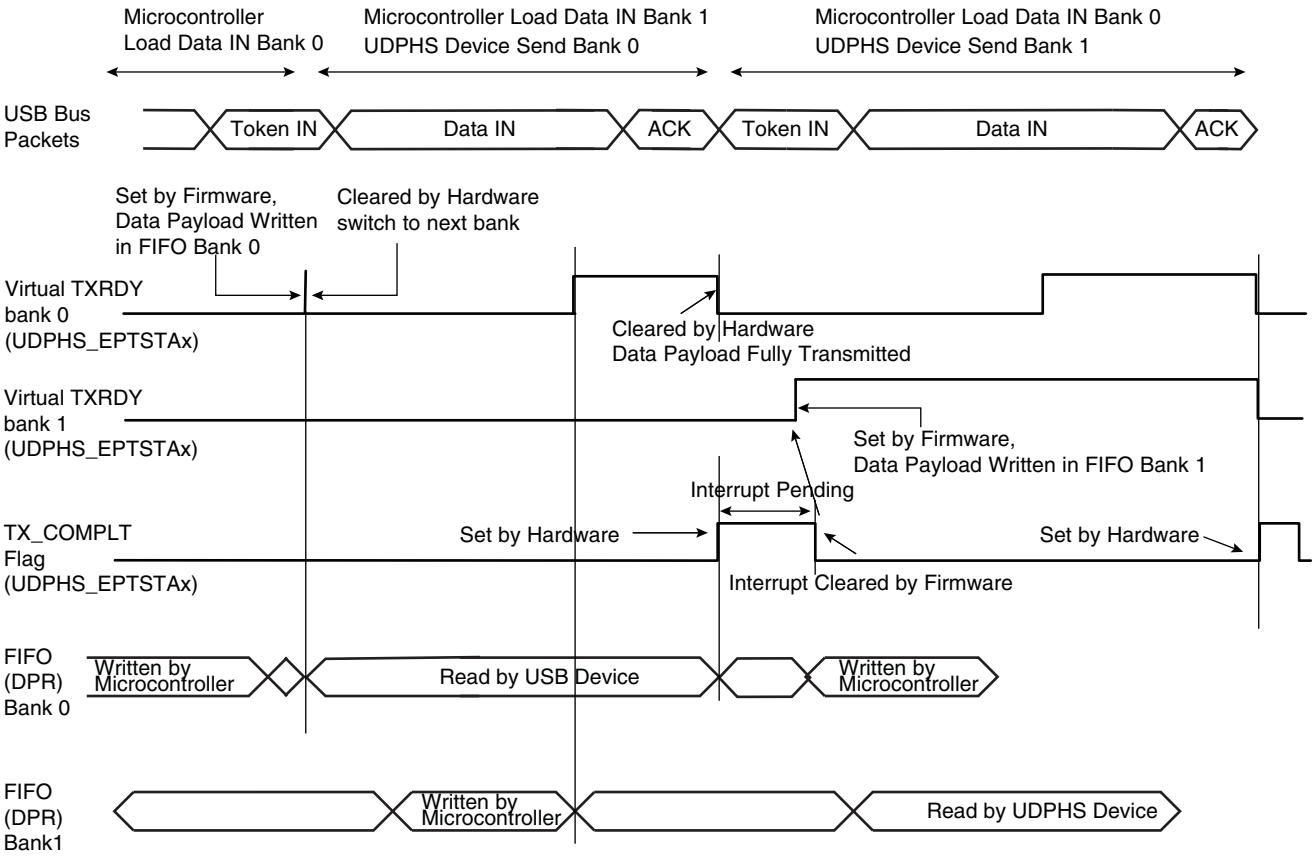
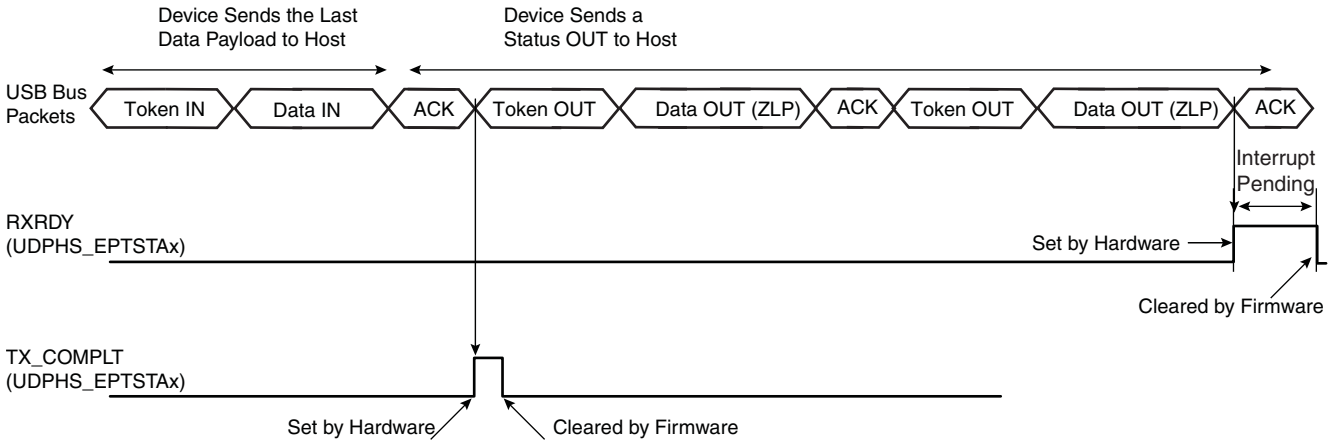
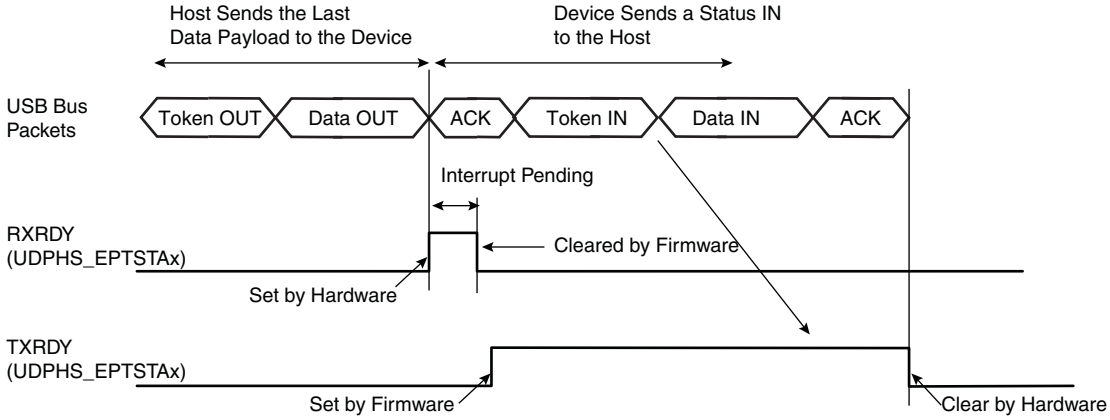


Figure 32-11.Data IN Followed By Status OUT Transfer at the End of a Control Transfer



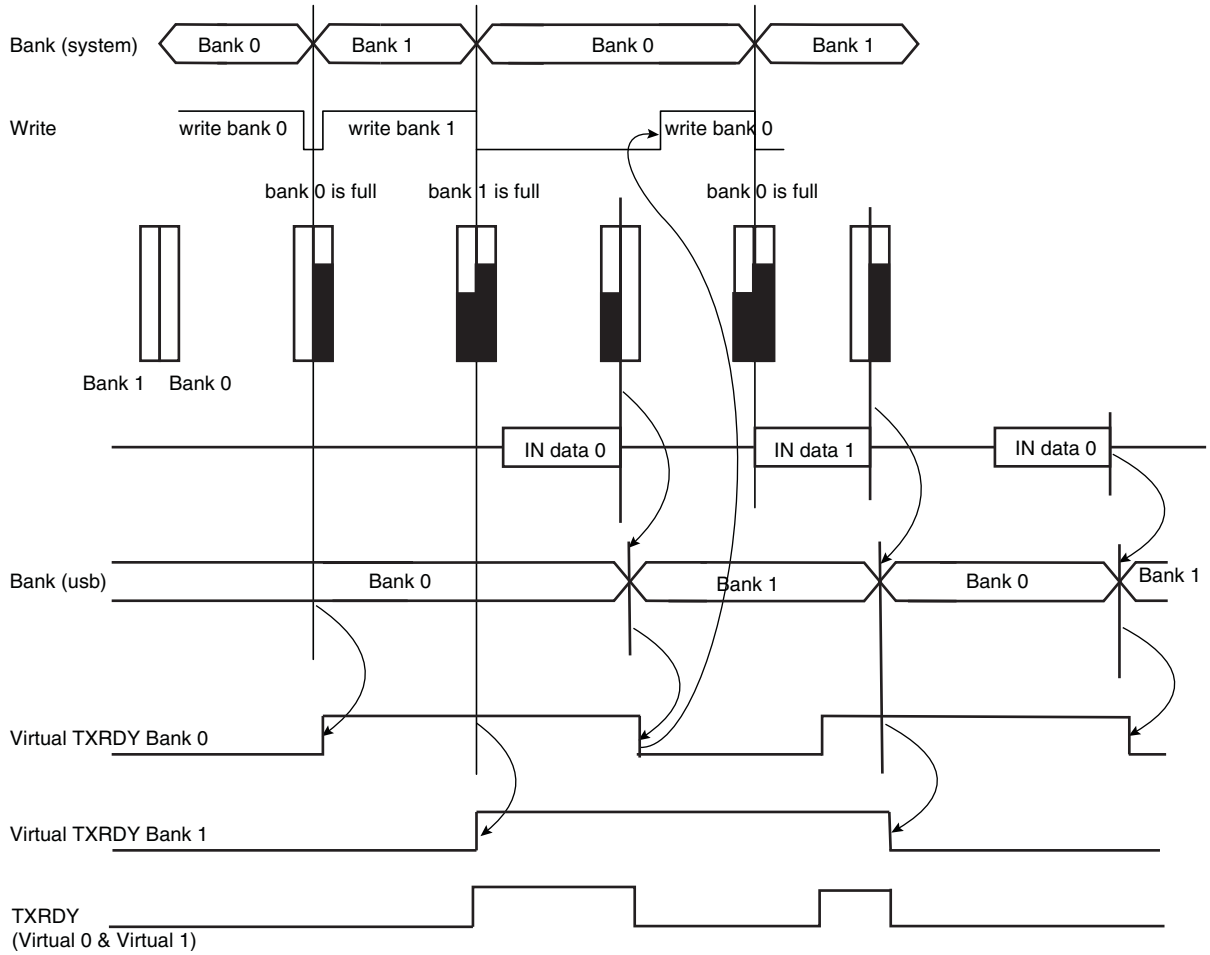
Note: A NAK handshake is always generated at the first status stage token.

Figure 32-12.Data OUT Followed by Status IN Transfer



Note: Before proceeding to the status stage, the software should determine that there is no risk of extra data from the host (data stage). If not certain (non-predictable data stage length), then the software should wait for a NAK-IN interrupt before proceeding to the status stage. This precaution should be taken to avoid collision in the FIFO.

Figure 32-13.Autovalid with DMA



Note: In the illustration above Autovalid validates a bank as full, although this might not be the case, in order to continue processing data and to send to DMA.

32.6.10.7 Isochronous IN

Isochronous-IN is used to transmit a stream of data whose timing is implied by the delivery rate. Isochronous transfer provides periodic, continuous communication between host and device.

It guarantees bandwidth and low latencies appropriate for telephony, audio, video, etc.

If the endpoint is not available (TXRDY_TRER = 0), then the device does not answer to the host. An ERR_FL_ISO interrupt is generated in the UDPHS_EPTSTAx register and once enabled, then sent to the CPU.

The STALL_SNT command bit is not used for an ISO-IN endpoint.

32.6.10.8 High Bandwidth Isochronous Endpoint Handling: IN Example

For high bandwidth isochronous endpoints, the DMA can be programmed with the number of transactions (BUFF_LENGTH field in UDPHS_DMACONTROLx) and the system should provide the required number of packets per microframe, otherwise, the host will notice a sequencing problem.

A response should be made to the first token IN recognized inside a microframe under the following conditions:

- If at least one bank has been validated, the correct DATAx corresponding to the programmed Number Of Transactions per Microframe (NB_TRANS) should be answered. In case of a subsequent missed or corrupted token IN inside the microframe, the USB 2.0 Core available data bank(s) that should normally have been transmitted during that microframe shall be flushed at its end. If this flush occurs, an error condition is flagged (ERR_FLUSH is set in UDPHS_EPTSTAx).

- If no bank is validated yet, the default DATA0 ZLP is answered and underflow is flagged (ERR_FL_ISO is set in UDPHS_EPTSTAx). Then, no data bank is flushed at microframe end.
- If no data bank has been validated at the time when a response should be made for the second transaction of NB_TRANS = 3 transactions microframe, a DATA1 ZLP is answered and underflow is flagged (ERR_FL_ISO is set in UDPHS_EPTSTAx). If and only if remaining untransmitted banks for that microframe are available at its end, they are flushed and an error condition is flagged (ERR_FLUSH is set in UDPHS_EPTSTAx).
- If no data bank has been validated at the time when a response should be made for the last programmed transaction of a microframe, a DATA0 ZLP is answered and underflow is flagged (ERR_FL_ISO is set in UDPHS_EPTSTAx). If and only if the remaining untransmitted data bank for that microframe is available at its end, it is flushed and an error condition is flagged (ERR_FLUSH is set in UDPHS_EPTSTAx).
- If at the end of a microframe no valid token IN has been recognized, no data bank is flushed and no error condition is reported.

At the end of a microframe in which at least one data bank has been transmitted, if less than NB_TRANS banks have been validated for that microframe, an error condition is flagged (ERR_TRANS is set in UDPHS_EPTSTAx).

Cases of Error (in UDPHS_EPTSTAx)

- ERR_FL_ISO: There was no data to transmit inside a microframe, so a ZLP is answered by default.
- ERR_FLUSH: At least one packet has been sent inside the microframe, but the number of token IN received is lesser than the number of transactions actually validated (TXRDY_TRER) and likewise with the NB_TRANS programmed.
- ERR_TRANS: At least one packet has been sent inside the microframe, but the number of token IN received is lesser than the number of programmed NB_TRANS transactions and the packets not requested were not validated.
- ERR_FL_ISO + ERR_FLUSH: At least one packet has been sent inside the microframe, but the data has not been validated in time to answer one of the following token IN.
- ERR_FL_ISO + ERR_TRANS: At least one packet has been sent inside the microframe, but the data has not been validated in time to answer one of the following token IN and the data can be discarded at the microframe end.
- ERR_FLUSH + ERR_TRANS: The first token IN has been answered and it was the only one received, a second bank has been validated but not the third, whereas NB_TRANS was waiting for three transactions.
- ERR_FL_ISO + ERR_FLUSH + ERR_TRANS: The first token IN has been treated, the data for the second Token IN was not available in time, but the second bank has been validated before the end of the microframe. The third bank has not been validated, but three transactions have been set in NB_TRANS.

32.6.10.9 Data OUT

32.6.10.10 Bulk OUT or Interrupt OUT

Like data IN, data OUT packets are sent by the host during the data or the status stage of control transfer or during an interrupt/bulk/isochronous OUT transfer. Data buffers are sent packet by packet under the control of the application or under the control of the DMA channel.

32.6.10.11 Bulk OUT or Interrupt OUT: Receiving a Packet Under Application Control (Host to Device)

Algorithm Description for Each Packet:

- The application enables an interrupt on RXRDY_TXKL.
- When an interrupt on RXRDY_TXKL is received, the application knows that UDPHS_EPTSTAx register BYTE_COUNT bytes have been received.
- The application reads the BYTE_COUNT bytes from the endpoint.
- The application clears RXRDY_TXKL.

Note: If the application does not know the size of the transfer, it may **not** be a good option to use AUTO_VALID. Because if a zero-length-packet is received, the RXRDY_TXKL is automatically cleared by the AUTO_VALID hardware and if the endpoint interrupt is triggered, the software will not find its originating flag when reading the UDPHS_EPTSTAx register.

Algorithm to Fill Several Packets:

- The application enables the interrupts of BUSY_BANK and AUTO_VALID.
- When a BUSY_BANK interrupt is received, the application knows that all banks available for the endpoint have been filled. Thus, the application can read all banks available.

If the application doesn't know the size of the receive buffer, instead of using the BUSY_BANK interrupt, the application must use RXRDY_TXKL.

32.6.10.12 Bulk OUT or Interrupt OUT: Sending a Buffer Using DMA (Host To Device)

To use the DMA setting, the AUTO_VALID field is mandatory.

See 32.6.10.6 Bulk IN or Interrupt IN: Sending a Buffer Using DMA (Device to Host) for more information.

DMA Configuration Example:

1. First program UDPHS_DMAADDRESSx with the address of the buffer that should be transferred.
2. Enable the interrupt of the DMA in UDPHS_IEN
3. Program the DMA Channelx Control Register:
   - Size of buffer to be sent.
   - END_B_EN: Can be used for OUT packet truncation (discarding of unbuffered packet data) at the end of DMA buffer.
   - END_BUFFIT: Generate an interrupt when BUFF_COUNT in the UDPHS_DMASTATUSx register reaches 0.
   - END_TR_EN: End of transfer enable, the UDPHS device can put an end to the current DMA transfer, in case of a short packet.
   - END_TR_IT: End of transfer interrupt enable, an interrupt is sent after the last USB packet has been transferred by the DMA, if the USB transfer ended with a short packet. (Beneficial when the receive size is unknown.)
   - CHANN_ENB: Run and stop at end of buffer.

For OUT transfer, the bank will be automatically cleared by hardware when the application has read all the bytes in the bank (the bank is empty).

Notes:
1. When a zero-length-packet is received, RXRDY_TXKL bit in UDPHS_EPTSTAx is cleared automatically by AUTO_VALID, and the application knows of the end of buffer by the presence of the END_TR_IT.
2. If the host sends a zero-length packet, and the endpoint is free, then the device sends an ACK. No data is written in the endpoint, the RXRDY_TXKL interrupt is generated, and the BYTE_COUNT field in UDPHS_EPTSTAx is null.

Figure 32-14.Data OUT Transfer for Endpoint with One Bank

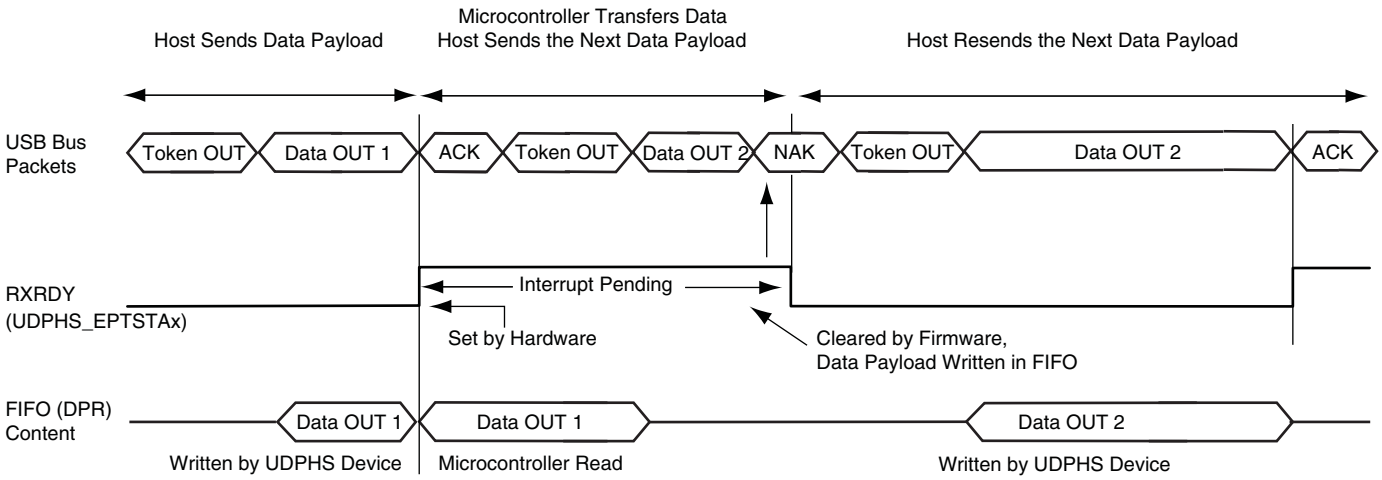
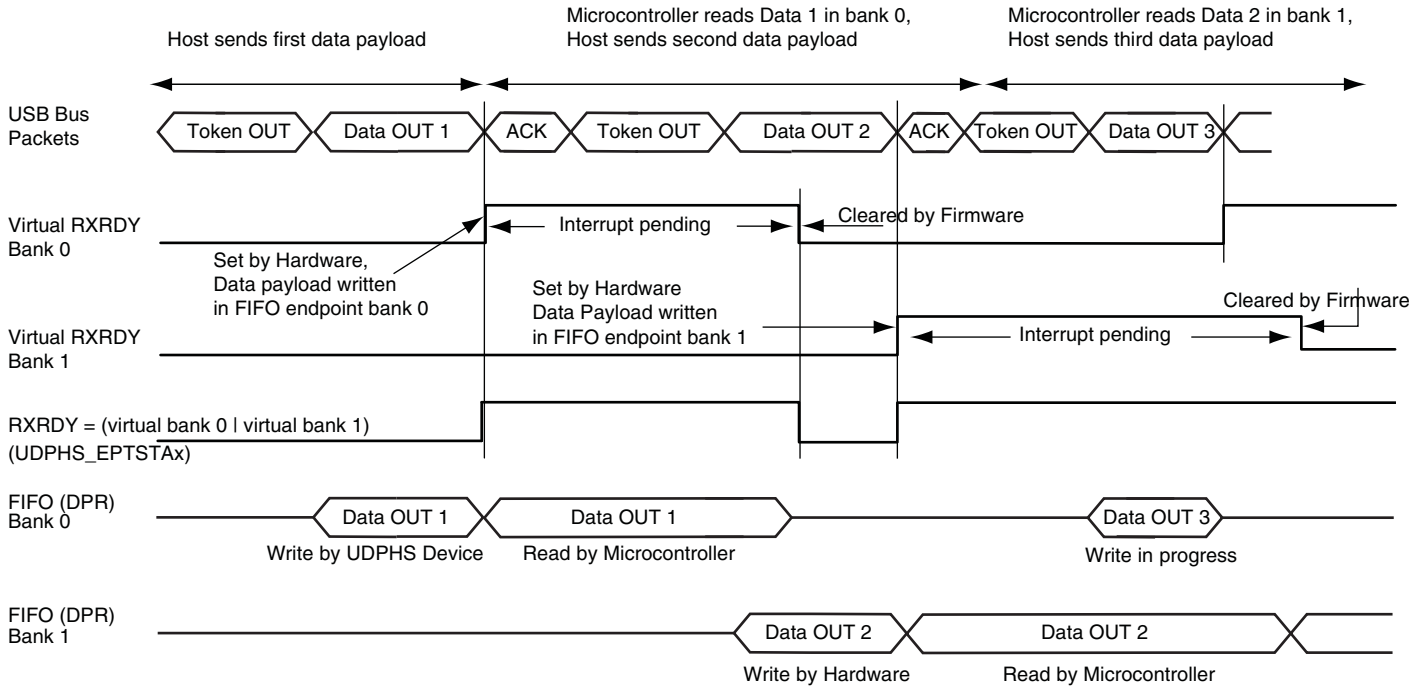
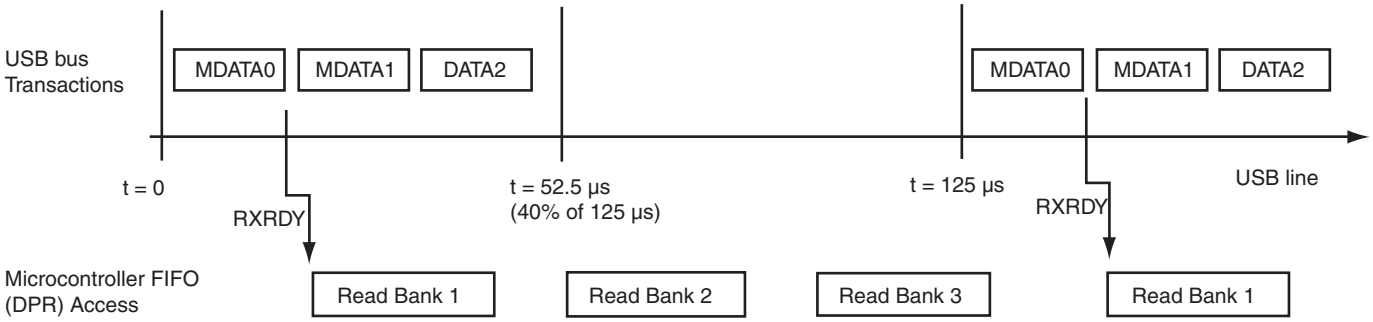


Figure 32-15.Data OUT Transfer for an Endpoint with Two Banks



32.6.10.13 High Bandwidth Isochronous Endpoint OUT

Figure 32-16.Bank Management, Example of Three Transactions per Microframe



USB 2.0 supports individual High Speed isochronous endpoints that require data rates up to 192 Mb/s (24 MB/s): 3x1024 data bytes per microframe.

To support such a rate, two or three banks may be used to buffer the three consecutive data packets. The microcontroller (or the DMA) should be able to empty the banks very rapidly (at least 24 MB/s on average).

NB_TRANS field in UDPHS_EPTCFGx register = Number Of Transactions per Microframe.

If NB_TRANS > 1 then it is High Bandwidth.

Example:

- If NB_TRANS = 3, the sequence should be either
  - MData0
  - MData0/Data1
  - MData0/Data1/Data2
- If NB_TRANS = 2, the sequence should be either

- MData0
- MData0/Data1

- If NB_TRANS = 1, the sequence should be
  - Data0

32.6.10.14 Isochronous Endpoint Handling: OUT Example

The user can ascertain the bank status (free or busy), and the toggle sequencing of the data packet for each bank with the UDPHS_EPTSTAx register in the three bit fields as follows:

- TOGGLESQ_STA: PID of the data stored in the current bank
- CURBK: Number of the bank currently being accessed by the microcontroller.
- BUSY_BANK_STA: Number of busy bank

This is particularly useful in case of a missing data packet.

If the inter-packet delay between the OUT token and the Data is greater than the USB standard, then the ISO-OUT transaction is ignored. (Payload data is not written, no interrupt is generated to the CPU.)

If there is a data CRC (Cyclic Redundancy Check) error, the payload is, none the less, written in the endpoint. The ERR_CRC_NTR flag is set in UDPHS_EPTSTAx register.

If the endpoint is already full, the packet is not written in the DPRAM. The ERR_FL_ISO flag is set in UDPHS_EPTSTAx.

If the payload data is greater than the maximum size of the endpoint, then the ERR_OVFLW flag is set. It is the task of the CPU to manage this error. The data packet is written in the endpoint (except the extra data).

If the host sends a Zero Length Packet, and the endpoint is free, no data is written in the endpoint, the RXRDY_TXKL flag is set, and the BYTE_COUNT field in UDPHS_EPTSTAx register is null.

The FRCESTALL command bit is unused for an isochonous endpoint.

Otherwise, payload data is written in the endpoint, the RXRDY_TXKL interrupt is generated and the BYTE_COUNT in UDPHS_EPTSTAx register is updated.

32.6.10.15 STALL

STALL is returned by a function in response to an IN token or after the data phase of an OUT or in response to a PING transaction. STALL indicates that a function is unable to transmit or receive data, or that a control pipe request is not supported.

- OUT

To stall an endpoint, set the FRCESTALL bit in UDPHS_EPTSETSTAx register and after the STALL_SNT flag has been set, set the TOGGLE_SEG bit in the UDPHS_EPTCLRSTAx register.

- IN

Set the FRCESTALL bit in UDPHS_EPTSETSTAx register.

Figure 32-17.Stall Handshake Data OUT Transfer

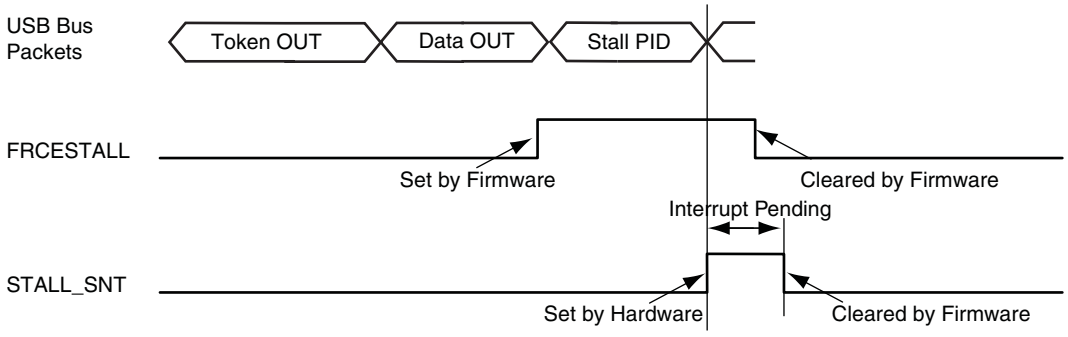
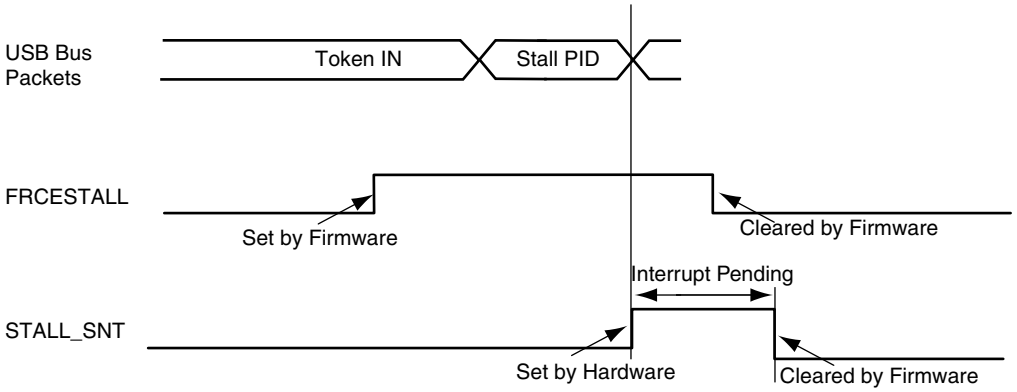


Figure 32-18.Stall Handshake Data IN Transfer



32.6.11 Speed Identification

The high speed reset is managed by the hardware.

At the connection, the host makes a reset which could be a classic reset (full speed) or a high speed reset.

At the end of the reset process (full or high), the ENDRESET interrupt is generated.

Then the CPU should read the SPEED bit in UDPHS_INTSTAx to ascertain the speed mode of the device.

32.6.12 USB V2.0 High Speed Global Interrupt

Interrupts are defined in Section 32.7.3 ”UDPHS Interrupt Enable Register” (UDPHS_IEN) and in Section 32.7.4 ”UDPHS Interrupt Status Register” (UDPHS_INTSTA).

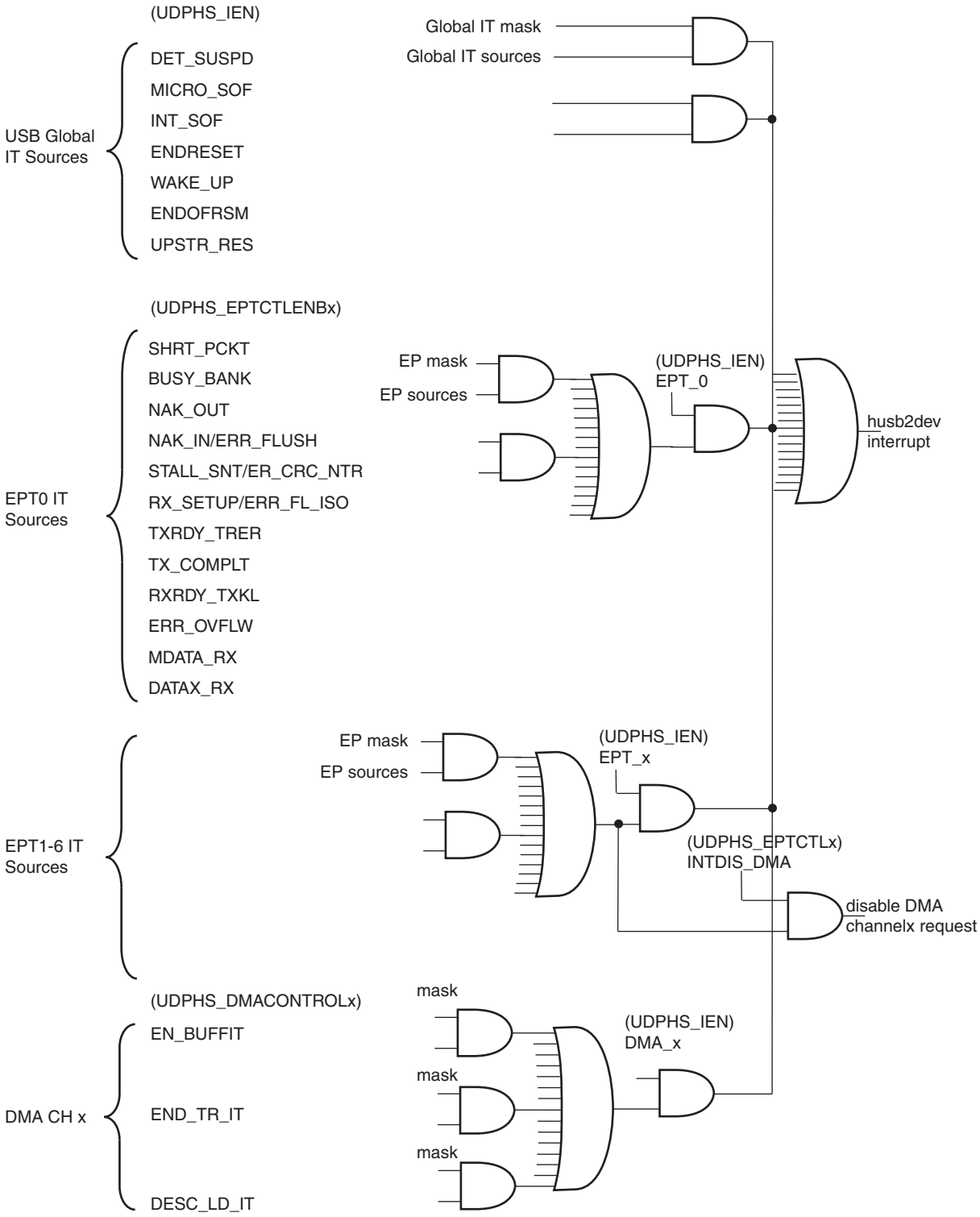
32.6.13 Endpoint Interrupts

Interrupts are enabled in UDPHS_IEN (see Section 32.7.3 ”UDPHS Interrupt Enable Register”) and individually masked in UDPHS_EPTCTLENBx (see Section 32.7.9 ”UDPHS Endpoint Control Enable Register (Control, Bulk, Interrupt Endpoints)”).

Table 32-5. Endpoint Interrupt Source Masks

SHRT_PCKT	Short Packet Interrupt
BUSY_BANK	Busy Bank Interrupt
NAK_OUT	NAKOUT Interrupt
NAK_IN/ERR_FLUSH	NAKIN/Error Flush Interrupt
STALL_SNT/ERR_CRC_NTR	Stall Sent/CRC error/Number of Transaction Error Interrupt
RX_SETUP/ERR_FL_ISO	Received SETUP/Error Flow Interrupt
TXRDY_TRER	TX Packet Read/Transaction Error Interrupt
TX_COMPLT	Transmitted IN Data Complete Interrupt
RXRDY_TXKL	Received OUT Data Interrupt
ERR_OVFLW	Overflow Error Interrupt
MDATA_RX	MDATA Interrupt
DATAX_RX	DATAx Interrupt

Figure 32-19.UDPHS Interrupt Control Interface

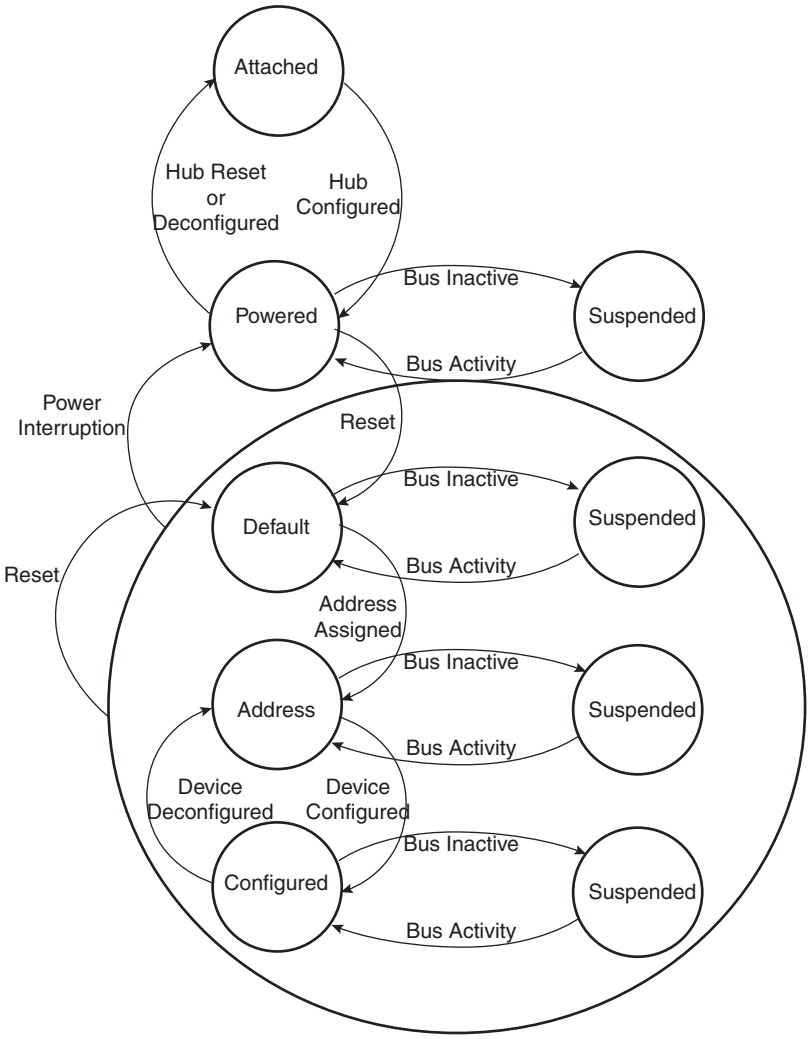


32.6.14 Power Modes

32.6.14.1 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 (USB Device Framework) of the Universal Serial Bus Specification, Rev 2.0.

Figure 32-20.UDPHS Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices may not consume more than 500 µA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake-up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake-up feature is not mandatory for all devices and must be negotiated with the host.

32.6.14.2 Not Powered State

Self powered devices can detect 5V VBUS using a PIO. When the device is not connected to a host, device power consumption can be reduced by the DETACH bit in UDPHS_CTRL. Disabling the transceiver is automatically done. HSDM, HSDP, FSDP and FSDP lines are tied to GND pull-downs integrated in the hub downstream ports.

32.6.14.3 Entering Attached State

When no device is connected, the USB FSDP and FSDM signals are tied to GND by 15 KΩ pull-downs integrated in the hub downstream ports. When a device is attached to an hub downstream port, the device connects a 1.5 KΩ pull-up on FSDP. The USB bus line goes into IDLE state, FSDP is pulled-up by the device 1.5 KΩ resistor to 3.3V and FSDM is pulled-down by the 15 KΩ resistor to GND of the host.

After pull-up connection, the device enters the powered state. The transceiver remains disabled until bus activity is detected.

In case of low power consumption need, the device can be stopped. When the device detects the VBUS, the software must enable the USB transceiver by enabling the EN_UDPHS bit in UDPHS_CTRL register.

The software can detach the pull-up by setting DETACH bit in UDPHS_CTRL register.

32.6.14.4 From Powered State to Default State (Reset)

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmasked flag ENDRESET is set in the UDPHS_IEN register and an interrupt is triggered.

Once the ENDRESET interrupt has been triggered, the device enters Default State. In this state, the UDPHS software must:

- Enable the default endpoint, setting the EPT_ENABL flag in the UDPHS_EPTCTLENB[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 in EPT_0 of the UDPHS_IEN register. The enumeration then begins by a control transfer.
- Configure the Interrupt Mask Register which has been reset by the USB reset detection
- Enable the transceiver.

In this state, the EN_UDPHS bit in UDPHS_CTRL register must be enabled.

32.6.14.5 From Default State to Address State (Address Assigned)

After a Set Address standard device request, the USB host peripheral enters the address state.

Warning: before the device enters address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDPHS device sets its new address once the TX_COMPLT flag in the UDPHS_EPTCTL[0] register has been received and cleared.

To move to address state, the driver software sets the DEV_ADDR field and the FADDR_EN flag in the UDPHS_CTRL register.

32.6.14.6 From Address State to Configured State (Device Configured)

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the BK_NUMBER, EPT_TYPE, EPT_DIR and EPT_SIZE fields in the UDPHS_EPTCFGx registers and enabling them by setting the EPT_ENABL flag in the UDPHS_EPTCTLENBx registers, and, optionally, enabling corresponding interrupts in the UDPHS_IEN register.

32.6.14.7 Entering Suspend State (Bus Activity)

When a Suspend (no bus activity on the USB bus) is detected, the DET_SUSPD signal in the UDPHS_STA register is set. This triggers an interrupt if the corresponding bit is set in the UDPHS_IEN register. This flag is cleared by writing to the UDPHS_CLRINT register. Then the device enters Suspend Mode.

In this state bus powered devices must drain less than 500 µA from the 5V VBUS. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The UDPHS device peripheral clocks can be switched off. Resume event is asynchronously detected.

32.6.14.8 Receiving a Host Resume

In Suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks disabled (however the pull-up should not be removed).

Once the resume is detected on the bus, the signal WAKE_UP in the UDPHS_INTSTA is set. It may generate an interrupt if the corresponding bit in the UDPHS_IEN register is set. This interrupt may be used to wake-up the core, enable PLL and main oscillators and configure clocks.

32.6.14.9 Sending an External Resume

In Suspend State it is possible to wake-up the host by sending an external resume.

The device waits at least 5 ms after being entered in Suspend State before sending an external resume.

The device must force a K state from 1 to 15 ms to resume the host.

32.6.15 Test Mode

A device must support the TEST_MODE feature when in the Default, Address or Configured High Speed device states.

TEST_MODE can be:

- Test_J
- Test_K
- Test_Packet
- Test_SEO_NAK

(See Section 32.7.7 “UDPHS Test Register” on page 564 for definitions of each test mode.)

```
const char test_packet_buffer[] = {
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,                 // JKJKJKJK * 9
  0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,                      // JJKKJJKK * 8
  0xEE,0xEE,0xEE,0xEE,0xEE,0xEE,0xEE,0xEE,                      // JJKKJJKK * 8
  0xFE,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, //
JJJJJJJKKKKKKK * 8
  0x7F,0xBF,0xDF,0xEF,0xF7,0xFB,0xFD,                           // JJJJJJJK * 8
  0xFC,0x7E,0xBF,0xDF,0xEF,0xF7,0xFB,0xFD,0x7E                  // {JKKKKKKK *
10}, JK
};
```

32.7 USB High Speed Device Port (UDPHS) User Interface

Table 32-6. Register Mapping

Offset	Register	Name	Access	Reset
0x00	UDPHS Control Register	UDPHS_CTRL	Read-write	0x0000_0200
0x04	UDPHS Frame Number Register	UDPHS_FNUM	Read-only	0x0000_0000
0x08 - 0x0C	Reserved	–	–	–
0x10	UDPHS Interrupt Enable Register	UDPHS_IEN	Read-write	0x0000_0010
0x14	UDPHS Interrupt Status Register	UDPHS_INTSTA	Read-only	0x0000_0000
0x18	UDPHS Clear Interrupt Register	UDPHS_CLRINT	Write-only	–
0x1C	UDPHS Endpoints Reset Register	UDPHS_EPTRST	Write-only	–
0x20 - 0xCC	Reserved	–	–	–
0xE0	UDPHS Test Register	UDPHS_TST	Read-write	0x0000_0000
0xE4 - 0xE8	Reserved	–	–	–
0x100 + endpoint * 0x20 + 0x00	UDPHS Endpoint Configuration Register	UDPHS_EPTCFG	Read-write	0x0000_0000
0x100 + endpoint * 0x20 + 0x04	UDPHS Endpoint Control Enable Register	UDPHS_EPTCTLENB	Write-only	–
0x100 + endpoint * 0x20 + 0x08	UDPHS Endpoint Control Disable Register	UDPHS_EPTCTLDIS	Write-only	–
0x100 + endpoint * 0x20 + 0x0C	UDPHS Endpoint Control Register	UDPHS_EPTCTL	Read-only	0x0000_0000(1)
0x100 + endpoint * 0x20 + 0x10	Reserved (for endpoint)	–	–	–
0x100 + endpoint * 0x20 + 0x14	UDPHS Endpoint Set Status Register	UDPHS_EPTSETSTA	Write-only	–
0x100 + endpoint * 0x20 + 0x18	UDPHS Endpoint Clear Status Register	UDPHS_EPTCLRSTA	Write-only	–
0x100 + endpoint * 0x20 + 0x1C	UDPHS Endpoint Status Register	UDPHS_EPTSTA	Read-only	0x0000_0040
0x120 - 0x1DC	UDPHS Endpoint1 to 6 (2) Registers			
0x300 + channel * 0x10 + 0x00	UDPHS DMA Next Descriptor Address Register	UDPHS_DMANXTDSC	Read-write	0x0000_0000
0x300 + channel * 0x10 + 0x04	UDPHS DMA Channel Address Register	UDPHS_DMAADDRESS	Read-write	0x0000_0000
0x300 + channel * 0x10 + 0x08	UDPHS DMA Channel Control Register	UDPHS_DMACONTROL	Read-write	0x0000_0000
0x300 + channel * 0x10 + 0x0C	UDPHS DMA Channel Status Register	UDPHS_DMASTATUS	Read-write	0x0000_0000
0x310 - 0x370	DMA Channel1 to 5 (3) Registers			

Notes:
1. The reset value for UDPHS_EPTCTL0 is 0x0000_0001.
2. The addresses for the UDPHS Endpoint registers shown here are for UDPHS Endpoint0. The structure of this group of registers is repeated successively for each endpoint according to the consecution of endpoint registers located between 0x120 and 0x1DC.
3. The DMA channel index refers to the corresponding EP number. When no DMA channel is assigned to one EP, the associated registers are reserved. This is the case for EP0, so DMA Channel 0 registers are reserved.

32.7.1 UDPHS Control Register

Name: UDPHS_CTRL

Address: 0xF803C000

Access: Read-write

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

- **DEV_ADDR: UDPHS Address**

This field contains the default address (0) after power-up or UDPHS bus reset (read), or it is written with the value set by a SET_ADDRESS request received by the device firmware (write).

- **FADDR_EN: Function Address Enable**

0 = Device is not in address state (read), or only the default function address is used (write).

1 = Device is in address state (read), or this bit is set by the device firmware after a successful status phase of a SET_ADDRESS transaction (write). When set, the only address accepted by the UDPHS controller is the one stored in the UDPHS Address field. It will not be cleared afterwards by the device firmware. It is cleared by hardware on hardware reset, or when UDPHS bus reset is received.

- **EN_UDPHS: UDPHS Enable**

0 = UDPHS is disabled (read), or this bit disables and resets the UDPHS controller (write). Switch the host to UTMI. .

1 = UDPHS is enabled (read), or this bit enables the UDPHS controller (write). Switch the host to UTMI.

- **DETACH: Detach Command**

0 = UDPHS is attached (read), or this bit pulls up the DP line (attach command) (write).

1 = UDPHS is detached, UTMI transceiver is suspended (read), or this bit simulates a detach on the UDPHS line and forces the UTMI transceiver into suspend state (Suspend M = 0) (write).

See PULLD_DIS description below.

- **REWAKEUP: Send Remote Wake Up**

0 = Remote Wake Up is disabled (read), or this bit has no effect (write).

1 = Remote Wake Up is enabled (read), or this bit forces an external interrupt on the UDPHS controller for Remote Wake UP purposes.

An Upstream Resume is sent only after the UDPHS bus has been in SUSPEND state for at least 5 ms.

This bit is automatically cleared by hardware at the end of the Upstream Resume.

- **PULLD_DIS: Pull-Down Disable**

When set, there is no pull-down on DP & DM. (DM Pull-Down = DP Pull-Down = 0).

Note: If the DETACH bit is also set, device DP & DM are left in high impedance state.

(See DETACH description above.)

DETACH	PULLD_DIS	DP	DM	Condition
0	0	Pull up	Pull down	Not recommended
0	1	Pull up	High impedance state	VBUS present
1	0	Pull down	Pull down	No VBUS
1	1	High impedance state	High impedance state	VBUS present & software disconnect

32.7.2 UDPHS Frame Number Register

Name: UDPHS_FNUM

Address: 0xF803C004

Access: Read-only

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

- **MICRO_FRAME_NUM: Microframe Number**

Number of the received microframe (0 to 7) in one frame.This field is reset at the beginning of each new frame (1 ms).

One microframe is received each 125 microseconds (1 ms/8).

- **FRAME_NUMBER: Frame Number as defined in the Packet Field Formats**

This field is provided in the last received SOF packet (see INT_SOF in the UDPHS Interrupt Status Register).

- **FNUM_ERR: Frame Number CRC Error**

This bit is set by hardware when a corrupted Frame Number in Start of Frame packet (or Micro SOF) is received.

This bit and the INT_SOF (or MICRO_SOF) interrupt are updated at the same time.

32.7.3 UDPHS Interrupt Enable Register

Name: UDPHS_IEN

Address: 0xF803C010

Access: Read-write

31	30	29	28	27	26	25	24
–	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0
7	6	5	4	3	2	1	0
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	–

- **DET_SUSPD: Suspend Interrupt Enable**

0 = Disable Suspend Interrupt.

1 = Enable Suspend Interrupt.

- **MICRO_SOF: Micro-SOF Interrupt Enable**

0 = Disable Micro-SOF Interrupt.

1 = Enable Micro-SOF Interrupt.

- **INT_SOF: SOF Interrupt Enable**

0 = Disable SOF Interrupt.

1 = Enable SOF Interrupt.

- **ENDRESET: End Of Reset Interrupt Enable**

0 = Disable End Of Reset Interrupt.

1 = Enable End Of Reset Interrupt. Automatically enabled after USB reset.

- **WAKE_UP: Wake Up CPU Interrupt Enable**

0 = Disable Wake Up CPU Interrupt.

1 = Enable Wake Up CPU Interrupt.

- **ENDOFRSM: End Of Resume Interrupt Enable**

0 = Disable Resume Interrupt.

1 = Enable Resume Interrupt.

- **UPSTR_RES: Upstream Resume Interrupt Enable**

0 = Disable Upstream Resume Interrupt.

1 = Enable Upstream Resume Interrupt.

- **EPT_x: Endpoint x Interrupt Enable**

0 = Disable the interrupts for this endpoint.

1 = Enable the interrupts for this endpoint.

- **DMA_x: DMA Channel x Interrupt Enable**

0 = Disable the interrupts for this channel.

1 = Enable the interrupts for this channel.

32.7.4 UDPHS Interrupt Status Register

Name: UDPHS_INTSTA

Address: 0xF803C014

Access: Read-only

31	30	29	28	27	26	25	24
–	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0
7	6	5	4	3	2	1	0
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	SPEED

- **SPEED: Speed Status**

0 = Reset by hardware when the hardware is in Full Speed mode.

1 = Set by hardware when the hardware is in High Speed mode

- **DET_SUSPD: Suspend Interrupt**

0 = Cleared by setting the DET_SUSPD bit in UDPHS_CLRINT register

1 = Set by hardware when a UDPHS Suspend (Idle bus for three frame periods, a J state for 3 ms) is detected. This triggers a UDPHS interrupt when the DET_SUSPD bit is set in UDPHS_IEN register.

- **MICRO_SOF: Micro Start Of Frame Interrupt**

0 = Cleared by setting the MICRO_SOF bit in UDPHS_CLRINT register.

1 = Set by hardware when an UDPHS micro start of frame PID (SOF) has been detected (every 125 us) or synthesized by the macro. This triggers a UDPHS interrupt when the MICRO_SOF bit is set in UDPHS_IEN. In case of detected SOF, the MICRO_FRAME_NUM field in UDPHS_FNUM register is incremented and the FRAME_NUMBER field doesn't change.

Note: The Micro Start Of Frame Interrupt (MICRO_SOF), and the Start Of Frame Interrupt (INT_SOF) are not generated at the same time.

- **INT_SOF: Start Of Frame Interrupt**

0 = Cleared by setting the INT_SOF bit in UDPHS_CLRINT.

1 = Set by hardware when an UDPHS Start Of Frame PID (SOF) has been detected (every 1 ms) or synthesized by the macro. This triggers a UDPHS interrupt when the INT_SOF bit is set in UDPHS_IEN register. In case of detected SOF, in High Speed mode, the MICRO_FRAME_NUMBER field is cleared in UDPHS_FNUM register and the FRAME_NUMBER field is updated.

- **ENDRESET: End Of Reset Interrupt**

0 = Cleared by setting the ENDRESET bit in UDPHS_CLRINT.

1 = Set by hardware when an End Of Reset has been detected by the UDPHS controller. This triggers a UDPHS interrupt when the ENDRESET bit is set in UDPHS_IEN.

- **WAKE_UP: Wake Up CPU Interrupt**

0 = Cleared by setting the WAKE_UP bit in UDPHS_CLRINT.

1 = Set by hardware when the UDPHS controller is in SUSPEND state and is re-activated by a filtered non-idle signal from the UDPHS line (not by an upstream resume). This triggers a UDPHS interrupt when the WAKE_UP bit is set in UDPHS_IEN register. When receiving this interrupt, the user has to enable the device controller clock prior to operation.

Note: this interrupt is generated even if the device controller clock is disabled.

- **ENDOFRSM: End Of Resume Interrupt**

0 = Cleared by setting the ENDOFRSM bit in UDPHS_CLRINT.

1 = Set by hardware when the UDPHS controller detects a good end of resume signal initiated by the host. This triggers a UDPHS interrupt when the ENDOFRSM bit is set in UDPHS_IEN.

- **UPSTR_RES: Upstream Resume Interrupt**

0 = Cleared by setting the UPSTR_RES bit in UDPHS_CLRINT.

1 = Set by hardware when the UDPHS controller is sending a resume signal called “upstream resume”. This triggers a UDPHS interrupt when the UPSTR_RES bit is set in UDPHS_IEN.

- **EPT_x: Endpoint x Interrupt**

0 = Reset when the UDPHS_EPTSTAx interrupt source is cleared.

1 = Set by hardware when an interrupt is triggered by the UDPHS_EPTSTAx register and this endpoint interrupt is enabled by the EPT_x bit in UDPHS_IEN.

- **DMA_x: DMA Channel x Interrupt**

0 = Reset when the UDPHS_DMASTATUSx interrupt source is cleared.

1 = Set by hardware when an interrupt is triggered by the DMA Channelx and this endpoint interrupt is enabled by the DMA_x bit in UDPHS_IEN.

32.7.5 UDPHS Clear Interrupt Register

Name: UDPHS_CLRINT

Address: 0xF803C018

Access: Write only

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

- **DET_SUSPD: Suspend Interrupt Clear**

0 = No effect.

1 = Clear the DET_SUSPD bit in UDPHS_INTSTA.

- **MICRO_SOF: Micro Start Of Frame Interrupt Clear**

0 = No effect.

1 = Clear the MICRO_SOF bit in UDPHS_INTSTA.

- **INT_SOF: Start Of Frame Interrupt Clear**

0 = No effect.

1 = Clear the INT_SOF bit in UDPHS_INTSTA.

- **ENDRESET: End Of Reset Interrupt Clear**

0 = No effect.

1 = Clear the ENDRESET bit in UDPHS_INTSTA.

- **WAKE_UP: Wake Up CPU Interrupt Clear**

0 = No effect.

1 = Clear the WAKE_UP bit in UDPHS_INTSTA.

- **ENDOFRSM: End Of Resume Interrupt Clear**

0 = No effect.

1 = Clear the ENDOFRSM bit in UDPHS_INTSTA.

- **UPSTR_RES: Upstream Resume Interrupt Clear**

0 = No effect.

1 = Clear the UPSTR_RES bit in UDPHS_INTSTA.

32.7.6 UDPHS Endpoints Reset Register

Name: UDPHS_EPTRST

Address: 0xF803C01C

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0

- **EPT_x: Endpoint x Reset**

0 = No effect.

1 = Reset the Endpointx state.

Setting this bit clears the Endpoint status UDPHS_EPTSTAx register, except for the TOGGLESQ_STA field.

32.7.7 UDPHS Test Register

Name: UDPHS_TST

Address: 0xF803C0E0

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	OPMODE2	TST_PKT	TST_K	TST_J	SPEED_CFG	

- **SPEED_CFG: Speed Configuration**

Speed Configuration:

Value	Name	Description
0	NORMAL	Normal Mode: The macro is in Full Speed mode, ready to make a High Speed identification, if the host supports it and then to automatically switch to High Speed mode
1		Reserved
2	HIGH_SPEED	Force High Speed: Set this value to force the hardware to work in High Speed mode. Only for debug or test purpose.
3	FULL_SPEED	Force Full Speed: Set this value to force the hardware to work only in Full Speed mode. In this configuration, the macro will not respond to a High Speed reset handshake.

- **TST_J: Test J Mode**

0 = No effect.

1 = Set to send the J state on the UDPHS line. This enables the testing of the high output drive level on the D+ line.

- **TST_K: Test K Mode**

0 = No effect.

1 = Set to send the K state on the UDPHS line. This enables the testing of the high output drive level on the D- line.

- **TST_PKT: Test Packet Mode**

0 = No effect.

1 = Set to repetitively transmit the packet stored in the current bank. This enables the testing of rise and fall times, eye patterns, jitter, and any other dynamic waveform specifications.

- **OPMODE2: OpMode2**

0 = No effect.

1 = Set to force the OpMode signal (UTMI interface) to “10”, to disable the bit-stuffing and the NRZI encoding.

Note: For the Test mode, Test_SE0_NAK (see Universal Serial Bus Specification, Revision 2.0: 7.1.20, Test Mode Support). Force the device in High Speed mode, and configure a bulk-type endpoint. Do not fill this endpoint for sending NAK to the host.

Upon command, a port's transceiver must enter the High Speed receive mode and remain in that mode until the exit action is taken. This enables the testing of output impedance, low level output voltage and loading characteristics. In addition, while in this mode, upstream facing ports (and only upstream facing ports) must respond to any IN token packet with a NAK handshake (only if the packet CRC is determined to be correct) within the normal allowed device response time. This enables testing of the device squelch level circuitry and, additionally, provides a general purpose stimulus/response test for basic functional testing.

32.7.8 UDPHS Endpoint Configuration Register

Name: UDPHS_EPTCFGx [x=0..6]

Address: 0xF803C100 [0], 0xF803C120 [1], 0xF803C140 [2], 0xF803C160 [3], 0xF803C180 [4], 0xF803C1A0 [5], 0xF803C1C0 [6]

Access: Read-write

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

- **EPT_SIZE: Endpoint Size**

Set this field according to the endpoint size in bytes (see Section 32.6.6 ”Endpoint Configuration”).

Endpoint Size (1)

Value	Name	Description
0	8	8 bytes
1	16	16 bytes
2	32	32 bytes
3	64	64 bytes
4	128	128 bytes
5	256	256 bytes
6	512	512 bytes
7	1024	1024 bytes

Note: 1. 1024 bytes is only for isochronous endpoint.

- **EPT_DIR: Endpoint Direction**

0 = Clear this bit to configure OUT direction for Bulk, Interrupt and Isochronous endpoints.

1 = Set this bit to configure IN direction for Bulk, Interrupt and Isochronous endpoints.

For Control endpoints this bit has no effect and should be left at zero.

- **EPT_TYPE: Endpoint Type**

Set this field according to the endpoint type (see Section 32.6.6 ”Endpoint Configuration”).

(Endpoint 0 should always be configured as control)

Endpoint Type

Value	Name	Description
0	CTRL8	Control endpoint
1	ISO	Isochronous endpoint
2	BULK	Bulk endpoint
3	INT	Interrupt endpoint

- **BK_NUMBER: Number of Banks**

Set this field according to the endpoint's number of banks (see Section 32.6.6 "Endpoint Configuration").

Number of Banks

Value	Name	Description
0	0	Zero bank, the endpoint is not mapped in memory
1	1	One bank (bank 0)
2	2	Double bank (Ping-Pong: bank0/bank1)
3	3	Triple bank (bank0/bank1/bank2)

- **NB_TRANS: Number Of Transaction per Microframe**

The Number of transactions per microframe is set by software.

Note: Meaningful for high bandwidth isochronous endpoint only.

- **EPT_MAPD: Endpoint Mapped**

0 = The user should reprogram the register with correct values.

1 = Set by hardware when the endpoint size (EPT_SIZE) and the number of banks (BK_NUMBER) are correct regarding:

  – The fifo max capacity (FIFO_MAX_SIZE in UDPHS_IPFEATURES register)
  – The number of endpoints/banks already allocated
  – The number of allowed banks for this endpoint

32.7.9 UDPHS Endpoint Control Enable Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCTLENBx [x=0..6]

Access: Write-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN	STALL_SNT	RX_SETUP	TXRDY	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
–	–	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

This register view is relevant only if EPT_TYPE=0x0, 0x2 or 0x3 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)” on page 576.

- **EPT_ENABL: Endpoint Enable**

0 = No effect.

1 = Enable endpoint according to the device configuration.

- **AUTO_VALID: Packet Auto-Valid Enable**

0 = No effect.

1 = Enable this bit to automatically validate the current packet and switch to the next bank for both IN and OUT transfers.

- **INTDIS_DMA: Interrupts Disable DMA**

0 = No effect.

1 = If set, when an enabled endpoint-originated interrupt is triggered, the DMA request is disabled.

- **NYET_DIS: NYET Disable (Only for High Speed Bulk OUT endpoints)**

0 = No effect.

1 = Forces an ACK response to the next High Speed Bulk OUT transfer instead of a NYET response.

- **ERR_OVFLW: Overflow Error Interrupt Enable**

0 = No effect.

1 = Enable Overflow Error Interrupt.

- **RXRDY_TXKL: Received OUT Data Interrupt Enable**

0 = No effect.

1 = Enable Received OUT Data Interrupt.

- **TX_COMPLT: Transmitted IN Data Complete Interrupt Enable**

0 = No effect.

1 = Enable Transmitted IN Data Complete Interrupt.

- **TXRDY: TX Packet Ready Interrupt Enable**

0 = No effect.

1 = Enable TX Packet Ready/Transaction Error Interrupt.

- **RX_SETUP: Received SETUP**

0 = No effect.

1 = Enable RX_SETUP Interrupt.

- **STALL_SNT: Stall Sent Interrupt Enable**

0 = No effect.

1 = Enable Stall Sent Interrupt.

- **NAK_IN: NAKIN Interrupt Enable**

0 = No effect.

1 = Enable NAKIN Interrupt.

- **NAK_OUT: NAKOUT Interrupt Enable**

0 = No effect.

1 = Enable NAKOUT Interrupt.

- **BUSY_BANK: Busy Bank Interrupt Enable**

0 = No effect.

1 = Enable Busy Bank Interrupt.

- **SHRT_PCKT: Short Packet Send/Short Packet Interrupt Enable**

For OUT endpoints:

0 = No effect.

1 = Enable Short Packet Interrupt.

For IN endpoints:

Guarantees short packet at end of DMA Transfer if the UDPHS_DMACONTROLx register END_B_EN and UDPHS_EPTCTLx register AUTOVALID bits are also set.

32.7.10 UDPHS Endpoint Control Enable Register (Isochronous Endpoints)

Name: UDPHS_EPTCTLENBx [x=0..6] (ISOENDPT)

Address: 0xF803C104 [0], 0xF803C124 [1], 0xF803C144 [2], 0xF803C164 [3], 0xF803C184 [4], 0xF803C1A4 [5], 0xF803C1C4 [6]

Access: Write-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
–	ERR_FLUSH	ERR_CRC_NTR	ERR_FL_ISO	TXRDY_TRER	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
MDATA_RX	DATAX_RX	–	–	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

This register view is relevant only if EPT_TYPE=0x1 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Control Register (Isochronous Endpoint)” on page 579.

- **EPT_ENABL: Endpoint Enable**

0 = No effect.

1 = Enable endpoint according to the device configuration.

- **AUTO_VALID: Packet Auto-Valid Enable**

0 = No effect.

1 = Enable this bit to automatically validate the current packet and switch to the next bank for both IN and OUT transfers.

- **INTDIS_DMA: Interrupts Disable DMA**

0 = No effect.

1 = If set, when an enabled endpoint-originated interrupt is triggered, the DMA request is disabled.

- **DATAX_RX: DATAx Interrupt Enable (Only for high bandwidth Isochronous OUT endpoints)**

0 = No effect.

1 = Enable DATAx Interrupt.

- **MDATA_RX: MDATA Interrupt Enable (Only for high bandwidth Isochronous OUT endpoints)**

0 = No effect.

1 = Enable MDATA Interrupt.

- **ERR_OVFLW: Overflow Error Interrupt Enable**

0 = No effect.

1 = Enable Overflow Error Interrupt.

- **RXRDY_TXKL: Received OUT Data Interrupt Enable**

0 = No effect.

1 = Enable Received OUT Data Interrupt.

- **TX_COMPLT: Transmitted IN Data Complete Interrupt Enable**

0 = No effect.

1 = Enable Transmitted IN Data Complete Interrupt.

- **TXRDY_TRER: TX Packet Ready/Transaction Error Interrupt Enable**

0 = No effect.

1 = Enable TX Packet Ready/Transaction Error Interrupt.

- **ERR_FL_ISO: Error Flow Interrupt Enable**

0 = No effect.

1 = Enable Error Flow ISO Interrupt.

- **ERR_CRC_NTR: ISO CRC Error/Number of Transaction Error Interrupt Enable**

0 = No effect.

1 = Enable Error CRC ISO/Error Number of Transaction Interrupt.

- **ERR_FLUSH: Bank Flush Error Interrupt Enable**

0 = No effect.

1 = Enable Bank Flush Error Interrupt.

- **BUSY_BANK: Busy Bank Interrupt Enable**

0 = No effect.

1 = Enable Busy Bank Interrupt.

- **SHRT_PCKT: Short Packet Send/Short Packet Interrupt Enable**

For OUT endpoints:

0 = No effect.

1 = Enable Short Packet Interrupt.

For IN endpoints:

Guarantees short packet at end of DMA Transfer if the UDPHS_DMACONTROLx register END_B_EN and UDPHS_EPTCTLx register AUTOVALID bits are also set.

32.7.11 UDPHS Endpoint Control Disable Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCTLDISx [x=0..6]

Address: 0xF803C108 [0], 0xF803C128 [1], 0xF803C148 [2], 0xF803C168 [3], 0xF803C188 [4], 0xF803C1A8 [5], 0xF803C1C8 [6]

Access: Write-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN	STALL_SNT	RX_SETUP	TXRDY	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
–	–	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_DISABL

This register view is relevant only if EPT_TYPE=0x0, 0x2 or 0x3 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)” on page 576.

- **EPT_DISABL: Endpoint Disable**

0 = No effect.

1 = Disable endpoint.

- **AUTO_VALID: Packet Auto-Valid Disable**

0 = No effect.

1 = Disable this bit to not automatically validate the current packet.

- **INTDIS_DMA: Interrupts Disable DMA**

0 = No effect.

1 = Disable the “Interrupts Disable DMA”.

- **NYET_DIS: NYET Enable (Only for High Speed Bulk OUT endpoints)**

0 = No effect.

1 = Let the hardware handle the handshake response for the High Speed Bulk OUT transfer.

- **ERR_OVFLW: Overflow Error Interrupt Disable**

0 = No effect.

1 = Disable Overflow Error Interrupt.

- **RXRDY_TXKL: Received OUT Data Interrupt Disable**

0 = No effect.

1 = Disable Received OUT Data Interrupt.

- **TX_COMPLT: Transmitted IN Data Complete Interrupt Disable**

0 = No effect.

1 = Disable Transmitted IN Data Complete Interrupt.

- **TXRDY: TX Packet Ready Interrupt Disable**

0 = No effect.

1 = Disable TX Packet Ready/Transaction Error Interrupt.

- **RX_SETUP: Received SETUP Interrupt Disable**

0 = No effect.

1 = Disable RX_SETUP Interrupt.

- **STALL_SNT: Stall Sent Interrupt Disable**

0 = No effect.

1 = Disable Stall Sent Interrupt.

- **NAK_IN: NAKIN Interrupt Disable**

0 = No effect.

1 = Disable NAKIN Interrupt.

- **NAK_OUT: NAKOUT Interrupt Disable**

0 = No effect.

1 = Disable NAKOUT Interrupt.

- **BUSY_BANK: Busy Bank Interrupt Disable**

0 = No effect.

1 = Disable Busy Bank Interrupt.

- **SHRT_PCKT: Short Packet Interrupt Disable**

For OUT endpoints:

0 = No effect.

1 = Disable Short Packet Interrupt.

For IN endpoints:

Never automatically add a zero length packet at end of DMA transfer.

32.7.12 UDPHS Endpoint Control Disable Register (Isochronous Endpoint)

Name: UDPHS_EPTCTLDISx [x=0..6] (ISOENDPT)

Address: 0xF803C108 [0], 0xF803C128 [1], 0xF803C148 [2], 0xF803C168 [3], 0xF803C188 [4], 0xF803C1A8 [5], 0xF803C1C8 [6]

Access: Write-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
–	ERR_FLUSH	ERR_CRC_NTR	ERR_FL_ISO	TXRDY_TRER	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
MDATA_RX	DATAX_RX	–	–	INTDIS_DMA	–	AUTO_VALID	EPT_DISABL

This register view is relevant only if EPT_TYPE=0x1 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Control Register (Isochronous Endpoint)” on page 579.

- **EPT_DISABL: Endpoint Disable**

0 = No effect.

1 = Disable endpoint.

- **AUTO_VALID: Packet Auto-Valid Disable**

0 = No effect.

1 = Disable this bit to not automatically validate the current packet.

- **INTDIS_DMA: Interrupts Disable DMA**

0 = No effect.

1 = Disable the “Interrupts Disable DMA”.

- **DATAX_RX: DATAx Interrupt Disable (Only for High Bandwidth Isochronous OUT endpoints)**

0 = No effect.

1 = Disable DATAx Interrupt.

- **MDATA_RX: MDATA Interrupt Disable (Only for High Bandwidth Isochronous OUT endpoints)**

0 = No effect.

1 = Disable MDATA Interrupt.

- **ERR_OVFLW: Overflow Error Interrupt Disable**

0 = No effect.

1 = Disable Overflow Error Interrupt.

- **RXRDY_TXKL: Received OUT Data Interrupt Disable**

0 = No effect.

1 = Disable Received OUT Data Interrupt.

- **TX_COMPLT: Transmitted IN Data Complete Interrupt Disable**

0 = No effect.

1 = Disable Transmitted IN Data Complete Interrupt.

- **TXRDY_TRER: TX Packet Ready/Transaction Error Interrupt Disable**

0 = No effect.

1 = Disable TX Packet Ready/Transaction Error Interrupt.

- **ERR_FL_ISO: Error Flow Interrupt Disable**

0 = No effect.

1 = Disable Error Flow ISO Interrupt.

- **ERR_CRC_NTR: ISO CRC Error/Number of Transaction Error Interrupt Disable**

0 = No effect.

1 = Disable Error CRC ISO/Error Number of Transaction Interrupt.

- **ERR_FLUSH: bank flush error Interrupt Disable**

0 = No effect.

1 = Disable Bank Flush Error Interrupt.

- **BUSY_BANK: Busy Bank Interrupt Disable**

0 = No effect.

1 = Disable Busy Bank Interrupt.

- **SHRT_PCKT: Short Packet Interrupt Disable**

For OUT endpoints:

0 = No effect.

1 = Disable Short Packet Interrupt.

For IN endpoints:

Never automatically add a zero length packet at end of DMA transfer.

32.7.13 UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCTLx [x=0..6]

Access: Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN	STALL_SNT	RX_SETUP	TXRDY	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
–	–	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

This register view is relevant only if EPT_TYPE=0x0, 0x2 or 0x3 in "UDPHS Endpoint Configuration Register" on page 566

- **EPT_ENABL: Endpoint Enable**

0 = If cleared, the endpoint is disabled according to the device configuration. Endpoint 0 should always be enabled after a hardware or UDPHS bus reset and participate in the device configuration.

1 = If set, the endpoint is enabled according to the device configuration.

- **AUTO_VALID: Packet Auto-Valid Enabled (Not for CONTROL Endpoints)**

Set this bit to automatically validate the current packet and switch to the next bank for both IN and OUT endpoints.

For IN Transfer:

If this bit is set, then the UDPHS_EPTSTAx register TXRDY bit is set automatically when the current bank is full and at the end of DMA buffer if the UDPHS_DMACONTROLx register END_B_EN bit is set.

The user may still set the UDPHS_EPTSTAx register TXRDY bit if the current bank is not full, unless the user wants to send a Zero Length Packet by software.

For OUT Transfer:

If this bit is set, then the UDPHS_EPTSTAx register RXRDY_TXKL bit is automatically reset for the current bank when the last packet byte has been read from the bank FIFO or at the end of DMA buffer if the UDPHS_DMACONTROLx register END_B_EN bit is set. For example, to truncate a padded data packet when the actual data transfer size is reached.

The user may still clear the UDPHS_EPTSTAx register RXRDY_TXKL bit, for example, after completing a DMA buffer by software if UDPHS_DMACONTROLx register END_B_EN bit was disabled or in order to cancel the read of the remaining data bank(s).

- **INTDIS_DMA: Interrupt Disables DMA**

If set, when an enabled endpoint-originated interrupt is triggered, the DMA request is disabled regardless of the UDPHS_IEN register EPT_x bit for this endpoint. Then, the firmware will have to clear or disable the interrupt source or clear this bit if transfer completion is needed.

If the exception raised is associated with the new system bank packet, then the previous DMA packet transfer is normally completed, but the new DMA packet transfer is not started (not requested).

If the exception raised is not associated to a new system bank packet (NAK_IN, NAK_OUT...), then the request cancellation may happen at any time and may immediately stop the current DMA transfer.

This may be used, for example, to identify or prevent an erroneous packet to be transferred into a buffer or to complete a DMA buffer by software after reception of a short packet.

- **NYET_DIS: NYET Disable (Only for High Speed Bulk OUT endpoints)**

0 = If cleared, this bit lets the hardware handle the handshake response for the High Speed Bulk OUT transfer.

1 = If set, this bit forces an ACK response to the next High Speed Bulk OUT transfer instead of a NYET response.

Note: According to the *Universal Serial Bus Specification, Rev 2.0* (8.5.1.1 NAK Responses to OUT/DATA During PING Protocol), a NAK response to an HS Bulk OUT transfer is expected to be an unusual occurrence.

- **ERR_OVFLW: Overflow Error Interrupt Enabled**

0 = Overflow Error Interrupt is masked.

1 = Overflow Error Interrupt is enabled.

- **RXRDY_TXKL: Received OUT Data Interrupt Enabled**

0 = Received OUT Data Interrupt is masked.

1 = Received OUT Data Interrupt is enabled.

- **TX_COMPLT: Transmitted IN Data Complete Interrupt Enabled**

0 = Transmitted IN Data Complete Interrupt is masked.

1 = Transmitted IN Data Complete Interrupt is enabled.

- **TXRDY: TX Packet Ready Interrupt Enabled**

0 = TX Packet Ready Interrupt is masked.

1 = TX Packet Ready Interrupt is enabled.

Caution: Interrupt source is active as long as the corresponding UDPHS_EPTSTAx register TXRDY flag remains low. If there are no more banks available for transmitting after the software has set UDPHS_EPTSTAx/TXRDY for the last transmit packet, then the interrupt source remains inactive until the first bank becomes free again to transmit at UDPHS_EPTSTAx/TXRDY hardware clear.

- **RX_SETUP: Received SETUP Interrupt Enabled**

0 = Received SETUP is masked.

1 = Received SETUP is enabled.

- **STALL_SNT: Stall Sent Interrupt Enabled**

0 = Stall Sent Interrupt is masked.

1 = Stall Sent Interrupt is enabled.

- **NAK_IN: NAKIN Interrupt Enabled**

0 = NAKIN Interrupt is masked.

1 = NAKIN Interrupt is enabled.

- **NAK_OUT: NAKOUT Interrupt Enabled**

0 = NAKOUT Interrupt is masked.

1 = NAKOUT Interrupt is enabled.

- **BUSY_BANK: Busy Bank Interrupt Enabled**

0 = BUSY_BANK Interrupt is masked.

1 = BUSY_BANK Interrupt is enabled.

For OUT endpoints: an interrupt is sent when all banks are busy.

For IN endpoints: an interrupt is sent when all banks are free.

- **SHRT_PCKT: Short Packet Interrupt Enabled**
  For OUT endpoints: send an Interrupt when a Short Packet has been received.

  0 = Short Packet Interrupt is masked.

  1 = Short Packet Interrupt is enabled.

  For IN endpoints: a Short Packet transmission is guaranteed upon end of the DMA Transfer, thus signaling a BULK or INTERRUPT end of transfer, but only if the UDPHS_DMACONTROLx register END_B_EN and UDPHS_EPTCTLx register AUTO_VALID bits are also set.

32.7.14 UDPHS Endpoint Control Register (Isochronous Endpoint)

Name: UDPHS_EPTCTLx [x=0..6] (ISOENDPT)

Address: 0xF803C10C [0], 0xF803C12C [1], 0xF803C14C [2], 0xF803C16C [3], 0xF803C18C [4], 0xF803C1AC [5], 0xF803C1CC [6]

Access: Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
–	ERR_FLUSH	ERR_CRC_NTR	ERR_FL_ISO	TXRDY_TRER	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
MDATA_RX	DATAX_RX	–	–	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

This register view is relevant only if EPT_TYPE=0x1 in “UDPHS Endpoint Configuration Register” on page 566

- **EPT_ENABL: Endpoint Enable**

0 = If cleared, the endpoint is disabled according to the device configuration. Endpoint 0 should always be enabled after a hardware or UDPHS bus reset and participate in the device configuration.

1 = If set, the endpoint is enabled according to the device configuration.

- **AUTO_VALID: Packet Auto-Valid Enabled**

Set this bit to automatically validate the current packet and switch to the next bank for both IN and OUT endpoints.

For IN Transfer:

If this bit is set, then the UDPHS_EPTSTAx register TXRDY_TRER bit is set automatically when the current bank is full and at the end of DMA buffer if the UDPHS_DMACONTROLx register END_B_EN bit is set.

The user may still set the UDPHS_EPTSTAx register TXRDY_TRER bit if the current bank is not full, unless the user wants to send a Zero Length Packet by software.

For OUT Transfer:

If this bit is set, then the UDPHS_EPTSTAx register RXRDY_TXKL bit is automatically reset for the current bank when the last packet byte has been read from the bank FIFO or at the end of DMA buffer if the UDPHS_DMACONTROLx register END_B_EN bit is set. For example, to truncate a padded data packet when the actual data transfer size is reached.

The user may still clear the UDPHS_EPTSTAx register RXRDY_TXKL bit, for example, after completing a DMA buffer by software if UDPHS_DMACONTROLx register END_B_EN bit was disabled or in order to cancel the read of the remaining data bank(s).

- **INTDIS_DMA: Interrupt Disables DMA**

If set, when an enabled endpoint-originated interrupt is triggered, the DMA request is disabled regardless of the UDPHS_IEN register EPT_x bit for this endpoint. Then, the firmware will have to clear or disable the interrupt source or clear this bit if transfer completion is needed.

If the exception raised is associated with the new system bank packet, then the previous DMA packet transfer is normally completed, but the new DMA packet transfer is not started (not requested).

If the exception raised is not associated to a new system bank packet (ex:ERR_FL_ISO), then the request cancellation may happen at any time and may immediately stop the current DMA transfer.

This may be used, for example, to identify or prevent an erroneous packet to be transferred into a buffer or to complete a DMA buffer by software after reception of a short packet, or to perform buffer truncation on ERR_FL_ISO interrupt for adaptive rate.

- **DATAX_RX: DATAx Interrupt Enabled (Only for High Bandwidth Isochronous OUT endpoints)**

0 = No effect.

1 = Send an interrupt when a DATA2, DATA1 or DATA0 packet has been received meaning the whole microframe data payload has been received.

- **MDATA_RX: MDATA Interrupt Enabled (Only for High Bandwidth Isochronous OUT endpoints)**

0 = No effect.

1 = Send an interrupt when an MDATA packet has been received and so at least one packet of the microframe data payload has been received.

- **ERR_OVFLW: Overflow Error Interrupt Enabled**

0 = Overflow Error Interrupt is masked.

1 = Overflow Error Interrupt is enabled.

- **RXRDY_TXKL: Received OUT Data Interrupt Enabled**

0 = Received OUT Data Interrupt is masked.

1 = Received OUT Data Interrupt is enabled.

- **TX_COMPLT: Transmitted IN Data Complete Interrupt Enabled**

0 = Transmitted IN Data Complete Interrupt is masked.

1 = Transmitted IN Data Complete Interrupt is enabled.

- **TXRDY_TRER: TX Packet Ready/Transaction Error Interrupt Enabled**

0 = TX Packet Ready/Transaction Error Interrupt is masked.

1 = TX Packet Ready/Transaction Error Interrupt is enabled.

Caution: Interrupt source is active as long as the corresponding UDPHS_EPTSTAx register TXRDY_TRER flag remains low. If there are no more banks available for transmitting after the software has set UDPHS_EPTSTAx/TXRDY_TRER for the last transmit packet, then the interrupt source remains inactive until the first bank becomes free again to transmit at UDPHS_EPTSTAx/TXRDY_TRER hardware clear.

- **ERR_FL_ISO: Error Flow Interrupt Enabled**

0 = Error Flow Interrupt is masked.

1 = Error Flow Interrupt is enabled.

- **ERR_CRC_NTR: ISO CRC Error/Number of Transaction Error Interrupt Enabled**

0 = ISO CRC error/number of Transaction Error Interrupt is masked.

1 = ISO CRC error/number of Transaction Error Interrupt is enabled.

- **ERR_FLUSH: Bank Flush Error Interrupt Enabled**

0 = Bank Flush Error Interrupt is masked.

1 = Bank Flush Error Interrupt is enabled.

- **BUSY_BANK: Busy Bank Interrupt Enabled**

0 = BUSY_BANK Interrupt is masked.

1 = BUSY_BANK Interrupt is enabled.

For OUT endpoints: An interrupt is sent when all banks are busy.

For IN endpoints: An interrupt is sent when all banks are free.

- **SHRT_PCKT: Short Packet Interrupt Enabled**

For OUT endpoints: send an Interrupt when a Short Packet has been received.

0 = Short Packet Interrupt is masked.

1 = Short Packet Interrupt is enabled.

For IN endpoints: a Short Packet transmission is guaranteed upon end of the DMA Transfer, thus signaling an end of isochronous (micro-)frame data, but only if the UDPHS_DMACONTROLx register END_B_EN and UDPHS_EPTCTLx register AUTO_VALID bits are also set.

32.7.15 UDPHS Endpoint Set Status Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTSETSTAx [x=0..6]

Access: Write-only

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

This register view is relevant only if EPT_TYPE=0x0, 0x2 or 0x3 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)” on page 586

- **FRCESTALL: Stall Handshake Request Set**

0 = No effect.

1 = Set this bit to request a STALL answer to the host for the next handshake

Refer to chapters 8.4.5 (Handshake Packets) and 9.4.5 (Get Status) of the *Universal Serial Bus Specification, Rev 2.0* for more information on the STALL handshake.

- **RXRDY_TXKL: KILL Bank Set (for IN Endpoint)**

0 = No effect.

1 = Kill the last written bank.

- **TXRDY: TX Packet Ready Set**

0 = No effect.

1 = Set this bit after a packet has been written into the endpoint FIFO for IN data transfers

  – This flag is used to generate a Data IN transaction (device to host).
  – Device firmware checks that it can write a data payload in the FIFO, checking that TXRDY is cleared.
  – Transfer to the FIFO is done by writing in the “Buffer Address” register.
  – Once the data payload has been transferred to the FIFO, the firmware notifies the UDPHS device setting TXRDY to one.
  – UDPHS bus transactions can start.
  – TXCOMP is set once the data payload has been received by the host.
  – Data should be written into the endpoint FIFO only after this bit has been cleared.
  – Set this bit without writing data to the endpoint FIFO to send a Zero Length Packet.

32.7.16 UDPHS Endpoint Set Status Register (Isochronous Endpoint)

Name: UDPHS_EPTSETSTAx [x=0..6] (ISOENDPT)

Address: 0xF803C114 [0], 0xF803C134 [1], 0xF803C154 [2], 0xF803C174 [3], 0xF803C194 [4], 0xF803C1B4 [5], 0xF803C1D4 [6]

Access: Write-only

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

This register view is relevant only if EPT_TYPE=0x1 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Status Register (Isochronous Endpoint)” on page 589.

- **RXRDY_TXKL: KILL Bank Set (for IN Endpoint)**

0 = No effect.

1 = Kill the last written bank.

- **TXRDY_TRER: TX Packet Ready Set**

0 = No effect.

1 = Set this bit after a packet has been written into the endpoint FIFO for IN data transfers

  – This flag is used to generate a Data IN transaction (device to host).
  – Device firmware checks that it can write a data payload in the FIFO, checking that TXRDY_TRER is cleared.
  – Transfer to the FIFO is done by writing in the “Buffer Address” register.
  – Once the data payload has been transferred to the FIFO, the firmware notifies the UDPHS device setting TXRDY_TRER to one.
  – UDPHS bus transactions can start.
  – TXCOMP is set once the data payload has been sent.
  – Data should be written into the endpoint FIFO only after this bit has been cleared.
  – Set this bit without writing data to the endpoint FIFO to send a Zero Length Packet.

32.7.17 UDPHS Endpoint Clear Status Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCLRSTAx [x=0..6]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN	STALL_SNT	RX_SETUP	–	TX_COMPLT	RXRDY_TXKL	–
7	6	5	4	3	2	1	0
–	TOGGLESQ	FRCESTALL	–	–	–	–	–

This register view is relevant only if EPT_TYPE=0x0, 0x2 or 0x3 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)” on page 586.

- **FRCESTALL: Stall Handshake Request Clear**

0 = No effect.

1 = Clear the STALL request. The next packets from host will not be STALLed.

- **TOGGLESQ: Data Toggle Clear**

0 = No effect.

1 = Clear the PID data of the current bank

For OUT endpoints, the next received packet should be a DATA0.

For IN endpoints, the next packet will be sent with a DATA0 PID.

- **RXRDY_TXKL: Received OUT Data Clear**

0 = No effect.

1 = Clear the RXRDY_TXKL flag of UDPHS_EPTSTAx.

- **TX_COMPLT: Transmitted IN Data Complete Clear**

0 = No effect.

1 = Clear the TX_COMPLT flag of UDPHS_EPTSTAx.

- **RX_SETUP: Received SETUP Clear**

0 = No effect.

1 = Clear the RX_SETUP flags of UDPHS_EPTSTAx.

- **STALL_SNT: Stall Sent Clear**

0 = No effect.

1 = Clear the STALL_SNT flags of UDPHS_EPTSTAx.

- **NAK_IN: NAKIN Clear**

0 = No effect.

1 = Clear the NAK_IN flags of UDPHS_EPTSTAx.

- **NAK_OUT: NAKOUT Clear**

0 = No effect.

1 = Clear the NAK_OUT flag of UDPHS_EPTSTAx.

32.7.18 UDPHS Endpoint Clear Status Register (Isochronous Endpoint)

Name: UDPHS_EPTCLRSTAx [x=0..6] (ISOENDPT)

Address: 0xF803C118 [0], 0xF803C138 [1], 0xF803C158 [2], 0xF803C178 [3], 0xF803C198 [4], 0xF803C1B8 [5], 0xF803C1D8 [6]

Access: Write-only

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

This register view is relevant only if EPT_TYPE=0x1 in “UDPHS Endpoint Configuration Register” on page 566

For additional Information, see “UDPHS Endpoint Status Register (Isochronous Endpoint)” on page 589.

- **TOGGLESQ: Data Toggle Clear**

0 = No effect.

1 = Clear the PID data of the current bank

For OUT endpoints, the next received packet should be a DATA0.

For IN endpoints, the next packet will be sent with a DATA0 PID.

- **RXRDY_TXKL: Received OUT Data Clear**

0 = No effect.

1 = Clear the RXRDY_TXKL flag of UDPHS_EPTSTAx.

- **TX_COMPLT: Transmitted IN Data Complete Clear**

0 = No effect.

1 = Clear the TX_COMPLT flag of UDPHS_EPTSTAx.

- **ERR_FL_ISO: Error Flow Clear**

0 = No effect.

1 = Clear the ERR_FL_ISO flags of UDPHS_EPTSTAx.

- **ERR_CRC_NTR: Number of Transaction Error Clear**

0 = No effect.

1 = Clear the ERR_CRC_NTR flags of UDPHS_EPTSTAx.

- **ERR_FLUSH: Bank Flush Error Clear**

0 = No effect.

1 = Clear the ERR_FLUSH flags of UDPHS_EPTSTAx.

32.7.19 UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTSTAx [x=0..6]

Access: Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT	BYTE_COUNT						
23	22	21	20	19	18	17	16
BYTE_COUNT				BUSY_BANK_STA		CURBK_CTLDIR	
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN	STALL_SNT	RX_SETUP	TXRDY	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW
7	6	5	4	3	2	1	0
TOGGLESQ_STA		FRCESTALL	–	–	–	–	–

This register view is relevant only if EPT_TYPE=0x0, 0x2 or 0x3 in "UDPHS Endpoint Configuration Register" on page 566

- **FRCESTALL: Stall Handshake Request**

0 = No effect.

1 = If set a STALL answer will be done to the host for the next handshake.

This bit is reset by hardware upon received SETUP.

- **TOGGLESQ_STA: Toggle Sequencing**

Toggle Sequencing:

– **IN Endpoint:** It indicates the PID Data Toggle that will be used for the next packet sent. This is not relative to the current bank.

– CONTROL and OUT endpoint:

These bits are set by hardware to indicate the PID data of the current bank:

Value	Name	Description
0	DATA0	DATA0
1	DATA1	DATA1
2	DATA2	Reserved for High Bandwidth Isochronous Endpoint
3	MDATA	Reserved for High Bandwidth Isochronous Endpoint

Notes:
1. In OUT transfer, the Toggle information is meaningful only when the current bank is busy (Received OUT Data = 1).
2. These bits are updated for OUT transfer:
   - A new data has been written into the current bank.
   - The user has just cleared the Received OUT Data bit to switch to the next bank.
3. This field is reset to DATA1 by the UDPHS_EPTCLRSTAx register TOGGLESQ bit, and by UDPHS_EPTCTLDISx (disable endpoint).

- **ERR_OVFLW: Overflow Error**

This bit is set by hardware when a new too-long packet is received.

Example: If the user programs an endpoint 64 bytes wide and the host sends 128 bytes in an OUT transfer, then the Overflow Error bit is set.

This bit is updated at the same time as the BYTE_COUNT field.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

- **RXRDY_TXKL: Received OUT Data/KILL Bank**
  - **Received OUT Data** (for OUT endpoint or Control endpoint):

This bit is set by hardware after a new packet has been stored in the endpoint FIFO.

This bit is cleared by the device firmware after reading the OUT data from the endpoint.

For multi-bank endpoints, this bit may remain active even when cleared by the device firmware, this if an other packet has been received meanwhile.

Hardware assertion of this bit may generate an interrupt if enabled by the UDPHS_EPTCTLx register RXRDY_TXKL bit.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

  - **KILL Bank** (for IN endpoint):
  - The bank is really cleared or the bank is sent, BUSY_BANK_STA is decremented.
  - The bank is not cleared but sent on the IN transfer, TX_COMPLT
  - The bank is not cleared because it was empty. The user should wait that this bit is cleared before trying to clear another packet.

Note: “Kill a packet” may be refused if at the same time, an IN token is coming and the current packet is sent on the UDPHS line. In this case, the TX_COMPLT bit is set. Take notice however, that if at least two banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. In fact, in that case, the current bank is sent (IN transfer) and the last bank is killed.

- **TX_COMPLT: Transmitted IN Data Complete**

This bit is set by hardware after an IN packet has been accepted (ACK’ed) by the host.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint), and by UDPHS_EPTCTLDISx (disable endpoint).

- **TXRDY: TX Packet Ready**

This bit is cleared by hardware after the host has acknowledged the packet.

For Multi-bank endpoints, this bit may remain clear even after software is set if another bank is available to transmit.

Hardware clear of this bit may generate an interrupt if enabled by the UDPHS_EPTCTLx register TXRDY bit.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint), and by UDPHS_EPTCTLDISx (disable endpoint).

- **RX_SETUP: Received SETUP**
  - (for Control endpoint only)

This bit is set by hardware when a valid SETUP packet has been received from the host.

It is cleared by the device firmware after reading the SETUP data from the endpoint FIFO.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint), and by UDPHS_EPTCTLDISx (disable endpoint).

- **STALL_SNT: Stall Sent**
  - (for Control, Bulk and Interrupt endpoints)

This bit is set by hardware after a STALL handshake has been sent as requested by the UDPHS_EPTSTAx register FRCESTALL bit.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

- **NAK_IN: NAK IN**

This bit is set by hardware when a NAK handshake has been sent in response to an IN request from the Host.

This bit is cleared by software.

- **NAK_OUT: NAK OUT**

This bit is set by hardware when a NAK handshake has been sent in response to an OUT or PING request from the Host.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by EPT_CTL_DISx (disable endpoint).

- **CURBK_CTLDIR: Current Bank/Control Direction**
  - **Current Bank** (not relevant for Control endpoint):

These bits are set by hardware to indicate the number of the current bank.

Value	Name	Description
0	BANK0	Bank 0 (or single bank)
1	BANK1	Bank 1
2	BANK2	Bank 2

Note: The current bank is updated each time the user:
- Sets the TX Packet Ready bit to prepare the next IN transfer and to switch to the next bank.
- Clears the received OUT data bit to access the next bank.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

  - **Control Direction** (for Control endpoint only):

0 = A Control Write is requested by the Host.

1 = A Control Read is requested by the Host.

Notes:
1. This bit corresponds with the 7th bit of the bmRequestType (Byte 0 of the Setup Data).
2. This bit is updated after receiving new setup data.

- **BUSY_BANK_STA: Busy Bank Number**

These bits are set by hardware to indicate the number of busy banks.

IN endpoint: It indicates the number of busy banks filled by the user, ready for IN transfer.

OUT endpoint: It indicates the number of busy banks filled by OUT transaction from the Host.

Value	Name	Description
0	1BUSYBANK	1 busy bank
1	2BUSYBANKS	2 busy banks
2	3BUSYBANKS	3 busy banks

- **BYTE_COUNT: UDPHS Byte Count**

Byte count of a received data packet.

This field is incremented after each write into the endpoint (to prepare an IN transfer).

This field is decremented after each reading into the endpoint (OUT transfer).

This field is also updated at RXRDY_TXKL flag clear with the next bank.

This field is also updated at TXRDY flag set with the next bank.

This field is reset by EPT_x of UDPHS_EPTRST register.

- **SHRT_PCKT: Short Packet**

An OUT Short Packet is detected when the receive byte count is less than the configured UDPHS_EPTCFGx register EPT_Size.

This bit is updated at the same time as the BYTE_COUNT field.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

32.7.20 UDPHS Endpoint Status Register (Isochronous Endpoint)

Name: UDPHS_EPTSTAx [x=0..6] (ISOENDPT)

Address: 0xF803C11C [0], 0xF803C13C [1], 0xF803C15C [2], 0xF803C17C [3], 0xF803C19C [4], 0xF803C1BC [5], 0xF803C1DC [6]

Access: Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT	BYTE_COUNT						

23	22	21	20	19	18	17	16
BYTE_COUNT				BUSY_BANK_STA		CURBK	

15	14	13	12	11	10	9	8
–	ERR_FLUSH	ERR_CRC_NTR	ERR_FL_ISO	TXRDY_TRER	TX_COMPLT	RXRDY_TXKL	ERR_OVFLW

7	6	5	4	3	2	1	0
TOGGLESQ_STA		–	–	–	–	–	–

This register view is relevant only if EPT_TYPE=0x1 in “UDPHS Endpoint Configuration Register” on page 566

- **TOGGLESQ_STA: Toggle Sequencing**

Toggle Sequencing:

– **IN Endpoint:** It indicates the PID Data Toggle that will be used for the next packet sent. This is not relative to the current bank.

– OUT endpoint:

These bits are set by hardware to indicate the PID data of the current bank:

Value	Name	Description
0	DATA0	DATA0
1	DATA1	DATA1
2	DATA2	Data2 (only for High Bandwidth Isochronous Endpoint)
3	MDATA	MData (only for High Bandwidth Isochronous Endpoint)

Notes:
1. In OUT transfer, the Toggle information is meaningful only when the current bank is busy (Received OUT Data = 1).
2. These bits are updated for OUT transfer:
   - A new data has been written into the current bank.
   - The user has just cleared the Received OUT Data bit to switch to the next bank.
3. For High Bandwidth Isochronous Out endpoint, it is recommended to check the UDPHS_EPTSTAx/TXRDY_TRER bit to know if the toggle sequencing is correct or not.
4. This field is reset to DATA1 by the UDPHS_EPTCLRSTAx register TOGGLESQ bit, and by UDPHS_EPTCTLDISx (disable endpoint).

- **ERR_OVFLW: Overflow Error**

This bit is set by hardware when a new too-long packet is received.

Example: If the user programs an endpoint 64 bytes wide and the host sends 128 bytes in an OUT transfer, then the Overflow Error bit is set.

This bit is updated at the same time as the BYTE_COUNT field.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

- **RXRDY_TXKL: Received OUT Data/KILL Bank**
  - **Received OUT Data** (for OUT endpoint or Control endpoint):

This bit is set by hardware after a new packet has been stored in the endpoint FIFO.

This bit is cleared by the device firmware after reading the OUT data from the endpoint.

For multi-bank endpoints, this bit may remain active even when cleared by the device firmware, this if an other packet has been received meanwhile.

Hardware assertion of this bit may generate an interrupt if enabled by the UDPHS_EPTCTLx register RXRDY_TXKL bit.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

  - **KILL Bank** (for IN endpoint):
  - The bank is really cleared or the bank is sent, BUSY_BANK_STA is decremented.
  - The bank is not cleared but sent on the IN transfer, TX_COMPLT
  - The bank is not cleared because it was empty. The user should wait that this bit is cleared before trying to clear another packet.

Note: "Kill a packet" may be refused if at the same time, an IN token is coming and the current packet is sent on the UDPHS line. In this case, the TX_COMPLT bit is set. Take notice however, that if at least two banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. In fact, in that case, the current bank is sent (IN transfer) and the last bank is killed.

- **TX_COMPLT: Transmitted IN Data Complete**

This bit is set by hardware after an IN packet has been sent.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint), and by UDPHS_EPTCTLDISx (disable endpoint).

- **TXRDY_TRER: TX Packet Ready/Transaction Error**
  - **TX Packet Ready**:

This bit is cleared by hardware, as soon as the packet has been sent.

For Multi-bank endpoints, this bit may remain clear even after software is set if another bank is available to transmit.

Hardware clear of this bit may generate an interrupt if enabled by the UDPHS_EPTCTLx register TXRDY_TRER bit.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint), and by UDPHS_EPTCTLDISx (disable endpoint).

  - **Transaction Error** (for high bandwidth isochronous OUT endpoints) (Read-Only):

This bit is set by hardware when a transaction error occurs inside one microframe.

If one toggle sequencing problem occurs among the n-transactions (n = 1, 2 or 3) inside a microframe, then this bit is still set as long as the current bank contains one "bad" n-transaction. (see "CURBK: Current Bank" on page 591) As soon as the current bank is relative to a new "good" n-transactions, then this bit is reset.

Notes:
1. A transaction error occurs when the toggle sequencing does not respect the *Universal Serial Bus Specification, Rev 2.0* (5.9.2 High Bandwidth Isochronous endpoints) (Bad PID, missing data....)
2. When a transaction error occurs, the user may empty all the "bad" transactions by clearing the Received OUT Data flag (RXRDY_TXKL).

If this bit is reset, then the user should consider that a new n-transaction is coming.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint), and by UDPHS_EPTCTLDISx (disable endpoint).

- **ERR_FL_ISO: Error Flow**

This bit is set by hardware when a transaction error occurs.

– Isochronous IN transaction is missed, the micro has no time to fill the endpoint (underflow).

– Isochronous OUT data is dropped because the bank is busy (overflow).

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

- **ERR_CRC_NTR: CRC ISO Error/Number of Transaction Error**

– **CRC ISO Error** (for Isochronous OUT endpoints) (Read-only):

This bit is set by hardware if the last received data is corrupted (CRC error on data).

This bit is updated by hardware when new data is received (Received OUT Data bit).

– **Number of Transaction Error** (for High Bandwidth Isochronous IN endpoints):

This bit is set at the end of a microframe in which at least one data bank has been transmitted, if less than the number of transactions per micro-frame banks (UDPHS_EPTCFGx register NB_TRANS) have been validated for transmission inside this microframe.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

- **ERR_FLUSH: Bank Flush Error**

– (for High Bandwidth Isochronous IN endpoints)

This bit is set when flushing unsent banks at the end of a microframe.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by EPT_CTL_DISx (disable endpoint).

- **CURBK: Current Bank**

– **Current Bank**:

These bits are set by hardware to indicate the number of the current bank.

Value	Name	Description
0	BANK0	Bank 0 (or single bank)
1	BANK1	Bank 1
2	BANK2	Bank 2

Note: The current bank is updated each time the user:
- Sets the TX Packet Ready bit to prepare the next IN transfer and to switch to the next bank.
- Clears the received OUT data bit to access the next bank.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

- **BUSY_BANK_STA: Busy Bank Number**

These bits are set by hardware to indicate the number of busy banks.

IN endpoint: It indicates the number of busy banks filled by the user, ready for IN transfer.

OUT endpoint: It indicates the number of busy banks filled by OUT transaction from the Host.

Value	Name	Description
0	1BUSYBANK	1 busy bank
1	2BUSYBANKS	2 busy banks
2	3BUSYBANKS	3 busy banks

- **BYTE_COUNT: UDPHS Byte Count**

Byte count of a received data packet.

This field is incremented after each write into the endpoint (to prepare an IN transfer).

This field is decremented after each reading into the endpoint (OUT transfer).

This field is also updated at RXRDY_TXKL flag clear with the next bank.

This field is also updated at TXRDY_TRER flag set with the next bank.

This field is reset by EPT_x of UDPHS_EPTRST register.

- **SHRT_PCKT: Short Packet**

An OUT Short Packet is detected when the receive byte count is less than the configured UDPHS_EPTCFGx register EPT_Size.

This bit is updated at the same time as the BYTE_COUNT field.

This bit is reset by UDPHS_EPTRST register EPT_x (reset endpoint) and by UDPHS_EPTCTLDISx (disable endpoint).

32.7.21 UDPHS DMA Channel Transfer Descriptor

The DMA channel transfer descriptor is loaded from the memory.

Be careful with the alignment of this buffer.

The structure of the DMA channel transfer descriptor is defined by three parameters as described below:

Offset 0:

The address must be aligned: 0xXXXX0

Next Descriptor Address Register: UDPHS_DMANXTDSCx

Offset 4:

The address must be aligned: 0xXXXX4

DMA Channelx Address Register: UDPHS_DMAADDRESSx

Offset 8:

The address must be aligned: 0xXXXX8

DMA Channelx Control Register: UDPHS_DMACONTROLx

To use the DMA channel transfer descriptor, fill the structures with the correct value (as described in the following pages).

Then write directly in UDPHS_DMANXTDSCx the address of the descriptor to be used first.

Then write 1 in the LDNXT_DSC bit of UDPHS_DMACONTROLx (load next channel transfer descriptor). The descriptor is automatically loaded upon Endpointx request for packet transfer.

32.7.22 UDPHS DMA Next Descriptor Address Register

Name: UDPHS_DMANXTDSCx [x = 0..5]

Address: 0xF803C300 [0], 0xF803C310 [1], 0xF803C320 [2], 0xF803C330 [3], 0xF803C340 [4], 0xF803C350 [5]

Access: Read-write

31	30	29	28	27	26	25	24
NXT_DSC_ADD							

23	22	21	20	19	18	17	16
NXT_DSC_ADD							

15	14	13	12	11	10	9	8
NXT_DSC_ADD							

7	6	5	4	3	2	1	0
NXT_DSC_ADD							

Note: Channel 0 is not used.

- **NXT_DSC_ADD: Next Descriptor Address**

This field points to the next channel descriptor to be processed. This channel descriptor must be aligned, so bits 0 to 3 of the address must be equal to zero.

32.7.23 UDPHS DMA Channel Address Register

Name: UDPHS_DMAADDRESSx [x = 0..5]

Address: 0xF803C304 [0], 0xF803C314 [1], 0xF803C324 [2], 0xF803C334 [3], 0xF803C344 [4], 0xF803C354 [5]

Access: Read-write

31	30	29	28	27	26	25	24
BUFF_ADD							

23	22	21	20	19	18	17	16
BUFF_ADD							

15	14	13	12	11	10	9	8
BUFF_ADD							

7	6	5	4	3	2	1	0
BUFF_ADD							

Note: Channel 0 is not used.

- **BUFF_ADD: Buffer Address**

This field determines the AHB bus starting address of a DMA channel transfer.

Channel start and end addresses may be aligned on any byte boundary.

The firmware may write this field only when the UDPHS_DMASTATUS register CHANN_ENB bit is clear.

This field is updated at the end of the address phase of the current access to the AHB bus. It is incrementing of the access byte width. The access width is 4 bytes (or less) at packet start or end, if the start or end address is not aligned on a word boundary.

The packet start address is either the channel start address or the next channel address to be accessed in the channel buffer.

The packet end address is either the channel end address or the latest channel address accessed in the channel buffer.

The channel start address is written by software or loaded from the descriptor, whereas the channel end address is either determined by the end of buffer or the UDPHS device, USB end of transfer if the UDPHS_DMACONTROLx register END_TR_EN bit is set.

32.7.24 UDPHS DMA Channel Control Register

Name: UDPHS_DMACONTROLx [x = 0..5]

Address: 0xF803C308 [0], 0xF803C318 [1], 0xF803C328 [2], 0xF803C338 [3], 0xF803C348 [4], 0xF803C358 [5]

Access: Read-write

31	30	29	28	27	26	25	24
BUFF_LENGTH							
23	22	21	20	19	18	17	16
BUFF_LENGTH							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
BURST_LCK	DESC_LD_IT	END_BUFFIT	END_TR_IT	END_B_EN	END_TR_EN	LDNXT_DSC	CHANN_ENB

Note: Channel 0 is not used.

- **CHANN_ENB: (Channel Enable Command)**

0 = DMA channel is disabled at and no transfer will occur upon request. This bit is also cleared by hardware when the channel source bus is disabled at end of buffer.

If the UDPHS_DMACONTROL register LDNXT_DSC bit has been cleared by descriptor loading, the firmware will have to set the corresponding CHANN_ENB bit to start the described transfer, if needed.

If the UDPHS_DMACONTROL register LDNXT_DSC bit is cleared, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both UDPHS_DMASTATUS register CHANN_ENB and CHANN_ACT flags read as 0.

If a channel request is currently serviced when this bit is cleared, the DMA FIFO buffer is drained until it is empty, then the UDPHS_DMASTATUS register CHANN_ENB bit is cleared.

If the LDNXT_DSC bit is set at or after this bit clearing, then the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded.

1 = UDPHS_DMASTATUS register CHANN_ENB bit will be set, thus enabling DMA channel data transfer. Then any pending request will start the transfer. This may be used to start or resume any requested transfer.

- **LDNXT_DSC: Load Next Channel Transfer Descriptor Enable (Command)**

0 = No channel register is loaded after the end of the channel transfer.

1 = The channel controller loads the next descriptor after the end of the current transfer, i.e. when the UDPHS_DMASTATUS/CHANN_ENB bit is reset.

If the UDPHS_DMA CONTROL/CHANN_ENB bit is cleared, the next descriptor is immediately loaded upon transfer request.

DMA Channel Control Command Summary

LDNXT_DSC	CHANN_ENB	Description
0	0	Stop now
0	1	Run and stop at end of buffer
1	0	Load next descriptor now
1	1	Run and link at end of buffer

- **END_TR_EN: End of Transfer Enable (Control)**

Used for OUT transfers only.

0 = USB end of transfer is ignored.

1 = UDPHS device can put an end to the current buffer transfer.

When set, a BULK or INTERRUPT short packet or the last packet of an ISOCHRONOUS (micro) frame (DATAX) will close the current buffer and the UDPHS_DMASTATUSx register END_TR_ST flag will be raised.

This is intended for UDPHS non-prenegotiated end of transfer (BULK or INTERRUPT) or ISOCHRONOUS microframe data buffer closure.

- **END_B_EN: End of Buffer Enable (Control)**

0 = DMA Buffer End has no impact on USB packet transfer.

1 = Endpoint can validate the packet (according to the values programmed in the UDPHS_EPTCTLx register AUTO_VALID and SHRT_PCKT fields) at DMA Buffer End, i.e. when the UDPHS_DMASTATUS register BUFF_COUNT reaches 0.

This is mainly for short packet IN validation initiated by the DMA reaching end of buffer, but could be used for OUT packet truncation (discarding of unwanted packet data) at the end of DMA buffer.

- **END_TR_IT: End of Transfer Interrupt Enable**

0 = UDPHS device initiated buffer transfer completion will not trigger any interrupt at UDPHS_STATUSx/END_TR_ST rising.

1 = An interrupt is sent after the buffer transfer is complete, if the UDPHS device has ended the buffer transfer.

Use when the receive size is unknown.

- **END_BUFFIT: End of Buffer Interrupt Enable**

0 = UDPHS_DMA_STATUSx/END_BF_ST rising will not trigger any interrupt.

1 = An interrupt is generated when the UDPHS_DMASTATUSx register BUFF_COUNT reaches zero.

- **DESC_LD_IT: Descriptor Loaded Interrupt Enable**

0 = UDPHS_DMASTATUSx/DESC_LDST rising will not trigger any interrupt.

1 = An interrupt is generated when a descriptor has been loaded from the bus.

- **BURST_LCK: Burst Lock Enable**

0 = The DMA never locks bus access.

1 = USB packets AHB data bursts are locked for maximum optimization of the bus bandwidth usage and maximization of fly-by AHB burst duration.

- **BUFF_LENGTH: Buffer Byte Length (Write-only)**

This field determines the number of bytes to be transferred until end of buffer. The maximum channel transfer size (64 Kbytes) is reached when this field is 0 (default value). If the transfer size is unknown, this field should be set to 0, but the transfer end may occur earlier under UDPHS device control.

When this field is written, The UDPHS_DMASTATUSx register BUFF_COUNT field is updated with the write value.

Notes:
1. Bits [31:2] are only writable when issuing a channel Control Command other than “Stop Now”.
2. For reliability it is highly recommended to wait for both UDPHS_DMASTATUSx register CHAN_ACT and CHAN_ENB flags are at 0, thus ensuring the channel has been stopped before issuing a command other than “Stop Now”.

32.7.25 UDPHS DMA Channel Status Register

Name: UDPHS_DMASTATUSx [x = 0..5]

Address: 0xF803C30C [0], 0xF803C31C [1], 0xF803C32C [2], 0xF803C33C [3], 0xF803C34C [4], 0xF803C35C [5]

Access: Read-write

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

Note: Channel 0 is not used.

- **CHANN_ENB: Channel Enable Status**

0 = If cleared, the DMA channel no longer transfers data, and may load the next descriptor if the UDPHS_DMACONTROLx register LDNXT_DSC bit is set.

When any transfer is ended either due to an elapsed byte count or a UDPHS device initiated transfer end, this bit is automatically reset.

1 = If set, the DMA channel is currently enabled and transfers data upon request.

This bit is normally set or cleared by writing into the UDPHS_DMACONTROLx register CHANN_ENB bit field either by software or descriptor loading.

If a channel request is currently serviced when the UDPHS_DMACONTROLx register CHANN_ENB bit is cleared, the DMA FIFO buffer is drained until it is empty, then this status bit is cleared.

- **CHANN_ACT: Channel Active Status**

0 = The DMA channel is no longer trying to source the packet data.

When a packet transfer is ended this bit is automatically reset.

1 = The DMA channel is currently trying to source packet data, i.e. selected as the highest-priority requesting channel.

When a packet transfer cannot be completed due to an END_BF_ST, this flag stays set during the next channel descriptor load (if any) and potentially until UDPHS packet transfer completion, if allowed by the new descriptor.

- **END_TR_ST: End of Channel Transfer Status**

0 = Cleared automatically when read by software.

1 = Set by hardware when the last packet transfer is complete, if the UDPHS device has ended the transfer.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **END_BF_ST: End of Channel Buffer Status**

0 = Cleared automatically when read by software.

1 = Set by hardware when the BUFF_COUNT downcount reach zero.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **DESC_LDST: Descriptor Loaded Status**

0 = Cleared automatically when read by software.

1 = Set by hardware when a descriptor has been loaded from the system bus.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **BUFF_COUNT: Buffer Byte Count**

This field determines the current number of bytes still to be transferred for this buffer.

This field is decremented from the AHB source bus access byte width at the end of this bus address phase.

The access byte width is 4 by default, or less, at DMA start or end, if the start or end address is not aligned on a word boundary.

At the end of buffer, the DMA accesses the UDPHS device only for the number of bytes needed to complete it.

This field value is reliable (stable) only if the channel has been stopped or frozen (UDPHS_EPTCTLx register NT_DIS_DMA bit is used to disable the channel request) and the channel is no longer active CHANN_ACT flag is 0.

Note: For OUT endpoints, if the receive buffer byte length (BUFF_LENGTH) has been defaulted to zero because the USB transfer length is unknown, the actual buffer byte length received will be 0x10000-BUFF_COUNT.

33. USB Host High Speed Port (UHPHS)

33.1 Description

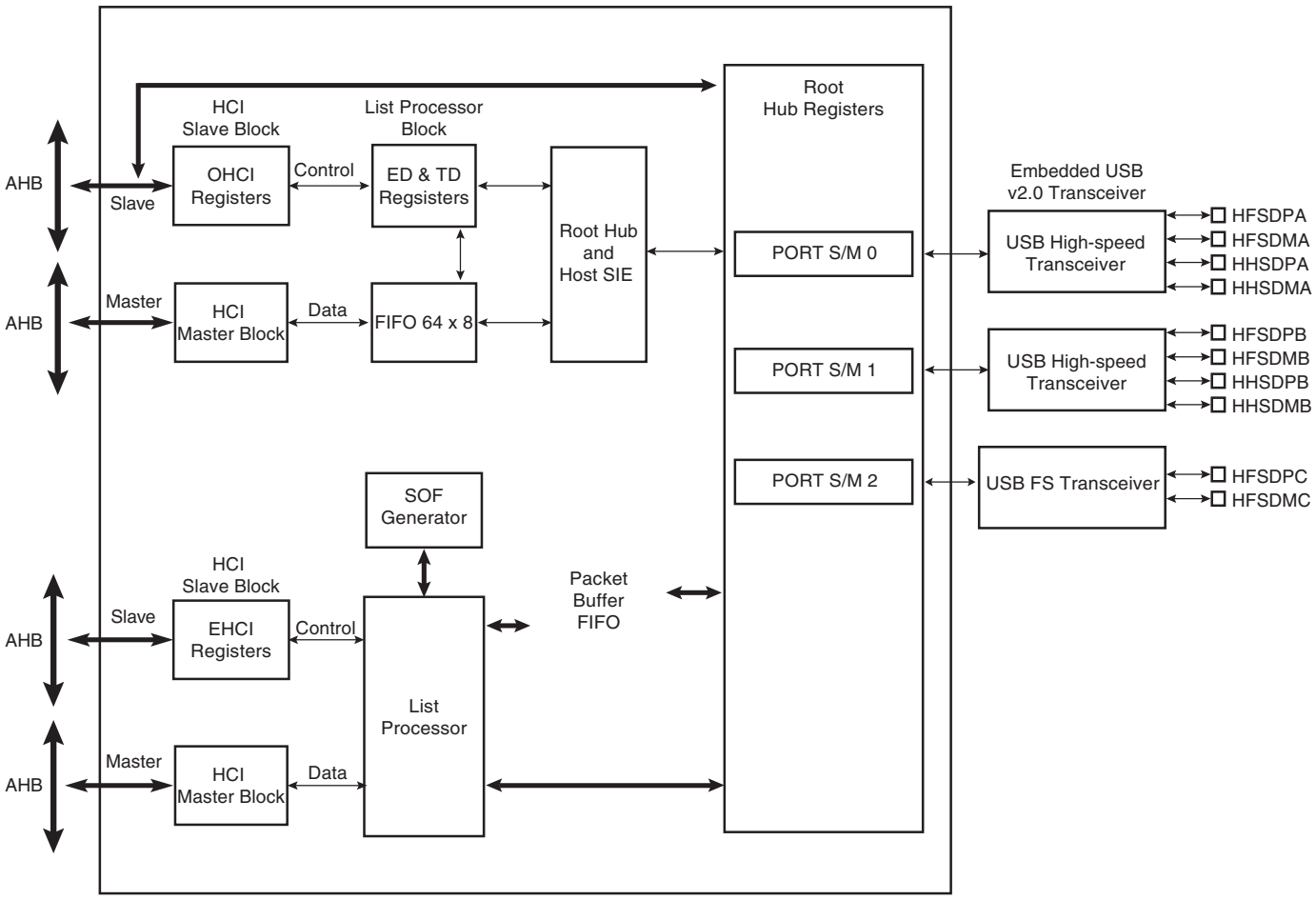
The USB Host High Speed Port (UHPHS) interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as Enhanced HCI protocol (Enhanced Host Controller Interface).

33.2 Embedded Characteristics

- Compliant with Enhanced HCI Rev 1.0 Specification
  - Compliant with USB V2.0 High-speed
  - Supports High-speed 480 Mbps
- Compliant with OpenHCI 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 USB devices
- Root Hub Integrated with 2 Downstream USB HS Ports and 1 FS Port
- Embedded USB Transceivers
- Supports Power Management
- 2 Hosts (A and B) High Speed (EHCI), Port A shared with UDPHS
- 1 Host (C) Full Speed only (OHCI)

33.3 Block Diagram

Figure 33-1. Block Diagram



Access to the USB host operational registers is achieved through the AHB bus slave interface. The Open HCI host controller and Enhanced HCI host controller initialize master DMA transfers through the AHB bus master interface as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory
- Access to the HC communication area
- Write status and retire transfer descriptor

Memory access errors (abort, misalignment) lead to an “Unrecoverable Error” indicated by the corresponding flag in the host controller operational registers.

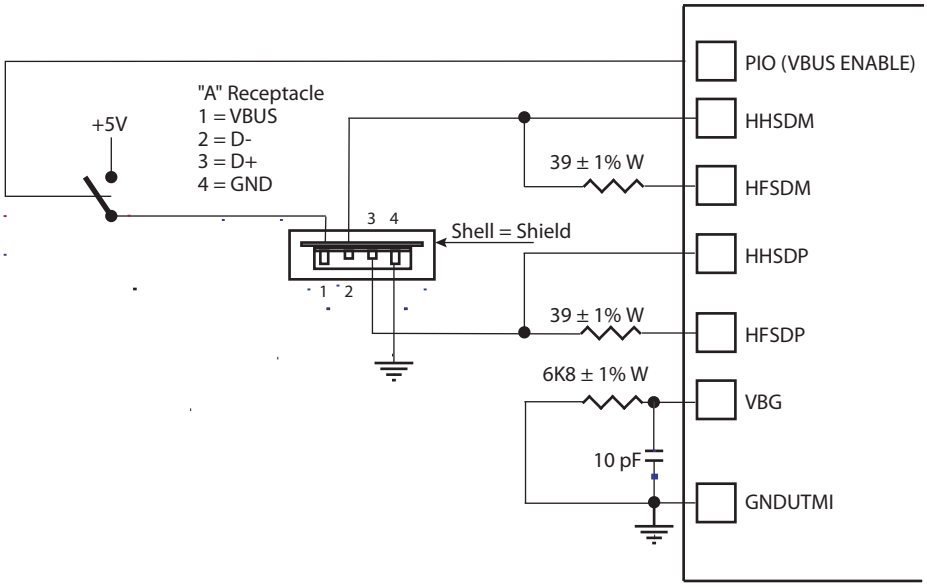
The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub’s operational registers. Device connection is automatically detected by the USB host port logic.

USB physical transceivers are integrated in the product and driven by the root hub’s ports.

Over current protection on ports can be activated by the USB host controller. Atmel’s standard product does not dedicate pads to external over current protection.

33.4 Typical Connection

Figure 33-2. Board Schematic to Interface UHP High-speed Host Controller



33.5 Product Dependencies

33.5.1 I/O Lines

HFSDPs, HFSDMs, HHSDPs and HHSDMs are not controlled by any PIO controllers. The embedded USB High Speed physical transceivers are controlled by the USB host controller.

One transceiver is shared with the USB High Speed Device (port A). The selection between Host Port A and USB Device is controlled by the UDPHS enable bit (EN_UDPHS) located in the UDPHS_CTRL control register.

In the case the port A is driven by the USB High Speed Device, the output signals are DFSDP, DFSDM, DHSDP and DHSDM. The transceiver is automatically selected for Device operation once the USB High Speed Device is enabled.

In the case the port A is driven by the USB High Speed Host, the output signals are HFSDPA, HFSDMA, HHSDPA and HHSDMA.

33.5.2 Power Management

The system embeds 2 transceivers.

The USB Host High Speed requires a 480 MHz clock for the embedded High-speed transceivers. This clock is provided by the UTMI PLL, it is UPLLCK.

In case power consumption is saved by stopping the UTMI PLL, high-speed operations are not possible. Nevertheless, OHCI Full-speed operations remain possible by selecting PLLACK as the input clock of OHCI.

The High-speed transceiver returns a 30 MHz clock to the USB Host controller.

The USB Host controller requires 48 MHz and 12 MHz clocks for OHCI full-speed operations. These clocks must be generated by a PLL with a correct accuracy of ± 0.25% thanks to USBDIV field.

Thus the USB Host peripheral receives three clocks from the Power Management Controller (PMC): the Peripheral Clock (MCK domain), the UHP48M and the UHP12M (built-in UHP48M divided by four) used by the OHCI to interface with the bus USB signals (Recovered 12 MHz domain) in Full-speed operations.

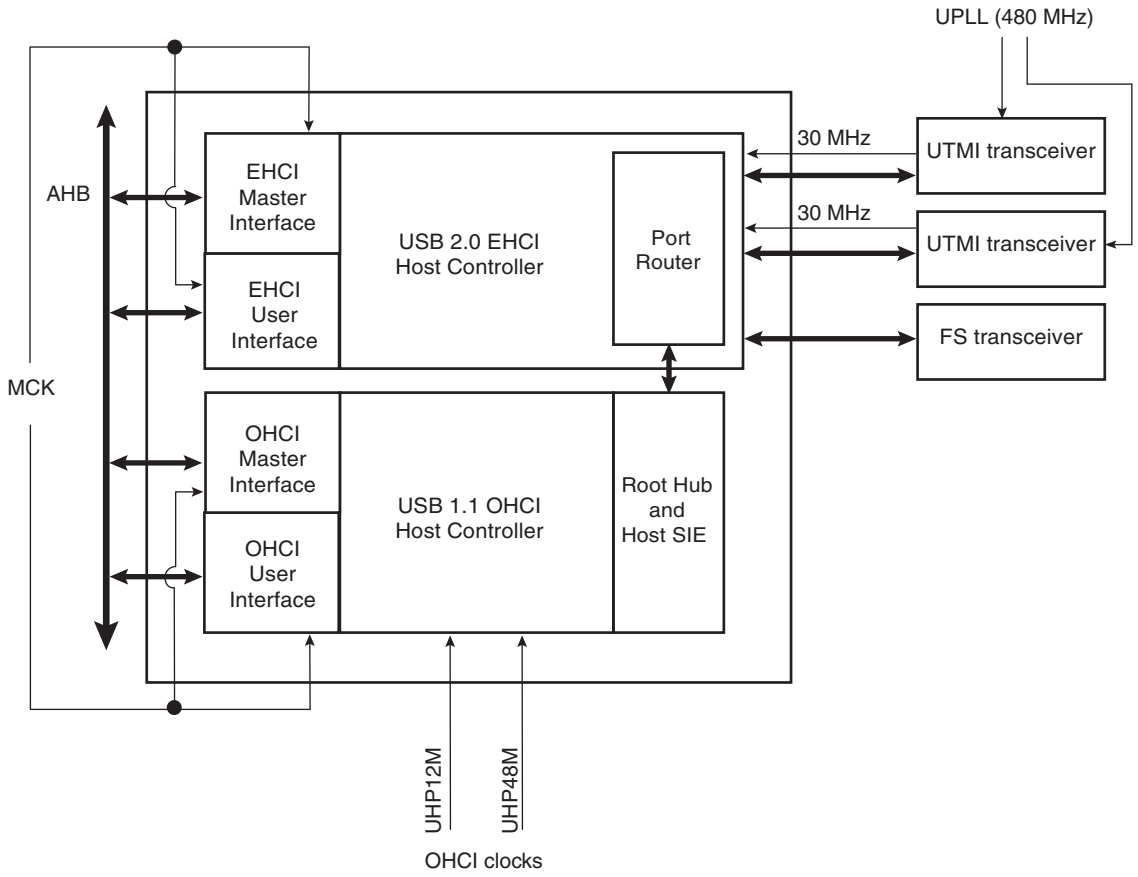
For High-speed operations, the user has to perform the following:

- Enable UHP peripheral clock, bit (1 << AT91C_ID_UHPHS) in PMC_PCER register.
- Write CKGR_PLLCOUNT field in PMC_UCKR register.
- Enable UPLL, bit AT91C_CKGR_UPLLEN in PMC_UCKR register.
- Wait until UTMI_PLL is locked. LOCKU bit in PMC_SR register
- Enable BIAS, bit AT91C_CKGR_BIASEN in PMC_UCKR register.
- Select UPLLCK as Input clock of OHCI part, USBS bit in PMC_USB register.
- Program the OHCI clocks (UHP48M and UHP12M) with USBDIV field in PMC_USB register. USBDIV must be 9 (division by 10) if UPLLCK is selected.
- Enable OHCI clocks, UHP bit in PMC_SCER register.

For OHCI Full-speed operations only, the user has to perform the following:

- Enable UHP peripheral clock, bit (1 << AT91C_ID_UHPHS) in PMC_PCER register.
- Select PLLACK as Input clock of OHCI part, USBS bit in PMC_USB register.
- Program the OHCI clocks (UHP48M and UHP12M) with USBDIV field in PMC_USB register. USBDIV value is to calculated regarding the PLLACK value and USB Full-speed accuracy.
- Enable the OHCI clocks, UHP bit in PMC_SCER register.

Figure 33-3. UHP Clock Trees



33.5.3 Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

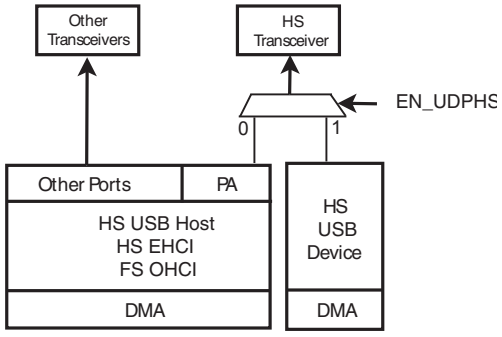
Handling USB host interrupts requires programming the AIC before configuring the UHP HS.

33.6 Functional Description

33.6.1 UTMI transceivers Sharing

The High Speed USB Host Port A is shared with the High Speed USB Device port and connected to the second UTMI transceiver. The selection between Host Port A and USB Device is controlled by the UDPHS enable bit (EN_UDPHS) located in the UDPHS_CTRL control register.

Figure 33-4. USB Selection



33.6.2 EHCI

The USB Host Port controller is fully compliant with the Enhanced HCI specification. The USB Host Port User Interface (registers description) can be found in the Enhanced HCI Rev 1.0 Specification available on http://www.intel.com/technology/usb/ehcispec.htm. The standard EHCI USB stack driver can be easily ported to Atmel's architecture in the same way all existing class drivers run, without hardware specialization.

33.6.3 OHCI

The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several Full-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 Open HCI 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's application. As an example, integrating an HID (Human Interface Device) class driver provides a plug & play feature for all USB keyboards and mouses.

34. High Speed MultiMedia Card Interface (HSMCI)

34.1 Description

The High Speed Multimedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

The HSMCI 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 HSMCI supports stream, block and multi block data read and write, and is compatible with the DMA Controller (DMAC), minimizing processor intervention for large buffer transfers.

The HSMCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 1 slot(s). Each slot may be used to interface with a High Speed MultiMedia Card bus (up to 30 Cards) or with an 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 High Speed 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 High Speed MultiMedia Card operations. The main differences between SD and High Speed MultiMedia Cards are the initialization process and the bus topology.

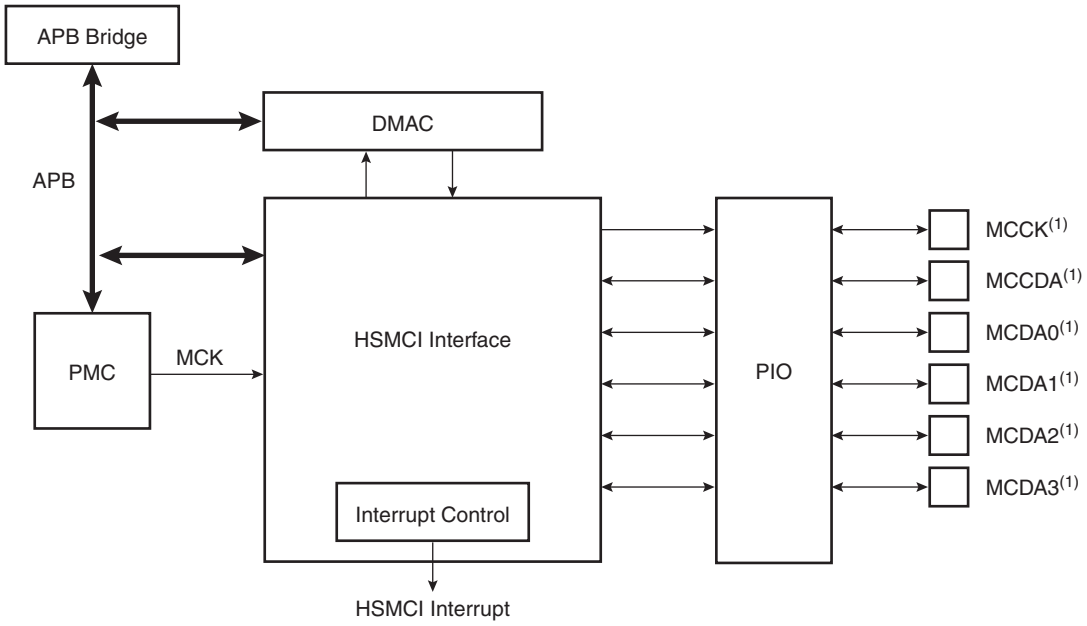
HSMCI fully supports CE-ATA Revision 1.1, built on the MMC System Specification v4.0. The module includes dedicated hardware to issue the command completion signal and capture the host command completion signal disable.

34.2 Embedded Characteristics

- Compatible with MultiMedia Card Specification Version 4.3
- Compatible with SD Memory Card Specification Version 2.0
- Compatible with SDIO Specification Version 2.0
- Compatible with CE-ATA Specification 1.1
- Cards Clock Rate Up to Master Clock Divided by 2
- Boot Operation Mode Support
- High Speed Mode Support
- Embedded Power Management to Slow Down Clock Rate When Not Used
- Supports 1 Multiplexed Slot(s)
  - Each Slot for either a High Speed MultiMedia Card Bus (Up to 30 Cards) or an SD Memory Card
- Support for Stream, Block and Multi-block Data Read and Write
- Supports Connection to DMA Controller (DMAC)
  - Minimizes Processor Intervention for Large Buffer Transfers
- Built in FIFO (from 16 to 256 bytes) with Large Memory Aperture Supporting Incremental Access
- Support for CE-ATA Completion Signal Disable Command
- Protection Against Unexpected Modification On-the-Fly of the Configuration Registers

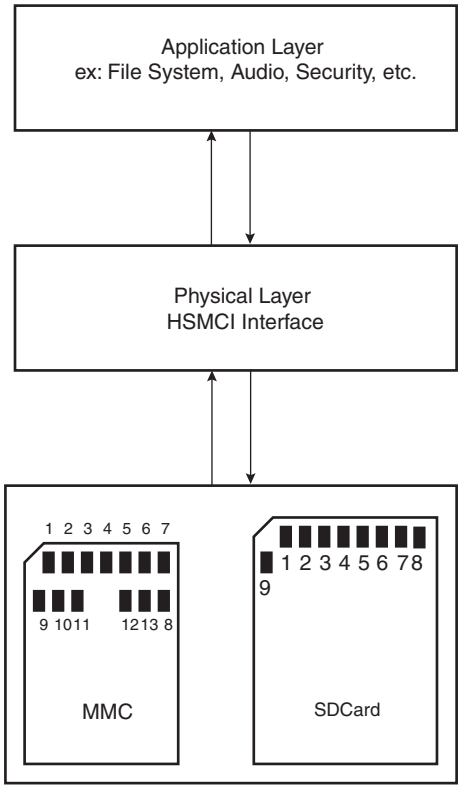
34.3 Block Diagram

Figure 34-1. Block Diagram



34.4 Application Block Diagram

Figure 34-2. Application Block Diagram



34.5 Pin Name List

Table 34-1. I/O Lines Description for 4-bit Configuration

Pin Name[1]	Pin Description	Type[2]	Comments
MCCDA	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0 - MCDA3	Data 0..3 of Slot A	I/O/PP	DAT[0..3] of an MMC DAT[0..3] of an SD Card/SDIO

Notes: 1. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

2. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

34.6 Product Dependencies

34.6.1 I/O Lines

The pins used for interfacing the High Speed MultiMedia Cards or SD Cards are multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to HSMCI pins.

Table 34-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
HSMCI0	MCI0_CDA	PA16	A
HSMCI0	MCI0_CK	PA17	A
HSMCI0	MCI0_DA0	PA15	A
HSMCI0	MCI0_DA1	PA18	A
HSMCI0	MCI0_DA2	PA19	A
HSMCI0	MCI0_DA3	PA20	A
HSMCI1	MCI1_CDA	PA12	B
HSMCI1	MCI1_CK	PA13	B
HSMCI1	MCI1_DA0	PA11	B
HSMCI1	MCI1_DA1	PA2	B
HSMCI1	MCI1_DA2	PA3	B
HSMCI1	MCI1_DA3	PA4	B

34.6.2 Power Management

The HSMCI is clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the HSMCI clock.

34.6.3 Interrupt

The HSMCI interface has an interrupt line connected to the interrupt controller.

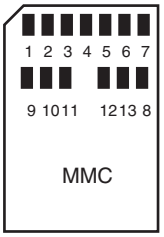
Handling the HSMCI interrupt requires programming the interrupt controller before configuring the HSMCI.

Table 34-3. Peripheral IDs

Instance	ID
HSMCI0	12
HSMCI1	26

34.7 Bus Topology

Figure 34-3. High Speed MultiMedia Memory Card Bus Topology



The High Speed MultiMedia Card communication is based on a 13-pin serial bus interface. It has three communication lines and four supply lines.

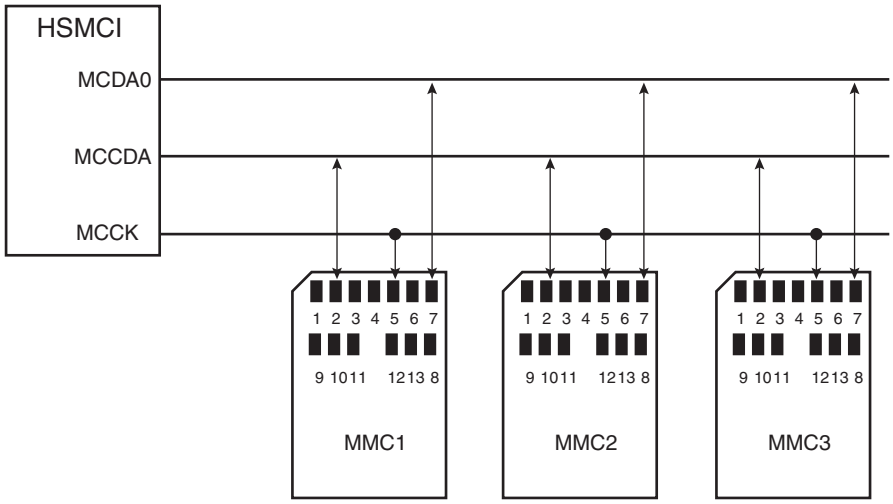
Table 34-4. Bus Topology

Pin Number	Name	Type[1]	Description	HSMCI Pin Name[2] (Slot z)
1	DAT[3]	I/O/PP	Data	MCDz3
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
8	DAT[1]	I/O/PP	Data 1	MCDz1
9	DAT[2]	I/O/PP	Data 2	MCDz2
10	DAT[4]	I/O/PP	Data 4	MCDz4
11	DAT[5]	I/O/PP	Data 5	MCDz5
12	DAT[6]	I/O/PP	Data 6	MCDz6
13	DAT[7]	I/O/PP	Data 7	MCDz7

Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

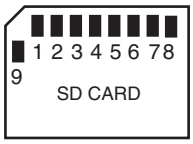
2. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 34-4. MMC Bus Connections (One Slot)



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

Figure 34-5. SD Memory Card Bus Topology



The SD Memory Card bus includes the signals listed in Table 34-5.

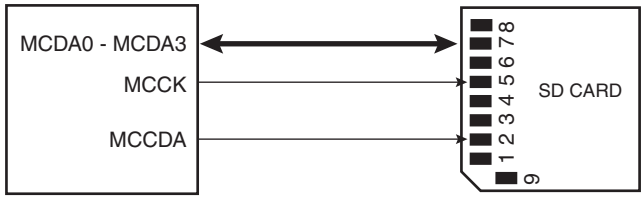
Table 34-5. SD Memory Card Bus Signals

Pin Number	Name	Type[1]	Description	HSMCI 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 HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 34-6. SD Card Bus Connections with One Slot



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

When the HSMCI is configured to operate with SD memory cards, the width of the data bus can be selected in the HSMCI_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 High Speed MultiMedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

34.8 High Speed MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based High Speed 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 High Speed MultiMedia Card System Specification. See also Table 34-6 on page 612.

High Speed MultiMedia Card 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 HSMCI 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 614.).

The HSMCI provides a set of registers to perform the entire range of High Speed MultiMedia Card operations.

34.8.1 Command - Response Operation

After reset, the HSMCI is disabled and becomes valid after setting the MCIEN bit in the HSMCI_CR Control Register.

The PWSEN bit saves power by dividing the HSMCI clock by 2^{PWSDIV} + 1 when the bus is inactive.

The two bits, RDPROOF and WRPROOF in the HSMCI Mode Register (HSMCI_MR) allow stopping the HSMCI Clock during read or write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

All the timings for High Speed MultiMedia Card are defined in the High Speed MultiMedia Card System Specification.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the HSMCI Command Register. The HSMCI_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 HSMCI_CMDR Control Register are described in Table 34-6 and Table 34-7.

Table 34-6. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Resp	Abbreviation	Command Description
CMD2	bcr(1)	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: 1. bcr means broadcast command with response.

Table 34-7. Fields and Values for HSMCI_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 HSMCI_ARGR contains the argument field of the command.

To send a command, the user must perform the following steps:

- Fill the argument register (HSMCI_ARGR) with the command argument.
- Set the command register (HSMCI_CMDR) (see Table 34-7).

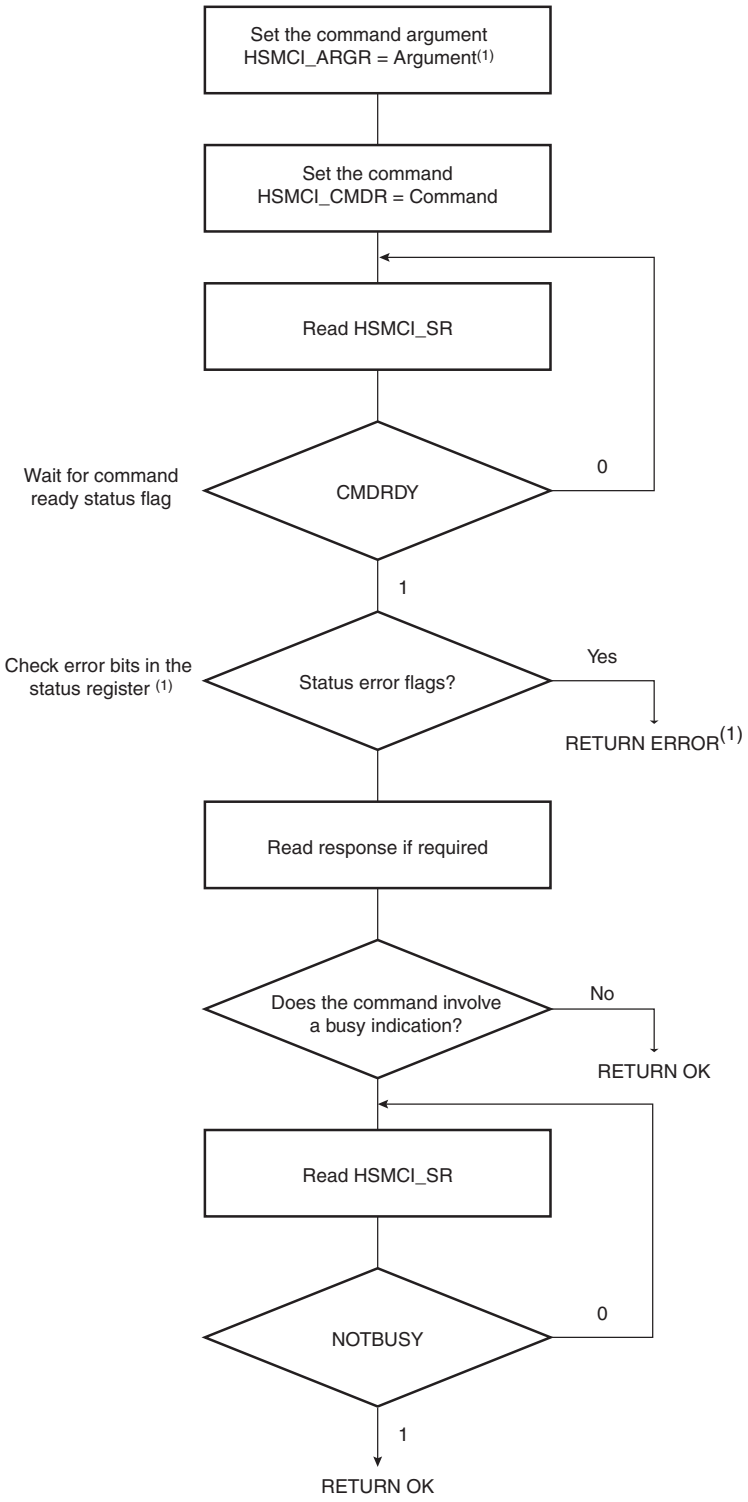
The command is sent immediately after writing the command register.

While the card maintains a busy indication (at the end of a STOP_TRANSMISSION command CMD12, for example), a new command shall not be sent. The NOTBUSY flag in the status register (HSMCI_SR) is asserted when the card releases the busy indication.

If the command requires a response, it can be read in the HSMCI Response Register (HSMCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The HSMCI 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 (HSMCI_IER) allows using an interrupt method.

Figure 34-7. Command/Response Functional Flow Diagram



Note: 1. If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the High Speed MultiMedia Card specification).

34.8.2 Data Transfer Operation

The High Speed MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kinds of transfer can be selected setting the Transfer Type (TRTYP) field in the HSMCI Command Register (HSMCI_CMDR).

These operations can be done using the features of the DMA Controller.

In all cases, the block length (BLKLEN field) must be defined either in the Mode Register HSMCI_MR, or in the Block Register HSMCI_BLKR. This field determines the size of the data block.

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):

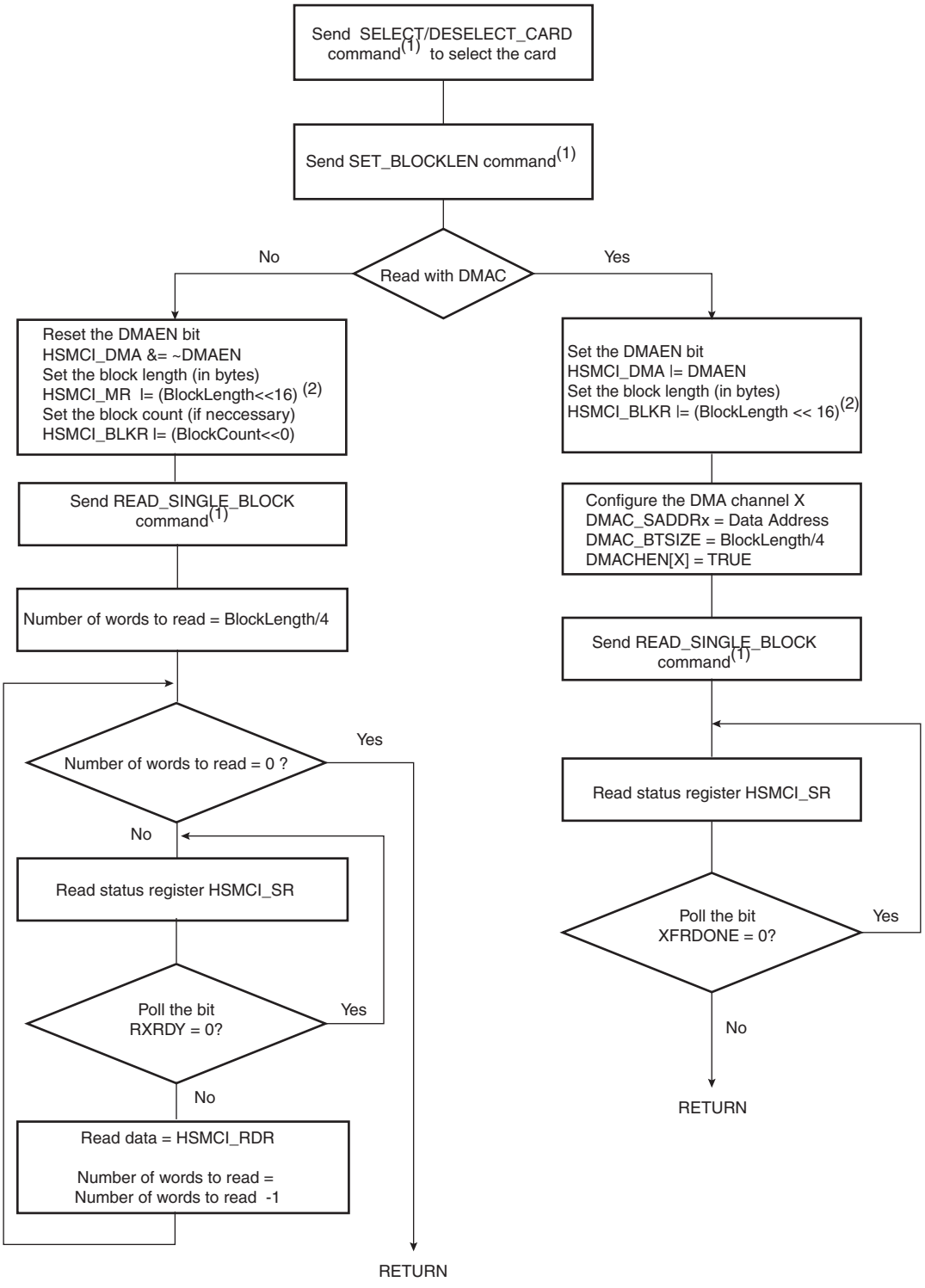
  The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

  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 HSMCI Block Register (HSMCI_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.

34.8.3 Read Operation

The following flowchart (Figure 34-8) shows how to read a single block with or without use of DMAC facilities. In this example, a polling method is used to wait for the end of read. Similarly, the user can configure the Interrupt Enable Register (HSMCI_IER) to trigger an interrupt at the end of read.

Figure 34-8. Read Functional Flow Diagram



Notes: 1. It is assumed that this command has been correctly sent (see Figure 34-7).
2. This field is also accessible in the HSMCI Block Register (HSMCI_BLKR).

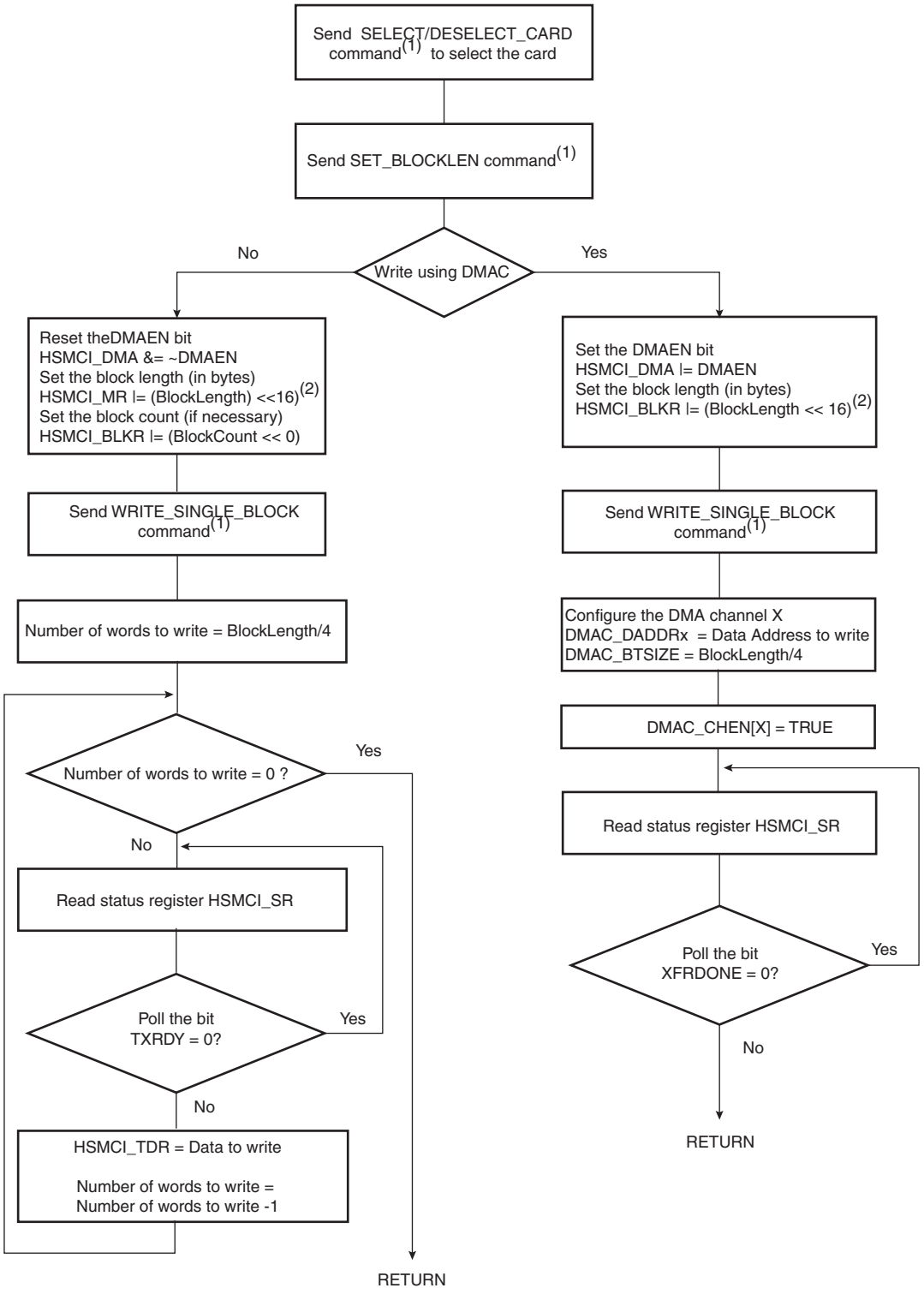
34.8.4 Write Operation

In write operation, the HSMCI Mode Register (HSMCI_MR) is used to define the padding value when writing non-multiple block size. If the bit PADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

If set, the bit DMAEN in the HSMCI_DMA register enables DMA transfer.

The following flowchart (Figure 34-9) shows how to write a single block with or without use of DMA facilities. Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (HSMCI_IMR).

Figure 34-9. Write Functional Flow Diagram

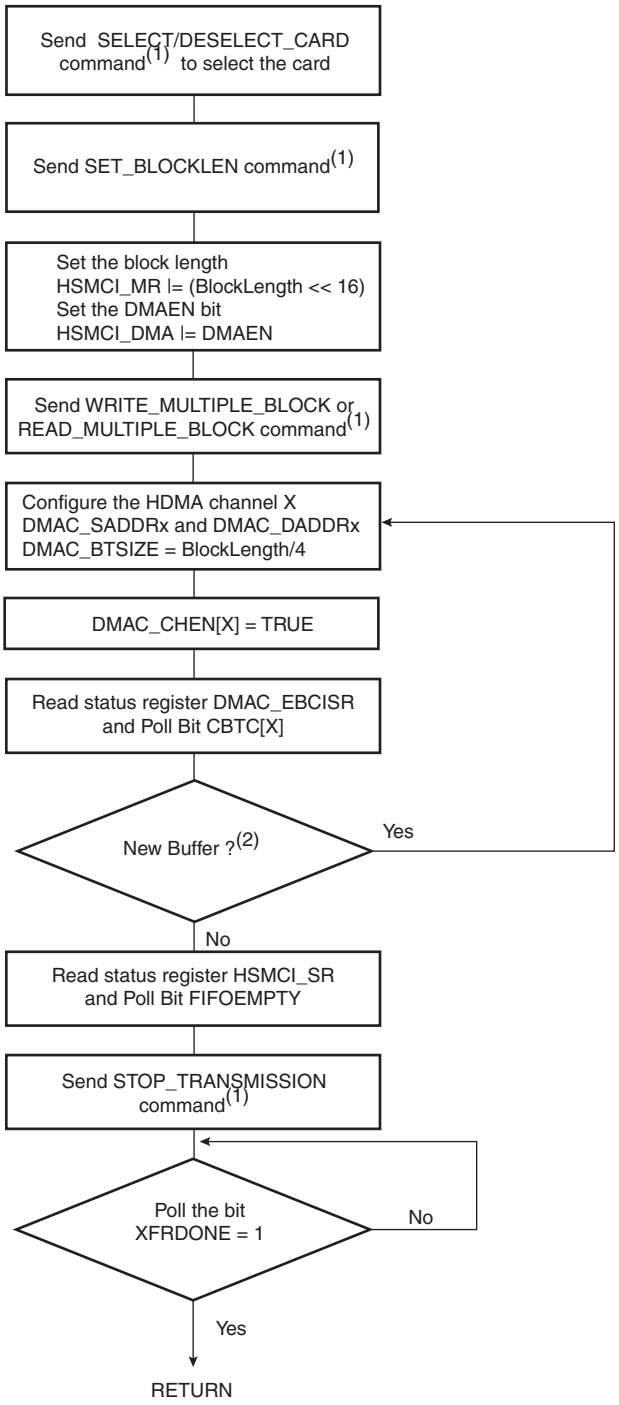


Note: 1. It is assumed that this command has been correctly sent (see Figure 34-7).

2. This field is also accessible in the HSMCI Block Register (HSMCI_BLKR).

The following flowchart (Figure 34-10) shows how to manage read multiple block and write multiple block transfers with the DMA Controller. Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (HSMCI_IMR).

Figure 34-10.Read Multiple Block and Write Multiple Block



Notes: 1. It is assumed that this command has been correctly sent (see Figure 34-7).
2. Handle errors reported in HSMCI_SR.

34.8.5 WRITE_SINGLE_BLOCK Operation using DMA Controller

1. Wait until the current command execution has successfully terminated.
3. Check that CMDRDY and NOTBUSY fields are asserted in HSMCI_SR
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.

4. Program the HSMCI_DMA register with the following fields:
   - OFFSET field with *dma_offset.*
   - CHKSIZE is user defined and set according to DMAC_DCSIZE.
   - DMAEN is set to true to enable DMA hardware handshaking in the HSMCI. This bit was previously set to false.
5. Issue a WRITE_SINGLE_BLOCK command writing HSMCI_ARG then HSMCI_CMDR.
6. Program the DMA Controller.
   1. Read the channel register to choose an available (disabled) channel.
   2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR register.
   3. Program the channel registers.
   4. The DMAC_SADDRx register for Channel x must be set to the location of the source data. When the first data location is not word aligned, the two LSB bits define the temporary value called *dma_offset.* The two LSB bits of DMAC_SADDRx must be set to 0.
   5. The DMAC_DADDRx register for Channel x must be set with the starting address of the HSMCI_FIFO address.
   6. Program the DMAC_CTRLAx register of Channel x with the following field's values:

   –DST_WIDTH is set to WORD.

   –SRC_WIDTH is set to WORD.

   –DCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

   –BTSIZE is programmed with *CEILING((block_length + dma_offset) / 4),* where the ceiling function is the function that returns the smallest integer not less than x.

   7. Program the DMAC_CTRLBx register for Channel x with the following field's values:

   –DST_INCR is set to INCR, the *block_length* value must not be larger than the HSMCI_FIFO aperture.

   –SRC_INCR is set to INCR.

   –FC field is programmed with memory to peripheral flow control mode.

   –both DST_DSCR and SRC_DSCR are set to 1 (descriptor fetch is disabled).

   –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA controller is able to prefetch data and write HSMCI simultaneously.

   8. Program the DMAC_CFGx register for Channel x with the following field's values:

   –FIFOCFG defines the watermark of the DMAC channel FIFO.

   –DST_H2SEL is set to true to enable hardware handshaking on the destination.

   –DST_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

   9. Enable Channel x, writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.
7. Wait for XFRDONE in the HSMCI_SR register.

34.8.6 READ_SINGLE_BLOCK Operation using DMA Controller

34.8.6.1 Block Length is Multiple of 4

1. Wait until the current command execution has successfully completed.
   1. Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Set RDPROOF bit in HSMCI_MR to avoid overflow.

5. Program HSMCI_DMA register with the following fields:
   - ROPT field is set to 0.
   - OFFSET field is set to 0.
   - CHKSIZE is user defined.
   - DMAEN is set to true to enable DMAC hardware handshaking in the HSMCI. This bit was previously set to false.
6. Issue a READ_SINGLE_BLOCK command.
7. Program the DMA controller.
   1. Read the channel register to choose an available (disabled) channel.
   2. Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR register.
   3. Program the channel registers.
   4. The DMAC_SADDRx register for Channel x must be set with the starting address of the HSMCI_FIFO address.
   5. The DMAC_DADDRx register for Channel x must be word aligned.
   6. Program the DMAC_CTRLAx register of Channel x with the following field's values:

   –DST_WIDTH is set to WORD.

   –SRC_WIDTH is set to WORD.

   –SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

   –BTSIZE is programmed with *block_length/4.*

   7. Program the DMAC_CTRLBx register for Channel x with the following field's values:

   –DST_INCR is set to INCR.

   –SRC_INCR is set to INCR.

   –FC field is programmed with peripheral to memory flow control mode.

   –both DST_DSCR and SRC_DSCR are set to 1 (descriptor fetch is disabled).

   –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA controller is able to prefetch data and write HSMCI simultaneously.

   8. Program the DMAC_CFGx register for Channel x with the following field's values:

   –FIFOCFG defines the watermark of the DMA channel FIFO.

   –SRC_H2SEL is set to true to enable hardware handshaking on the destination.

   –SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

   –Enable Channel x, writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.
8. Wait for XFRDONE in the HSMCI_SR register.

34.8.6.2 Block Length is Not Multiple of 4 and Padding Not Used (ROPT field in HSMCI_DMA register set to 0)

In the previous DMA transfer flow (block length multiple of 4), the DMA controller is configured to use only WORD AHB access. When the block length is no longer a multiple of 4 this is no longer true. The DMA controller is programmed to copy exactly the block length number of bytes using 2 transfer descriptors.

1. Use the previous step until READ_SINGLE_BLOCK then
2. Program the DMA controller to use a two descriptors linked list.

1. Read the channel register to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR register.
3. Program the channel registers in the Memory for the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W, standing for LLI word oriented transfer.
4. The LLI_W.DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
5. The LLI_W.DMAC_DADDRx field in the memory must be word aligned.
6. Program LLI_W.DMAC_CTRLAx with the following field's values:
   –DST_WIDTH is set to WORD.
   –SRC_WIDTH is set to WORD.
   –SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.
   –BTSIZE is programmed with *block_length/4.* If BTSIZE is zero, this descriptor is skipped later.
7. Program LLI_W.DMAC_CTRLBx with the following field's values:
   –DST_INCR is set to INCR
   –SRC_INCR is set to INCR
   –FC field is programmed with peripheral to memory flow control mode.
   –SRC_DSCR is set to zero. (descriptor fetch is enabled for the SRC)
   –DST_DSCR is set to one. (descriptor fetch is disabled for the DST)
   –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, DMA controller is able to prefetch data and write HSMCI simultaneously.
8. Program the LLI_W.DMAC_CFGx register for Channel x with the following field's values:
   –FIFOCFG defines the watermark of the DMA channel FIFO.
   –DST_REP is set to zero meaning that address are contiguous.
   –SRC_H2SEL is set to true to enable hardware handshaking on the destination.
   –SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.
9. Program LLI_W.DMAC_DSCRx with the address of LLI_B descriptor. And set DSCRx_IF to the AHB Layer ID. This operation actually links the Word oriented descriptor on the second byte oriented descriptor. When *block_length[1:0]* is equal to 0 (multiple of 4) LLI_W.DMAC_DSCRx points to 0, only LLI_W is relevant.
10. Program the channel registers in the Memory for the second descriptor. This descriptor will be byte oriented. This descriptor is referred to as LLI_B, standing for LLI Byte oriented.
11. The LLI_B.DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
12. The LLI_B.DMAC_DADDRx is not relevant if previous word aligned descriptor was enabled. If 1, 2 or 3 bytes are transferred that address is user defined and not word aligned.
13. Program LLI_B.DMAC_CTRLAx with the following field's values:
   –DST_WIDTH is set to BYTE.
   –SRC_WIDTH is set to BYTE.
   –SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.
   –BTSIZE is programmed with *block_length[1:0].* (last 1, 2, or 3 bytes of the buffer).
14. Program LLI_B.DMAC_CTRLBx with the following field's values:
   –DST_INCR is set to INCR
   –SRC_INCR is set to INCR

–FC field is programmed with peripheral to memory flow control mode.

–Both SRC_DSCR and DST_DSCR are set to 1 (descriptor fetch is disabled) or Next descriptor location points to 0.

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, DMA Controller is able to prefetch data and write HSMCI simultaneously.

15. Program LLI_B.DMAC_CFGx memory location for Channel x with the following field's values:

–FIFOCFG defines the watermark of the DMA channel FIFO.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

16. Program LLI_B.DMAC_DSCR with 0.
17. Program the DMAC_CTRLBx register for Channel x with 0. its content is updated with the LLI fetch operation.
18. Program DMAC_DSCRx with the address of LLI_W if *block_length* greater than 4 else with address of LLI_B.
19. Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

3. Wait for XFRDONE in the HSMCI_SR register.

34.8.6.3 Block Length is Not Multiple of 4, with Padding Value (ROPT field in HSMCI_DMA register set to 1)

When the ROPT field is set to one, The DMA Controller performs only WORD access on the bus to transfer a non-multiple of 4 block length. Unlike previous flow, in which the transfer size is rounded to the nearest multiple of 4.

1. Program the HSMCI Interface, see previous flow.
   - ROPT field is set to 1.
2. Program the DMA Controller
   1. Read the channel register to choose an available (disabled) channel.
   2. Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR register.
   3. Program the channel registers.
   4. The DMAC_SADDRx register for Channel x must be set with the starting address of the HSMCI_FIFO address.
   5. The DMAC_DADDRx register for Channel x must be word aligned.
   6. Program the DMAC_CTRLAx register of Channel x with the following field's values:

   –DST_WIDTH is set to WORD

   –SRC_WIDTH is set to WORD

   –SCSIZE must be set according to the value of HSMCI_DMA.CHKSIZE Field.

   –BTSIZE is programmed with *CEILING(block_length/4).*

   7. Program the DMAC_CTRLBx register for Channel x with the following field's values:

   –DST_INCR is set to INCR

   –SRC_INCR is set to INCR

   –FC field is programmed with peripheral to memory flow control mode.

   –both DST_DSCR and SRC_DSCR are set to 1. (descriptor fetch is disabled)

   –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

   8. Program the DMAC_CFGx register for Channel x with the following field's values:

   –FIFOCFG defines the watermark of the DMA channel FIFO.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

–Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

3. Wait for XFRDONE in the HSMCI_SR register.

34.8.7 WRITE_MULTIPLE_BLOCK

34.8.7.1 One Block per Descriptor

1. Wait until the current command execution has successfully terminated.
    1. Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Program the HSMCI_DMA register with the following fields:
    - OFFSET field with *dma_offset.*
    - CHKSIZE is user defined.
    - DMAEN is set to true to enable DMAC hardware handshaking in the HSMCI. This bit was previously set to false.
5. Issue a WRITE_MULTIPLE_BLOCK command.
6. Program the DMA Controller to use a list of descriptors. Each descriptor transfers one block of data. Block *n* of data is transferred with descriptor LLI(n).
    1. Read the channel register to choose an available (disabled) channel.
    2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR register.
    3. Program a List of descriptors.
    4. The LLI(n).DMAC_SADDRx memory location for Channel x must be set to the location of the source data. When the first data location is not word aligned, the two LSB bits define the temporary value called *dma_offset.* The two LSB bits of LLI(n).DMAC_SADDRx must be set to 0.
    5. The LLI(n).DMAC_DADDRx register for Channel x must be set with the starting address of the HSMCI_FIFO address.
    6. Program the LLI(n).DMAC_CTRLAx register of Channel x with the following field's values:

    –DST_WIDTH is set to WORD.

    –SRC_WIDTH is set to WORD.

    –DCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

    –BTSIZE is programmed with *CEILING((block_length + dma_offset)/4).*

    7. Program the LLI(n).DMAC_CTRLBx register for Channel x with the following field's values:

    –DST_INCR is set to INCR.

    –SRC_INCR is set to INCR.

    –DST_DSCR is set to 0 (fetch operation is enabled for the destination).

    –SRC_DSCR is set to 1 (source address is contiguous).

    –FC field is programmed with memory to peripheral flow control mode.

    –Both DST_DSCR and SRC_DSCR are set to 1 (descriptor fetch is disabled).

    –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, DMA Controller is able to prefetch data and write HSMCI simultaneously.

    8. Program the LLI(n).DMAC_CFGx register for Channel x with the following field's values:

    –FIFOCFG defines the watermark of the DMA channel FIFO.

–DST_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_REP is set to 0. (contiguous memory access at block boundary)

–DST_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

9. If LLI(n) is the last descriptor, then LLI(n).DSCR points to 0 else LLI(n) points to the start address of LLI(n+1).
10. Program DMAC_CTRLBx for the Channel Register x with 0. Its content is updated with the LLI fetch operation.
11. Program DMAC_DSCRx for the Channel Register x with the address of the first descriptor LLI(0).
12. Enable Channel x writing one to DMAC_CHER[x]. The DMA is ready and waiting for request.

7. Poll CBTC[x] bit in the DMAC_EBCISR Register.
8. If a new list of buffers shall be transferred, repeat step 6. Check and handle HSMCI errors.
9. Poll FIFOEMPTY field in the HSMCI_SR.
10. Send The STOP_TRANSMISSION command writing HSMCI_ARG then HSMCI_CMDR.
11. Wait for XFRDONE in the HSMCI_SR register.

34.8.8 READ_MULTIPLE_BLOCK

34.8.8.1 Block Length is a Multiple of 4

1. Wait until the current command execution has successfully terminated.
    1. Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Set RDPROOF bit in HSMCI_MR to avoid overflow.
5. Program the HSMCI_DMA register with the following fields:
    - ROPT field is set to 0.
    - OFFSET field is set to 0.
    - CHKSIZE is user defined.
    - DMAEN is set to true to enable DMAC hardware handshaking in the HSMCI. This bit was previously set to false.
6. Issue a READ_MULTIPLE_BLOCK command.
7. Program the DMA Controller to use a list of descriptors:
    1. Read the channel register to choose an available (disabled) channel.
    2. Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_EBCISR register.
    3. Program the channel registers in the Memory with the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W(n), standing for LLI word oriented transfer for block *n*.
    4. The LLI_W(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
    5. The LLI_W(n).DMAC_DADDRx field in the memory must be word aligned.
    6. Program LLI_W(n).DMAC_CTRLAx with the following field's values:

    –DST_WIDTH is set to WORD

    –SRC_WIDTH is set to WORD

    –SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

    –BTSIZE is programmed with *block_length/4.*

    7. Program LLI_W(n).DMAC_CTRLBx with the following field's values:

    –DST_INCR is set to INCR.

–SRC_INCR is set to INCR.

–FC field is programmed with peripheral to memory flow control mode.

–SRC_DSCR is set to 0 (descriptor fetch is enabled for the SRC).

–DST_DSCR is set to TRUE (descriptor fetch is disabled for the DST).

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

8. Program the LLI_W(n).DMAC_CFGx register for Channel x with the following field's values:

–FIFOCFG defines the watermark of the DMA channel FIFO.

–DST_REP is set to zero. Addresses are contiguous.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

9. Program LLI_W(n).DMAC_DSCRx with the address of LLI_W(n+1) descriptor. And set the DSCRx_IF to the AHB Layer ID. This operation actually links descriptors together. If LLI_W(n) is the last descriptor then LLI_W(n).DMAC_DSCRx points to 0.
10. Program the DMAC_CTRLBx register for Channel x with 0. its content is updated with the LLI Fetch operation.
11. Program DMAC_DSCRx register for Channel x with the address of LLI_W(0).
12. Enable Channel x writing one to DMAC_CHER[x]. The DMA is ready and waiting for request.

8. Poll CBTC[x] bit in the DMAC_EBCISR Register.
9. If a new list of buffer shall be transferred repeat step 6. Check and handle HSMCI errors.
10. Poll FIFOEMPTY field in the HSMCI_SR.
11. Send The STOP_TRANSMISSION command writing the HSMCI_ARG then the HSMCI_CMDR.
12. Wait for XFRDONE in the HSMCI_SR register.

34.8.8.2 Block Length is Not Multiple of 4. (ROPT field in HSMCI_DMA register set to 0)

Two DMA Transfer descriptors are used to perform the HSMCI block transfer.

1. Use the previous step to configure the HSMCI to perform a READ_MULTIPLE_BLOCK command.
2. Issue a READ_MULTIPLE_BLOCK command.
3. Program the DMA Controller to use a list of descriptors.
    1. Read the channel register to choose an available (disabled) channel.
    2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR register.
    3. For every block of data repeat the following procedure:
    4. Program the channel registers in the Memory for the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W(n) standing for LLI word oriented transfer for block *n*.
    5. The LLI_W(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
    6. The LLI_W(n).DMAC_DADDRx field in the memory must be word aligned.
    7. Program LLI_W(n).DMAC_CTRLAx with the following field's values:

    –DST_WIDTH is set to WORD.

    –SRC_WIDTH is set to WORD.

    –SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

    –BTSIZE is programmed with *block_length/4.* If BTSIZE is zero, this descriptor is skipped later.

    8. Program LLI_W(n).DMAC_CTRLBx with the following field's values:

    –DST_INCR is set to INCR.

–SRC_INCR is set to INCR.

–FC field is programmed with peripheral to memory flow control mode.

–SRC_DSCR is set to 0 (descriptor fetch is enabled for the SRC).

–DST_DSCR is set to TRUE (descriptor fetch is disabled for the DST).

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

9. Program the LLI_W(n).DMAC_CFGx register for Channel x with the following field's values:

–FIFOCFG defines the watermark of the DMA channel FIFO.

–DST_REP is set to zero. Address are contiguous.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

10. Program LLI_W(n).DMAC_DSCRx with the address of LLI_B(n) descriptor. And set the DSCRx_IF to the AHB Layer ID. This operation actually links the Word oriented descriptor on the second byte oriented descriptor. When *block_length[1:0]* is equal to 0 (multiple of 4) LLI_W(n).DMAC_DSCRx points to 0, only LLI_W(n) is relevant.
11. Program the channel registers in the Memory for the second descriptor. This descriptor will be byte oriented. This descriptor is referred to as LLI_B(n), standing for LLI Byte oriented.
12. The LLI_B(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
13. The LLI_B(n).DMAC_DADDRx is not relevant if previous word aligned descriptor was enabled. If 1, 2 or 3 bytes are transferred, that address is user defined and not word aligned.
14. Program LLI_B(n).DMAC_CTRLAx with the following field's values:

–DST_WIDTH is set to BYTE.

–SRC_WIDTH is set to BYTE.

–SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

–BTSIZE is programmed with *block_length[1:0].* (last 1, 2, or 3 bytes of the buffer).

15. Program LLI_B(n).DMAC_CTRLBx with the following field's values:

–DST_INCR is set to INCR.

–SRC_INCR is set to INCR.

–FC field is programmed with peripheral to memory flow control mode.

–Both SRC_DSCR and DST_DSCR are set to 1 (descriptor fetch is disabled) or Next descriptor location points to 0.

–DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

16. Program LLI_B(n).DMAC_CFGx memory location for Channel x with the following field's values:

–FIFOCFG defines the watermark of the DMAC channel FIFO.

–SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller

17. Program LLI_B(n).DMAC_DSCR with address of descriptor LLI_W(n+1). If LLI_B(n) is the last descriptor, then program LLI_B(n).DMAC_DSCR with 0.
18. Program the DMAC_CTRLBx register for Channel x with 0, its content is updated with the LLI Fetch operation.
19. Program DMAC_DSCRx with the address of LLI_W(0) if *block_length* is greater than 4 else with address of LLI_B(0).
20. Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

4. Enable DMADONE interrupt in the HSMCI_IER register.
5. Poll CBTC[x] bit in the DMAC_EBCISR Register.
6. If a new list of buffers shall be transferred, repeat step 7. Check and handle HSMCI errors.
7. Poll FIFOEMPTY field in the HSMCI_SR.
8. Send The STOP_TRANSMISSION command writing HSMCI_ARG then HSMCI_CMDR.
9. Wait for XFRDONE in the HSMCI_SR register.

34.8.8.3 Block Length is Not a Multiple of 4. (ROPT field in HSMCI_DMA register set to 1)

One DMA Transfer descriptor is used to perform the HSMCI block transfer, the DMA writes a rounded up value to the nearest multiple of 4.

1. Use the previous step to configure the HSMCI to perform a READ_MULTIPLE_BLOCK.
2. Set the ROPT field to 1 in the HSMCI_DMA register.
3. Issue a READ_MULTIPLE_BLOCK command.
4. Program the DMA controller to use a list of descriptors:
    1. Read the channel register to choose an available (disabled) channel.
    2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_EBCISR register.
    3. Program the channel registers in the Memory with the first descriptor. This descriptor will be word oriented. This descriptor is referred to as LLI_W(n), standing for LLI word oriented transfer for block *n*.
    4. The LLI_W(n).DMAC_SADDRx field in memory must be set with the starting address of the HSMCI_FIFO address.
    5. The LLI_W(n).DMAC_DADDRx field in the memory must be word aligned.
    6. Program LLI_W(n).DMAC_CTRLAx with the following field's values:

    –DST_WIDTH is set to WORD.

    –SRC_WIDTH is set to WORD.

    –SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

    –BTSIZE is programmed with *Ceiling(block_length/4)*.

    7. Program LLI_W(n).DMAC_CTRLBx with the following field's values:

    –DST_INCR is set to INCR

    –SRC_INCR is set to INCR

    –FC field is programmed with peripheral to memory flow control mode.

    –SRC_DSCR is set to 0. (descriptor fetch is enabled for the SRC)

    –DST_DSCR is set to TRUE. (descriptor fetch is disabled for the DST)

    –DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

    8. Program the LLI_W(n).DMAC_CFGx register for Channel x with the following field's values:

    –FIFOCFG defines the watermark of the DMA channel FIFO.

    –DST_REP is set to zero. Address are contiguous.

    –SRC_H2SEL is set to true to enable hardware handshaking on the destination.

–SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller.

9. Program LLI_W(n).DMAC_DSCRx with the address of LLI_W(n+1) descriptor. And set the DSCRx_IF to the AHB Layer ID. This operation actually links descriptors together. If LLI_W(n) is the last descriptor then LLI_W(n).DMAC_DSCRx points to 0.
10. Program the DMAC_CTRLBx register for Channel x with 0. its content is updated with the LLI Fetch operation.
11. Program the DMAC_DSCRx register for Channel x with the address of LLI_W(0).
12. Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

5. Poll CBTC[x] bit in the DMAC_EBCISR register.
6. If a new list of buffers shall be transferred repeat step 7. Check and handle HSMCI errors.
7. Poll FIFOEMPTY field in the HSMCI_SR.
8. Send The STOP_TRANSMISSION command writing the HSMCI_ARG then the HSMCI_CMDR.
9. Wait for XFRDONE in the HSMCI_SR register.

34.9 SD/SDIO Card Operation

The High Speed 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 MultiMedia 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 High Speed 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 High Speed MultiMedia Card is the initialization process.

The SD/SDIO Card Register (HSMCI_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).

34.9.1 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 HSMCI Command Register (HSMCI_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 HSMCI Block Register (HSMCI_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 HSMCI Command Register.

34.9.2 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 HSMCI Interrupt Enable Register. The SDIO interrupt is sampled regardless of the currently selected slot.

34.10 CE-ATA Operation

CE-ATA maps the streamlined ATA command set onto the MMC interface. The ATA task file is mapped onto MMC register space.

CE-ATA utilizes five MMC commands:

- GO_IDLE_STATE (CMD0): used for hard reset.
- STOP_TRANSMISSION (CMD12): causes the ATA command currently executing to be aborted.
- FAST_IO (CMD39): Used for single register access to the ATA taskfile registers, 8 bit access only.
- RW_MULTIPLE_REGISTERS (CMD60): used to issue an ATA command or to access the control/status registers.
- RW_MULTIPLE_BLOCK (CMD61): used to transfer data for an ATA command.

CE-ATA utilizes the same MMC command sequences for initialization as traditional MMC devices.

34.10.1 Executing an ATA Polling Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8kB of DATA.
2. Read the ATA status register until DRQ is set.
3. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
4. Read the ATA status register until DRQ && BSY are set to 0.

34.10.2 Executing an ATA Interrupt Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8kB of DATA with nIEN field set to zero to enable the command completion signal in the device.
2. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
3. Wait for Completion Signal Received Interrupt.

34.10.3 Aborting an ATA Command

If the host needs to abort an ATA command prior to the completion signal it must send a special command to avoid potential collision on the command line. The SPCMD field of the HSMCI_CMDR must be set to 3 to issue the CE-ATA completion Signal Disable Command.

34.10.4 CE-ATA Error Recovery

Several methods of ATA command failure may occur, including:

- No response to an MMC command, such as RW_MULTIPLE_REGISTER (CMD60).
- CRC is invalid for an MMC command or response.
- CRC16 is invalid for an MMC data packet.
- ATA Status register reflects an error by setting the ERR bit to one.
- The command completion signal does not arrive within a host specified time out period.

Error conditions are expected to happen infrequently. Thus, a robust error recovery mechanism may be used for each error event.

The recommended error recovery procedure after a timeout is:

- Issue the command completion signal disable if nIEN was cleared to zero and the RW_MULTIPLE_BLOCK (CMD61) response has been received.
- Issue STOP_TRANSMISSION (CMD12) and successfully receive the R1 response.
- Issue a software reset to the CE-ATA device using FAST_IO (CMD39).

If STOP_TRANMISSION (CMD12) is successful, then the device is again ready for ATA commands. However, if the error recovery procedure does not work as expected or there is another timeout, the next step is to issue GO_IDLE_STATE (CMD0) to the device. GO_IDLE_STATE (CMD0) is a hard reset to the device and completely resets all device states.

Note that after issuing GO_IDLE_STATE (CMD0), all device initialization needs to be completed again. If the CE-ATA device completes all MMC commands correctly but fails the ATA command with the ERR bit set in the ATA Status register, no error recovery action is required. The ATA command itself failed implying that the device could not complete the action requested, however, there was no communication or protocol failure. After the device signals an error by setting the ERR bit to one in the ATA Status register, the host may attempt to retry the command.

34.11 HSMCI Boot Operation Mode

In boot operation mode, the processor can read boot data from the slave (MMC device) by keeping the CMD line low after power-on before issuing CMD1. The data can be read from either the boot area or user area, depending on register setting.

34.11.1 Boot Procedure, Processor Mode

1. Configure the HSMCI data bus width programming SDCBUS Field in the HSMCI_SDCR register. The BOOT_BUS_WIDTH field located in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks, writing BLKLEN and BCNT fields of the HSMCI_BLKR Register.
3. Issue the Boot Operation Request command by writing to the HSMCI_CMDR register with SPCMD field set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
4. The BOOT_ACK field located in the HSMCI_CMDR register must be set to one, if the BOOT_ACK field of the MMC device located in the Extended CSD register is set to one.
5. Host processor can copy boot data sequentially as soon as the RXRDY flag is asserted.
6. When Data transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR register with SPCMD field set to BOOTEND.

34.11.2 Boot Procedure DMA Mode

1. Configure the HSMCI data bus width by programming SDCBUS Field in the HSMCI_SDCR register. The BOOT_BUS_WIDTH field in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks by writing BLKLEN and BCNT fields of the HSMCI_BLKR register.
3. Enable DMA transfer in the HSMCI_DMA register.
4. Configure DMA controller, program the total amount of data to be transferred and enable the relevant channel.
5. Issue the Boot Operation Request command by writing to the HSMCI_CMDR register with SPCND set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
6. DMA controller copies the boot partition to the memory.
7. When DMA transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR register with SPCMD field set to BOOTEND.

34.12 HSMCI Transfer Done Timings

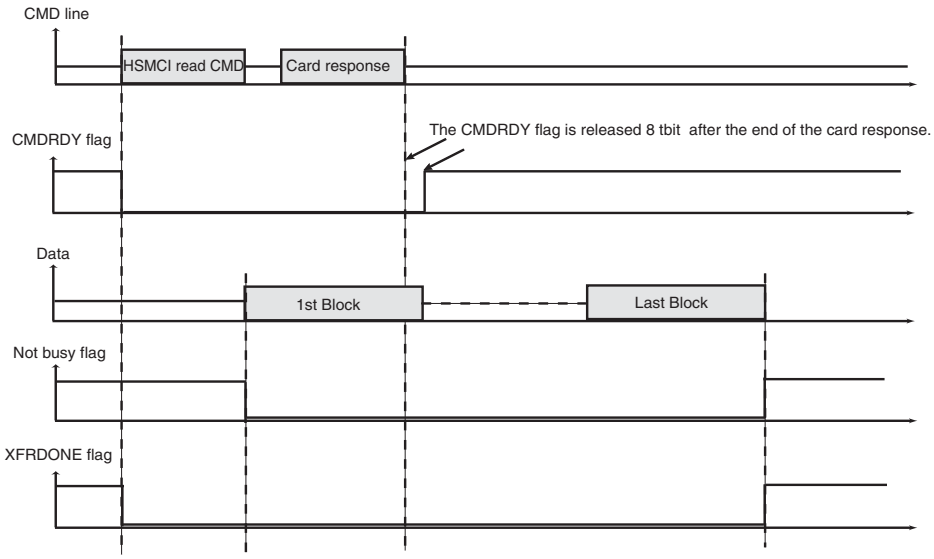
34.12.1 Definition

The XFRDONE flag in the HSMCI_SR indicates exactly when the read or write sequence is finished.

34.12.2 Read Access

During a read access, the XFRDONE flag behaves as shown in Figure 34-11.

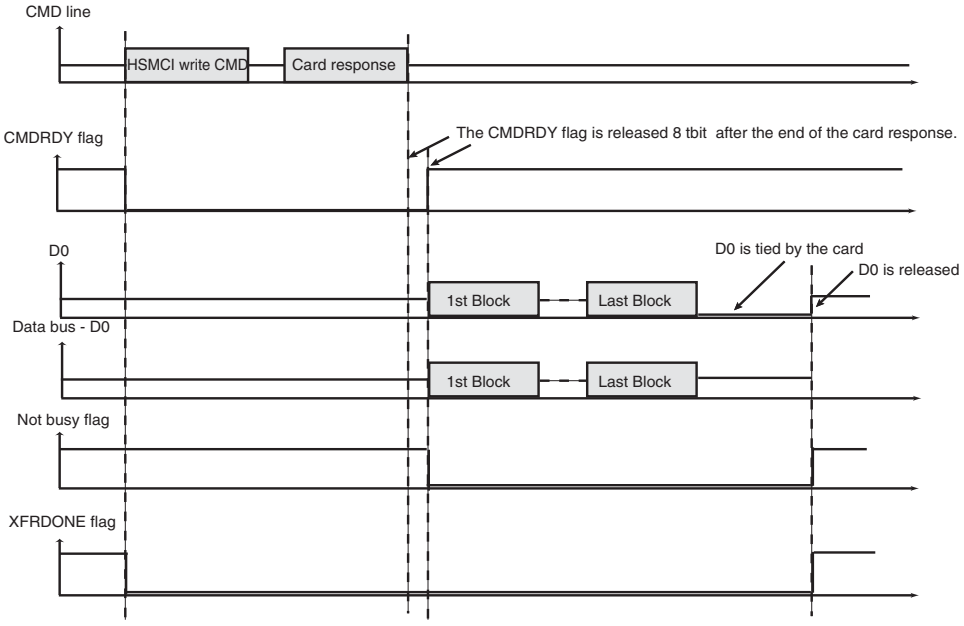
Figure 34-11.XFRDONE During a Read Access



34.12.3 Write Access

During a write access, the XFRDONE flag behaves as shown in Figure 34-12.

Figure 34-12.XFRDONE During a Write Access



34.13 Write Protection Registers

To prevent any single software error that may corrupt HSMCI behavior, the entire HSMCI address space from address offset 0x000 to 0x00FC can be write-protected by setting the WPEN bit in the "H SMCI Write Protect Mode Register" (HSMCI_WPMR).

If a write access to anywhere in the HSMCI address space from address offset 0x000 to 0x00FC is detected, then the WPVS flag in the HSMCI Write Protect Status Register (HSMCI_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the HSMCI Write Protect Mode Register (HSMCI_WPMR) with the appropriate access key, WPKEY.

The protected registers are:

- "HSMCI Mode Register" on page 635
- "HSMCI Data Timeout Register" on page 636
- "HSMCI SDCard/SDIO Register" on page 637
- "HSMCI Completion Signal Timeout Register" on page 642
- "HSMCI DMA Configuration Register" on page 655
- "HSMCI Configuration Register" on page 656

34.14 High Speed MultiMedia Card Interface (HSMCI) User Interface

Table 34-8. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	HSMCI_CR	Write	–
0x04	Mode Register	HSMCI_MR	Read-write	0x0
0x08	Data Timeout Register	HSMCI_DTOR	Read-write	0x0
0x0C	SD/SDIO Card Register	HSMCI_SDCR	Read-write	0x0
0x10	Argument Register	HSMCI_ARGR	Read-write	0x0
0x14	Command Register	HSMCI_CMDR	Write	–
0x18	Block Register	HSMCI_BLKR	Read-write	0x0
0x1C	Completion Signal Timeout Register	HSMCI_CSTOR	Read-write	0x0
0x20	Response Register(1)	HSMCI_RSPR	Read	0x0
0x24	Response Register(1)	HSMCI_RSPR	Read	0x0
0x28	Response Register(1)	HSMCI_RSPR	Read	0x0
0x2C	Response Register(1)	HSMCI_RSPR	Read	0x0
0x30	Receive Data Register	HSMCI_RDR	Read	0x0
0x34	Transmit Data Register	HSMCI_TDR	Write	–
0x38 - 0x3C	Reserved	–	–	–
0x40	Status Register	HSMCI_SR	Read	0xC0E5
0x44	Interrupt Enable Register	HSMCI_IER	Write	–
0x48	Interrupt Disable Register	HSMCI_IDR	Write	–
0x4C	Interrupt Mask Register	HSMCI_IMR	Read	0x0
0x50	DMA Configuration Register	HSMCI_DMA	Read-write	0x00
0x54	Configuration Register	HSMCI_CFG	Read-write	0x00
0x58-0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	HSMCI_WPMR	Read-write	–
0xE8	Write Protection Status Register	HSMCI_WPSR	Read-only	–
0xEC - 0xFC	Reserved	–	–	–
0x100-0x1FC	Reserved	–	–	–
0x200	FIFO Memory Aperture0	HSMCI_FIFO0	Read-write	0x0
...	...	...	...	...
0x5FC	FIFO Memory Aperture255	HSMCI_FIFO255	Read-write	0x0

Notes: 1. The Response Register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

34.14.1 HSMCI Control Register

Name: HSMCI_CR

Address: 0xF0008000 (0), 0xF000C000 (1)

Access: Write-only

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

- **MCIEN: Multi-Media Interface Enable**

0 = No effect.

1 = Enables the Multi-Media Interface if MCDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0 = No effect.

1 = Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0 = No effect.

1 = Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field (Mode Register, HSMCI_MR).

- **PWSDIS: Power Save Mode Disable**

0 = No effect.

1 = Disables the Power Saving Mode.

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the HSMCI. A software triggered hardware reset of the HSMCI interface is performed.

34.14.2 HSMCI Mode Register

Name: HSMCI_MR

Address: 0xF0008004 (0), 0xF000C004 (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “H SMCI Write Protect Mode Register” on page 657.

- **CLKDIV: Clock Divider**

High Speed MultiMedia Card Interface clock (MCCK or HSMCI_CK) is Master Clock (MCK) divider by ({CLKDIV,CLKODD}+2).

- **PWSDIV: Power Saving Divider**

High Speed 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 HSMCI_CR (HSMCI_PWSEN bit).

- **RDPROOF: Read Proof Enable**

Enabling Read Proof allows to stop the HSMCI 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 HSMCI 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.

- **FBYTE: Force Byte Transfer**

Enabling Force Byte Transfer allow byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported.

Warning: BLKLEN value depends on FBYTE.

0 = Disables Force Byte Transfer.

1 = Enables Force Byte Transfer.

- **PADV: Padding Value**

0 = 0x00 value is used when padding data in write transfer.

1 = 0xFF value is used when padding data in write transfer.

PADV may be only in manual transfer.

- **CLKODD: Clock divider is odd**

This field is the least significant bit of the clock divider and indicates the clock divider parity.

34.14.3 HSMCI Data Timeout Register

Name: HSMCI_DTOR

Address: 0xF0008008 (0), 0xF000C008 (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “H SMCI Write Protect Mode Register” on page 657.

- **DTOCYC: Data Timeout Cycle Number**

These fields determine the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

- **DTOMUL: Data Timeout Multiplier**

Multiplier is defined by DTOMUL as shown in the following table:

Value	Name	Description
0	1	DTOCYC
1	16	DTOCYC x 16
2	128	DTOCYC x 128
3	256	DTOCYC x 256
4	1024	DTOCYC x 1024
5	4096	DTOCYC x 4096
6	65536	DTOCYC x 65536
7	1048576	DTOCYC x 1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the HSMCI Status Register (HSMCI_SR) rises.

34.14.4 HSMCI SDCard/SDIO Register

Name: HSMCI_SDCR

Address: 0xF000800C (0), 0xF000C00C (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “H SMCI Write Protect Mode Register” on page 657.

- **SDCSEL: SDCard/SDIO Slot**

Value	Name	Description
0	SLOTA	Slot A is selected.
1	SLOTB	–
2	SLOTC	–
3	SLOTD	–

- **SDCBUS: SDCard/SDIO Bus Width**

Value	Name	Description
0	1	1 bit
1	–	Reserved
2	4	4 bit
3	8	8 bit

34.14.5 HSMCI Argument Register

Name: HSMCI_ARGR

Address: 0xF0008010 (0), 0xF000C010 (1)

Access: Read-write

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

- **ARG: Command Argument**

34.14.6 HSMCI Command Register

Name: HSMCI_CMDR

Address: 0xF0008014 (0), 0xF000C014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	BOOT_ACK	ATACS	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 HSMCI_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**

This is the command index.

- **RSPTYP: Response Type**

Value	Name	Description
0	NORESP	No response.
1	48_BIT	48-bit response.
2	136_BIT	136-bit response.
3	R1B	R1b response type

- **SPCMD: Special Command**

Value	Name	Description
0	STD	Not a special CMD.
1	INIT	Initialization CMD: 74 clock cycles for initialization sequence.
2	SYNC	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
3	CE_ATA	CE-ATA Completion Signal disable Command. The host cancels the ability for the device to return a command completion signal on the command line.
4	IT_CMD	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
5	IT_RESP	Interrupt response: Corresponds to the Interrupt Mode (CMD40).
6	BOR	Boot Operation Request. Start a boot operation mode, the host processor can read boot data from the MMC device directly.
7	EBO	End Boot Operation. This command allows the host processor to terminate the boot operation mode.

- **OPDCMD: Open Drain Command**

0 (PUSHPULL) = Push pull command.

1 (OPENDRAIN) = Open drain command.

- **MAXLAT: Max Latency for Command to Response**

0 (5) = 5-cycle max latency.

1 (64) = 64-cycle max latency.

- **TRCMD: Transfer Command**

Value	Name	Description
0	NO_DATA	No data transfer
1	START_DATA	Start data transfer
2	STOP_DATA	Stop data transfer
3	–	Reserved

- **TRDIR: Transfer Direction**

0 (WRITE) = Write.

1 (READ) = Read.

- **TRTYP: Transfer Type**

Value	Name	Description
0	SINGLE	MMC/SD Card Single Block
1	MULTIPLE	MMC/SD Card Multiple Block
2	STREAM	MMC Stream
4	BYTE	SDIO Byte
5	BLOCK	SDIO Block

- **IOSPCMD: SDIO Special Command**

Value	Name	Description
0	STD	Not an SDIO Special Command
1	SUSPEND	SDIO Suspend Command
2	RESUME	SDIO Resume Command

- **ATACS: ATA with Command Completion Signal**

0 (NORMAL) = Normal operation mode.

1 (COMPLETION) = This bit indicates that a completion signal is expected within a programmed amount of time (HSMCI_CSTOR).

- **BOOT_ACK: Boot Operation Acknowledge.**

The master can choose to receive the boot acknowledge from the slave when a Boot Request command is issued. When set to one this field indicates that a Boot acknowledge is expected within a programmable amount of time defined with DTOMUL and DTOCYC fields located in the HSMCI_DTOR register. If the acknowledge pattern is not received then an acknowledge timeout error is raised. If the acknowledge pattern is corrupted then an acknowledge pattern error is set.

34.14.7 HSMCI Block Register

Name: HSMCI_BLKR

Address: 0xF0008018 (0), 0xF000C018 (1)

Access: Read-write

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

- **BCNT: MMC/SDIO Block Count - SDIO Byte Count**

This field determines the number of data byte(s) or block(s) to transfer.

The transfer data type and the authorized values for BCNT field are determined by the TRTYP field in the HSMCI Command Register (HSMCI_CMDR).

When TRTYP=1 (MMC/SDCARD Multiple Block), BCNT can be programmed from 1 to 65535, 0 corresponds to an infinite block transfer.

When TRTYP=4 (SDIO Byte), BCNT can be programmed from 1 to 511, 0 corresponds to 512-byte transfer. Values in range 512 to 65536 are forbidden.

When TRTYP=5 (SDIO Block), BCNT can be programmed from 1 to 511, 0 corresponds to an infinite block transfer. Values in range 512 to 65536 are forbidden.

Warning: In SDIO Byte and Block modes (TRTYP=4 or 5), writing the 7 last bits of BCNT field with a value which differs from 0 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 HSMCI Mode Register (HSMCI_MR).

Bits 16 and 17 must be set to 0 if FBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

34.14.8 HSMCI Completion Signal Timeout Register

Name: HSMCI_CSTOR

Address: 0xF000801C (0), 0xF000C01C (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “H SMCI Write Protect Mode Register” on page 657.

- **CSTOCYC: Completion Signal Timeout Cycle Number**

These fields determine the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

- **CSTOMUL: Completion Signal Timeout Multiplier**

These fields determine the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

These fields determine the maximum number of Master Clock cycles that the HSMCI waits between the end of the data transfer and the assertion of the completion signal. The data transfer comprises data phase and the optional busy phase. If a non-DATA ATA command is issued, the HSMCI starts waiting immediately after the end of the response until the completion signal.

Multiplier is defined by CSTOMUL as shown in the following table:

Value	Name	Description
0	1	CSTOCYC x 1
1	16	CSTOCYC x 16
2	128	CSTOCYC x 128
3	256	CSTOCYC x 256
4	1024	CSTOCYC x 1024
5	4096	CSTOCYC x 4096
6	65536	CSTOCYC x 65536
7	1048576	CSTOCYC x 1048576

If the data time-out set by CSTOCYC and CSTOMUL has been exceeded, the Completion Signal Time-out Error flag (CSTOE) in the HSMCI Status Register (HSMCI_SR) rises.

34.14.9 HSMCI Response Register

Name: HSMCI_RSPR

Address: 0xF0008020 (0), 0xF000C020 (1)

Access: Read-only

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

- **RSP: Response**

Note: 1. The response register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C).
N depends on the size of the response.

34.14.10 HSMCI Receive Data Register

Name: HSMCI_RDR

Address: 0xF0008030 (0), 0xF000C030 (1)

Access: Read-only

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

- **DATA: Data to Read**

34.14.11 HSMCI Transmit Data Register

Name: HSMCI_TDR

Address: 0xF0008034 (0), 0xF000C034 (1)

Access: Write-only

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

- **DATA: Data to Write**

34.14.12 HSMCI Status Register

Name: HSMCI_SR

Address: 0xF0008040 (0), 0xF000C040 (1)

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	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 HSMCI_CMDR.

- **RXRDY: Receiver Ready**

0 = Data has not yet been received since the last read of HSMCI_RDR.

1 = Data has been received since the last read of HSMCI_RDR.

- **TXRDY: Transmit Ready**

0= The last data written in HSMCI_TDR has not yet been transferred in the Shift Register.

1= The last data written in HSMCI_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 HSMCI_SR.

1 = A data block transfer has ended, including the CRC16 Status transmission.
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: HSMCI Not Busy**

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.

Refer to the MMC or SD Specification for more details concerning the busy behavior.

For all the read operations, the NOTBUSY flag is cleared at the end of the host command.
For the Infinite Read Multiple Blocks, the NOTBUSY flag is set at the end of the STOP_TRANSMISSION host command (CMD12).

For the Single Block Reads, the NOTBUSY flag is set at the end of the data read block.
For the Multiple Block Reads with pre-defined block count, the NOTBUSY flag is set at the end of the last received data block.

The NOTBUSY flag allows to deal with these different states.

0 = The HSMCI is not ready for new data transfer. Cleared at the end of the card response.

1 = The HSMCI 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.

- **SDIOIRQA: SDIO Interrupt for Slot A**

0 = No interrupt detected on SDIO Slot A.

1 = An SDIO Interrupt on Slot A occurred. Cleared when reading the HSMCI_SR.

- **SDIOWAIT: SDIO Read Wait Operation Status**

0 = Normal Bus operation.

1 = The data bus has entered IO wait state.

- **CSRCV: CE-ATA Completion Signal Received**

0 = No completion signal received since last status read operation.

1 = The device has issued a command completion signal on the command line. Cleared by reading in the HSMCI_SR register.

- **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 HSMCI_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 HSMCI_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 HSMCI_CMDR.

- **RTOE: Response Time-out Error**

0 = No error.

1 = The response time-out set by MAXLAT in the HSMCI_CMDR has been exceeded. Cleared when writing in the HSMCI_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 HSMCI_SR register.

- **DTOE: Data Time-out Error**

0 = No error.

1 = The data time-out set by DTOCYC and DTOMUL in HSMCI_DTOR has been exceeded. Cleared by reading in the HSMCI_SR register.

- **CSTOE: Completion Signal Time-out Error**

0 = No error.

1 = The completion signal time-out set by CSTOCYC and CSTOMUL in HSMCI_CSTOR has been exceeded. Cleared by reading in the HSMCI_SR register. Cleared by reading in the HSMCI_SR register.

- **BLKOVRE: DMA Block Overrun Error**

0 = No error.

1 = A new block of data is received and the DMA controller has not started to move the current pending block, a block overrun is raised. Cleared by reading in the HSMCI_SR register.

- **DMADONE: DMA Transfer done**

0 = DMA buffer transfer has not completed since the last read of the HSMCI_SR register.

1 = DMA buffer transfer has completed.

- **FIFOEMPTY: FIFO empty flag**

0 = FIFO contains at least one byte.

1 = FIFO is empty.

- **XFRDONE: Transfer Done flag**

0 = A transfer is in progress.

1 = Command Register is ready to operate and the data bus is in the idle state.

- **ACKRCV: Boot Operation Acknowledge Received**

0 = No Boot acknowledge received since the last read of the status register.

1 = A Boot acknowledge signal has been received. Cleared by reading the HSMCI_SR register.

- **ACKRCVE: Boot Operation Acknowledge Error**

0 = No error

1 = Corrupted Boot Acknowledge signal received.

- **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.

When FERRCTRL in HSMCI_CFG is set to 1, OVRE becomes reset after read.

- **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 or when setting FERRCTRL in HSMCI_CFG to 1.

When FERRCTRL in HSMCI_CFG is set to 1, UNRE becomes reset after read.

34.14.13 HSMCI Interrupt Enable Register

Name: HSMCI_IER

Address: 0xF0008044 (0), 0xF000C044 (1)

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	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**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Enable**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Enable**
- **CSRCV: Completion Signal Received 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**
- **CSTOE: Completion Signal Timeout Error Interrupt Enable**
- **BLKOVRE: DMA Block Overrun Error Interrupt Enable**
- **DMADONE: DMA Transfer completed Interrupt Enable**

- **FIFOEMPTY: FIFO empty Interrupt enable**
- **XFRDONE: Transfer Done Interrupt enable**
- **ACKRCV: Boot Acknowledge Interrupt Enable**
- **ACKRCVE: Boot Acknowledge Error Interrupt Enable**
- **OVRE: Overrun Interrupt Enable**
- **UNRE: Underrun Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

34.14.14 HSMCI Interrupt Disable Register

Name: HSMCI_IDR

Address: 0xF0008048 (0), 0xF000C048 (1)

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	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**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Disable**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Disable**
- **CSRCV: Completion Signal received 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**
- **CSTOE: Completion Signal Time out Error Interrupt Disable**
- **BLKOVRE: DMA Block Overrun Error Interrupt Disable**
- **DMADONE: DMA Transfer completed Interrupt Disable**

- **FIFOEMPTY: FIFO empty Interrupt Disable**
- **XFRDONE: Transfer Done Interrupt Disable**
- **ACKRCV: Boot Acknowledge Interrupt Disable**
- **ACKRCVE: Boot Acknowledge Error Interrupt Disable**
- **OVRE: Overrun Interrupt Disable**
- **UNRE: Underrun Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

34.14.15 HSMCI Interrupt Mask Register

Name: HSMCI_IMR

Address: 0xF000804C (0), 0xF000C04C (1)

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	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**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Mask**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Mask**
- **CSRCV: Completion Signal Received 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**
- **CSTOE: Completion Signal Time-out Error Interrupt Mask**
- **BLKOVRE: DMA Block Overrun Error Interrupt Mask**
- **DMADONE: DMA Transfer Completed Interrupt Mask**

- **FIFOEMPTY: FIFO Empty Interrupt Mask**
- **XFRDONE: Transfer Done Interrupt Mask**
- **ACKRCV: Boot Operation Acknowledge Received Interrupt Mask**
- **ACKRCVE: Boot Operation Acknowledge Error Interrupt Mask**
- **OVRE: Overrun Interrupt Mask**
- **UNRE: Underrun Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

34.14.16 HSMCI DMA Configuration Register

Name: HSMCI_DMA

Address: 0xF0008050 (0), 0xF000C050 (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “H SMCI Write Protect Mode Register” on page 657.

- **OFFSET: DMA Write Buffer Offset**

This field indicates the number of discarded bytes when the DMA writes the first word of the transfer.

- **CHKSIZE: DMA Channel Read and Write Chunk Size**

The CHKSIZE field indicates the number of data available when the DMA chunk transfer request is asserted.

Value	Name	Description
0	1	1 data available
1	4	4 data available
2	8	8 data available
3	16	16 data available
–	–	Reserved

- **DMAEN: DMA Hardware Handshaking Enable**

0 = DMA interface is disabled.

1 = DMA Interface is enabled.

Note: To avoid unpredictable behavior, DMA hardware handshaking must be disabled when CPU transfers are performed.

- **ROPT: Read Optimization with padding**

0: BLKLEN bytes are moved from the Memory Card to the system memory, two DMA descriptors are used when the transfer size is not a multiple of 4.

1: Ceiling(BLKLEN/4) * 4 bytes are moved from the Memory Card to the system memory, only one DMA descriptor is used.

34.14.17 HSMCI Configuration Register

Name: HSMCI_CFG

Address: 0xF0008054 (0), 0xF000C054 (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “H SMCI Write Protect Mode Register” on page 657.

- **FIFOMODE: HSMCI Internal FIFO control mode**

0 = A write transfer starts when a sufficient amount of data is written into the FIFO.

When the block length is greater than or equal to 3/4 of the HSMCI internal FIFO size, then the write transfer starts as soon as half the FIFO is filled. When the block length is greater than or equal to half the internal FIFO size, then the write transfer starts as soon as one quarter of the FIFO is filled. In other cases, the transfer starts as soon as the total amount of data is written in the internal FIFO.

1 = A write transfer starts as soon as one data is written into the FIFO.

- **FERRCTRL: Flow Error flag reset control mode**

0= When an underflow/overflow condition flag is set, a new Write/Read command is needed to reset the flag.

1= When an underflow/overflow condition flag is set, a read status resets the flag.

- **HSMODE: High Speed Mode**

0= Default bus timing mode.

1= If set to one, the host controller outputs command line and data lines on the rising edge of the card clock. The Host driver shall check the high speed support in the card registers.

- **LSYNC: Synchronize on the last block**

0= The pending command is sent at the end of the current data block.

1= The pending command is sent at the end of the block transfer when the transfer length is not infinite. (block count shall be different from zero)

34.14.18H SMCI Write Protect Mode Register

Name: HSMCI_WPMR

Address: 0xF00080E4 (0), 0xF000C0E4 (1)

Access: Read-write

31	30	29	28	27	26	25	24
WP_KEY (0x4D => “M”)							
23	22	21	20	19	18	17	16
WP_KEY (0x43 => C”)							
15	14	13	12	11	10	9	8
WP_KEY (0x49 => “I”)							
7	6	5	4	3	2	1	0
							WP_EN

- **WP_EN: Write Protection Enable**

0 = Disables the Write Protection if WP_KEY corresponds to 0x4D4349 (“MCI’ in ASCII).

1 = Enables the Write Protection if WP_KEY corresponds to 0x4D4349 (“MCI’ in ASCII).

- **WP_KEY: Write Protection Key password**

Should be written at value **0x4D4349** (ASCII code for “MCI”). Writing any other value in this field has no effect.

Protects the registers:

- “HSMCI Mode Register” on page 635
- “HSMCI Data Timeout Register” on page 636
- “HSMCI SDCard/SDIO Register” on page 637
- “HSMCI Completion Signal Timeout Register” on page 642
- “HSMCI DMA Configuration Register” on page 655
- “HSMCI Configuration Register” on page 656

34.14.19HSMCI Write Protect Status Register

Name: HSMCI_WPSR

Address: 0xF00080E8 (0), 0xF000C0E8 (1)

Access: Read-only

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

- **WP_VS: Write Protection Violation Status**

Value	Name	Description
0	NONE	No Write Protection Violation occurred since the last read of this register (WP_SR)
1	WRITE	Write Protection detected unauthorized attempt to write a control register had occurred (since the last read.)
2	RESET	Software reset had been performed while Write Protection was enabled (since the last read).
3	BOTH	Both Write Protection violation and software reset with Write Protection enabled have occurred since the last read.

- **WP_VSRC: Write Protection Violation SouRCe**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

34.14.20HSMCI FIFOx Memory Aperture

Name: HSMCI_FIFOx[x=0..255]

Address: 0xF0008200 (0), 0xF000C200 (1)

Access: Read-write

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 or Data to Write**

35. Serial Peripheral Interface (SPI)

35.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves" which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

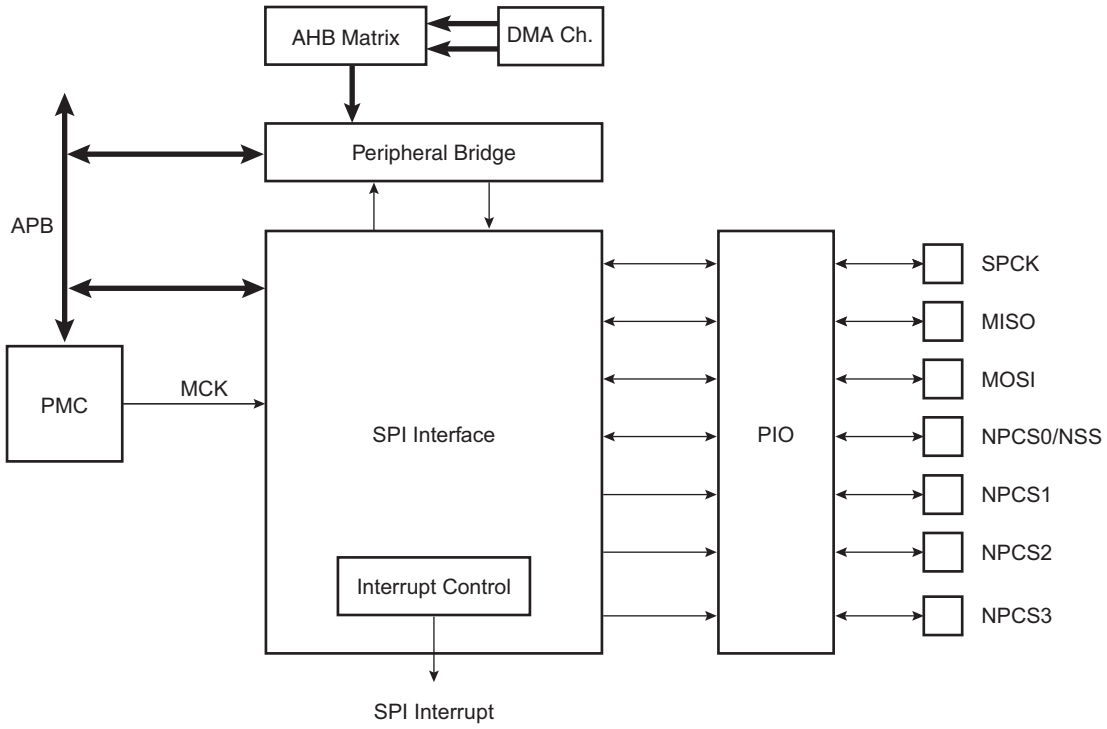
- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

35.2 Embedded Characteristics

- Supports Communication with Serial External Devices
  - Master Mode can drive SPCK up to peripheral clock (bounded by maximum bus clock divided by 2)
  - Slave Mode operates on SPCK, asynchronously to Core and Bus ClockFour Chip Selects with External Decoder Support Allow Communication with Up to 15 Peripherals
  - 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 Coprocessors
- Master or Slave Serial Peripheral Bus Interface
  - 8-bit to 16-bit Programmable Data Length Per Chip Select
  - Programmable Phase and Polarity Per Chip Select
  - Programmable Transfer Delay Between Consecutive Transfers and Delay before SPI Clock per Chip Select
  - Programmable Delay Between Chip Selects
  - Selectable Mode Fault Detection
- Connection to DMA Channel Capabilities Optimizes Data Transfers
  - One channel for the Receiver, One Channel for the Transmitter

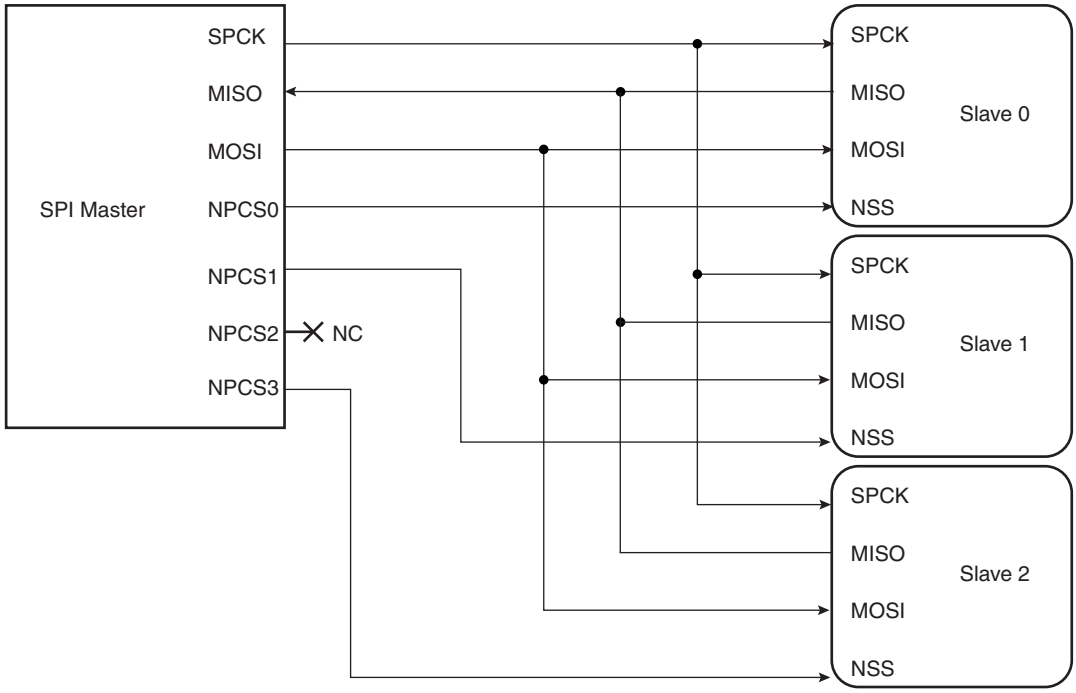
35.3 Block Diagram

Figure 35-1. Block Diagram



35.4 Application Block Diagram

Figure 35-2. Application Block Diagram: Single Master/Multiple Slave Implementation



35.5 Signal Description

Table 35-1. 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

35.6 Product Dependencies

35.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

Table 35-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_MISO	PA11	A
SPI0	SPI0_MOSI	PA12	A

Table 35-2. I/O Lines

SPI0	SPI0_NPCS0	PA14	A
SPI0	SPI0_NPCS1	PA7	B
SPI0	SPI0_NPCS2	PA1	B
SPI0	SPI0_NPCS3	PB3	B
SPI0	SPI0_SPCK	PA13	A
SPI1	SPI1_MISO	PA21	B
SPI1	SPI1_MOSI	PA22	B
SPI1	SPI1_NPCS0	PA8	B
SPI1	SPI1_NPCS1	PA0	B
SPI1	SPI1_NPCS2	PA31	B
SPI1	SPI1_NPCS3	PA30	B
SPI1	SPI1_SPCK	PA23	B

35.6.2 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.

35.6.3 Interrupt

The SPI interface has an interrupt line connected to the Interrupt Controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

Table 35-3. Peripheral IDs

Instance	ID
SPI0	13
SPI1	14

35.6.4 Direct Memory Access Controller (DMAC)

The SPI interface can be used in conjunction with the DMAC in order to reduce processor overhead. For a full description of the DMAC, refer to the corresponding section in the full datasheet.

35.7 Functional Description

35.7.1 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.

35.7.2 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.

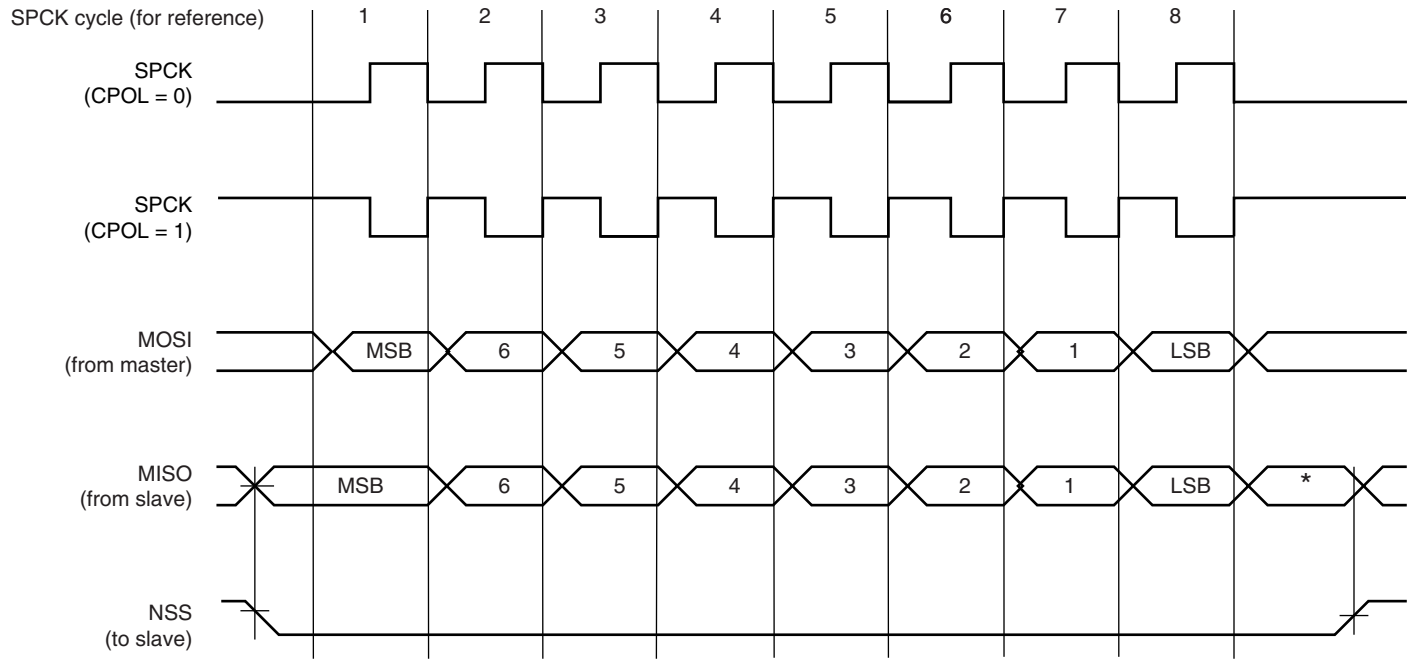
Table 35-4 shows the four modes and corresponding parameter settings.

Table 35-4. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA	Shift SPCK Edge	Capture SPCK Edge	SPCK Inactive Level
0	0	1	Falling	Rising	Low
1	0	0	Rising	Falling	Low
2	1	1	Rising	Falling	High
3	1	0	Falling	Rising	High

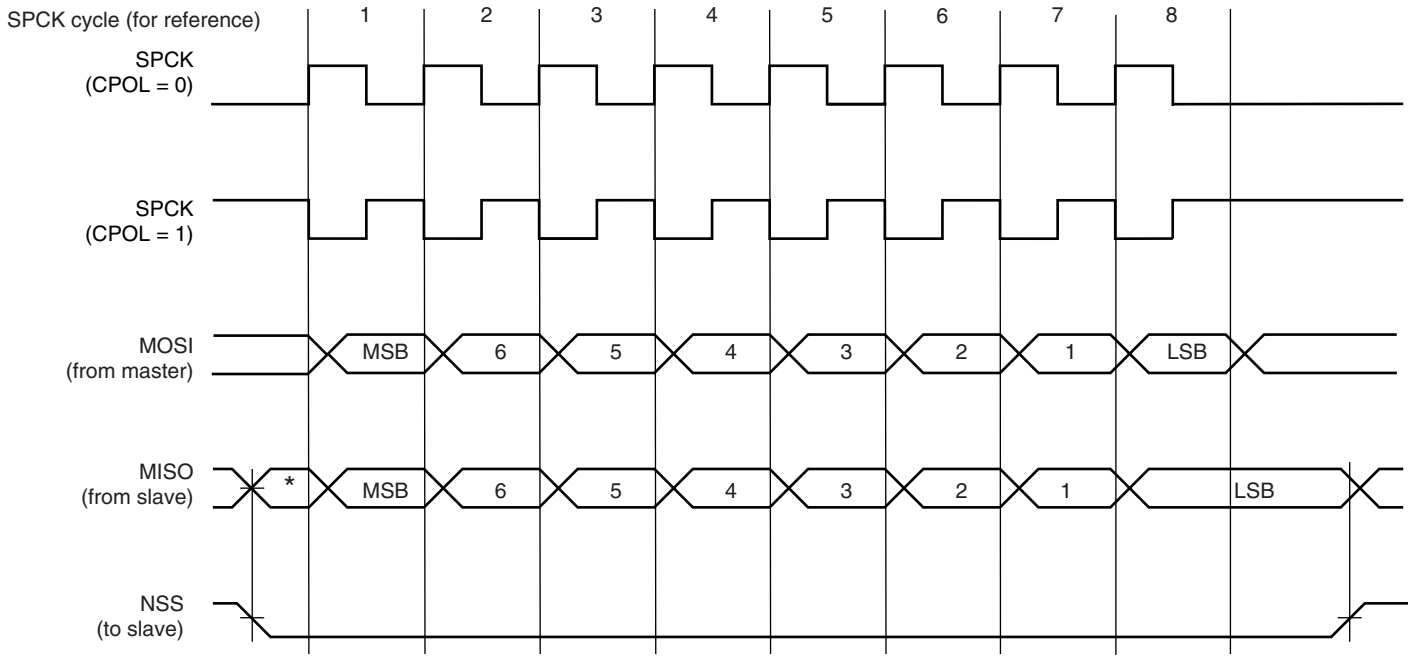
Figure 35-3 and Figure 35-4 show examples of data transfers.

Figure 35-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Figure 35-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

35.7.3 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. Receiving data cannot occur without transmitting data. If receiving mode is not needed, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the status register can be discarded.

Before writing the TDR, the PCS field in the SPI_MR register must be set in order to select a slave.

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 DMAchannel.

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.

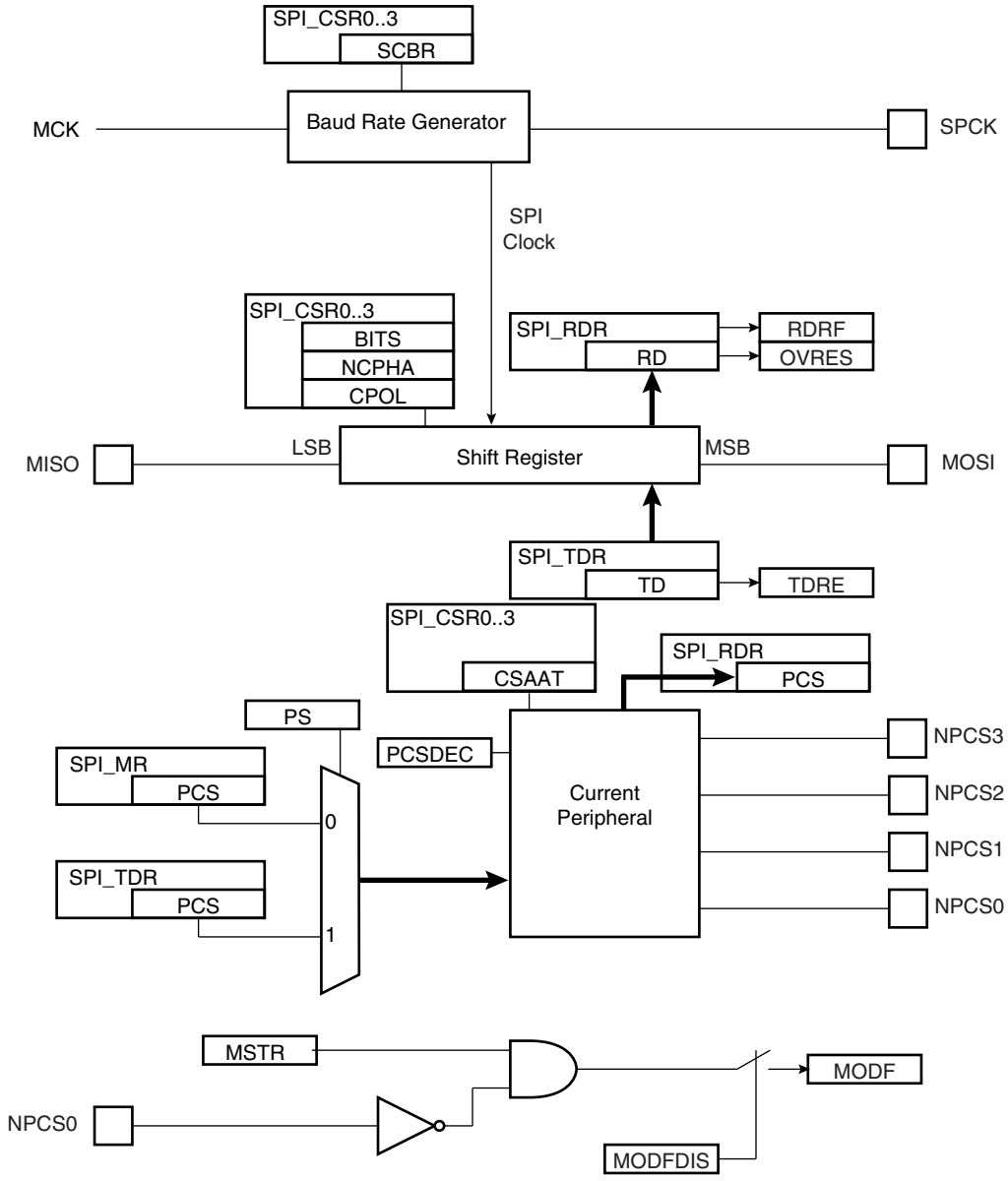
The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

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 35-5, shows a block diagram of the SPI when operating in Master Mode. Figure 35-6 on page 667 shows a flow chart describing how transfers are handled.

35.7.3.1 Master Mode Block Diagram

Figure 35-5. Master Mode Block Diagram



35.7.3.2 Master Mode Flow Diagram

Figure 35-6. Master Mode Flow Diagram

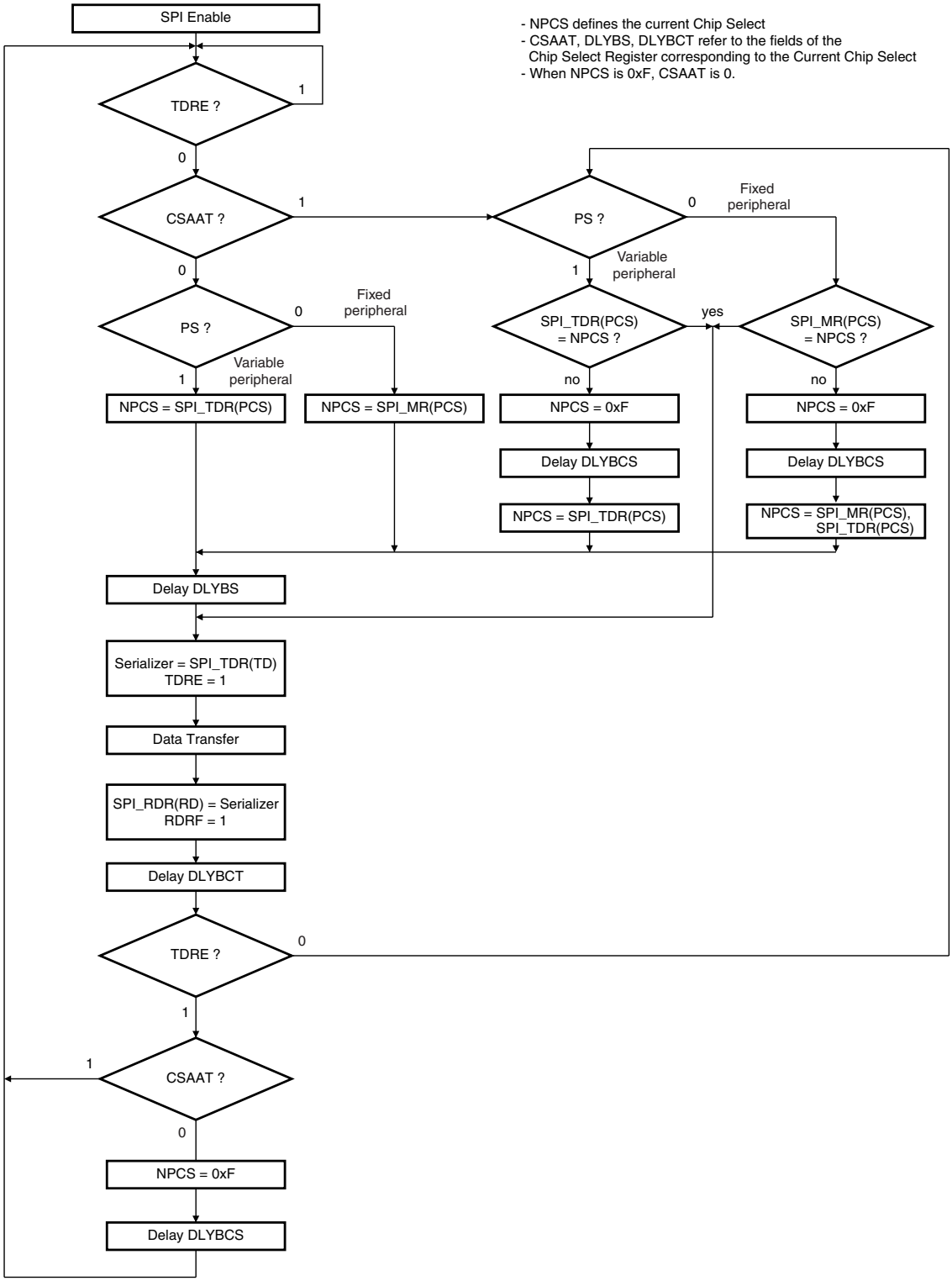
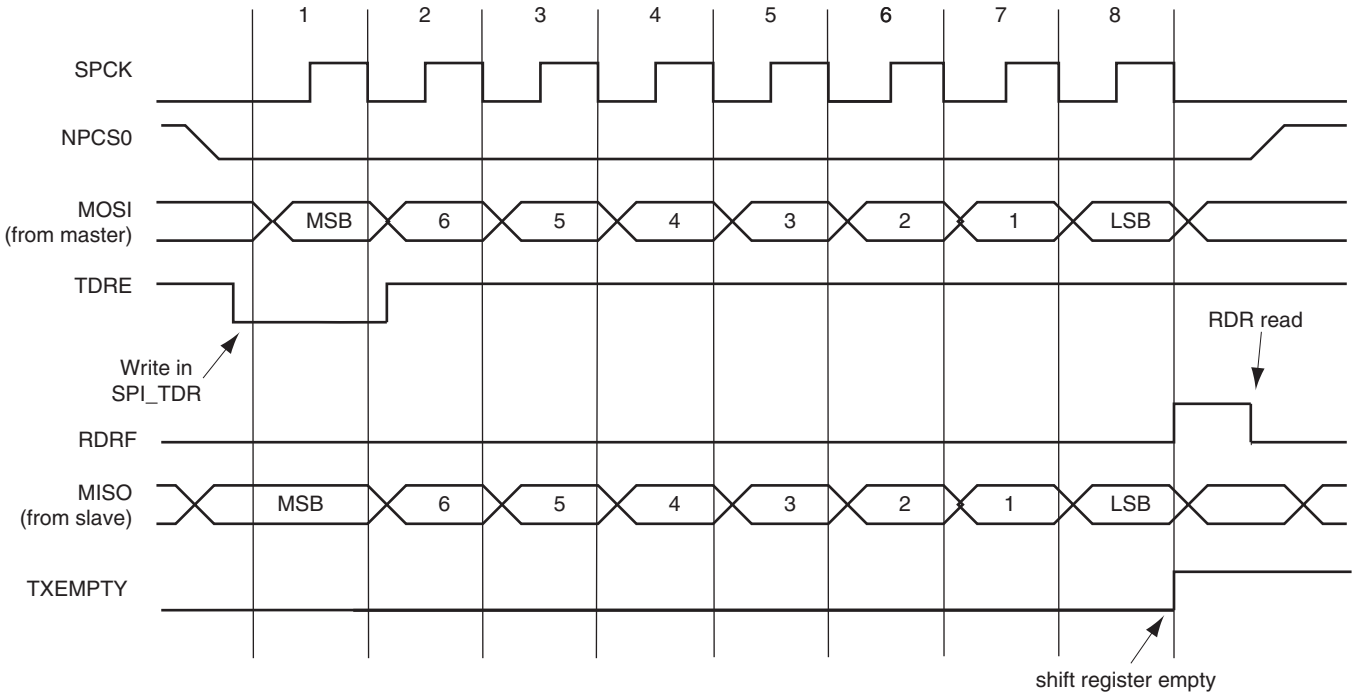


Figure 35-7 shows Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags behavior within the SPI_SR (Status Register) during an 8-bit data transfer in fixed mode and no Peripheral Data Controller involved.

Figure 35-7. Status Register Flags Behavior



35.7.3.3 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.

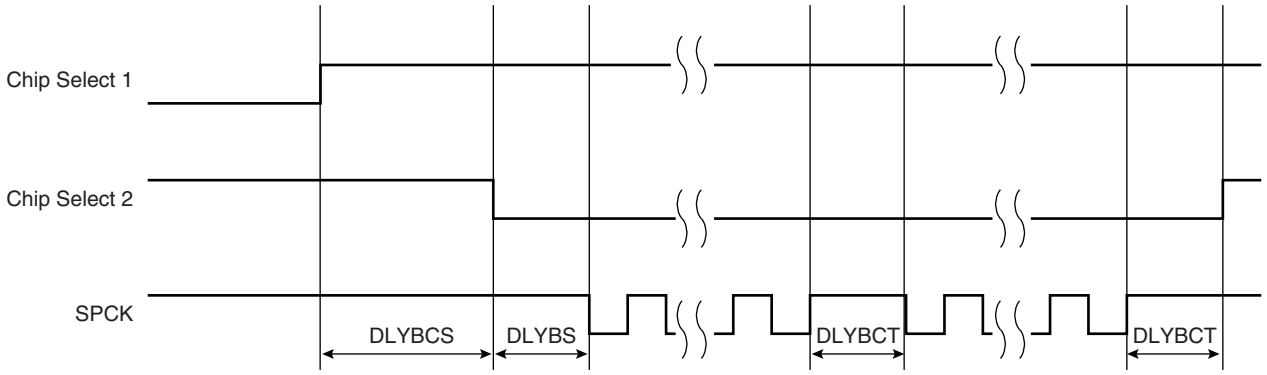
35.7.3.4 Transfer Delays

Figure 35-8 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 35-8. Programmable Delays



35.7.3.5 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.

- Fixed Peripheral Select: SPI exchanges data with only 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: Data can be exchanged with more than one peripheral without having to reprogram the NPCS field in the SPI_MR register.

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 value to write in the SPI_TDR register as the following format.

[xxxxxxx(7-bit) + LASTXFER(1-bit)[(1)]+ xxxx(4-bit) + PCS (4-bit) + DATA (8 to 16-bit)] with PCS equals to the chip select to assert as defined in Section 35.8.4 (SPI Transmit Data Register) and LASTXFER bit at 0 or 1 depending on CSAAT bit.

Note: 1. Optional.

CSAAT, LASTXFER bits are discussed in Section 35.7.3.9 ”Peripheral Deselection with DMAC”.

If LASTXFER is used, the command must be issued before writing the last character. Instead of LASTXFER, the user can use the SPIDIS command. After the end of the DMA transfer, wait for the TXEMPTY flag, then write SPIDIS into the SPI_CR register (this will not change the configuration register values); the NPCS will be deactivated after the last character transfer. Then, another DMA transfer can be started if the SPIEN was previously written in the SPI_CR register.

35.7.3.6 SPI Direct Access Memory Controller (DMAC)

In both fixed and variable mode the Direct Memory Access Controller (DMAC) can be used to reduce processor overhead.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the DMAC 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 DMAC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

35.7.3.7 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 1 of up to 16 decoder/demultiplexer. 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., one NPCS line driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

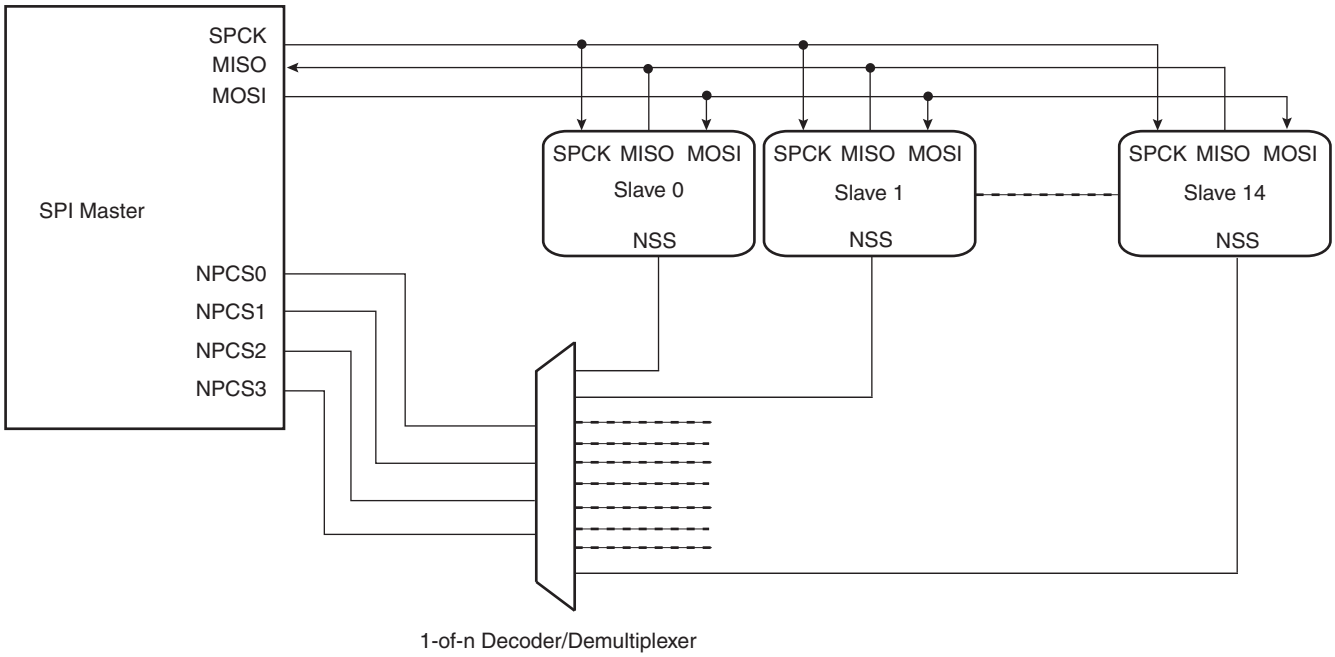
When operating with decoding, the SPI directly outputs the value defined by the PCS field on NPCS lines of either the Mode Register or the Transmit Data Register (depending on PS).

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. Figure 35-9 below shows such an implementation.

If the CSAAT bit is used, with or without the DMAC, the Mode Fault detection for NPCS0 line must be disabled. This is not needed for all other chip select lines since Mode Fault Detection is only on NPCS0.

Figure 35-9. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation



35.7.3.8 Peripheral Deselection without DMA

During a transfer of more than one data on a Chip Select without the DMA, the SPI_TDR is loaded by the processor, the flag TDRE rises as soon as the content of the SPI_TDR is transferred into the internal shift register. When this flag is detected high, the SPI_TDR can be reloaded. If this reload by the processor occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. But depending on the application software handling the SPI status register flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the SPI_TDR in time to keep the chip select active (low). A null Delay Between Consecutive Transfer (DLYBCT) value in the SPI_CSR register, will

give even less time for the processor to reload the SPI_TDR. With some SPI slave peripherals, requiring the chip select line to remain active (low) during a full set of transfers might lead to communication errors.

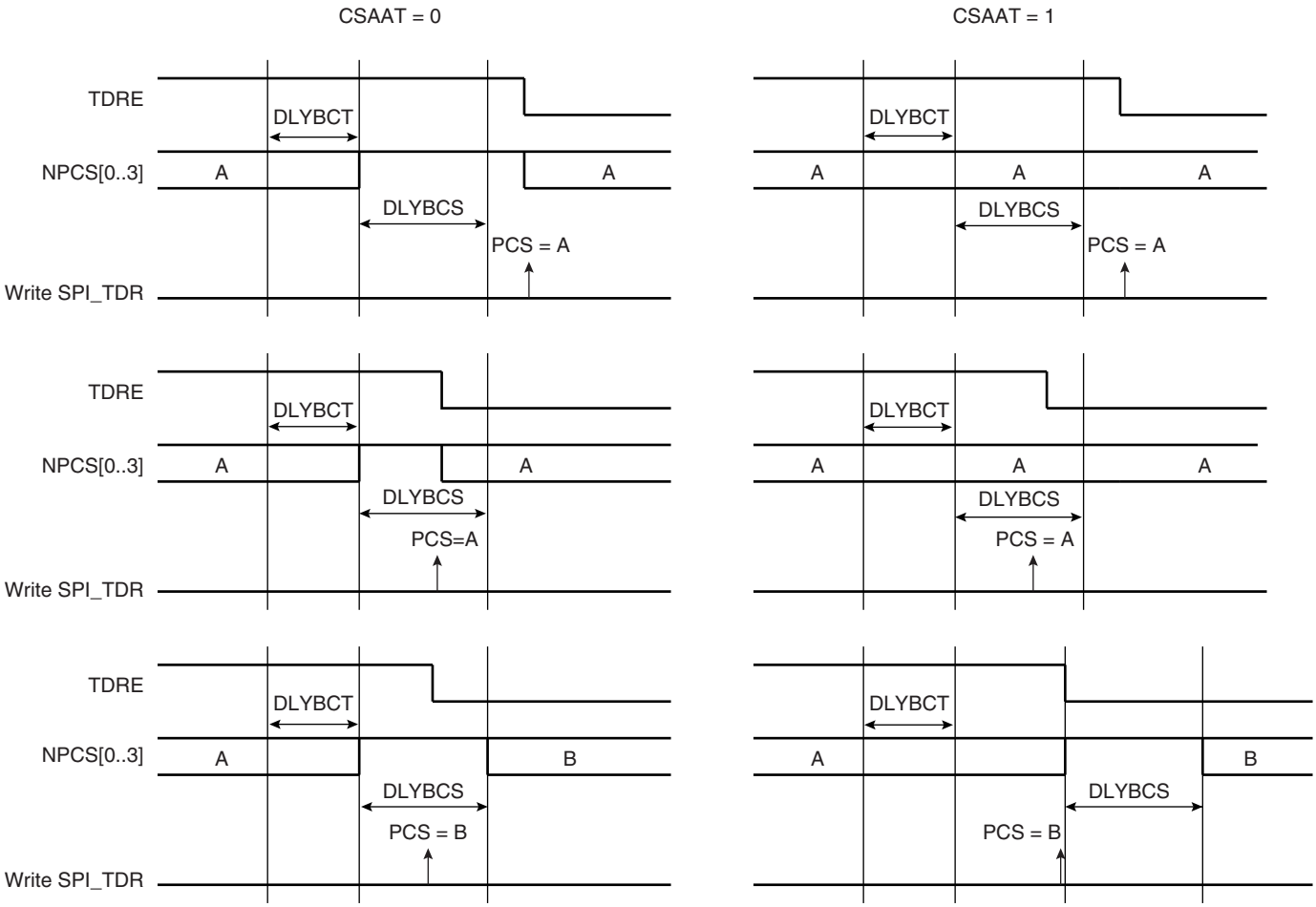
To facilitate interfacing with such devices, the Chip Select Register [CSR0...CSR3] can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another chip select is required. Even if the SPI_TDR is not reloaded the chip select will remain active. To have the chip select line to raise at the end of the transfer the Last transfer Bit (LASTXFER) in the SPI_MR register must be set at 1 before writing the last data to transmit into the SPI_TDR.

35.7.3.9 Peripheral Deselection with DMAC

When the Direct Memory Access Controller is used, the chip select line will remain low during the whole transfer since the TDRE flag is managed by the DMAC itself. The reloading of the SPI_TDR by the DMAC is done as soon as TDRE flag is set to one. In this case the use of CSAAT bit might not be needed. However, it may happen that when other DMAC channels connected to other peripherals are in use as well, the SPI DMAC might be delayed by another (DMAC with a higher priority on the bus). Having DMAC buffers in slower memories like flash memory or SDRAM compared to fast internal SRAM, may lengthen the reload time of the SPI_TDR by the DMAC as well. This means that the SPI_TDR might not be reloaded in time to keep the chip select line low. In this case the chip select line may toggle between data transfer and according to some SPI Slave devices, the communication might get lost. The use of the CSAAT bit might be needed.

Figure 35-10 shows different peripheral deselection cases and the effect of the CSAAT bit.

Figure 35-10.Peripheral Deselection



35.7.3.10 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. In this case, multi-master configuration, NPCS0, MOSI, MISO and SPCK pins must be configured in open drain (through the PIO controller). 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).

35.7.4 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 35.8.9 “SPI Chip Select Register” on page 684.)

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.

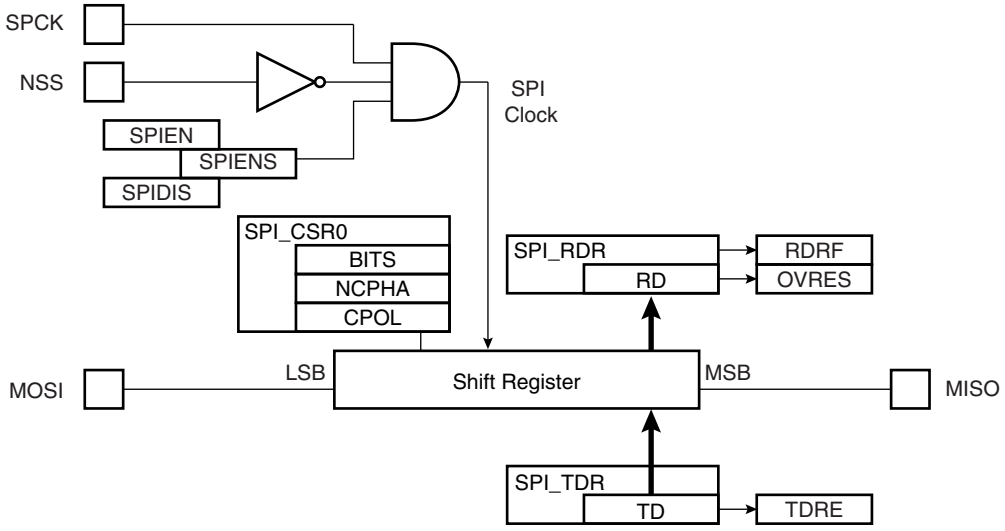
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets 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 35-11 shows a block diagram of the SPI when operating in Slave Mode.

Figure 35-11.Slave Mode Functional Bloc Diagram



35.7.5 Write Protected Registers

To prevent any single software error that may corrupt SPI behavior, the registers listed below can be write-protected by setting the WPEN bit in the SPI Write Protection Mode Register (SPI_WPMR).

If a write access in a write-protected register is detected, then the WPVS flag in the SPI Write Protection Status Register (SPI_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically reset after reading the SPI Write Protection Status Register (SPI_WPSR).

List of the write-protected registers:

Section 35.8.2 ”SPI Mode Register”

Section 35.8.9 ”SPI Chip Select Register”

35.8 Serial Peripheral Interface (SPI) User Interface

Table 35-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	---
0x04	Mode Register	SPI_MR	Read-write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	---
0x10	Status Register	SPI_SR	Read-only	0x000000F0
0x14	Interrupt Enable Register	SPI_IER	Write-only	---
0x18	Interrupt Disable Register	SPI_IDR	Write-only	---
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20 - 0x2C	Reserved			
0x30	Chip Select Register 0	SPI_CSR0	Read-write	0x0
0x34	Chip Select Register 1	SPI_CSR1	Read-write	0x0
0x38	Chip Select Register 2	SPI_CSR2	Read-write	0x0
0x3C	Chip Select Register 3	SPI_CSR3	Read-write	0x0
0x4C - 0xE0	Reserved	–	–	–
0xE4	Write Protection Control Register	SPI_WPMR	Read-write	0x0
0xE8	Write Protection Status Register	SPI_WPSR	Read-only	0x0
0x00E8 - 0x00F8	Reserved	–	–	–
0x00FC	Reserved	–	–	–

35.8.1 SPI Control Register

Name: SPI_CR

Address: 0xF0000000 (0), 0xF0004000 (1)

Access: Write-only

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

- **SPIEN: SPI Enable**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0 = No effect.

1 = Disables the SPI.

As soon as SPIDIS is set, SPI finishes its transfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in slave mode after software reset.

- **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.

Refer to Section 35.7.3.5 ”Peripheral Selection”for more details.

35.8.2 SPI Mode Register

Name: SPI_MR

Address: 0xF0000004 (0), 0xF0004004 (1)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in "SPI Write Protection Mode Register".

- **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.

- **WDRBT: Wait Data Read Before Transfer**

0 = No Effect. In master mode, a transfer can be initiated whatever the state of the Receive Data Register is.

1 = In Master Mode, a transfer can start only if the Receive Data Register is empty, i.e. does not contain any unread data. This mode prevents overrun error in reception.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled.

LLB controls the local loopback on the data serializer for testing in Master Mode only. (MISO is internally connected on MOSI.)

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don’t care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is less than or equal to six, six MCK periods will be inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

35.8.3 SPI Receive Data Register

Name: SPI_RDR

Address: 0xF0000008 (0), 0xF0004008 (1)

Access: Read-only

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

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

Note: When using variable peripheral select mode (PS = 1 in SPI_MR) it is mandatory to also set the WDRBT field to 1 if the SPI_RDR PCS field is to be processed.

35.8.4 SPI Transmit Data Register

Name: SPI_TDR

Address: 0xF000000C (0), 0xF000400C (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS: Peripheral Chip Select**

This field is only used if Variable Peripheral Select is active (PS = 1).

If PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don’t care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

This field is only used if Variable Peripheral Select is active (PS = 1).

35.8.5 SPI Status Register

Name: SPI_SR

Address: 0xF0000010 (0), 0xF0004010 (1)

Access: Read-only

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

- **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.

35.8.6 SPI Interrupt Enable Register

Name: SPI_IER

Address: 0xF0000014 (0), 0xF0004014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	–	–	–	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**
- **NSSR: NSS Rising Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**

35.8.7 SPI Interrupt Disable Register

Name: SPI_IDR

Address: 0xF0000018 (0), 0xF0004018 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	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**
- **NSSR: NSS Rising Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**

35.8.8 SPI Interrupt Mask Register

Name: SPI_IMR

Address: 0xF000001C (0), 0xF000401C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	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**
- **NSSR: NSS Rising Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**

35.8.9 SPI Chip Select Register

Name: SPI_CSRx[x=0..3]

Address: 0xF0000030 (0), 0xF0004030 (1)

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in "SPI Write Protection Mode Register".

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 35.8.9 "SPI Chip Select Register" on page 684.)

The BITS field determines the number of data bits transferred. Reserved values should not be used.

Value	Name	Description
0	8_BIT	8 bits for transfer
1	9_BIT	9 bits for transfer
2	10_BIT	10 bits for transfer
3	11_BIT	11 bits for transfer
4	12_BIT	12 bits for transfer
5	13_BIT	13 bits for transfer
6	14_BIT	14 bits for transfer
7	15_BIT	15 bits for transfer

Value	Name	Description
8	16_BIT	16 bits for transfer
9	–	Reserved
10	–	Reserved
11	–	Reserved
12	–	Reserved
13	–	Reserved
14	–	Reserved
15	–	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.

Note: If one of the SCBR fields inSPI_CSRx is set to 1, the other SCBR fields in SPI_CSRx must be set to 1 as well, if they are required to process transfers. If they are not used to transfer data, they can be set at any value.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS valid to the first valid SPCK transition.

When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

Otherwise, the following equations determine the delay:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

35.8.10 SPI Write Protection Mode Register

Name: SPI_WPMR

Address: 0xF00000E4 (0), 0xF00040E4 (1)

Access: Read-write

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

- **WPEN: Write Protection Enable**

0: The Write Protection is Disabled

1: The Write Protection is Enabled

- **WPKEY: Write Protection Key Password**

If a value is written in WPEN, the value is taken into account only if WPKEY is written with “SPI” (SPI written in ASCII Code, ie 0x535049 in hexadecimal).

List of the write-protected registers:

Section 35.8.2 ”SPI Mode Register”

Section 35.8.9 ”SPI Chip Select Register”

35.8.11 SPI Write Protection Status Register

Name: SPI_WPSR

Address: 0xF00000E8 (0), 0xF00040E8 (1)

Access: Read-only

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

- **WPVS: Write Protection Violation Status**

0 = No Write Protect Violation has occurred since the last read of the SPI_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the SPI_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

This Field indicates the APB Offset of the register concerned by the violation (SPI_MR or SPI_CSRx)

36. Timer Counter (TC)

36.1 Description

The Timer Counter (TC) includes six identical 32-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 TC channels.

The Block Control Register allows the channels to be started simultaneously with the same instruction.

The Block Mode Register defines the external clock inputs for each channel, allowing them to be chained.

Table 36-1 gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2.

Table 36-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5(1)	SLCK

Note: 1. When Slow Clock is selected for Master Clock (CSS = 0 in PMC Master Clock Register), TIMER_CLOCK5 input is equivalent to Master Clock.

36.2 Embedded Characteristics

- Provides six 32-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 acting as trigger event
- Internal interrupt signal
- Two global registers that act on all TC channels

36.3 Block Diagram

Figure 36-1. Timer Counter Block Diagram

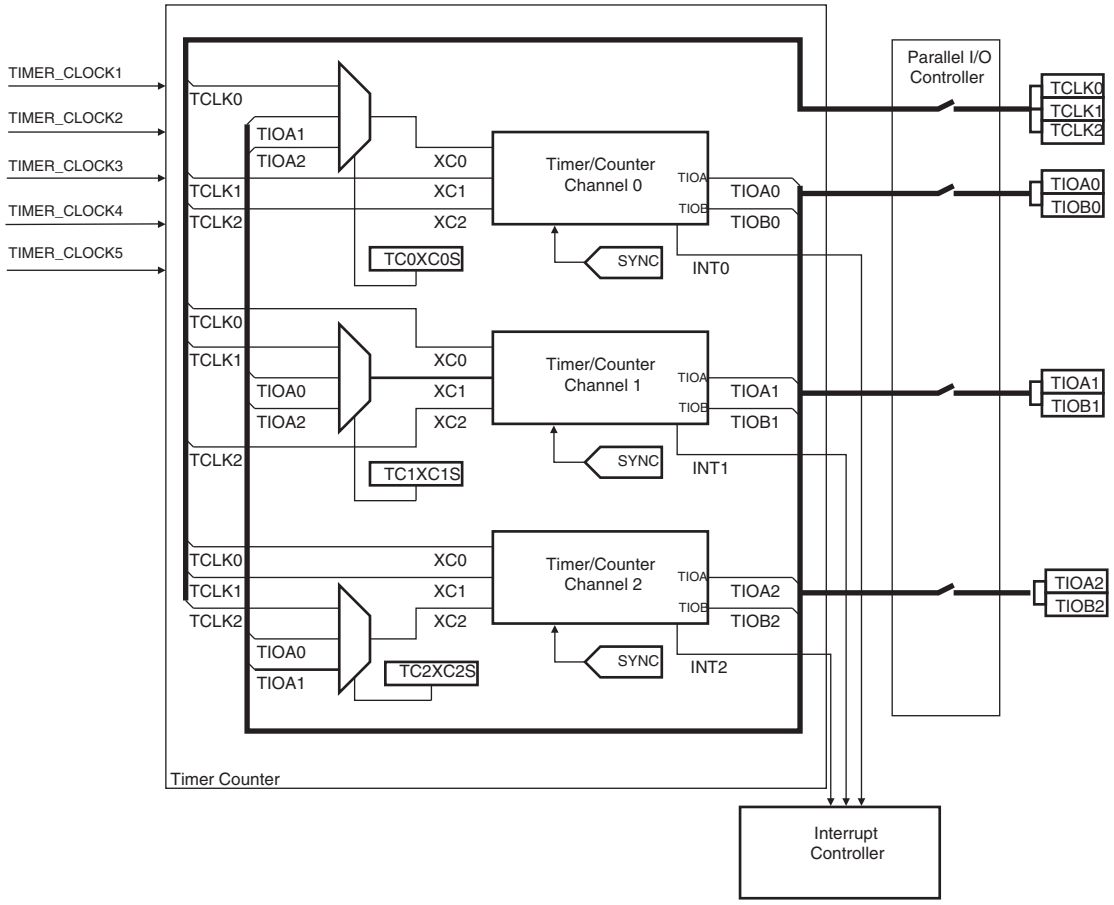


Table 36-2. Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output (internal signal)
	SYNC	Synchronization Input Signal (from configuration register)

36.4 Pin Name List

Table 36-3. TC pin list

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

36.5 Product Dependencies

36.5.1 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 TC pins to their peripheral functions.

Table 36-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TC0	TCLK0	PA24	A
TC0	TCLK1	PA25	A
TC0	TCLK2	PA26	A
TC0	TIOA0	PA21	A
TC0	TIOA1	PA22	A
TC0	TIOA2	PA23	A
TC0	TIOB0	PA27	A
TC0	TIOB1	PA28	A
TC0	TIOB2	PA29	A
TC1	TCLK3	PC4	C
TC1	TCLK4	PC7	C
TC1	TCLK5	PC14	C
TC1	TIOA3	PC2	C
TC1	TIOA4	PC5	C
TC1	TIOA5	PC12	C
TC1	TIOB3	PC3	C
TC1	TIOB4	PC6	C
TC1	TIOB5	PC13	C

36.5.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer Counter clock.

36.5.3 Interrupt

The TC has an interrupt line connected to the Interrupt Controller (IC). Handling the TC interrupt requires programming the IC before configuring the TC.

36.6 Functional Description

36.6.1 TC Description

The six channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 36-5 on page 703.

36.6.2 32-bit Counter

Each channel is organized around a 32-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the COVFS bit in TC_SR (Status Register) is set.

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

36.6.3 Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the internal I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC_BMR (Block Mode). See Figure 36-2 ”Clock Chaining Selection”.

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

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5
- External clock signals: XC0, XC1 or XC2

This selection is made by the TCCLKS bits in the TC Channel Mode Register.

The selected clock can be inverted with the CLKI bit in TC_CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2). See Figure 36-3 ”Clock Selection”

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the master clock period. The external clock frequency must be at least 2.5 times lower than the master clock

Figure 36-2. Clock Chaining Selection

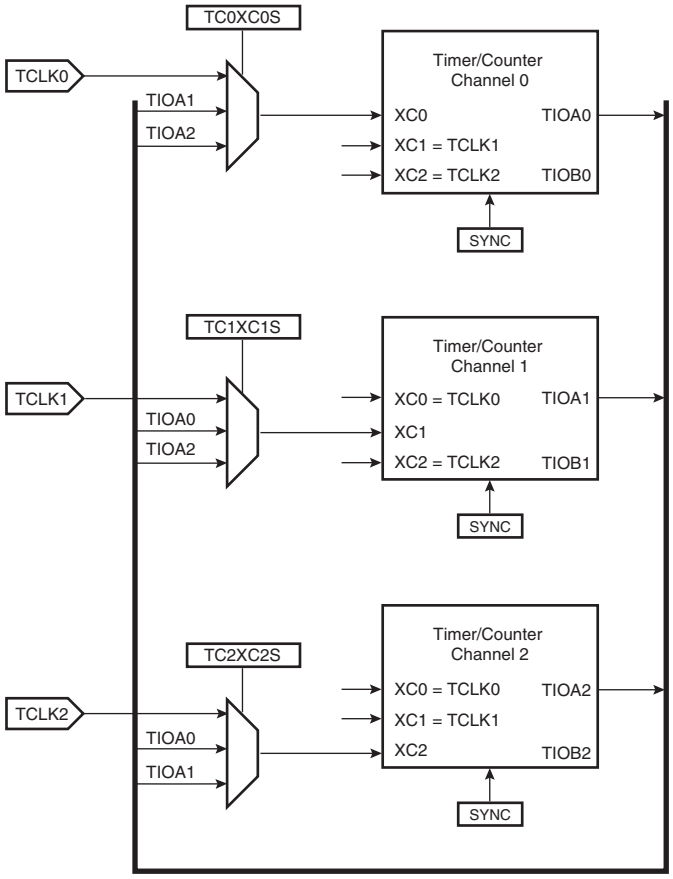
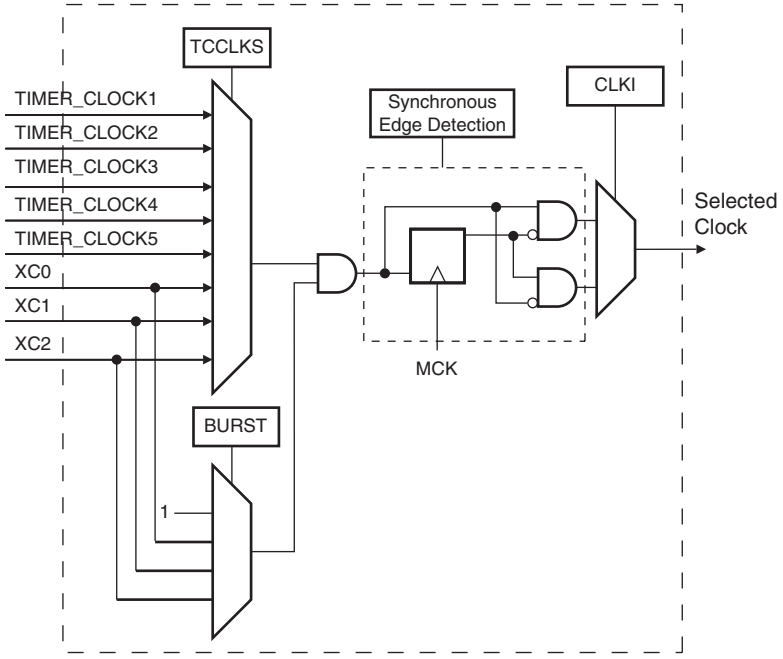


Figure 36-3. Clock Selection

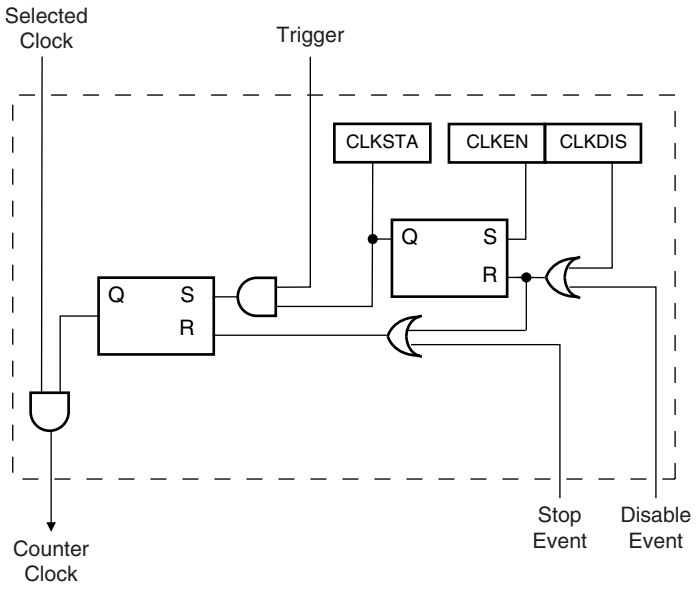


36.6.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 36-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the Control Register. In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in TC_CMR. In Waveform Mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the Control Register can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the Status Register.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture Mode (LDBSTOP = 1 in TC_CMR) or a RC compare event in Waveform Mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands have effect only if the clock is enabled.

Figure 36-4. Clock Control



36.6.5 TC Operating Modes

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register.

In Capture Mode, TIOA and TIOB are configured as inputs.

In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

36.6.6 Trigger

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

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

The following triggers are common to both modes:

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

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

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

36.6.7 Capture Operating Mode

This mode is entered by clearing the WAVE parameter in TC_CMR (Channel Mode Register).

Capture Mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 36-5 shows the configuration of the TC channel when programmed in Capture Mode.

36.6.8 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 selected edge for the loading of register A, and the LDRB parameter defines the TIOA selected edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

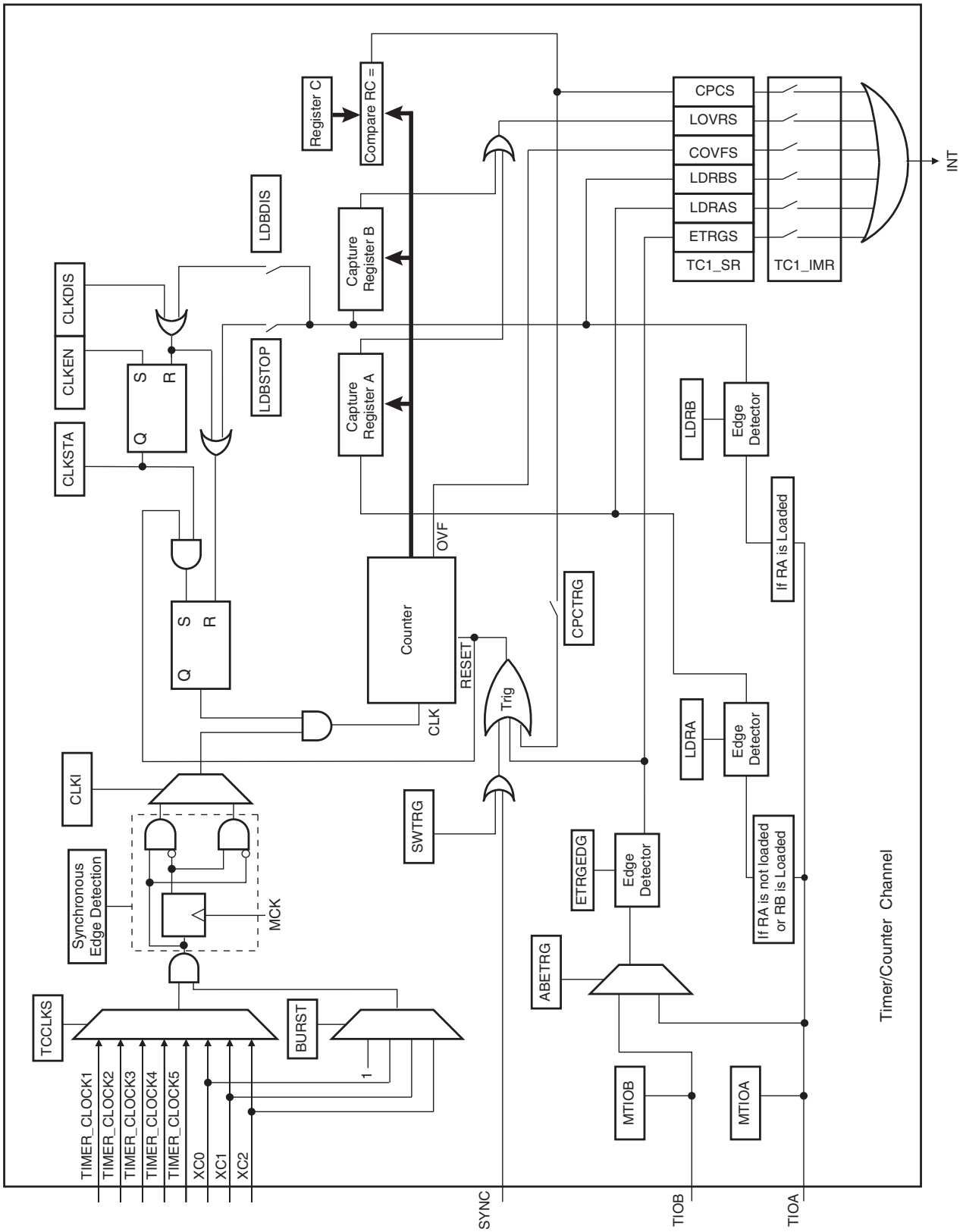
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

36.6.9 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 the TC_CMR register selects TIOA or TIOB input signal as an external trigger . The ETRGEDG parameter defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Figure 36-5. Capture Mode



36.6.10 Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in TC_CMR (Channel Mode Register).

In Waveform Operating Mode the TC channel generates 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 36-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

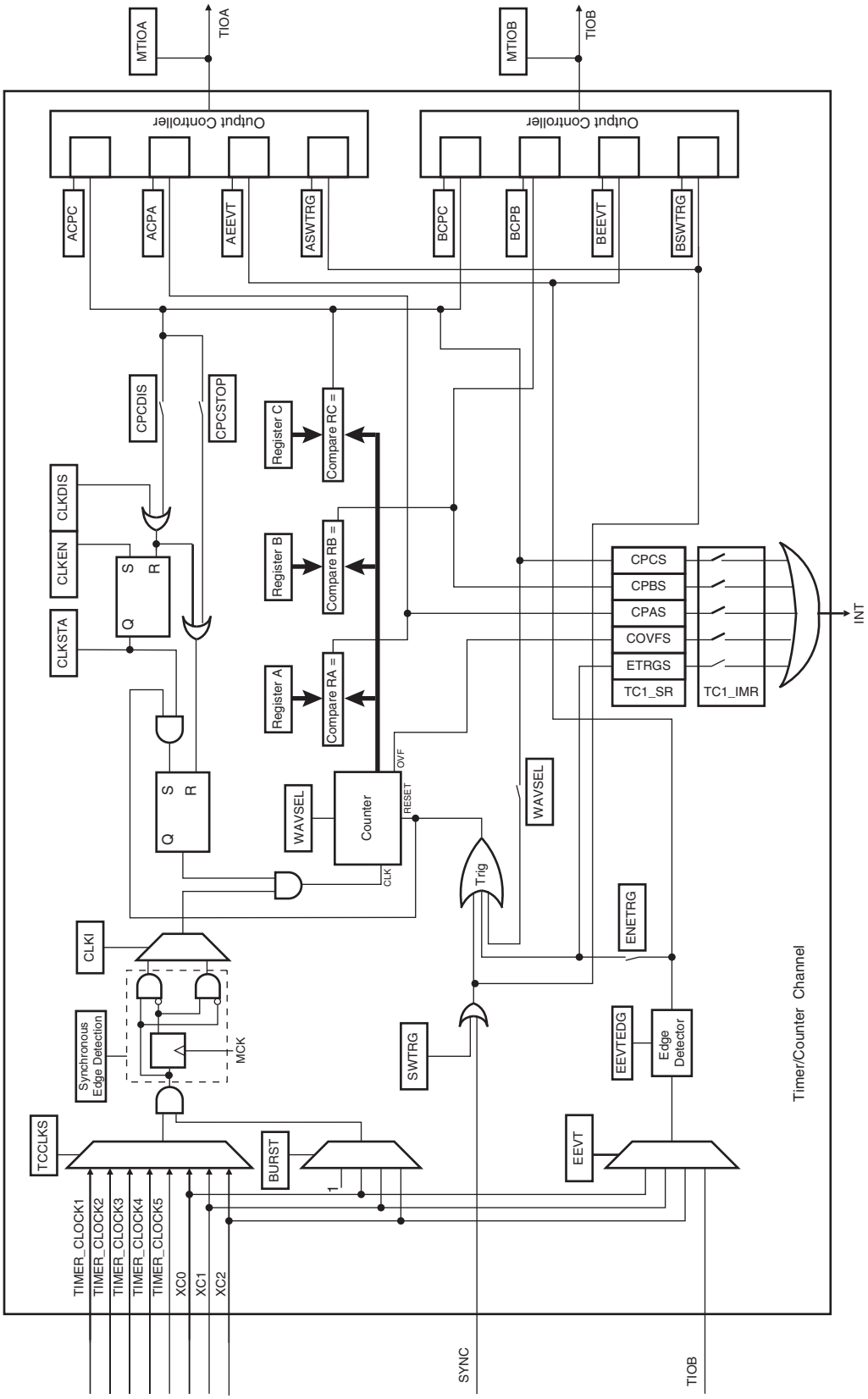
36.6.11 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Figure 36-6. Waveform Mode



36.6.11.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 36-7.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 36-8.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 36-7. WAVSEL= 00 without trigger

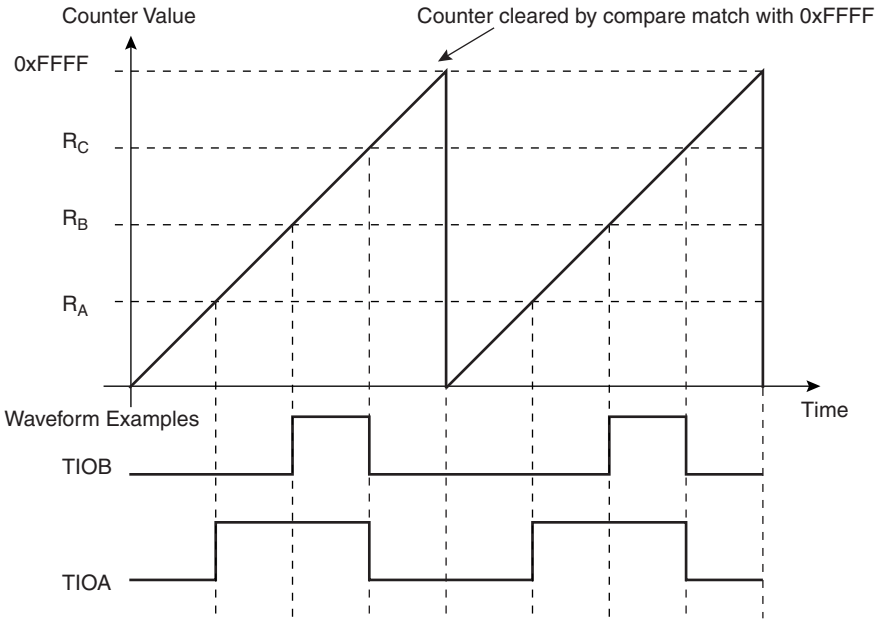
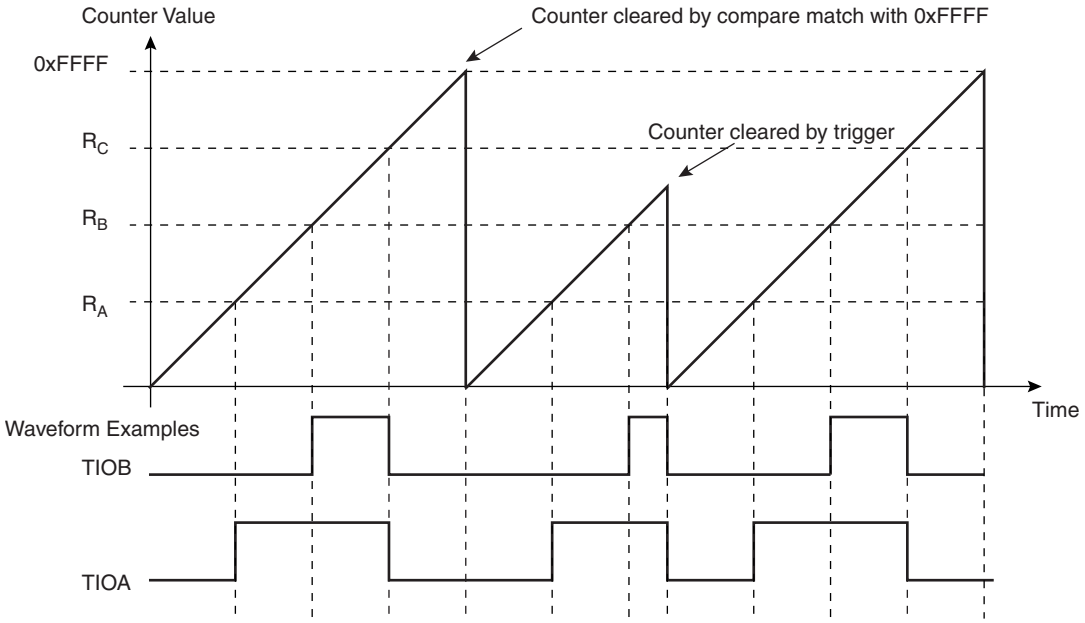


Figure 36-8. WAVSEL= 00 with trigger



36.6.11.2 WAVSEL = 10

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 36-9.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 36-10.

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

Figure 36-9. WAVSEL = 10 Without Trigger

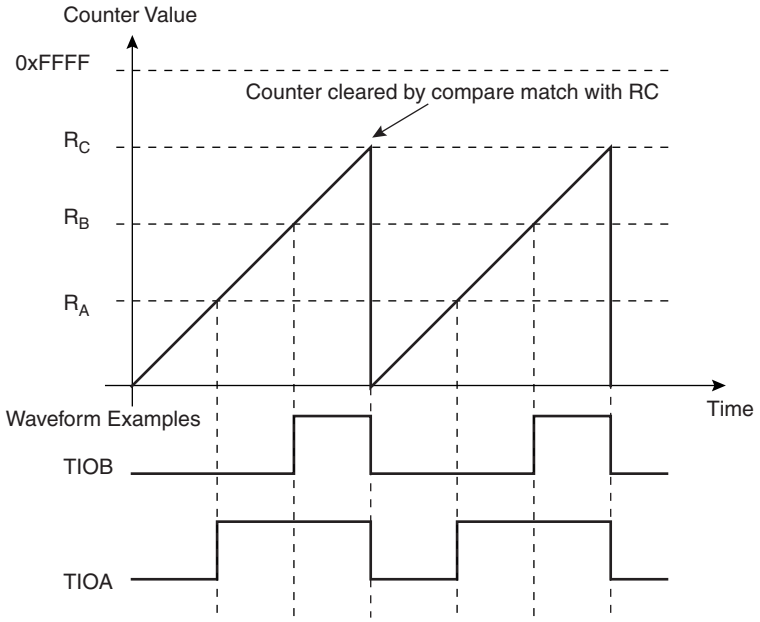
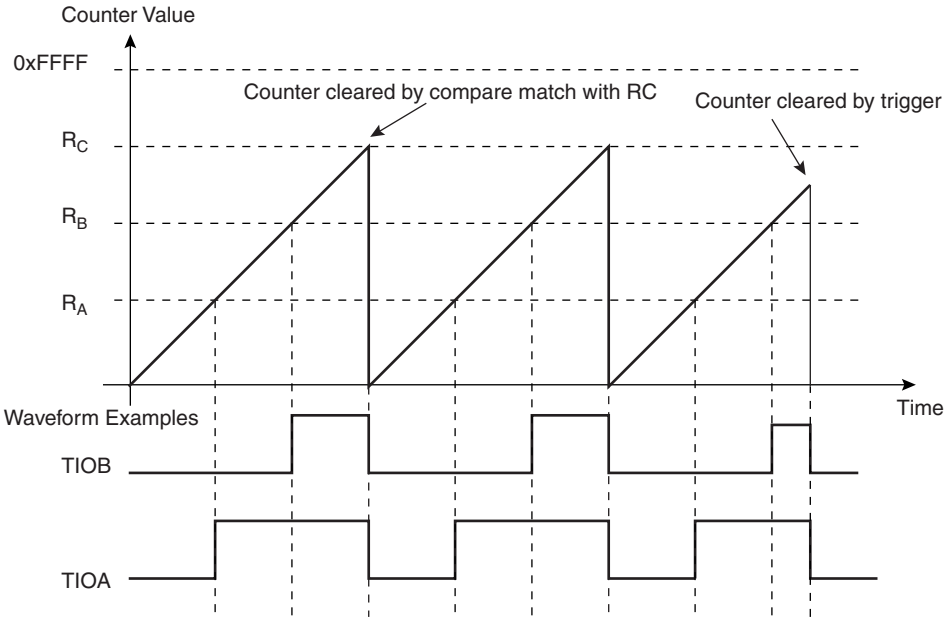


Figure 36-10.WAVSEL = 10 With Trigger



36.6.11.3 WAVSEL = 01

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 36-11.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 36-12.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 36-11.WAVSEL = 01 Without Trigger

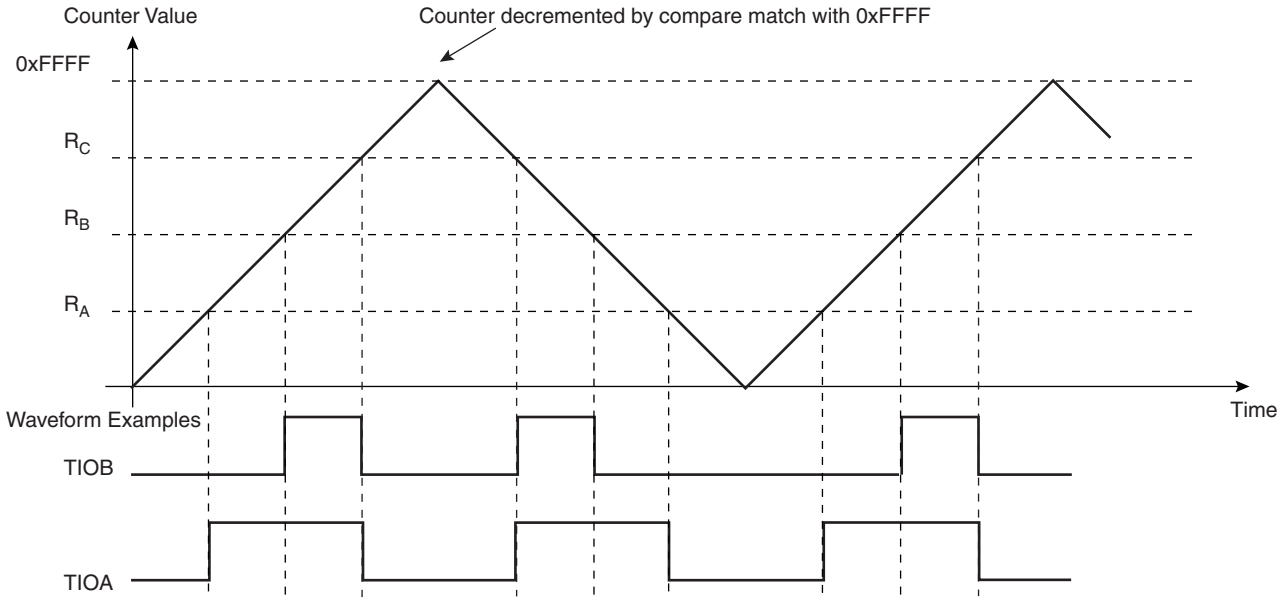
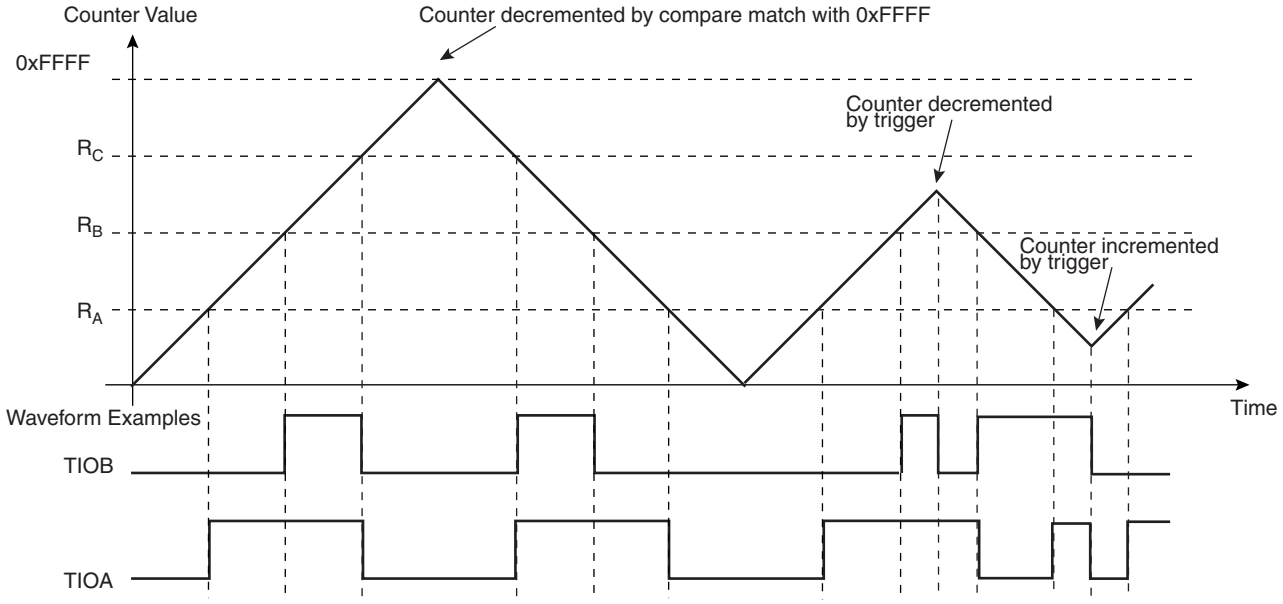


Figure 36-12.WAVSEL = 01 With Trigger



36.6.11.4 WAVSEL = 11

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 36-13.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 36-14.

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

Figure 36-13.WAVSEL = 11 Without Trigger

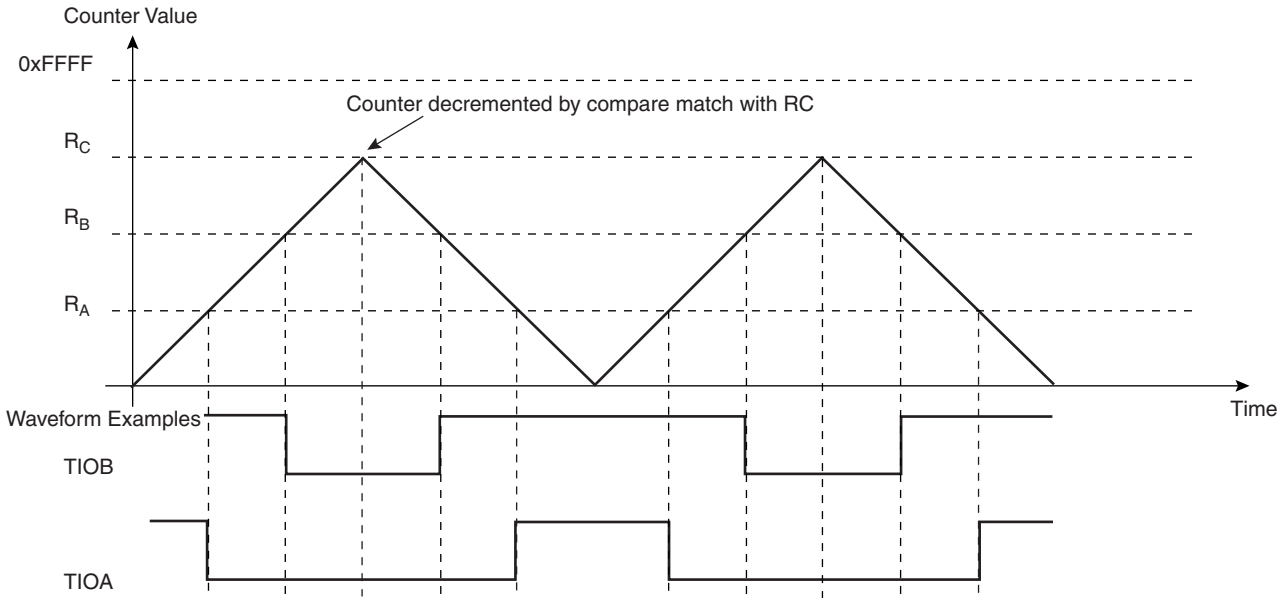
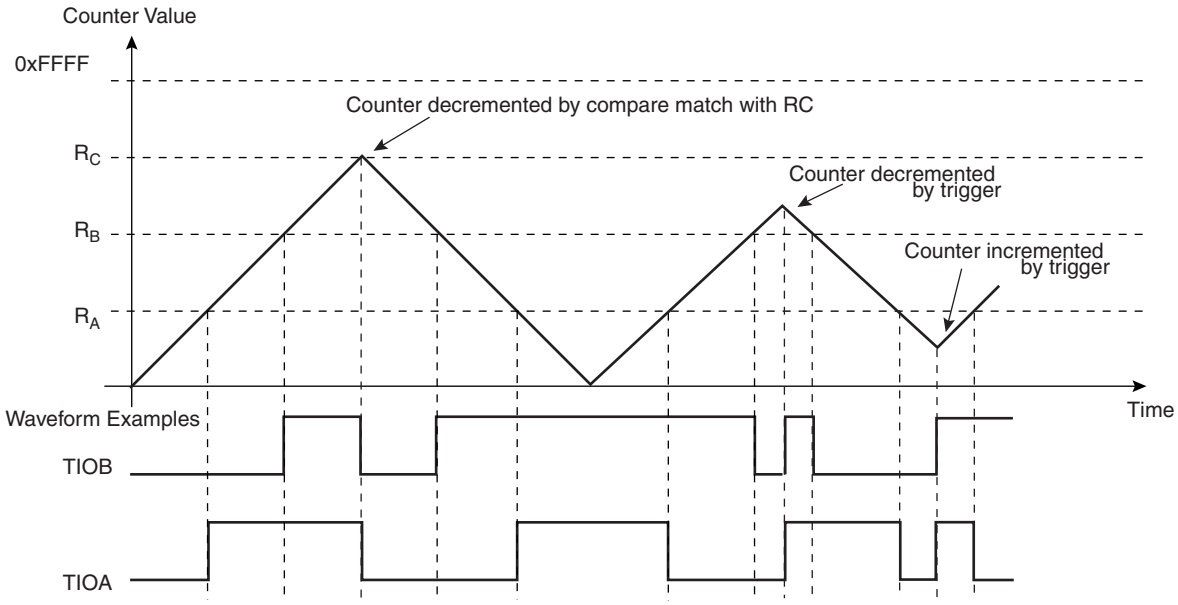


Figure 36-14.WAVSEL = 11 With Trigger



36.6.12 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.

36.6.13 Output Controller

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

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

36.7 Timer Counter (TC) User Interface

Table 36-5. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00 + channel * 0x40 + 0x00	Channel Control Register	TC_CCR	Write-only	–
0x00 + channel * 0x40 + 0x04	Channel Mode Register	TC_CMR	Read-write	0
0x00 + channel * 0x40 + 0x08	Reserved			
0x00 + channel * 0x40 + 0x0C	Reserved			
0x00 + channel * 0x40 + 0x10	Counter Value	TC_CV	Read-only	0
0x00 + channel * 0x40 + 0x14	Register A	TC_RA	Read-write[2]	0
0x00 + channel * 0x40 + 0x18	Register B	TC_RB	Read-write[2]	0
0x00 + channel * 0x40 + 0x1C	Register C	TC_RC	Read-write	0
0x00 + channel * 0x40 + 0x20	Status Register	TC_SR	Read-only	0
0x00 + channel * 0x40 + 0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x00 + channel * 0x40 + 0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x00 + channel * 0x40 + 0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xC0	Block Control Register	TC_BCR	Write-only	–
0xC4	Block Mode Register	TC_BMR	Read-write	0
0xC8 - 0xD4	Reserved			
0xD8	Reserved			
0xE4	Reserved			
0xE8 - 0xFC	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if WAVE = 0

36.7.1 TC Channel Control Register

Name: TC_CCRx [x=0..2]

Address: 0xF8008000 (0)[0], 0xF8008040 (0)[1], 0xF8008080 (0)[2], 0xF800C000 (1)[0], 0xF800C040 (1)[1], 0xF800C080 (1)[2]

Access: Write-only

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

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

1 = A software trigger is performed: the counter is reset and the clock is started.

36.7.2 TC Channel Mode Register: Capture Mode

Name: TC_CMRx [x=0..2] (WAVE = 0)

Address: 0xF8008004 (0)[0], 0xF8008044 (0)[1], 0xF8008084 (0)[2], 0xF800C004 (1)[0], 0xF800C044 (1)[1], 0xF800C084 (1)[2]

Access: Read-write

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

- **TCCLKS: Clock Selection**

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: TCLK1
1	TIMER_CLOCK2	Clock selected: TCLK2
2	TIMER_CLOCK3	Clock selected: TCLK3
3	TIMER_CLOCK4	Clock selected: TCLK4
4	TIMER_CLOCK5	Clock selected: TCLK5
5	XC0	Clock selected: XC0
6	XC1	Clock selected: XC1
7	XC2	Clock selected: 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**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	XC0	XC0 is ANDed with the selected clock.
2	XC1	XC1 is ANDed with the selected clock.
3	XC2	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**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	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: Waveform Mode**

0 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

- **LDRB: RB Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

36.7.3 TC Channel Mode Register: Waveform Mode

Name: TC_CMRx [x=0..2] (WAVE = 1)

Access: Read-write

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

- **TCCLKS: Clock Selection**

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: TCLK1
1	TIMER_CLOCK2	Clock selected: TCLK2
2	TIMER_CLOCK3	Clock selected: TCLK3
3	TIMER_CLOCK4	Clock selected: TCLK4
4	TIMER_CLOCK5	Clock selected: TCLK5
5	XC0	Clock selected: XC0
6	XC1	Clock selected: XC1
7	XC2	Clock selected: 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**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	XC0	XC0 is ANDed with the selected clock.
2	XC1	XC1 is ANDed with the selected clock.
3	XC2	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**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

- **EEVT: External Event Selection**

Signal selected as external event.

Value	Name	Description	TIOB Direction
0	TIOB	TIOB[1]	Input
1	XC0	XC0	Output
2	XC1	XC1	Output
3	XC2	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**

Value	Name	Description
0	UP	UP mode without automatic trigger on RC Compare
1	UPDOWN	UPDOWN mode without automatic trigger on RC Compare
2	UP_RC	UP mode with automatic trigger on RC Compare
3	UPDOWN_RC	UPDOWN mode with automatic trigger on RC Compare

- **WAVE: Waveform Mode**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

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

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

36.7.4 TC Counter Value Register

Name: TC_CVx [x=0..2]

Address: 0xF8008010 (0)[0], 0xF8008050 (0)[1], 0xF8008090 (0)[2], 0xF800C010 (1)[0], 0xF800C050 (1)[1], 0xF800C090 (1)[2]

Access: Read-only

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

- **CV: Counter Value**

CV contains the counter value in real time.

36.7.5 TC Register A

Name: TC_RAx [x=0..2]

Address: 0xF8008014 (0)[0], 0xF8008054 (0)[1], 0xF8008094 (0)[2], 0xF800C014 (1)[0], 0xF800C054 (1)[1], 0xF800C094 (1)[2]

Access: Read-only if WAVE = 0, Read-write if WAVE = 1

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

- **RA: Register A**

RA contains the Register A value in real time.

36.7.6 TC Register B

Name: TC_RBx [x=0..2]

Address: 0xF8008018 (0)[0], 0xF8008058 (0)[1], 0xF8008098 (0)[2], 0xF800C018 (1)[0], 0xF800C058 (1)[1], 0xF800C098 (1)[2]

Access: Read-only if WAVE = 0, Read-write if WAVE = 1

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

- **RB: Register B**

RB contains the Register B value in real time.

36.7.7 TC Register C

Name: TC_RCx [x=0..2]

Address: 0xF800801C (0)[0], 0xF800805C (0)[1], 0xF800809C (0)[2], 0xF800C01C (1)[0], 0xF800C05C (1)[1], 0xF800C09C (1)[2]

Access: Read-write

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

- **RC: Register C**

RC contains the Register C value in real time.

36.7.8 TC Status Register

Name: TC_SRx [x=0..2]

Address: 0xF8008020 (0)[0], 0xF8008060 (0)[1], 0xF80080A0 (0)[2], 0xF800C020 (1)[0], 0xF800C060 (1)[1], 0xF800C0A0 (1)[2]

Access: Read-only

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

- **COVFS: Counter Overflow Status**

0 = No counter overflow has occurred since the last read of the Status Register.

1 = A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status**

0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.

- **CPAS: RA Compare Status**

0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPBS: RB Compare Status**

0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPCS: RC Compare Status**

0 = RC Compare has not occurred since the last read of the Status Register.

1 = RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status**

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- **LDRBS: RB Loading Status**

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- **ETRGS: External Trigger Status**

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

36.7.9 TC Interrupt Enable Register

Name: TC_IERx [x=0..2]

Address: 0xF8008024 (0)[0], 0xF8008064 (0)[1], 0xF80080A4 (0)[2], 0xF800C024 (1)[0], 0xF800C064 (1)[1], 0xF800C0A4 (1)[2]

Access: Write-only

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

- **COVFS: Counter Overflow**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.

36.7.10 TC Interrupt Disable Register

Name: TC_IDRx [x=0..2]

Address: 0xF8008028 (0)[0], 0xF8008068 (0)[1], 0xF80080A8 (0)[2], 0xF800C028 (1)[0], 0xF800C068 (1)[1], 0xF800C0A8 (1)[2]

Access: Write-only

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

- **COVFS: Counter Overflow**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

36.7.11 TC Interrupt Mask Register

Name: TC_IMRx [x=0..2]

Address: 0xF800802C (0)[0], 0xF800806C (0)[1], 0xF80080AC (0)[2], 0xF800C02C (1)[0], 0xF800C06C (1)[1], 0xF800C0AC (1)[2]

Access: Read-only

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

- **COVFS: Counter Overflow**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.

36.7.12 TC Block Control Register

Name: TC_BCR

Address: 0xF80080C0 (0), 0xF800C0C0 (1)

Access: Write-only

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

- **SYNC: Synchro Command**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

36.7.13 TC Block Mode Register

Name: TC_BMR

Address: 0xF80080C4 (0), 0xF800C0C4 (1)

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

- **TC0XC0S: External Clock Signal 0 Selection**

Value	Name	Description
0	TCLK0	Signal connected to XC0: TCLK0
1	–	Reserved
2	TIOA1	Signal connected to XC0: TIOA1
3	TIOA2	Signal connected to XC0: TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

Value	Name	Description
0	TCLK1	Signal connected to XC1: TCLK1
1	–	Reserved
2	TIOA0	Signal connected to XC1: TIOA0
3	TIOA2	Signal connected to XC1: TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

Value	Name	Description
0	TCLK2	Signal connected to XC2: TCLK2
1	–	Reserved
2	TIOA1	Signal connected to XC2: TIOA1
3	TIOA2	Signal connected to XC2: TIOA2

37. Pulse Width Modulation Controller (PWM)

37.1 Description

The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM macrocell master clock.

All PWM macrocell accesses are made through APB mapped registers.

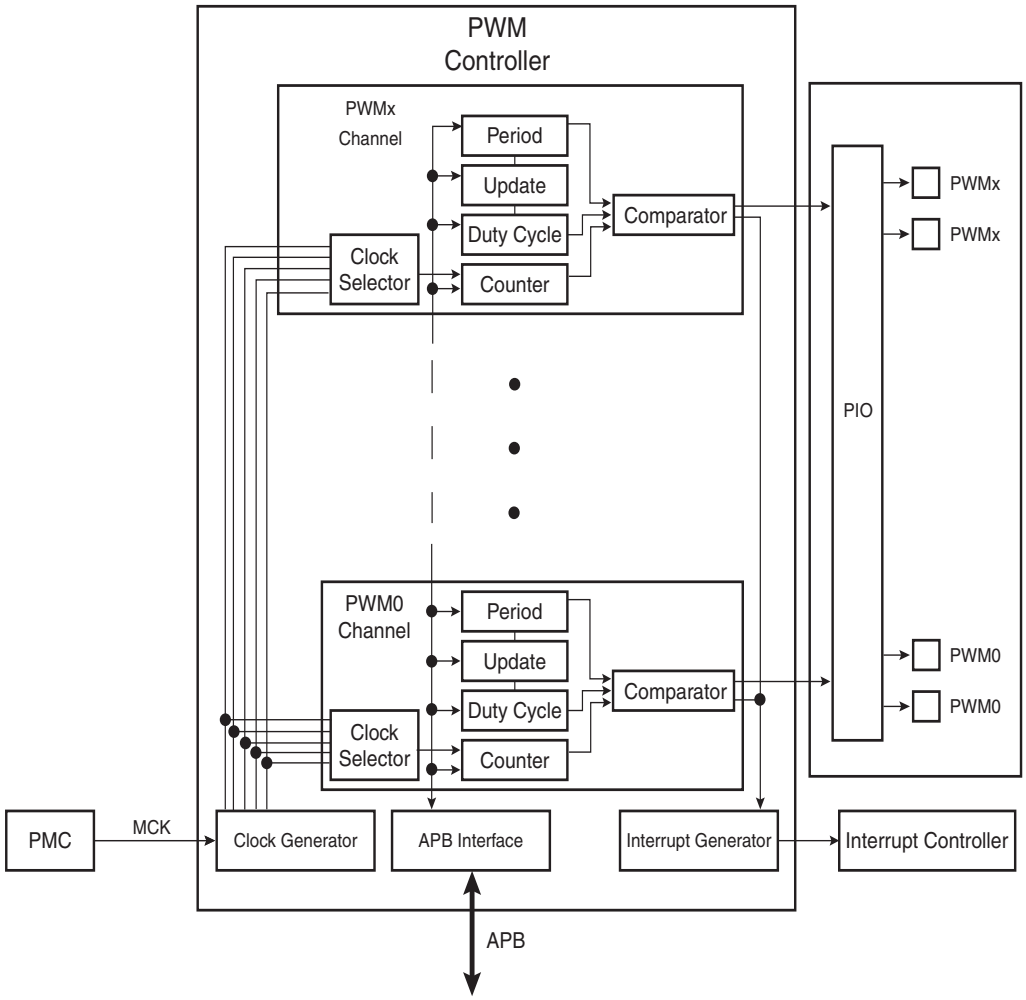
Channels can be synchronized, to generate non overlapped waveforms. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period or the duty-cycle.

37.2 Embedded characteristics

- 4 Channels
- One 16-bit Counter Per Channel
- Common Clock Generator Providing Thirteen Different Clocks
  - A Modulo n Counter Providing Eleven Clocks
  - Two Independent Linear Dividers Working on Modulo n Counter Outputs
- Independent Channels
  - Independent Enable Disable Command for Each Channel
  - Independent Clock Selection for Each Channel
  - Independent Period and Duty Cycle for Each Channel
  - Double Buffering of Period or Duty Cycle for Each Channel
  - Programmable Selection of The Output Waveform Polarity for Each Channel
  - Programmable Center or Left Aligned Output Waveform for Each Channel Block Diagram

37.3 Block Diagram

Figure 37-1. Pulse Width Modulation Controller Block Diagram



37.4 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 37-1. I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

37.5 Product Dependencies

37.5.1 I/O Lines

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

All of the PWM outputs may or may not be enabled. If an application requires only four channels, then only four PIO lines will be assigned to PWM outputs.

Table 37-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
PWM	PWM0	PB11	B
PWM	PWM0	PC10	C
PWM	PWM0	PC18	C
PWM	PWM1	PB12	B
PWM	PWM1	PC11	C
PWM	PWM1	PC19	C
PWM	PWM2	PB13	B
PWM	PWM2	PC20	C
PWM	PWM3	PB14	B
PWM	PWM3	PC21	C

37.5.2 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.

All the PWM registers except PWM_CDTY and PWM_CPRD can be read without the PWM peripheral clock enabled. All the registers can be written without the peripheral clock enabled.

37.5.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the PWM interrupt requires the Interrupt Controller to be programmed first. Note that it is not recommended to use the PWM interrupt line in edge sensitive mode.

Table 37-3. Peripheral IDs

Instance	ID
PWM	18

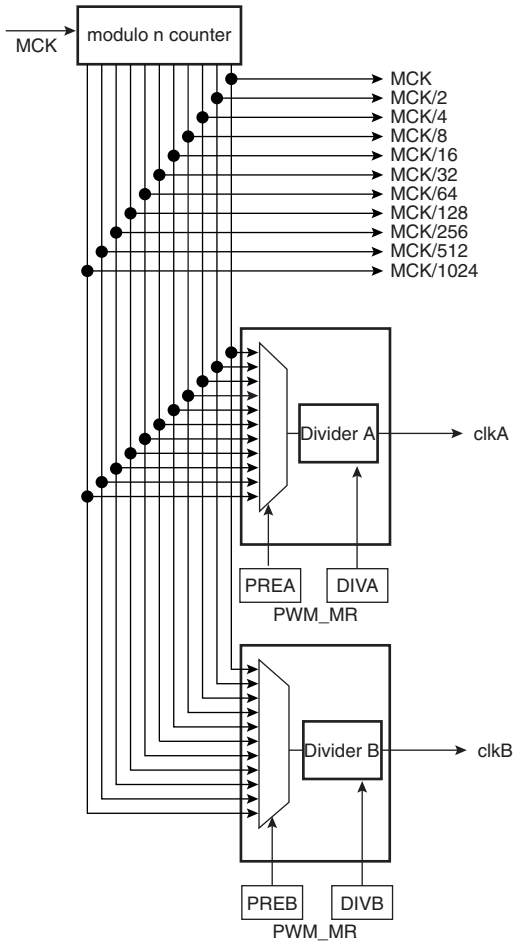
37.6 Functional Description

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.

37.6.1 PWM Clock Generator

Figure 37-2. 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).

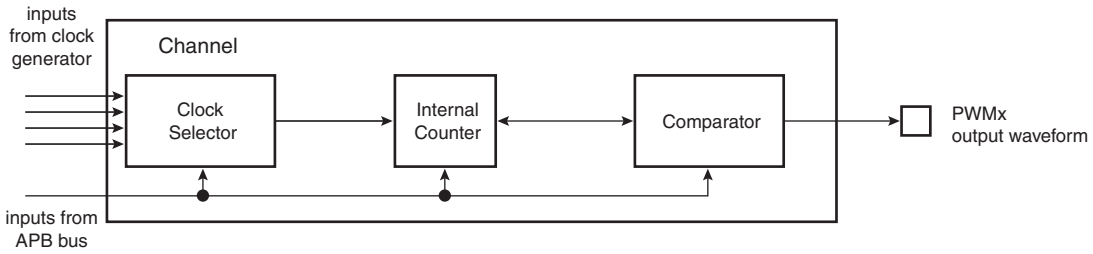
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

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.

37.6.2 PWM Channel

37.6.2.1 Block Diagram

Figure 37-3. 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 37.6.1 “PWM Clock Generator” on page 724.
- 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.

37.6.2.2 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 is:

$$\frac{(X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(X*CPRD*DIVA)}{MCK} \text{ or } \frac{(X*CPRD*DIVB)}{MCK}$$

If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
By using the Master Clock (MCK) divided by an X given prescaler value
(with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula is:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2*X*CPRD*DIVA)}{MCK} \text{ or } \frac{(2*X*CPRD*DIVB)}{MCK}$$

- 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:

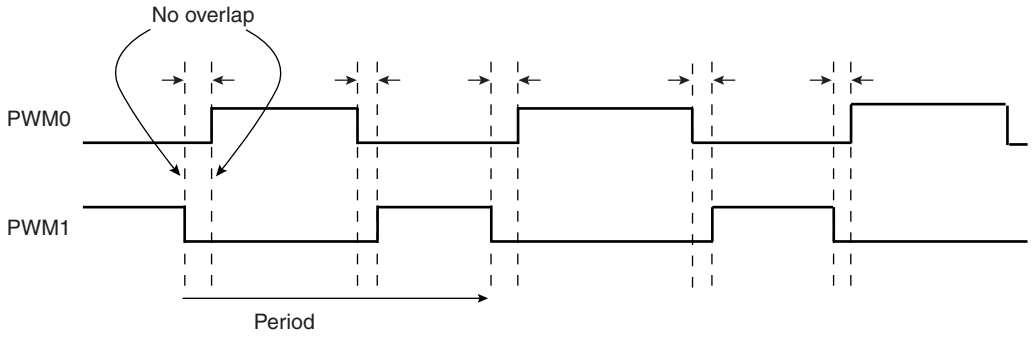
$$\text{duty cycle} = \frac{(period - 1/\text{fchannel_x_clock} \times CDTY)}{period}$$

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 37-4. Non Overlapped Center Aligned Waveforms



Note: See Figure 37-5 on page 727 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:

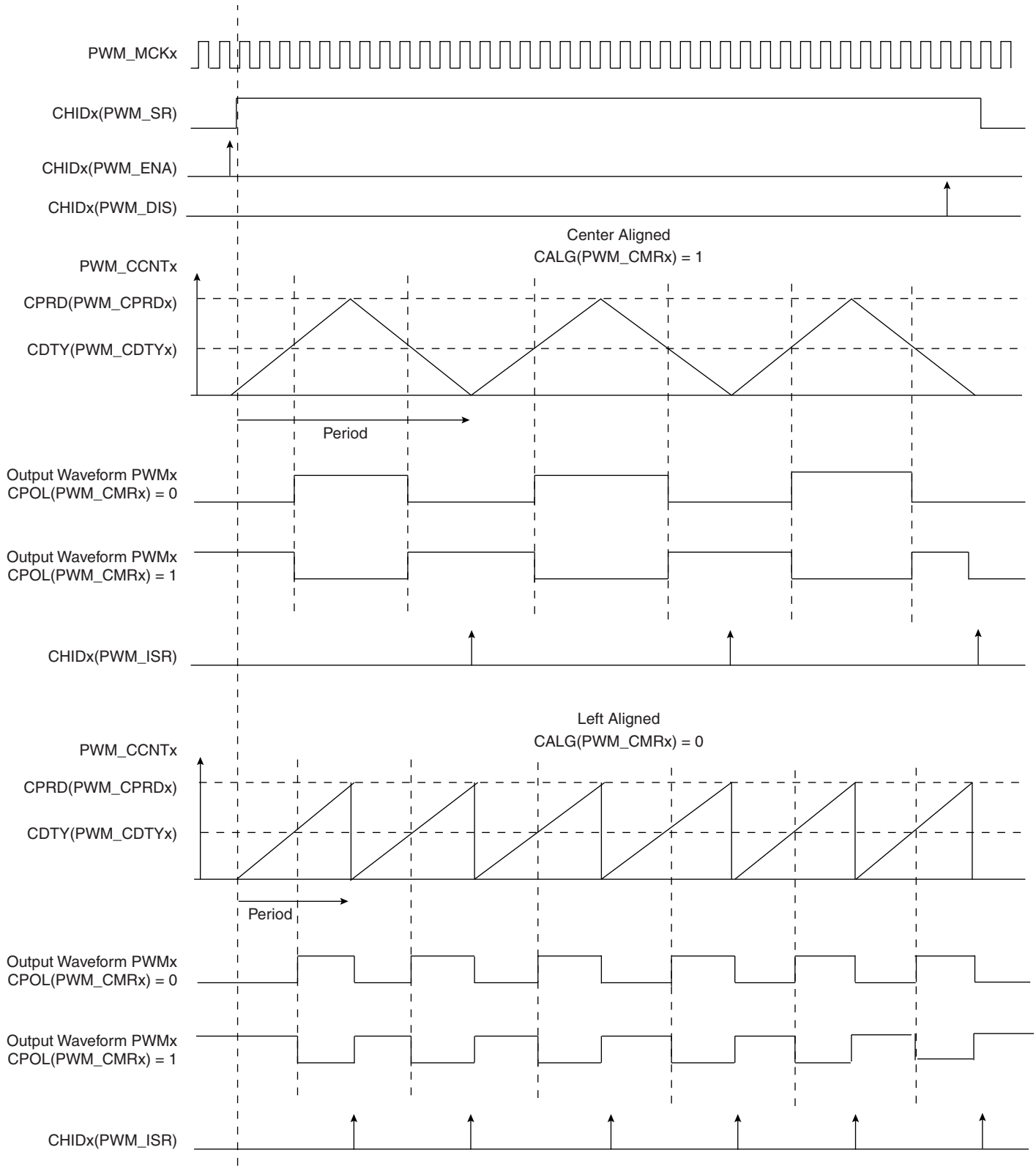
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

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 37-5. Waveform Properties



37.6.3 PWM Controller Operations

37.6.3.1 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.

37.6.3.2 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.

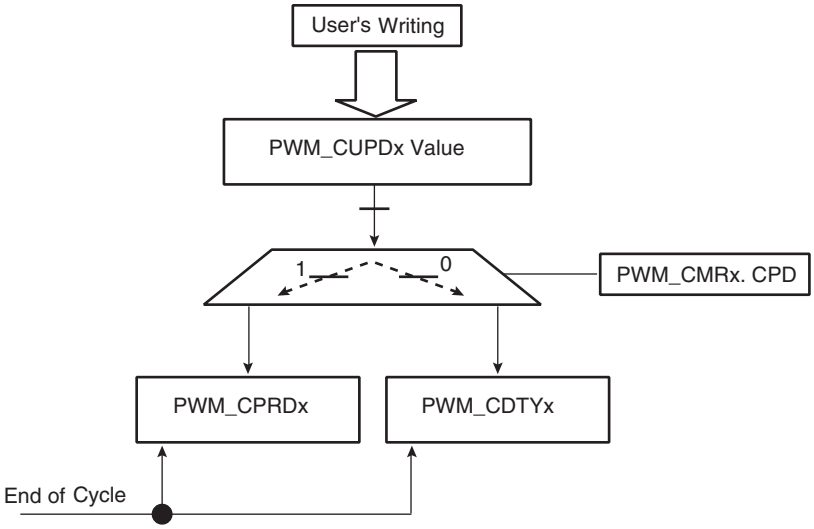
For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value between 1 up to 14 in PWM_CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

37.6.3.3 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 37-6. Synchronized Period or Duty Cycle Update



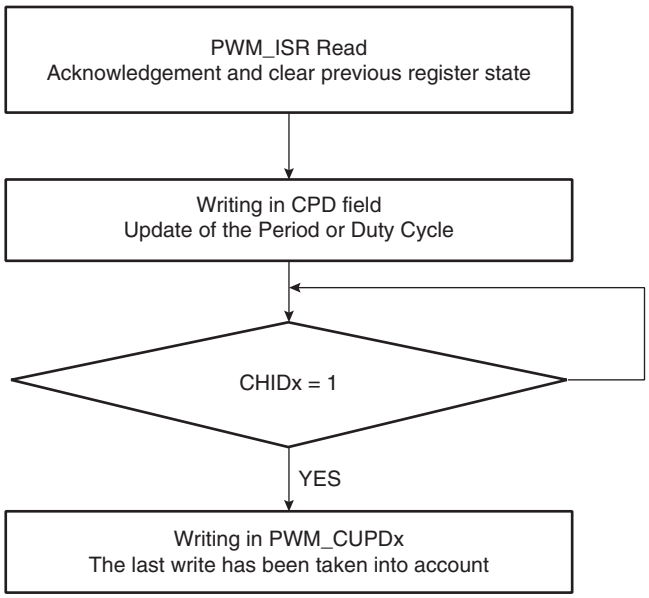
To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 37-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

Figure 37-7. Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

37.6.3.4 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.

37.7 Pulse Width Modulation Controller (PWM) User Interface

Table 37-4. Register Mapping[1]

Offset	Register	Name	Access	Reset
0x00	PWM Mode Register	PWM_MR	Read-write	0
0x04	PWM Enable Register	PWM_ENA	Write-only	-
0x08	PWM Disable Register	PWM_DIS	Write-only	-
0x0C	PWM Status Register	PWM_SR	Read-only	0
0x10	PWM Interrupt Enable Register	PWM_IER	Write-only	-
0x14	PWM Interrupt Disable Register	PWM_IDR	Write-only	-
0x18	PWM Interrupt Mask Register	PWM_IMR	Read-only	0
0x1C	PWM Interrupt Status Register	PWM_ISR	Read-only	0
0x20 - 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.

37.7.1 PWM Mode Register

Name: PWM_MR

Address: 0xF8034000

Access: Read/Write

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

23	22	21	20	19	18	17	16
DIVB							

15	14	13	12	11	10	9	8
–	–	–	–	PREA			

7	6	5	4	3	2	1	0
DIVA							

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

Value	Name	Description
0	CLK_OFF	CLKA, CLKB clock is turned off
1	CLK_DIV1	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**

Value	Name	Description
0000	MCK	Master Clock
0001	MCKDIV2	Master Clock divided by 2
0010	MCKDIV4	Master Clock divided by 4
0011	MCKDIV8	Master Clock divided by 8
0100	MCKDIV16	Master Clock divided by 16
0101	MCKDIV32	Master Clock divided by 32
0110	MCKDIV64	Master Clock divided by 64
0111	MCKDIV128	Master Clock divided by 128
1000	MCKDIV256	Master Clock divided by 256
1001	MCKDIV512	Master Clock divided by 512
1010	MCKDIV1024	Master Clock divided by 1024

Values which are not listed in the table must be considered as “reserved”.

37.7.2 PWM Enable Register

Name: PWM_ENA

Address: 0xF8034004

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Enable PWM output for channel x.

37.7.3 PWM Disable Register

Name: PWM_DIS

Address: 0xF8034008

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Disable PWM output for channel x.

37.7.4 PWM Status Register

Name: PWM_SR

Address: 0xF803400C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

37.7.5 PWM Interrupt Enable Register

Name: PWM_IER

Address: 0xF8034010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Enable interrupt for PWM channel x.

37.7.6 PWM Interrupt Disable Register

Name: PWM_IDR

Address: 0xF8034014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Disable interrupt for PWM channel x.

37.7.7 PWM Interrupt Mask Register

Name: PWM_IMR

Address: 0xF8034018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

37.7.8 PWM Interrupt Status Register

Name: PWM_ISR

Address: 0xF803401C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No new channel period has been achieved since the last read of the PWM_ISR register.

1 = At least one new channel period has been achieved since the last read of the PWM_ISR register.

Note: Reading PWM_ISR automatically clears CHIDx flags.

37.7.9 PWM Channel Mode Register

Name: PWM_CMR[0..3]

Address: 0xF8034200 [0], 0xF8034220 [1], 0xF8034240 [2], 0xF8034260 [3]

Access: Read/Write

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

- **CPRE: Channel Pre-scaler**

Value	Name	Description
0000	MCK	Master Clock
0001	MCKDIV2	Master Clock divided by 2
0010	MCKDIV4	Master Clock divided by 4
0011	MCKDIV8	Master Clock divided by 8
0100	MCKDIV16	Master Clock divided by 16
0101	MCKDIV32	Master Clock divided by 32
0110	MCKDIV64	Master Clock divided by 64
0111	MCKDIV128	Master Clock divided by 128
1000	MCKDIV256	Master Clock divided by 256
1001	MCKDIV512	Master Clock divided by 512
1010	MCKDIV1024	Master Clock divided by 1024
1011	CLKA	Clock A
1100	CLKB	Clock B

Values which are not listed in the table must be considered as “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.

37.7.10 PWM Channel Duty Cycle Register

Name: PWM_CDTY[0..3]

Address: 0xF8034204 [0], 0xF8034224 [1], 0xF8034244 [2], 0xF8034264 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
CDTY							

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

Only the first 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).

37.7.11 PWM Channel Period Register

Name: PWM_CPRD[0..3]

Address: 0xF8034208 [0], 0xF8034228 [1], 0xF8034248 [2], 0xF8034268 [3]

Access: Read/Write

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

Only the first 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 is:

$$\frac{(X \times CPRD)}{MCK}$$

– 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 is:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

37.7.12 PWM Channel Counter Register

Name: PWM_CCNT[0..3]

Address: 0xF803420C [0], 0xF803422C [1], 0xF803424C [2], 0xF803426C [3]

Access: Read-only

31	30	29	28	27	26	25	24
CNT							

23	22	21	20	19	18	17	16
CNT							

15	14	13	12	11	10	9	8
CNT							

7	6	5	4	3	2	1	0
CNT							

- **CNT: Channel Counter Register**

Internal counter value. This register is reset when:

- The channel is enabled (writing CHIDx in the PWM_ENA register).
- The counter reaches CPRD value defined in the PWM_CPRDx register if the waveform is left aligned.

37.7.13 PWM Channel Update Register

Name: PWM_CUPD[0..3]

Address: 0xF8034210 [0], 0xF8034230 [1], 0xF8034250 [2], 0xF8034270 [3]

Access: Write-only

31	30	29	28	27	26	25	24
CUPD							

23	22	21	20	19	18	17	16
CUPD							

15	14	13	12	11	10	9	8
CUPD							

7	6	5	4	3	2	1	0
CUPD							

CUPD: Channel Update Register

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.

When CPD field of PWM_CMRx register = 0, the duty-cycle (CDTY of PWM_CDTYx register) is updated with the CUPD value at the beginning of the next period.

When CPD field of PWM_CMRx register = 1, the period (CPRD of PWM_CPRDx register) is updated with the CUPD value at the beginning of the next period.

38. Two-wire Interface (TWI)

38.1 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 38-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I^2C compatible device.

Table 38-1. Atmel TWI compatibility with I^2C Standard

I^2C Standard	Atmel TWI
Standard Mode Speed (100 KHz)	Supported
Fast Mode Speed (400 KHz)	Supported
7 or 10 bits Slave Addressing	Supported
START BYTE(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope control and input filtering (Fast mode)	Not Supported
Clock stretching	Supported
Multi Master Capability	Supported

Note: 1. START + b000000001 + Ack + Sr

38.2 Embedded Characteristics

- Three TWIs
- Compatible with Atmel Two-wire Interface Serial Memory and I²C Compatible Devices[1]
- One, Two or Three Bytes for Slave Address
- Sequential Read-write Operations
- Master, Multi-master and Slave Mode Operation
- Bit Rate: Up to 400 Kbits
- General Call Supported in Slave mode
- SMBUS Quick Command Supported in Master Mode
- Connection to DMA Controller (DMA) Channel Capabilities optimizes Data Transfers in Master Mode Only

Note: 1. See Table 38-1 for details on compatibility with I²C Standard.

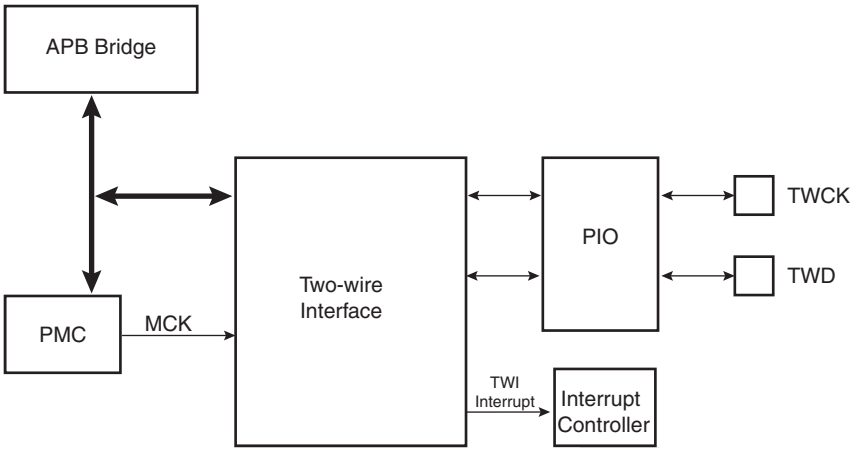
38.3 List of Abbreviations

Table 38-2. Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

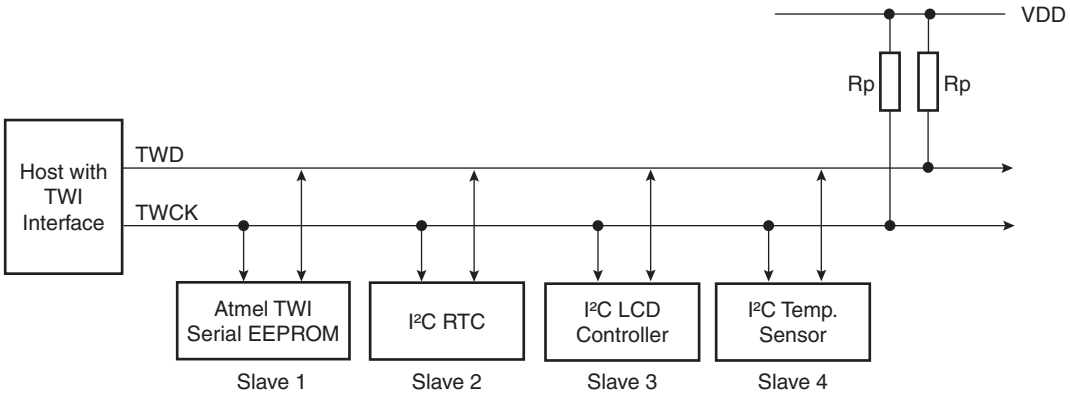
38.4 Block Diagram

Figure 38-1. Block Diagram



38.5 Application Block Diagram

Figure 38-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

38.5.1 I/O Lines Description

Table 38-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

38.6 Product Dependencies

38.6.1 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 38-2 on page 745). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following step:

- Program the PIO controller to dedicate TWD and TWCK as peripheral lines.

The user must not program TWD and TWCK as open-drain. It is already done by the hardware.

Table 38-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TWI0	TWCK0	PA31	A
TWI0	TWD0	PA30	A
TWI1	TWCK1	PC1	C
TWI1	TWD1	PC0	C
TWI2	TWCK2	PB5	B
TWI2	TWD2	PB4	B

38.6.2 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.

38.6.3 Interrupt

The TWI interface has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TWI.

Table 38-5. Peripheral IDs

Instance	ID
TWI0	9
TWI1	10
TWI2	11

38.7 Functional Description

38.7.1 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 38-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 38-3).

- 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 38-3. START and STOP Conditions

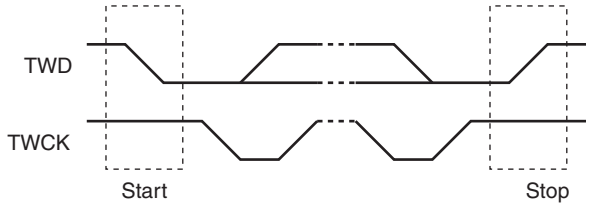
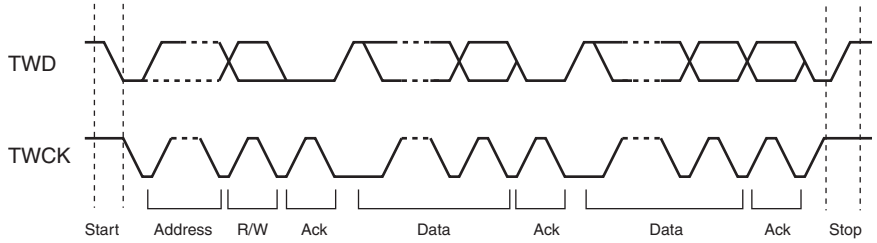


Figure 38-4. Transfer Format



38.7.2 Modes of Operation

The TWI has different 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.

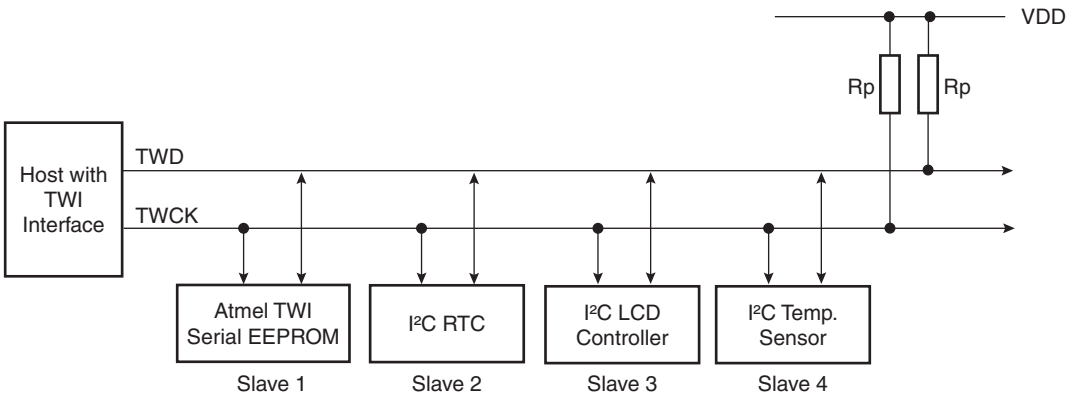
38.8 Master Mode

38.8.1 Definition

The Master is the device that starts a transfer, generates a clock and stops it.

38.8.2 Application Block Diagram

Figure 38-5. Master Mode Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

38.8.3 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.

38.8.4 Master Transmitter Mode

After the master initiates a Start condition when writing into the Transmit Holding Register, TWI_THR, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction, 0 in this case (MREAD = 0 in TWI_MMR).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the Not Acknowledge bit (NACK**)** in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI_IER). If the slave acknowledges the byte, the data written in the TWI_THR, is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR.

While no new data is written in the TWI_THR, the Serial Clock Line is tied low. When new data is written in the TWI_THR, the SCL is released and the data is sent. To generate a STOP event, the STOP command must be performed by writing in the STOP field of TWI_CR.

After a Master Write transfer, the Serial Clock line is stretched (tied low) while no new data is written in the TWI_THR or until a STOP command is performed.

See Figure 38-6, Figure 38-7, and Figure 38-8.

Figure 38-6. Master Write with One Data Byte

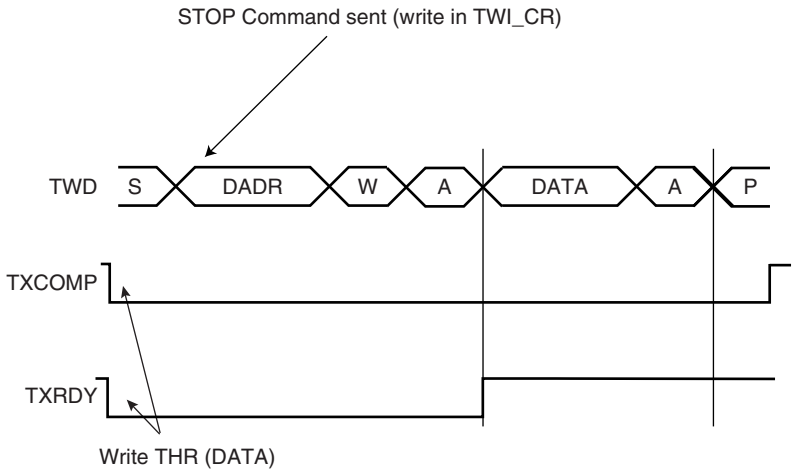


Figure 38-7. Master Write with Multiple Data Bytes

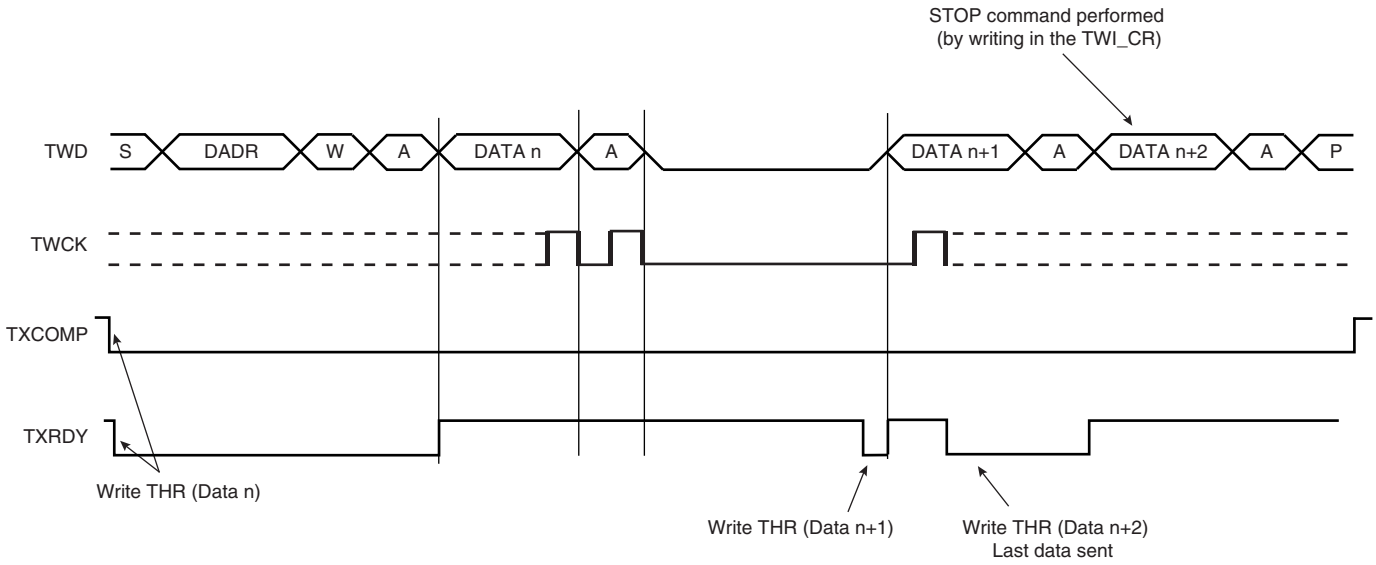
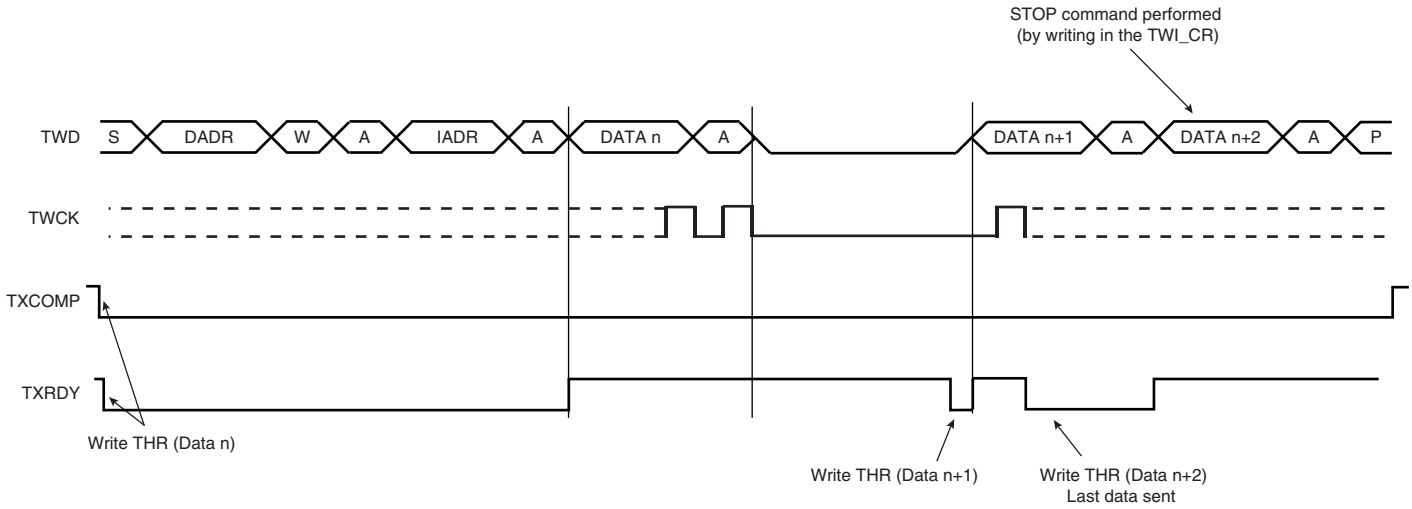


Figure 38-8. Master Write with One Byte Internal Address and Multiple Data Bytes



38.8.5 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 38-9. 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 38-9. 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 38-10. For Internal Address usage see Section 38.8.6.

If the receive holding register (TWI_RHR) is full (RXRDY high) and the master is receiving data, the Serial Clock Line will be tied low before receiving the last bit of the data and until the TWI_RHR register is read. Once the TWI_RHR register is read, the master will stop stretching the Serial Clock Line and end the data reception. See Figure 38-11.

Warning: When receiving multiple bytes in master read mode, if the next-to-last access is not read (the RXRDY flag remains high), the last access will not be completed until TWI_RHR is read. The last access stops on the next-to-last bit (clock stretching). When the TWI_RHR register is read there is only half a bit period to send the stop bit command, else another read access might occur (spurious access).

A possible workaround is to raise the STOP BIT command before reading the TWI_RHR on the next-to-last access (within IT handler).

Figure 38-9. Master Read with One Data Byte

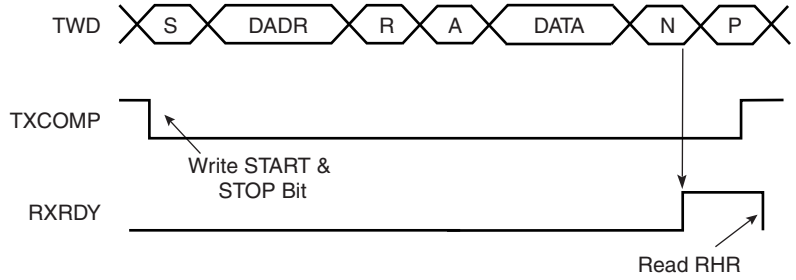


Figure 38-10.Master Read with Multiple Data Bytes

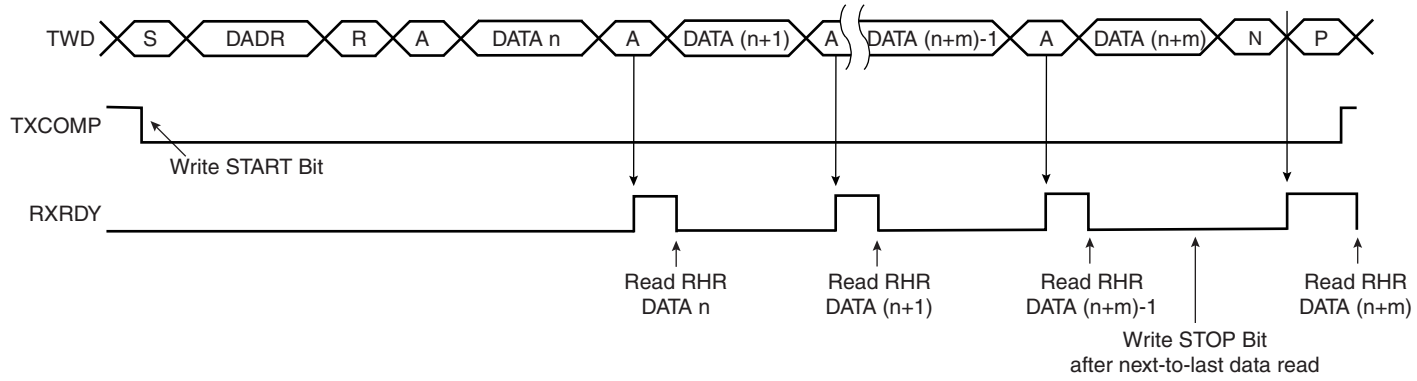
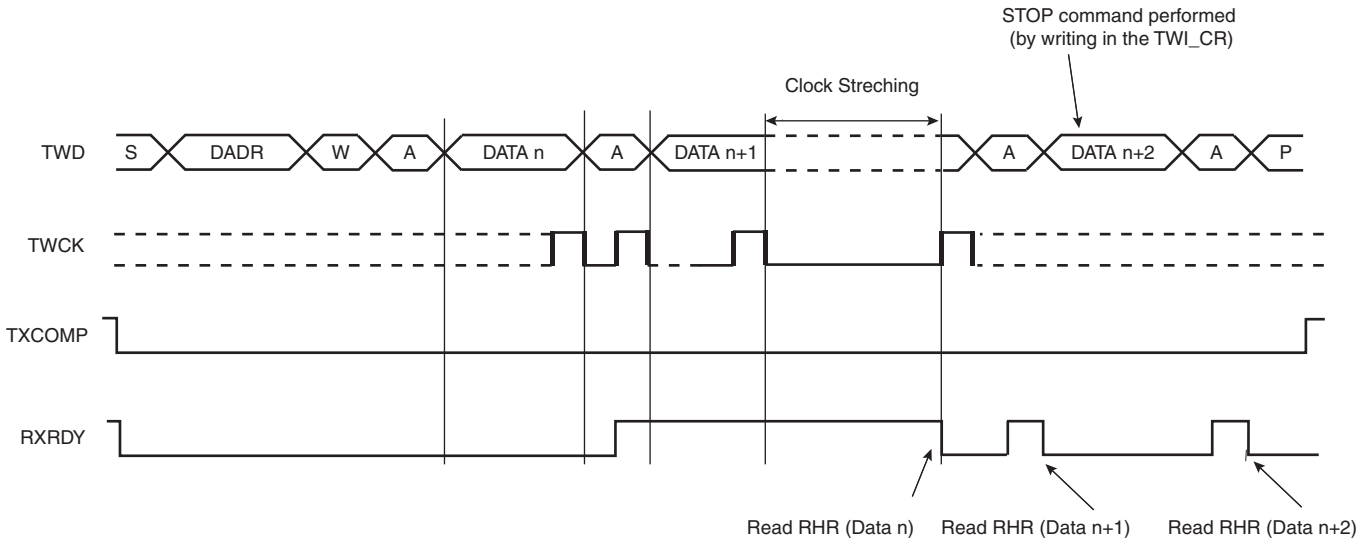


Figure 38-11.Master Read Clock Stretching with Multiple Data Bytes



38.8.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

38.8.6.1 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 I^2C fully-compatible devices. See Figure 38-13. See Figure 38-12 and Figure 38-14 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 38-12.Master Write with One, Two or Three Bytes Internal Address and One Data Byte

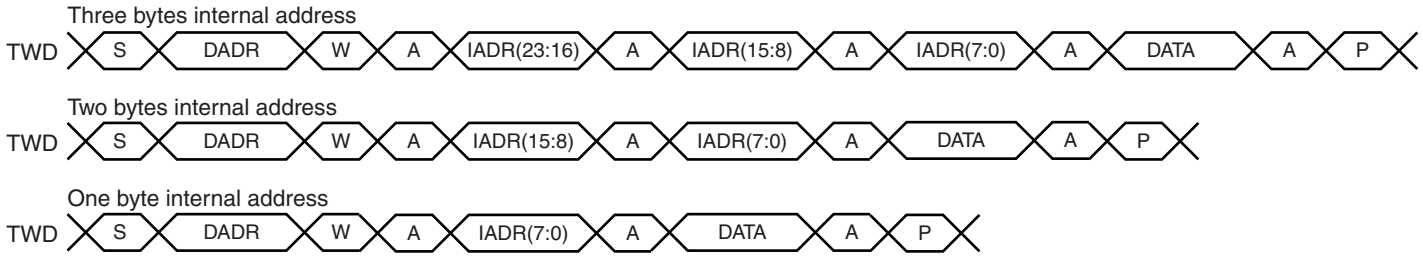
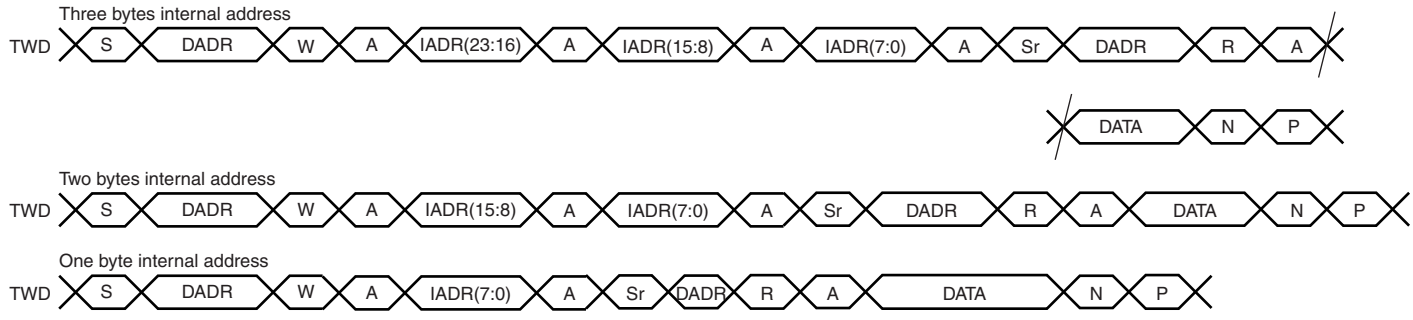


Figure 38-13.Master Read with One, Two or Three Bytes Internal Address and One Data Byte



38.8.6.2 10-bit Slave Addressing

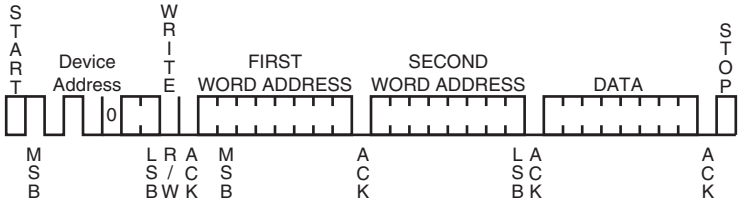
For a slave address higher than 7 bits, the user must configure the address size (IADRSZ) and set the other slave address bits in the internal address register (TWI_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

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 38-14 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 38-14. Internal Address Usage



38.8.7 Using the DMA Controller

The use of the DMA significantly reduces the CPU load.

To assure correct implementation, respect the following programming sequence.

38.8.7.1 Data Transmit with the DMA

1. Initialize the DMA (channels, memory pointers, size, etc.);
2. Configure the master mode (DADR, CKDIV, etc.).
3. Enable the DMA.
4. Wait for the DMA BTC flag.
5. Disable the DMA.

38.8.7.2 Data Receive with the DMA

The PDC transfer size must be defined with the buffer size minus 2. The two remaining characters must be managed without PDC to ensure that the exact number of bytes are received whatever the system bus latency conditions encountered during the end of buffer transfer period.

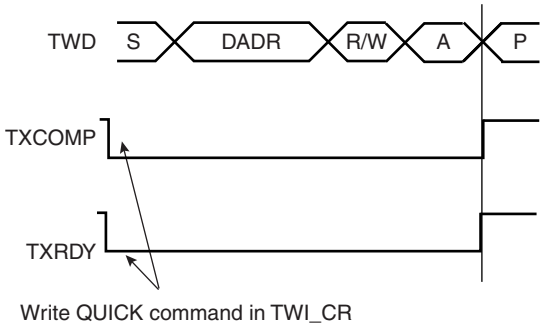
1. Initialize the DMA (channels, memory pointers, size -2, etc.);
2. Configure the master mode (DADR, CKDIV, etc.).
3. Enable the DMA.
4. Wait for the DMA BTC flag.
5. Disable the DMA.
6. Wait for the RXRDY flag in the TWI_SR register
7. Set the STOP command in TWI_CR
8. Read the penultimate character in TWI_RHR
9. Wait for the RXRDY flag in the TWI_SR register
10. Read the last character in TWI_RHR

38.8.8 SMBUS Quick Command (Master Mode Only)

The TWI interface can perform a Quick Command:

1. Configure the master mode (DADR, CKDIV, etc.).
2. Write the MREAD bit in the TWI_MMR register at the value of the one-bit command to be sent.
3. Start the transfer by setting the QUICK bit in the TWI_CR.

Figure 38-15.SMBUS Quick Command



38.8.9 Read-write Flowcharts

The following flowcharts shown in Figure 38-17 on page 756, Figure 38-18 on page 757, Figure 38-19 on page 758, Figure 38-20 on page 759 and Figure 38-21 on page 760 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 38-16.TWI Write Operation with Single Data Byte without Internal Address

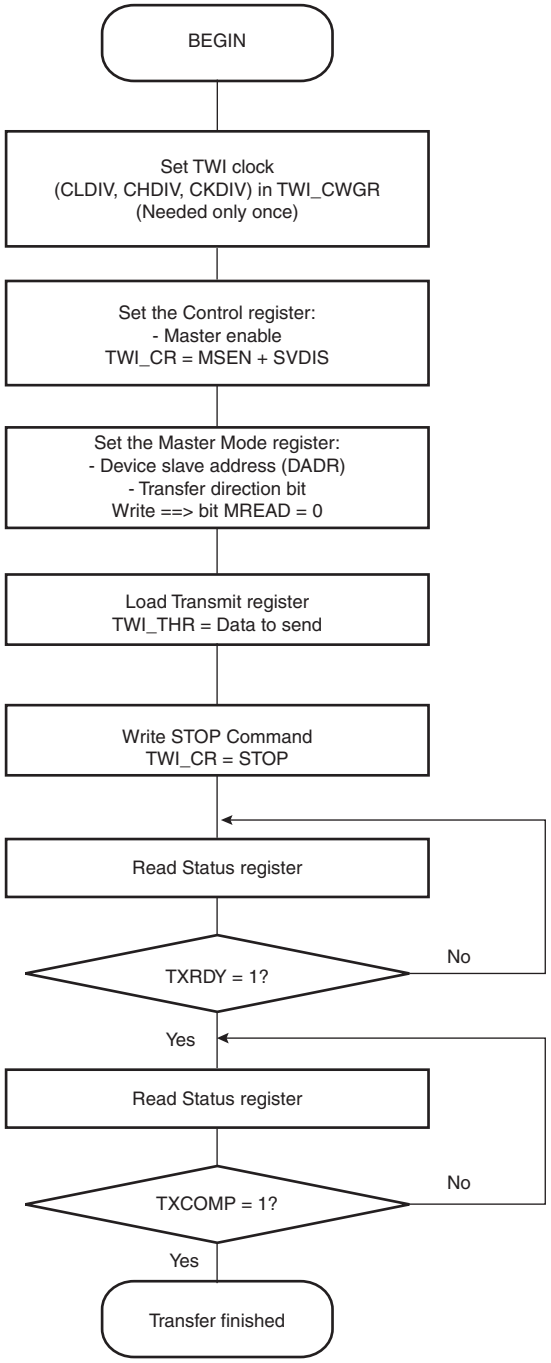


Figure 38-17.TWI Write Operation with Single Data Byte and Internal Address

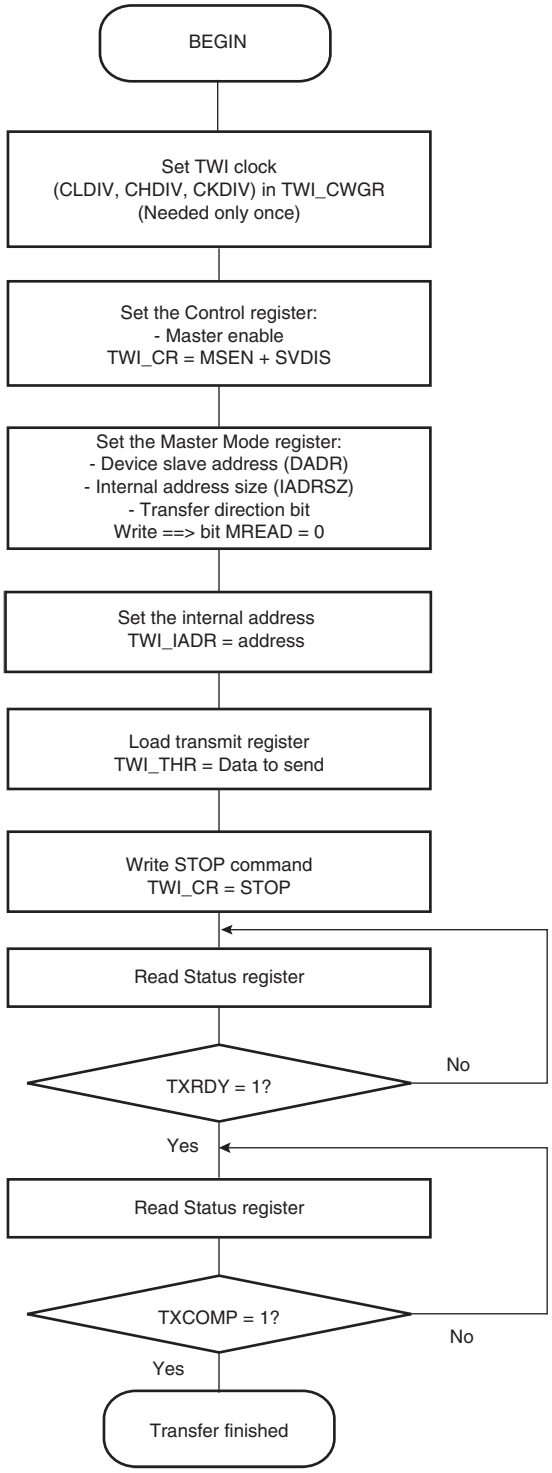


Figure 38-18.TWI Write Operation with Multiple Data Bytes with or without Internal Address

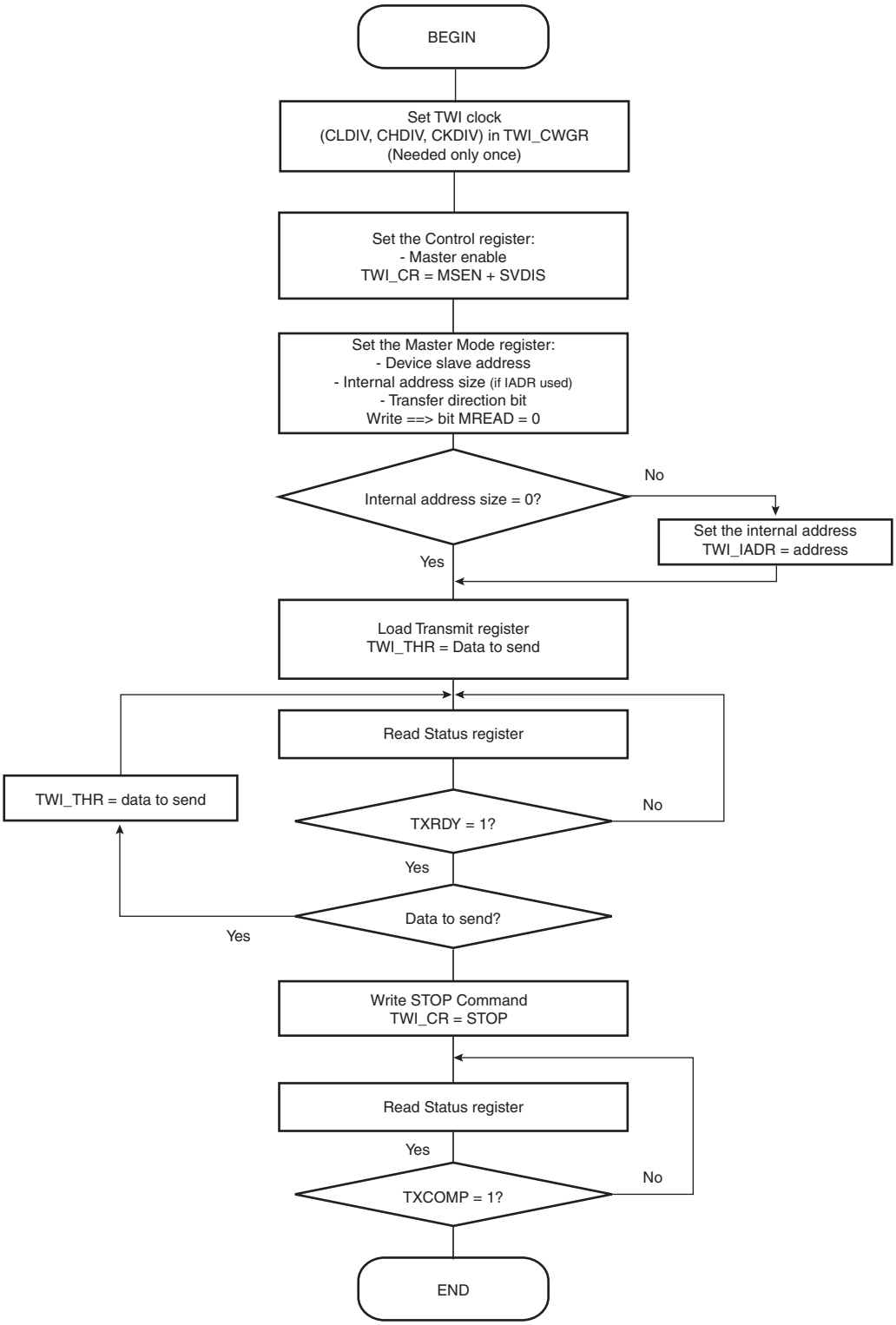


Figure 38-19.TWI Read Operation with Single Data Byte without Internal Address

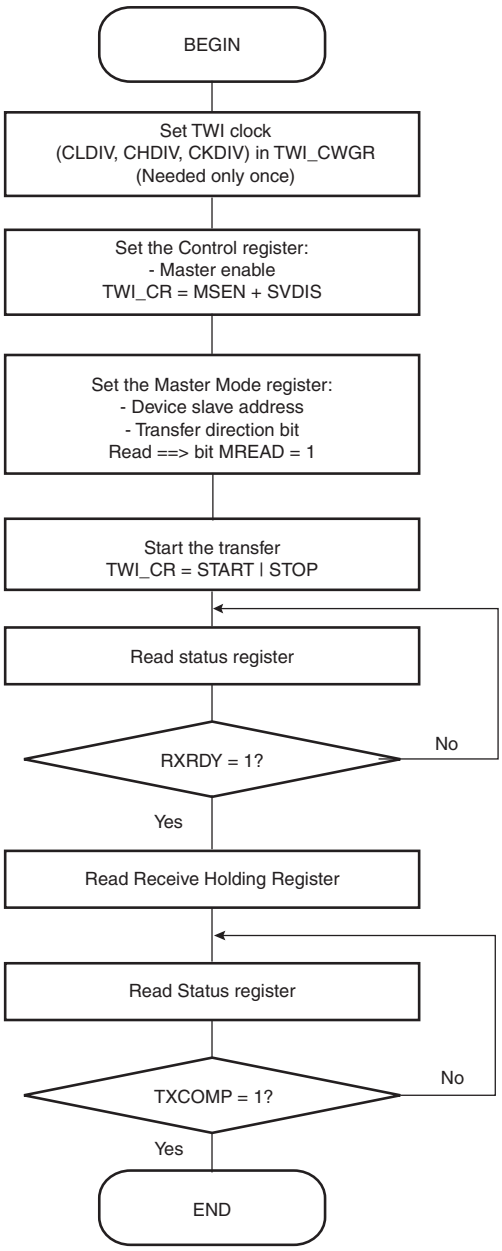


Figure 38-20.TWI Read Operation with Single Data Byte and Internal Address

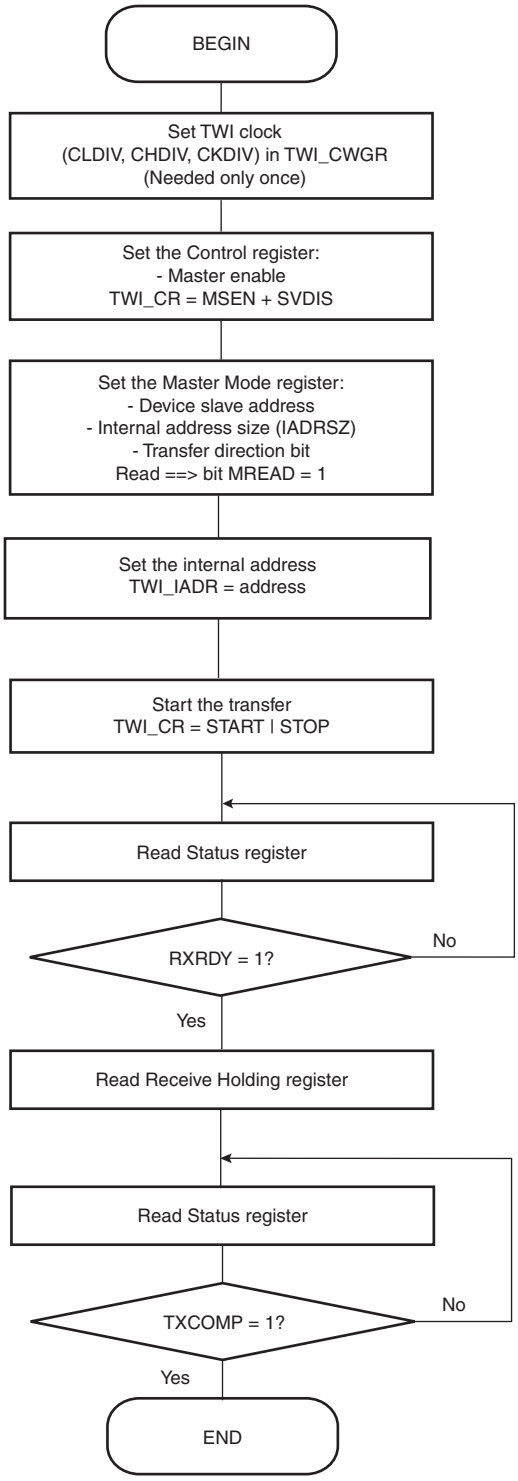
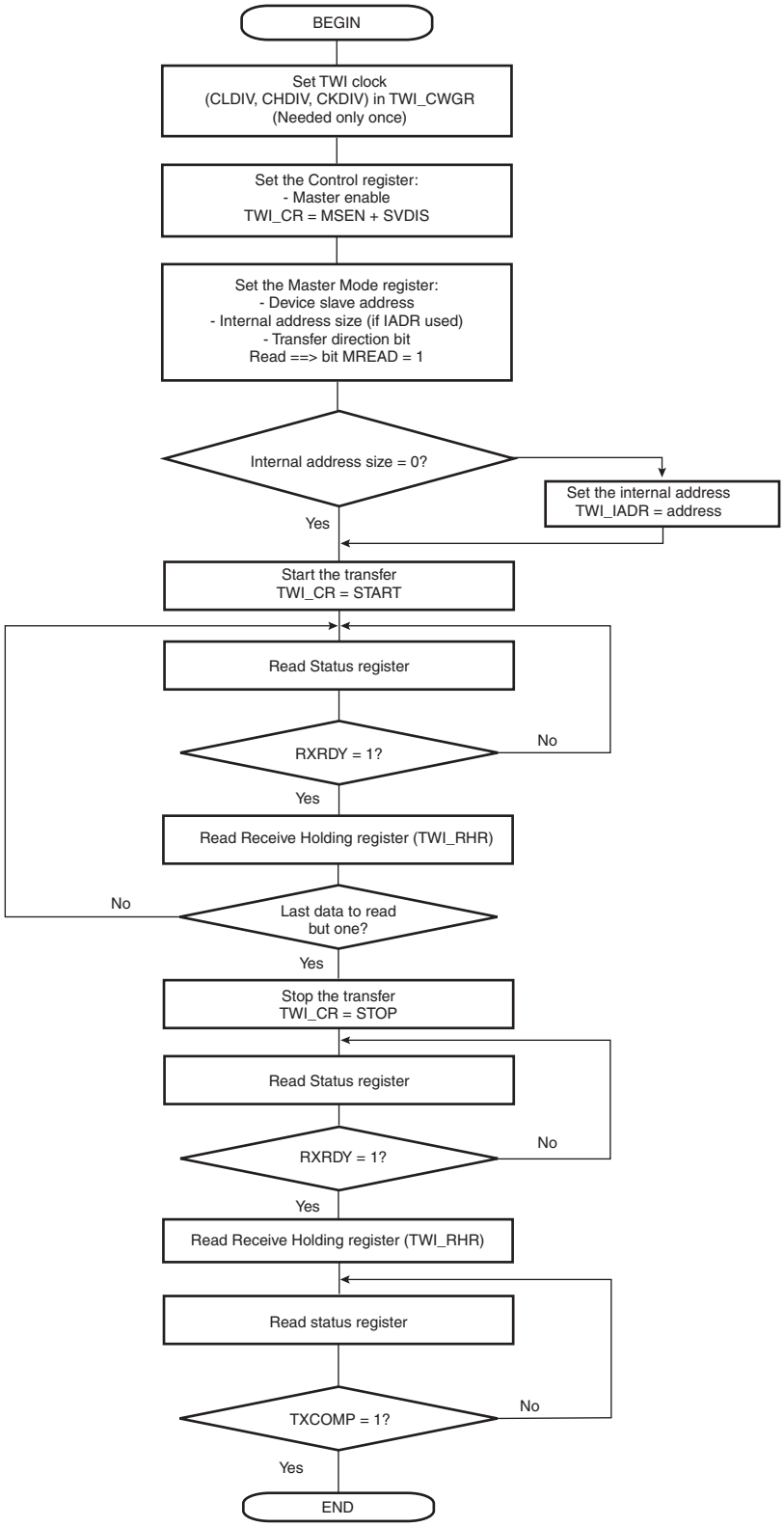


Figure 38-21.TWI Read Operation with Multiple Data Bytes with or without Internal Address



38.9 Multi-master Mode

38.9.1 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 38-23 on page 762.

38.9.2 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.

38.9.2.1 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 38-22 on page 762).

Note: The state of the bus (busy or free) is not indicated in the user interface.

38.9.2.2 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 38-22.Programmer Sends Data While the Bus is Busy

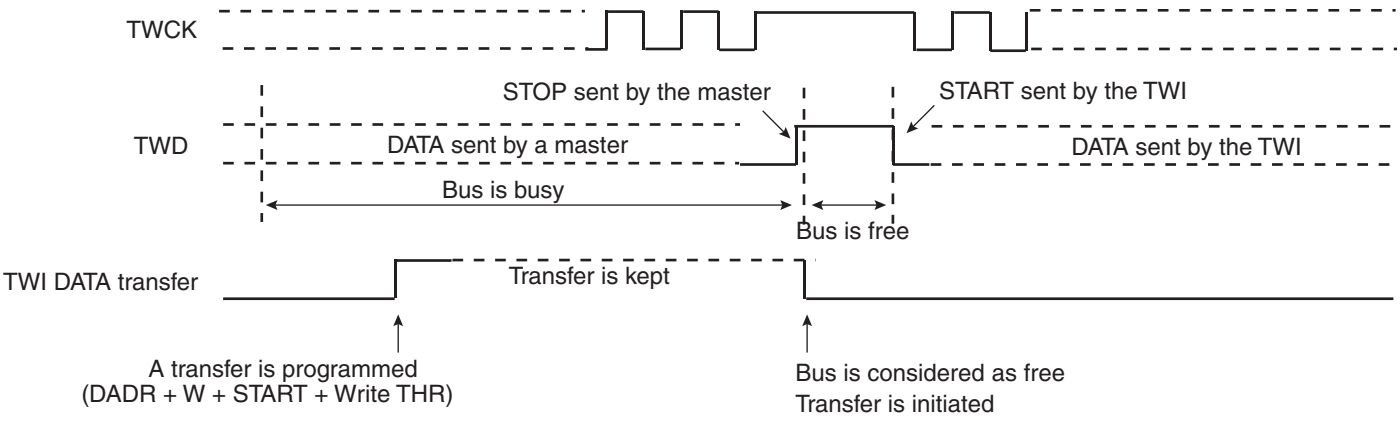
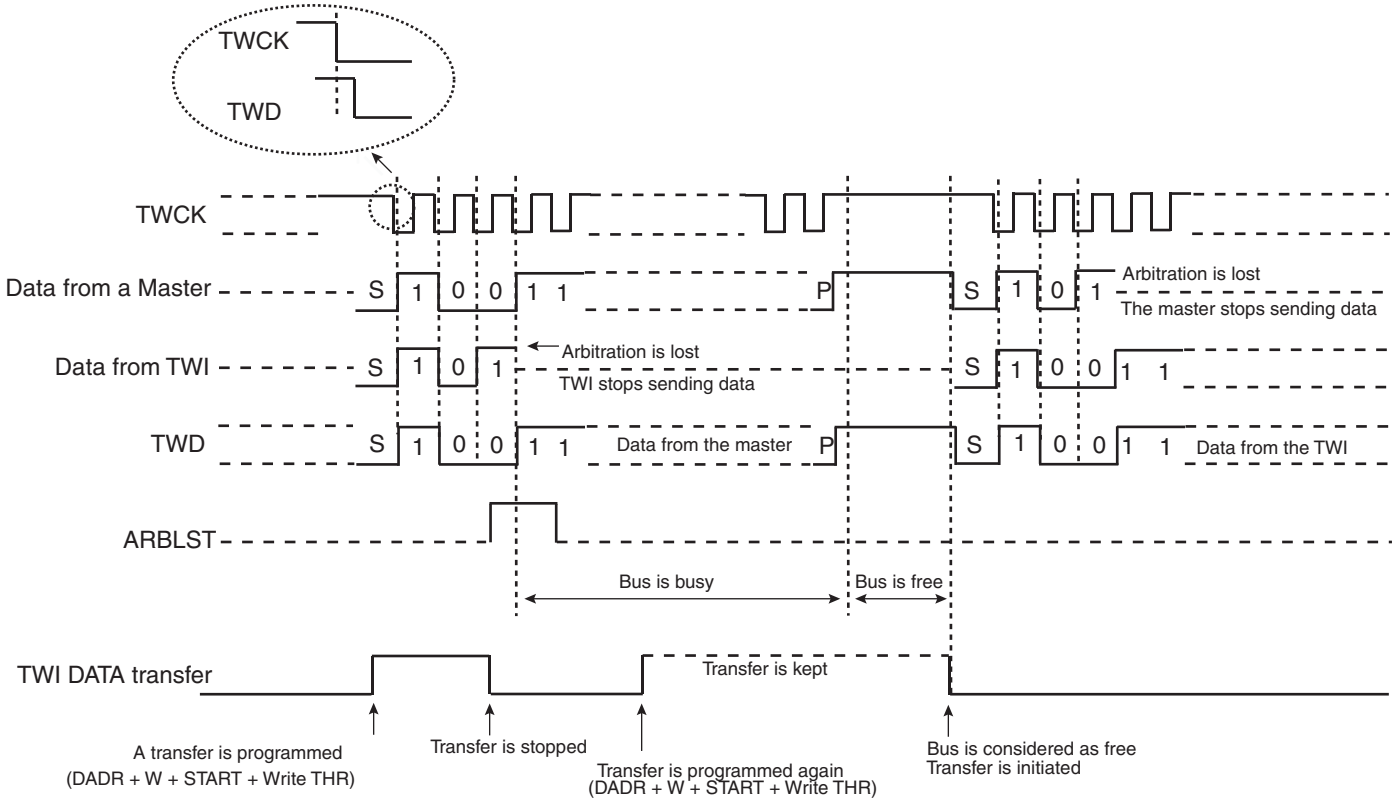
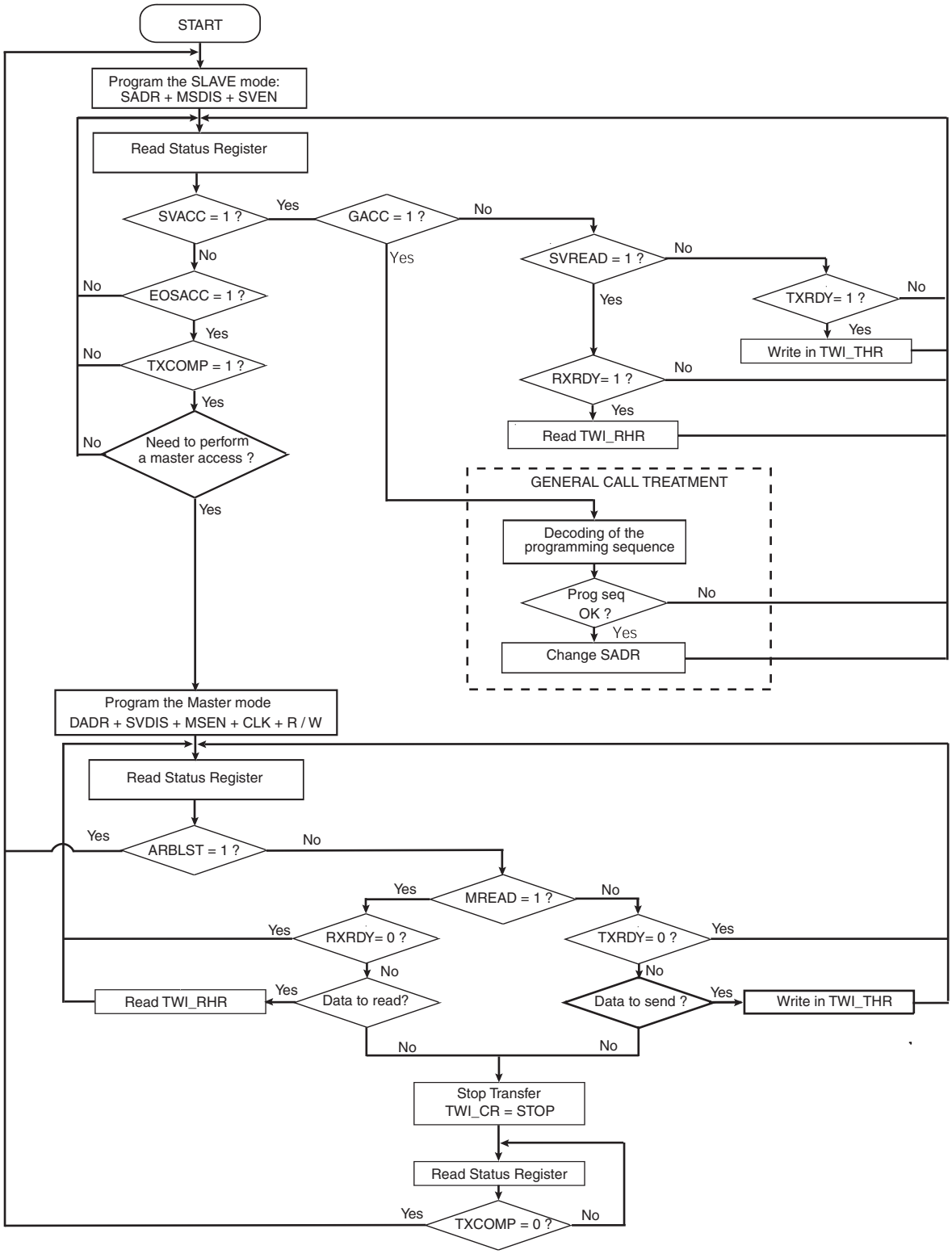


Figure 38-23.Arbitration Cases



The flowchart shown in Figure 38-24 on page 763 gives an example of read and write operations in Multi-master mode.

Figure 38-24.Multi-master Flowchart



38.10 Slave Mode

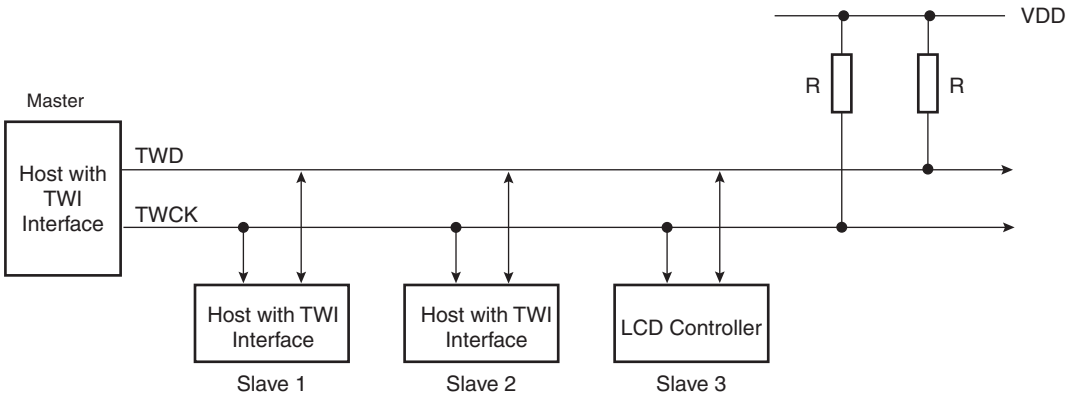
38.10.1 Definition

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).

38.10.2 Application Block Diagram

Figure 38-25.Slave Mode Typical Application Block Diagram



38.10.3 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.

38.10.4 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.

38.10.4.1 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 38-26 on page 765.

38.10.4.2 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 38-27 on page 766.

38.10.4.3 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 38-29 on page 767 and Figure 38-30 on page 768.

38.10.4.4 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 38-28 on page 766.

38.10.5 Data Transfer

38.10.5.1 Read Operation

The read mode is defined as a data requirement from the master.

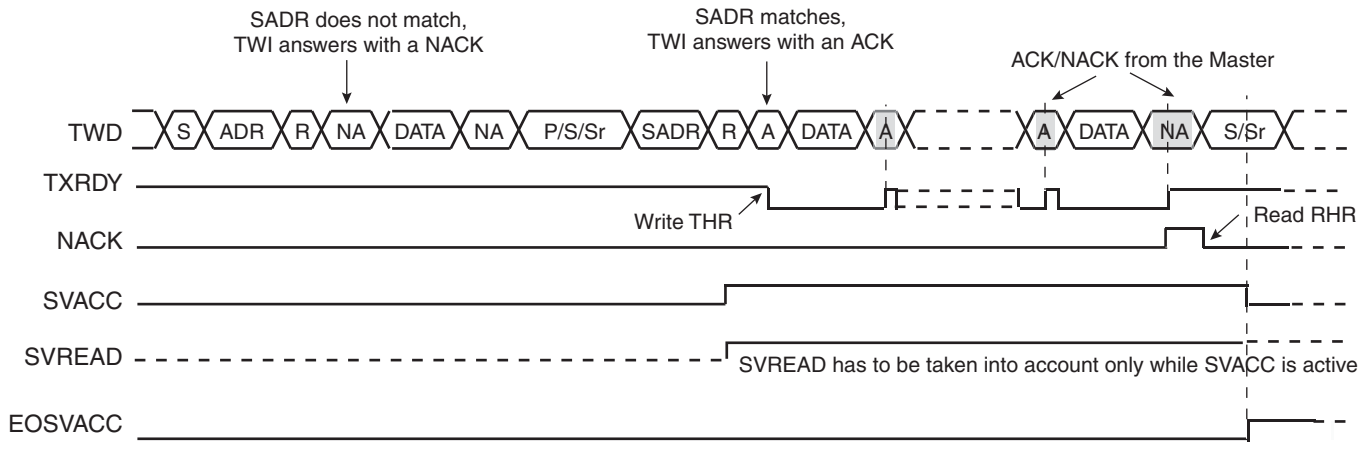
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

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 38-26 on page 765 describes the write operation.

Figure 38-26.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.

38.10.5.2 Write Operation

The write mode is defined as a data transmission from the master.

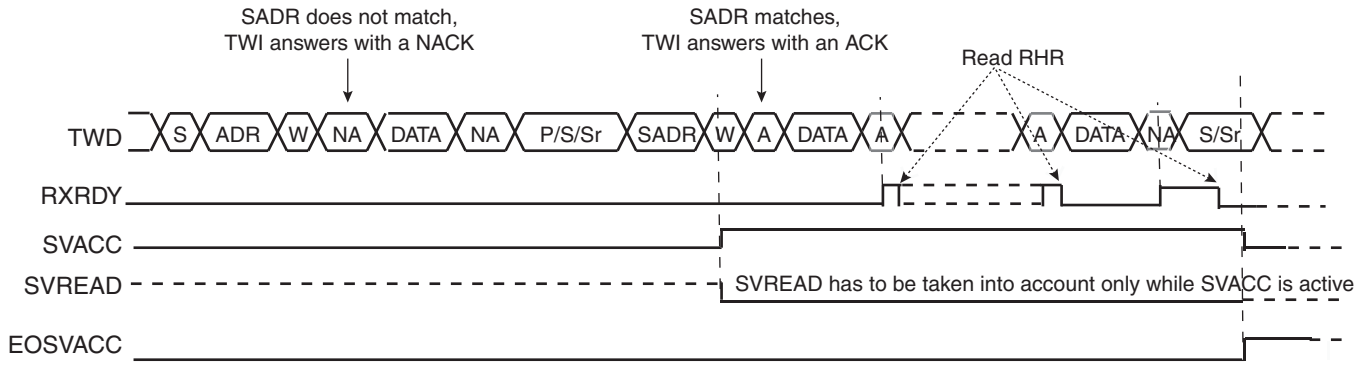
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

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 38-27 on page 766 describes the Write operation.

Figure 38-27.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.

38.10.5.3 General Call

The general call is performed in order to change the address of the slave.

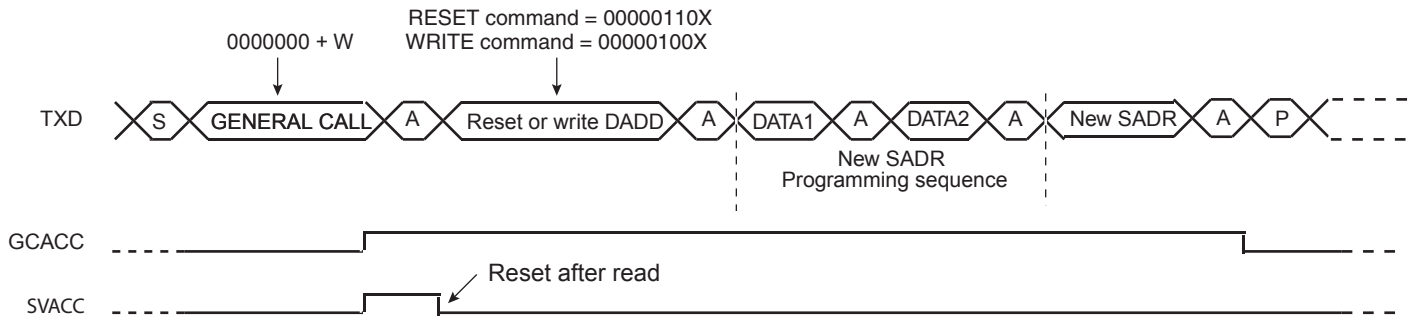
If a GENERAL CALL is detected, GACC is set.

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 38-28 on page 766 describes the General Call access.

Figure 38-28.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.

38.10.5.4 Clock Synchronization

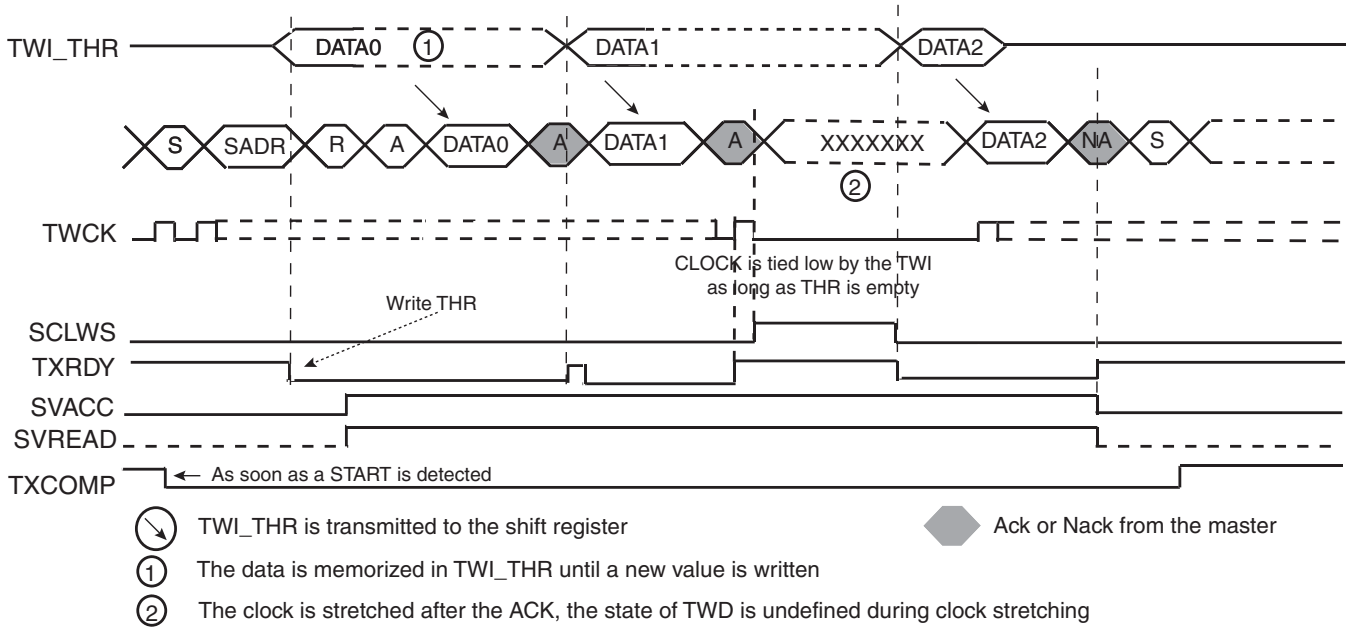
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

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 38-29 on page 767 describes the clock synchronization in Read mode.

Figure 38-29.Clock Synchronization in Read Mode



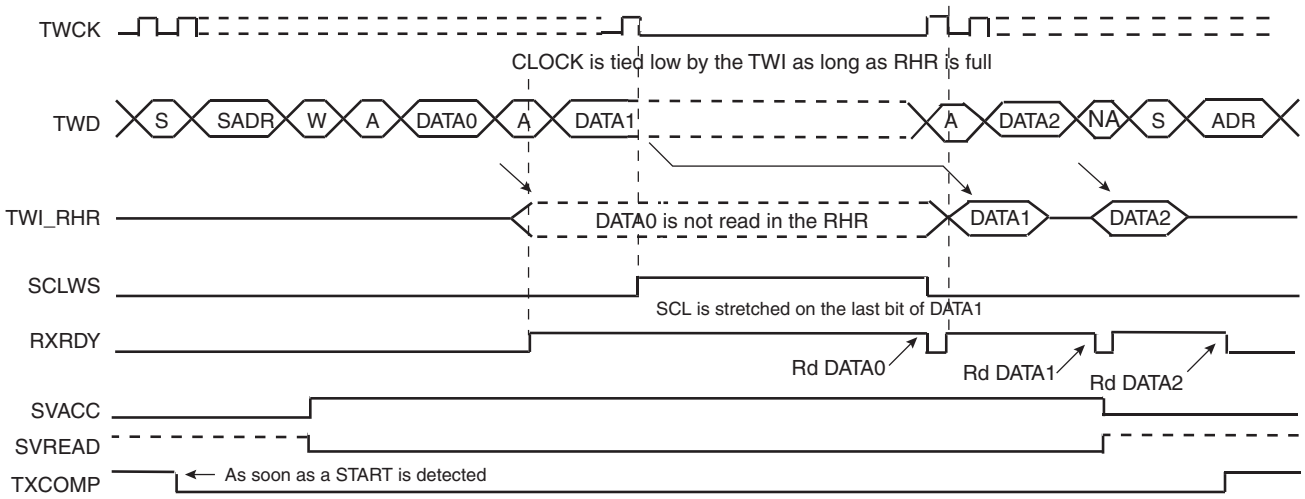
Notes:
1. TXRDY is reset when data has been written in the TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock synchronization mechanism is started.

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 38-30 on page 768 describes the clock synchronization in Read mode.

Figure 38-30.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.

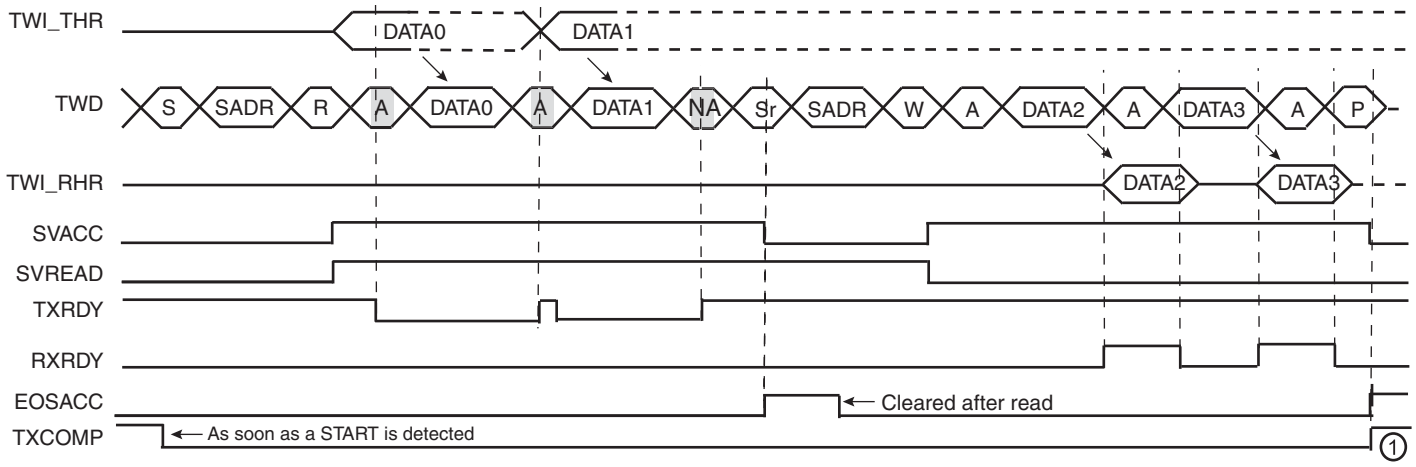
38.10.5.5 Reversal after a Repeated Start

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 38-31 on page 769 describes the repeated start + reversal from Read to Write mode.

Figure 38-31.Repeated Start + Reversal from Read to Write Mode

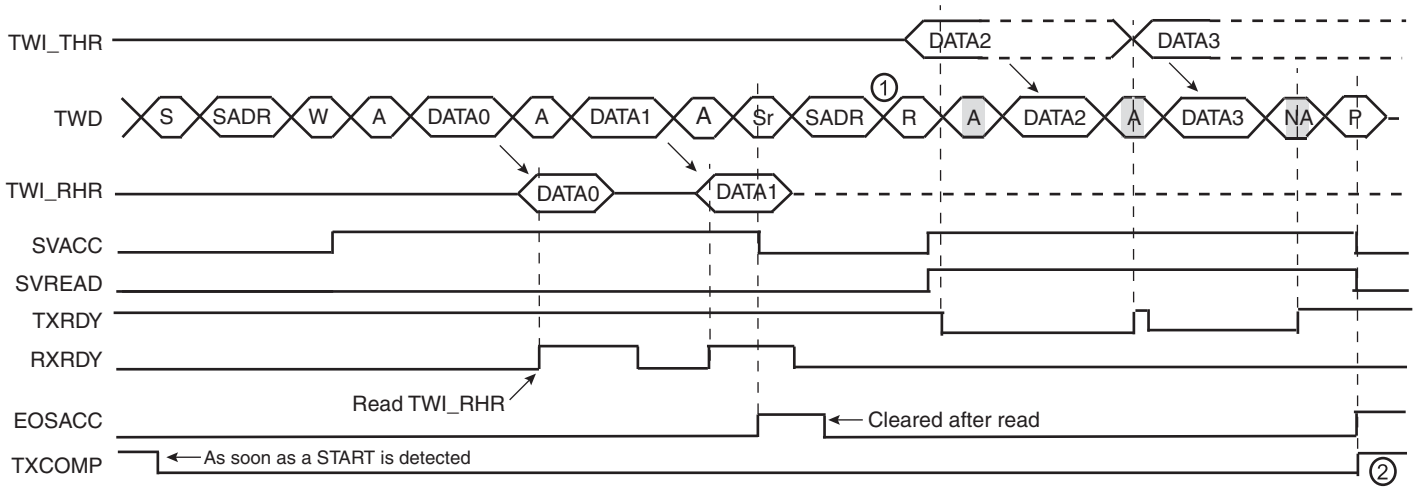


Note: 1. TXCOMP is only set at the end of the transmission because after the repeated start, SAD is detected again.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command. Figure 38-32 on page 769 describes the repeated start + reversal from Write to Read mode.

Figure 38-32.Repeated Start + Reversal from Write to Read Mode



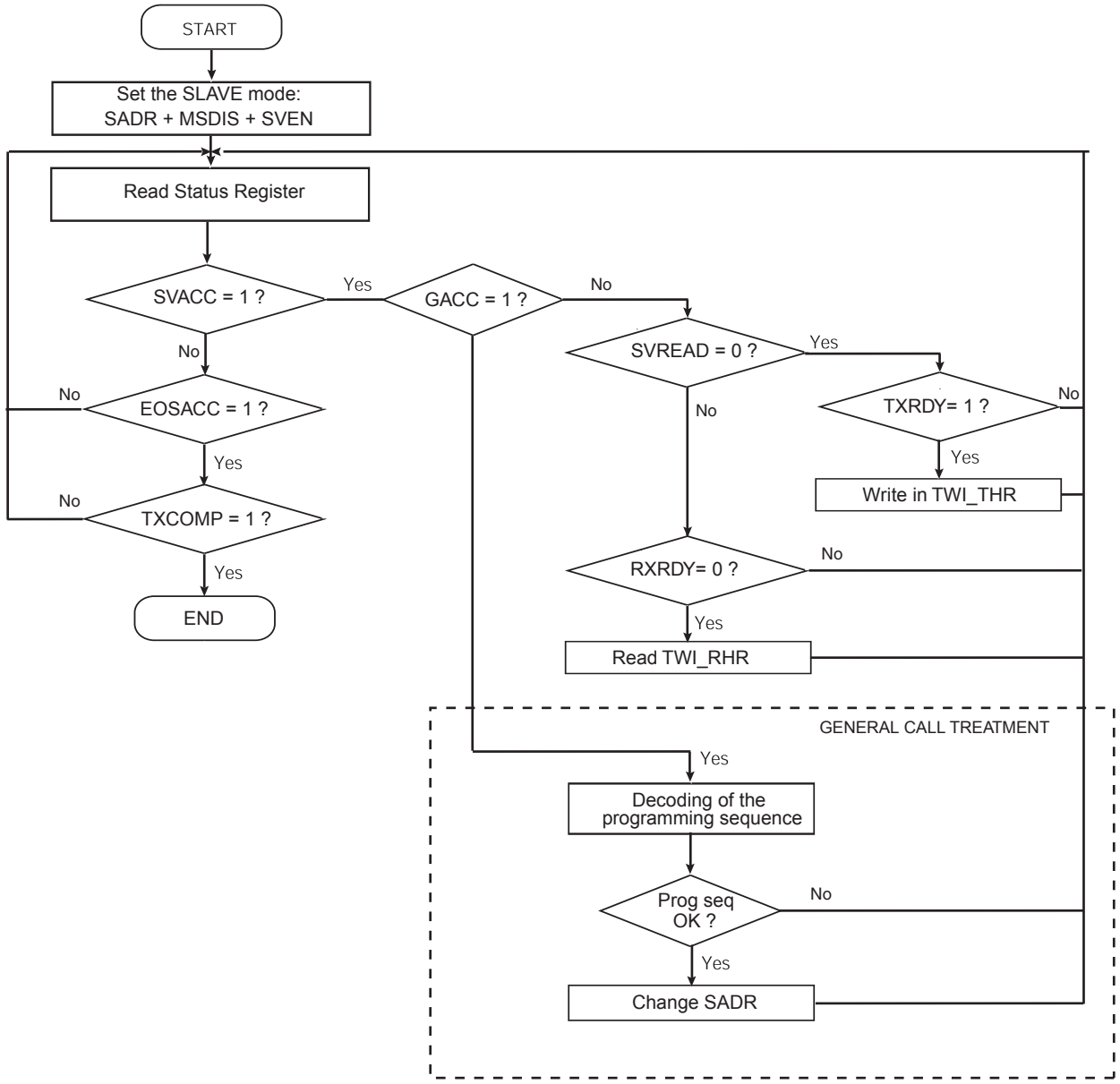
Notes: 1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.

2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

38.10.6 Read Write Flowcharts

The flowchart shown in Figure 38-33 on page 770 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 38-33.Read Write Flowchart in Slave Mode



38.11 Write Protection System

In order to bring security to the TWI, a write protection system has been implemented.

The write protection mode prevents the write of "TWI Clock Waveform Generator Register" and "TWI Slave Mode Register". When this mode is enabled and one of the protected registers is written, an error is generated in the "TWI Write Protection Status Register" and the register write request is canceled. When a write protection error occurs the WPROTERR flag is set and the address of the corresponding canceled register write is available in the WPROTADRR field of the TWI_WPROT_STATUS register.

Due to the nature of the write protection feature, enabling and disabling the write protection mode requires the use of a security code. Thus when enabling or disabling the write protection mode the SECURITY_CODE field of the "TWI Write Protection Mode Register" must be filled with the "TWI" ASCII code (0x545749) otherwise the register write will be canceled.

38.12 Two-wire Interface (TWI) User Interface

Table 38-6. 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
0x14 - 0x1C	Reserved	–	–	–
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
0xE4	Protection Mode Register	TWI_WPROT_MODE	Read-write	0x00000000
0xE8	Protection Status Register	TWI_WPROT_STATUS	Read-only	0x00000000
0xEC - 0xFC(1)	Reserved	–	–	–

Note: 1. All unlisted offset values are considered as “reserved”.

38.12.1 TWI Control Register

Name: TWI_CR

Address: 0xF8010000 (0), 0xF8014000 (1), 0xF8018000 (2)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	QUICK	SVDIS	SVEN	MSDIS	MSEN	STOP	START

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWI_THR).

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read mode.

  – In single data byte master read, the START and STOP must both be set.
  – In multiple data bytes master read, the STOP must be set after the last data received but one.
  – In master read mode, if a NACK bit is received, the STOP is automatically performed.
  – In master data write operation, a STOP condition will be sent after the transmission of the current data is finished.

- **MSEN: TWI Master Mode Enabled**

0 = No effect.

1 = If MSDIS = 0, the master mode is enabled.

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWI Master Mode Disabled**

0 = No effect.

1 = The master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWI Slave Mode Enabled**

0 = No effect.

1 = If SVDIS = 0, the slave mode is enabled.

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWI Slave Mode Disabled**

0 = No effect.

1 = The slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **QUICK: SMBUS Quick Command**

0 = No effect.

1 = If Master mode is enabled, a SMBUS Quick Command is sent.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

38.12.2 TWI Master Mode Register

Name: TWI_MMR

Address: 0xF8010004 (0), 0xF8014004 (1), 0xF8018004 (2)

Access: Read-write

Reset: 0x00000000

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

- **IADRSZ: Internal Device Address Size**

Value	Name	Description
0	NONE	No internal device address
1	1_BYTE	One-byte internal device address
2	2_BYTE	Two-byte internal device address
3	3_BYTE	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.

38.12.3 TWI Slave Mode Register

Name: TWI_SMR

Address: 0xF8010008 (0), 0xF8014008 (1), 0xF8018008 (2)

Access: Read-write

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
–	SADR						

15	14	13	12	11	10	9	8
–	–	–	–	–	–		

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in the “TWI Write Protection Mode Register”.

- **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.

38.12.4 TWI Internal Address Register

Name: TWI_IADR

Address: 0xF801000C (0), 0xF801400C (1), 0xF801800C (2)

Access: Read-write

Reset: 0x00000000

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

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

38.12.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Address: 0xF8010010 (0), 0xF8014010 (1), 0xF8018010 (2)

Access: Read-write

Reset: 0x00000000

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

This register can only be written if the WPEN bit is cleared in the "TWI Write Protection Mode Register".

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.

38.12.6 TWI Status Register

Name: TWI_SR

Address: 0xF8010020 (0), 0xF8014020 (1), 0xF8018020 (2)

Access: Read-only

Reset: 0x0000F009

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
				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 38-8 on page 750 and in Figure 38-10 on page 751.

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 38-29 on page 767, Figure 38-30 on page 768, Figure 38-31 on page 769 and Figure 38-32 on page 769.

- **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 38-10 on page 751.

RXRDY behavior in Slave mode can be seen in Figure 38-27 on page 766, Figure 38-30 on page 768, Figure 38-31 on page 769 and Figure 38-32 on page 769.

- **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 38.8.4 on page 748.

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 38-26 on page 765, Figure 38-29 on page 767, Figure 38-31 on page 769 and Figure 38-32 on page 769.

- **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 38-26 on page 765, Figure 38-27 on page 766, Figure 38-31 on page 769 and Figure 38-32 on page 769.

- **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 38-26 on page 765, Figure 38-27 on page 766, Figure 38-31 on page 769 and Figure 38-32 on page 769.

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0 = No General Call has been detected.

1 = A General Call has been detected. After the detection of General Call, if need be, the programmer may acknowledge this access and decode the following bytes and respond according to the value of the bytes.

GACC behavior can be seen in Figure 38-28 on page 766.

- **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 38-29 on page 767 and Figure 38-30 on page 768.

- **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 38-31 on page 769 and Figure 38-32 on page 769

38.12.7 TWI Interrupt Enable Register

Name: TWI_IER

Address: 0xF8010024 (0), 0xF8014024 (1), 0xF8018024 (2)

Access: Write-only

Reset: 0x00000000

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

38.12.8 TWI Interrupt Disable Register

Name: TWI_IDR

Address: 0xF8010028 (0), 0xF8014028 (1), 0xF8018028 (2)

Access: Write-only

Reset: 0x00000000

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

38.12.9 TWI Interrupt Mask Register

Name: TWI_IMR

Address: 0xF801002C (0), 0xF801402C (1), 0xF801802C (2)

Access: Read-only

Reset: 0x00000000

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

38.12.10TWI Receive Holding Register

Name: TWI_RHR

Address: 0xF8010030 (0), 0xF8014030 (1), 0xF8018030 (2)

Access: Read-only

Reset: 0x00000000

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

- **RXDATA: Master or Slave Receive Holding Data**

38.12.11TWI Transmit Holding Register

Name: TWI_THR

Address: 0xF8010034 (0), 0xF8014034 (1), 0xF8018034 (2)

Access: Read-write

Reset: 0x00000000

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

- **TXDATA: Master or Slave Transmit Holding Data**

38.12.12TWI Write Protection Mode Register

Name: TWI_WPROT_MODE

Address: 0xF80100E4 (0), 0xF80140E4 (1), 0xF80180E4 (2)

Access: Read-write

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

- **SECURITY_CODE: Write protection mode security code**

This security code is needed to set/reset the WPROT bit value (see Section 38.11 “Write Protection System” for details).

Must be filled with **0x545749** (ASCII code for TWI).

- **WPROT: Write protection bit**

Enables/Disables write protection mode.

The write protected registers are:

“TWI Clock Waveform Generator Register”

“TWI Slave Mode Register”

38.12.13TWI Write Protection Status Register

Name: TWI_WPROT_STATUS

Address: 0xF80100E8 (0), 0xF80140E8 (1), 0xF80180E8 (2)

Access: Read-only

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

- **WPROTADDR: Write Protection Error Address**

Indicates the address of the register write request which generated the error.

- **WPROTERR: Write Protection Error**

Indicates a write protection error.

39. Universal Synchronous Asynchronous Receiver Transmitter (USART)

39.1 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485, LIN, and SPI 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 DMA Controller, which enables data transfers to the transmitter and from the receiver. The DMAC provides chained buffer management without any intervention of the processor.

39.2 Embedded Characteristics

- Programmable Baud Rate Generator
- 5-bit to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB-first or LSB-first
  - Optional Break Generation and Detection
  - By-8 or by-16 Over-sampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- 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
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to Internal Clock Frequency MCK/6
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 specifications
  - Master or Slave
  - Processing of frames with up to 256 data bytes
  - Response Data length can be configurable or defined automatically by the Identifier
  - Self synchronization in Slave node configuration
  - Automatic processing and verification of the “Synch Break” and the “Synch Field”
  - The “Synch Break” is detected even if it is partially superimposed with a data byte
  - Automatic Identifier parity calculation/sending and verification
  - Parity sending and verification can be disabled
  - Automatic Checksum calculation/sending and verification
  - Checksum sending and verification can be disabled
  - Support both “Classic” and “Enhanced” checksum types
  - Full LIN error checking and reporting
  - Frame Slot Mode: the Master allocates slots to the scheduled frames automatically.
  - Generation of the Wakeup signal
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two DMA Controller Channels (DMAC)
- Offers Buffer Transfer without Processor Intervention

39.3 Block Diagram

Figure 39-1. USART Block Diagram

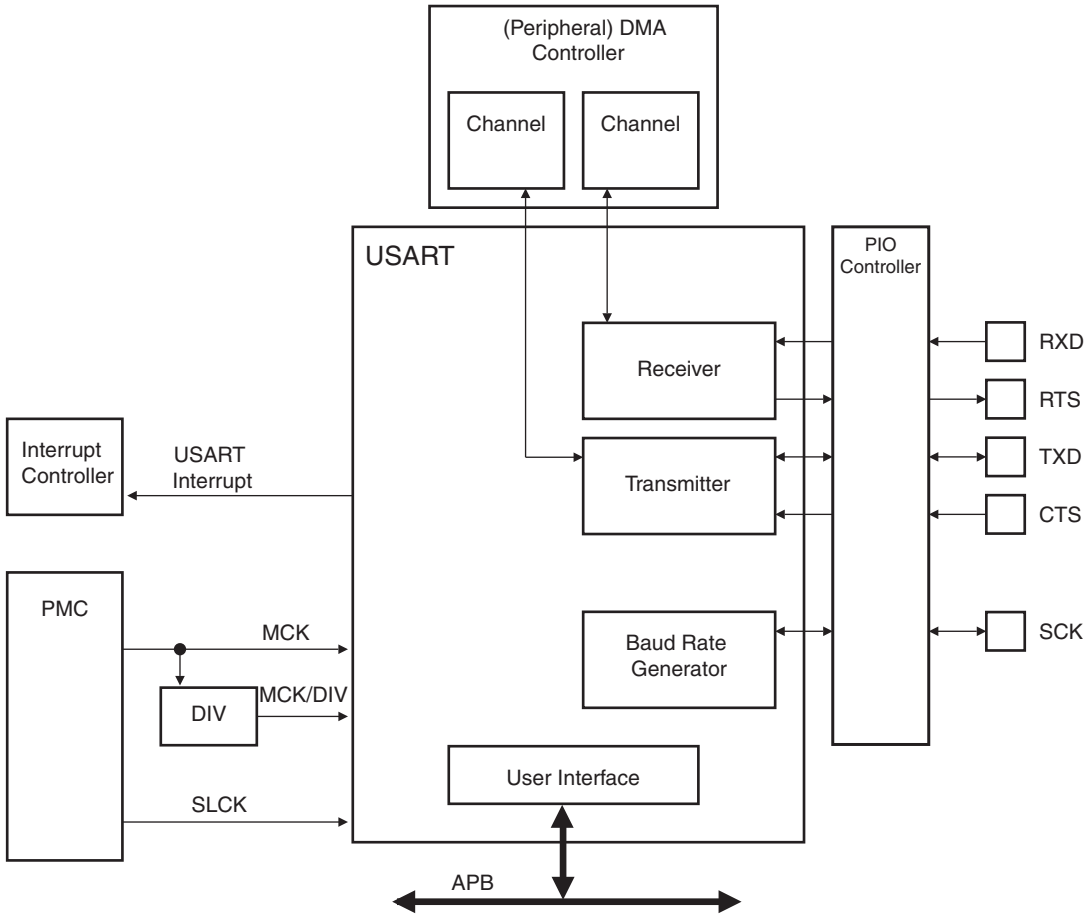
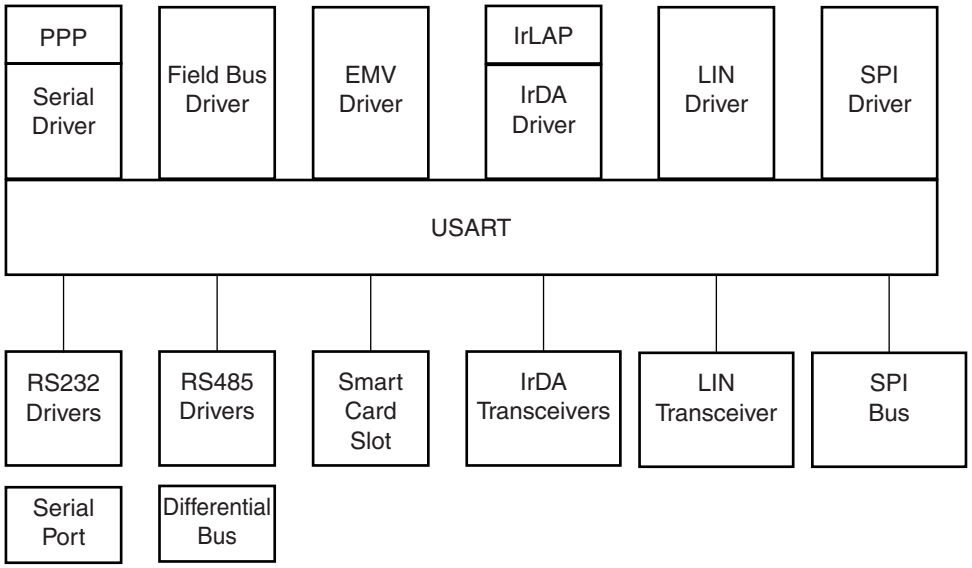


Table 39-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
RXD	RXD	MOSI	MISO
TXD	TXD	MISO	MOSI
RTS	RTS	–	CS
CTS	CTS	CS	–

39.4 Application Block Diagram

Figure 39-2. Application Block Diagram



39.5 I/O Lines Description

Table 39-2. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	
TXD	Transmit Serial Data or Master Out Slave In (MOSI) in SPI Master Mode or Master In Slave Out (MISO) in SPI Slave Mode	I/O	
RXD	Receive Serial Data or Master In Slave Out (MISO) in SPI Master Mode or Master Out Slave In (MOSI) in SPI Slave Mode	Input	
CTS	Clear to Send or Slave Select (NSS) in SPI Slave Mode	Input	Low
RTS	Request to Send or Slave Select (NSS) in SPI Master Mode	Output	Low

39.6 Product Dependencies

39.6.1 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.

Table 39-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PA3	A
USART0	RTS0	PA2	A
USART0	RXD0	PA1	A
USART0	SCK0	PA4	A
USART0	TXD0	PA0	A
USART1	CTS1	PC28	C
USART1	RTS1	PC27	C
USART1	RXD1	PA6	A
USART1	SCK1	PC29	C
USART1	TXD1	PA5	A
USART2	CTS2	PB1	B
USART2	RTS2	PB0	B
USART2	RXD2	PA8	A
USART2	SCK2	PB2	B
USART2	TXD2	PA7	A

Table 39-3. I/O Lines

USART3	CTS3	PC25	B
USART3	RTS3	PC24	B
USART3	RXD3	PC23	B
USART3	SCK3	PC26	B
USART3	TXD3	PC22	B

39.6.2 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.

39.6.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Interrupt Controller. using the USART interrupt requires the Interrupt Controller to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

Table 39-4. Peripheral IDs

Instance	ID
USART0	5
USART1	6
USART2	7
USART3	8

39.7 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5-bit to 9-bit full-duplex asynchronous serial communication
  - MSB-first 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-first 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
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to Internal Clock Frequency MCK/6
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 specifications
  - Master or Slave
  - Processing of frames with up to 256 data bytes
  - Response Data length can be configurable or defined automatically by the Identifier
  - Self synchronization in Slave node configuration
  - Automatic processing and verification of the “Synch Break” and the “Synch Field”
  - The “Synch Break” is detected even if it is partially superimposed with a data byte
  - Automatic Identifier parity calculation/sending and verification
  - Parity sending and verification can be disabled
  - Automatic Checksum calculation/sending and verification
  - Checksum sending and verification can be disabled
  - Support both “Classic” and “Enhanced” checksum types
  - Full LIN error checking and reporting
  - Frame Slot Mode: the Master allocates slots to the scheduled frames automatically.
  - Generation of the Wakeup signal
- Test modes
  - Remote loopback, local loopback, automatic echo

39.7.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

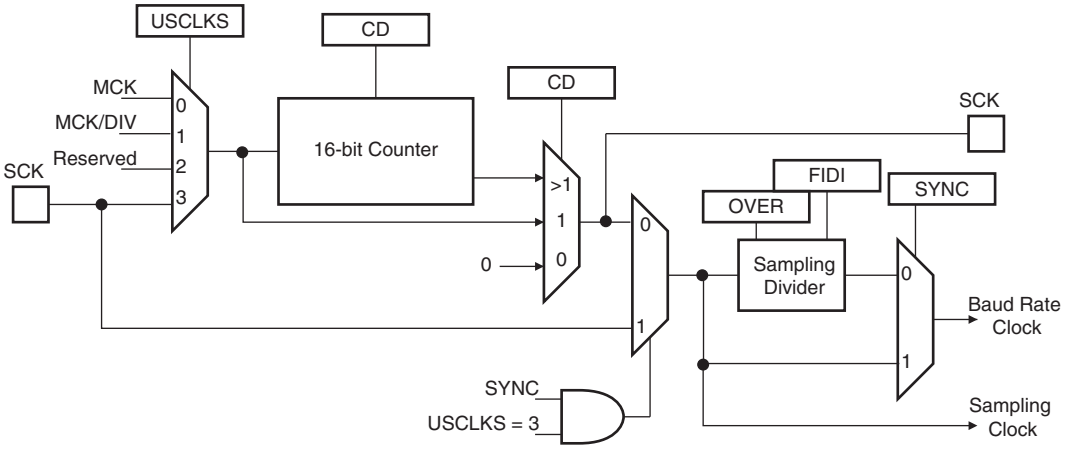
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- 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 to 0, the Baud Rate Generator does not generate any clock. If CD is programmed to 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 in USART mode, or 6 times lower in SPI mode.

Figure 39-3. Baud Rate Generator



39.7.1.1 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 to 1.

Baud Rate Calculation Example

Table 39-5 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 39-5. 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%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

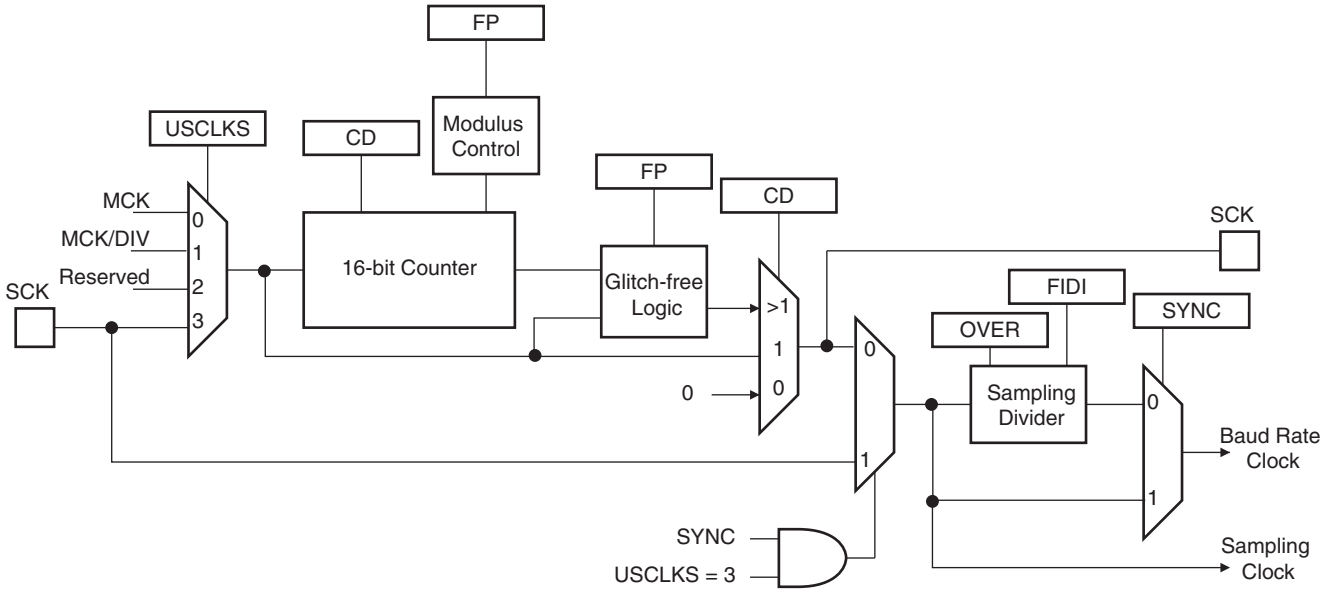
39.7.1.2 Fractional Baud Rate in Asynchronous Mode

The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain Baud Rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the Baud Rate Generator Register (US_BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2-Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

Figure 39-4. Fractional Baud Rate Generator



39.7.1.3 Baud Rate in Synchronous Mode or SPI 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. In synchronous mode master (USCLKS = 0 or 1, CLK0 set to 1), the receive part limits the SCK maximum frequency to MCK/4.5 in USART mode, or MCK/6 in SPI mode.

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.

39.7.1.4 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 39-6.

Table 39-6. 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 39-7.

Table 39-7. 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 39-8 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 39-8. 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

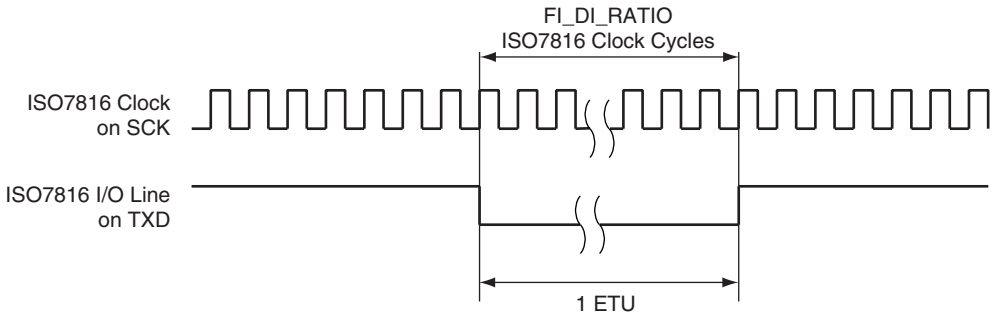
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. 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 39-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 39-5. Elementary Time Unit (ETU)



39.7.2 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.

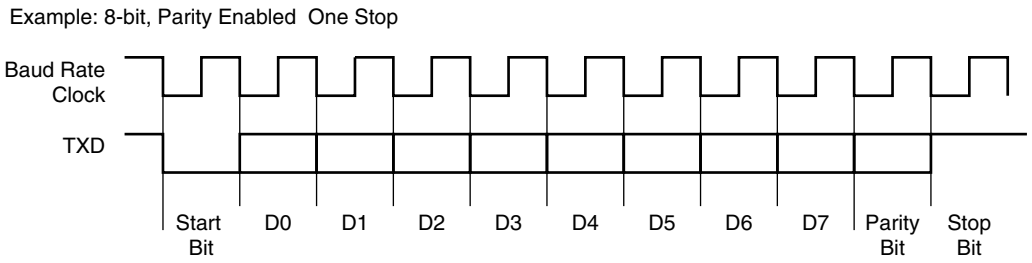
39.7.3 Synchronous and Asynchronous Modes

39.7.3.1 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 to 1, the most significant bit is sent first. If written to 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.

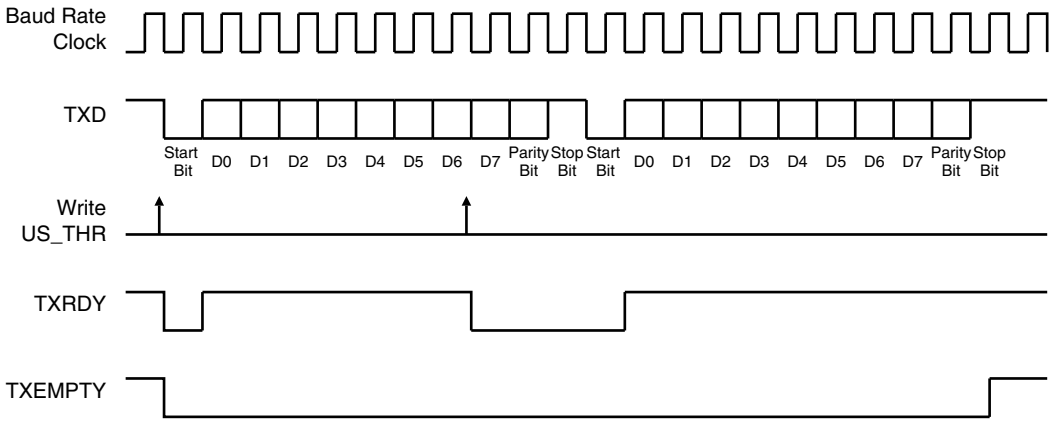
Figure 39-6. Character Transmit



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.

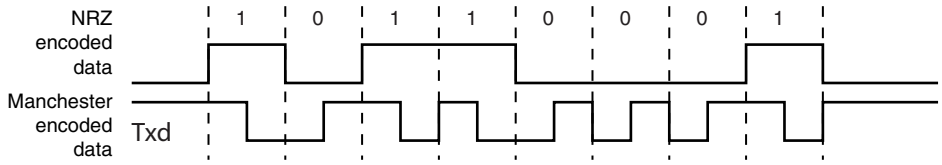
Figure 39-7. Transmitter Status



39.7.3.2 Manchester Encoder

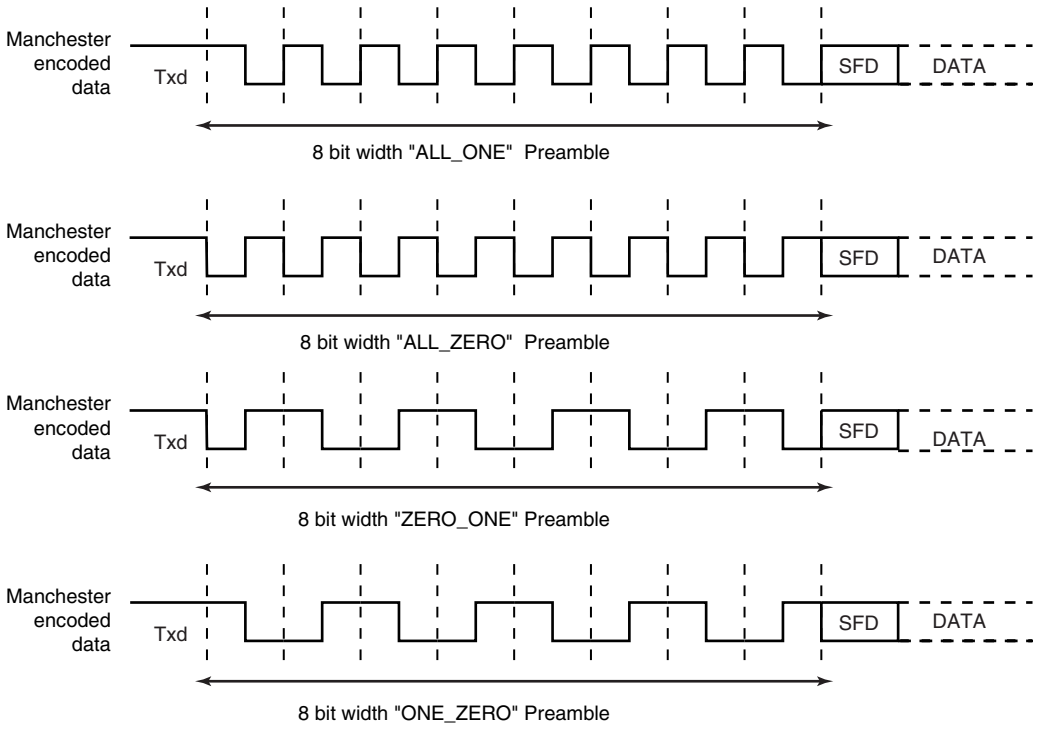
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero 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 39-8 illustrates this coding scheme.

Figure 39-8. NRZ to Manchester Encoding



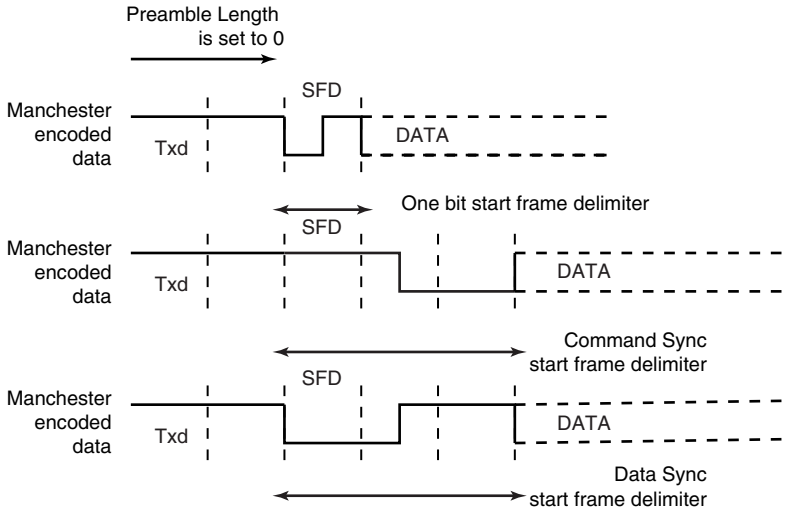
The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble 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 39-9 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 39-9. 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 39-10 illustrates these patterns. If the start frame delimiter, also known as the start bit, is one bit, (ONEBIT to 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 to 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.

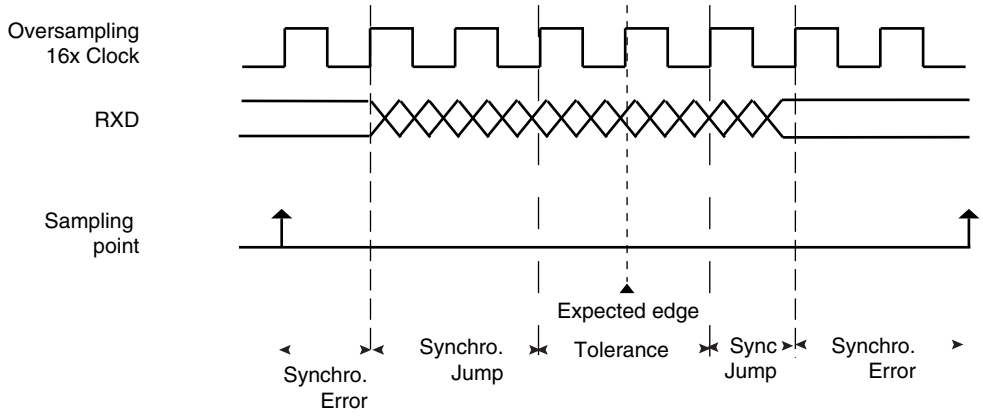
Figure 39-10.Start Frame Delimiter



Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the USART_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 39-11.Bit Resynchronization



39.7.3.3 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US_MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time to 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 to 0), a start is detected at the eighth sample to 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER to 1), a start bit is detected at the fourth sample to 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 39-12 and Figure 39-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 39-12.Asynchronous Start Detection

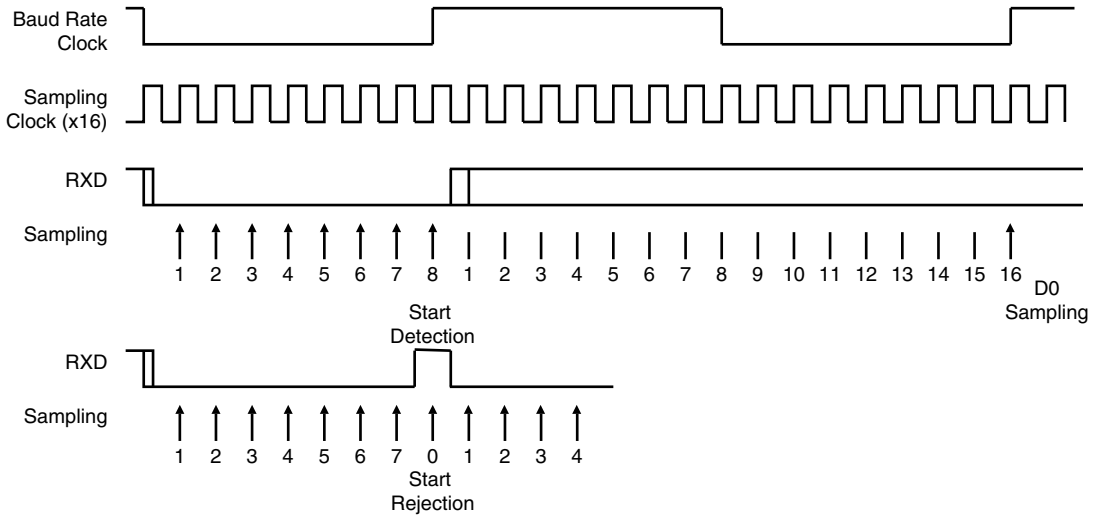
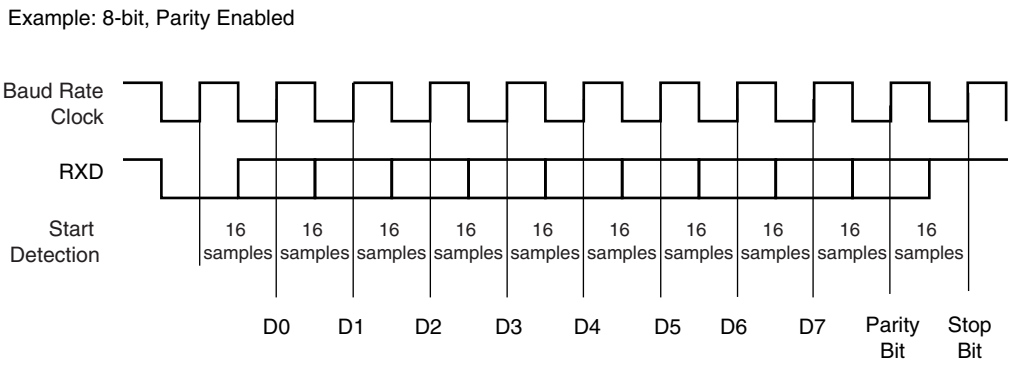


Figure 39-13.Asynchronous Character Reception



39.7.3.4 Manchester Decoder

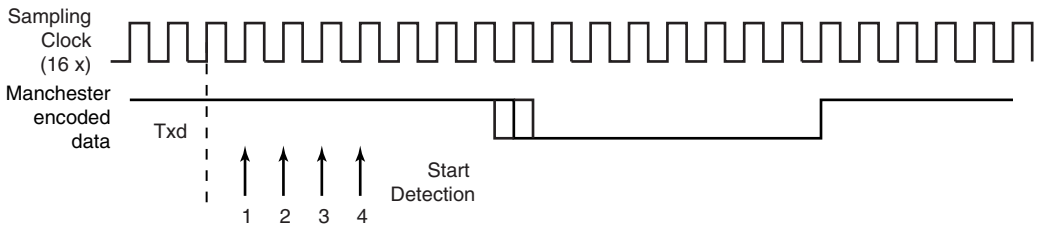
When the MAN field in US_MR register is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in US_MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL field in US_MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in US_MAN. See Figure 39-9 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 to zero, a start bit is detected. See Figure 39-14. The sample pulse rejection mechanism applies.

In order to increase the compatibility the RXIDLV bit in the US_MAN register allows to inform the USART block of the Rx line idle state value (Rx line undriven), it can be either level one (pull-up) or level zero (pull-down). By default this bit is set to one (Rx line is at level 1 if undriven).

Figure 39-14.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 39-15 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 to 1. See Figure 39-16 for an example of Manchester error detection during data phase.

Figure 39-15.Preamble Pattern Mismatch

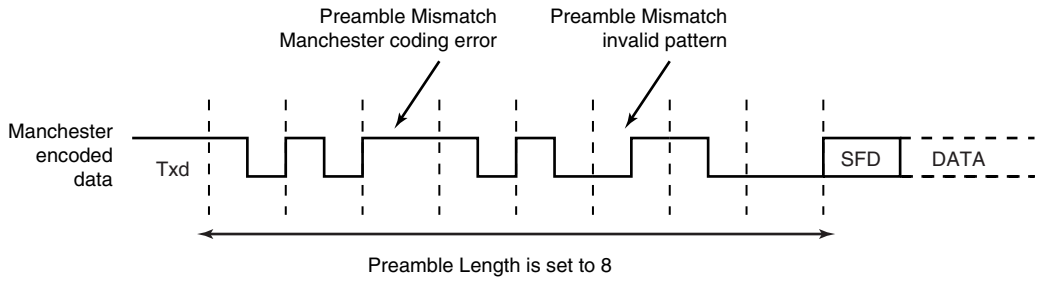
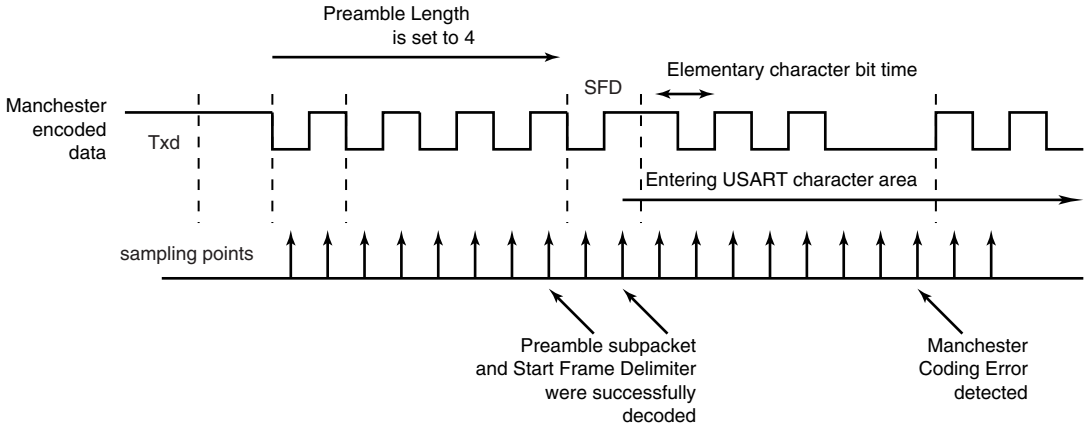


Figure 39-16.Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field to 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.

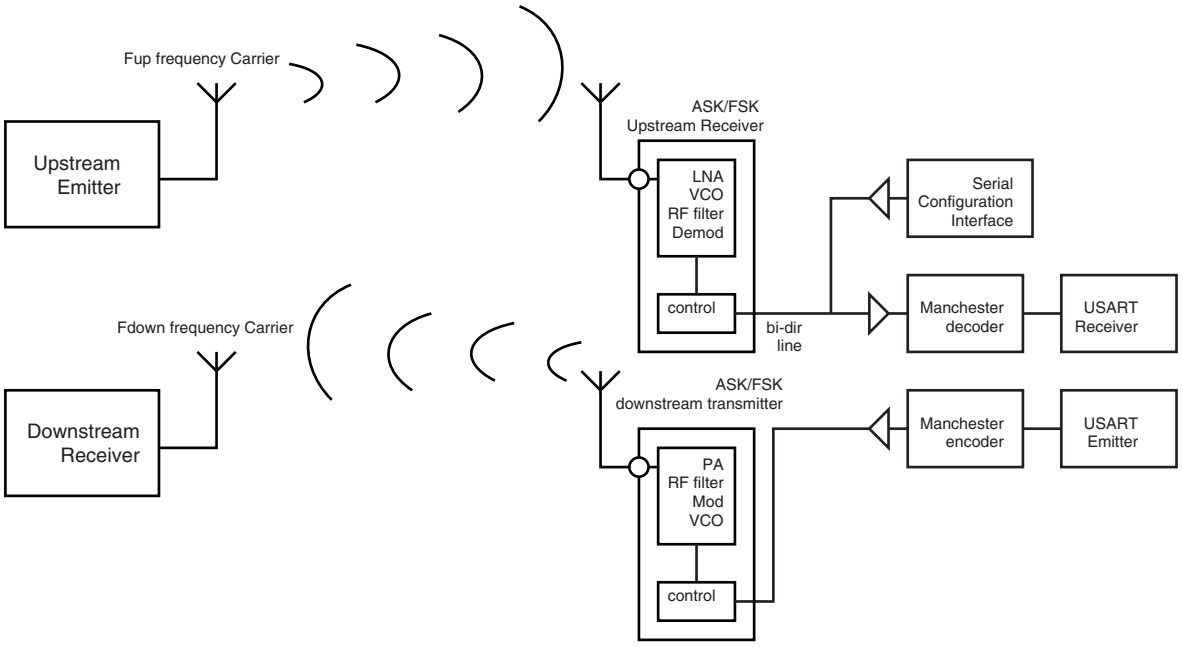
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

39.7.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 39-17.

Figure 39-17.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 39-18 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 39-19.

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 39-18.ASK Modulator Output

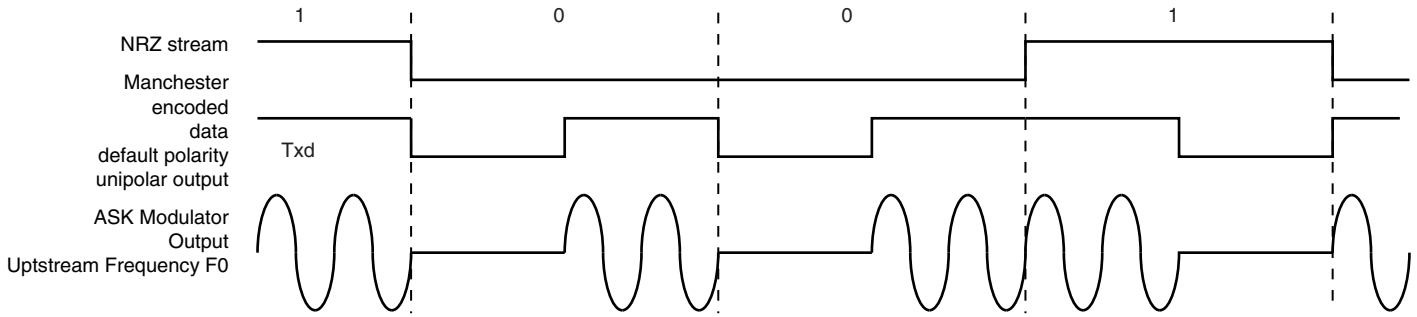
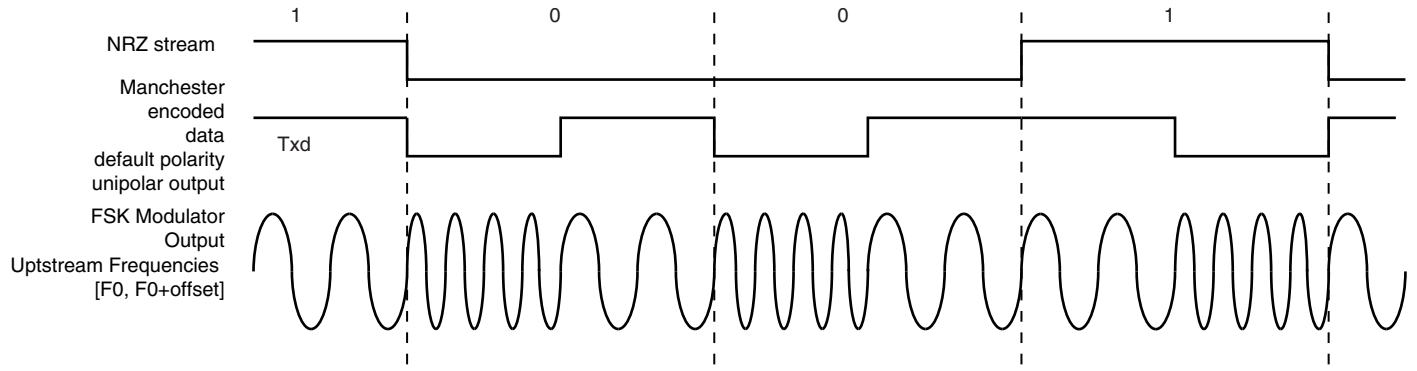


Figure 39-19.FSK Modulator Output



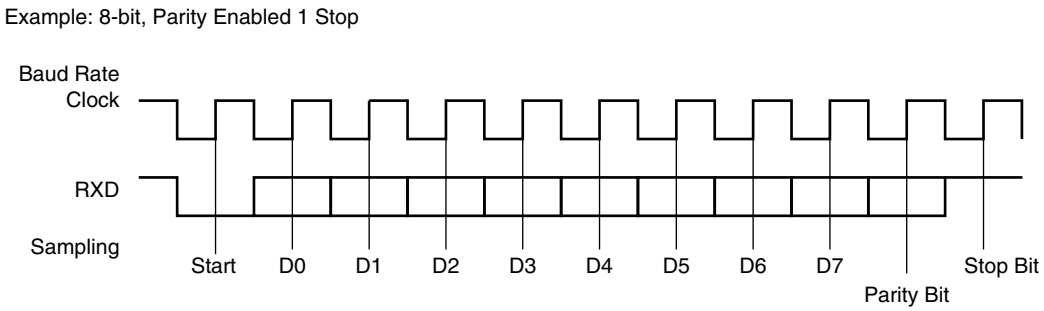
39.7.3.6 Synchronous Receiver

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 39-20 illustrates a character reception in synchronous mode.

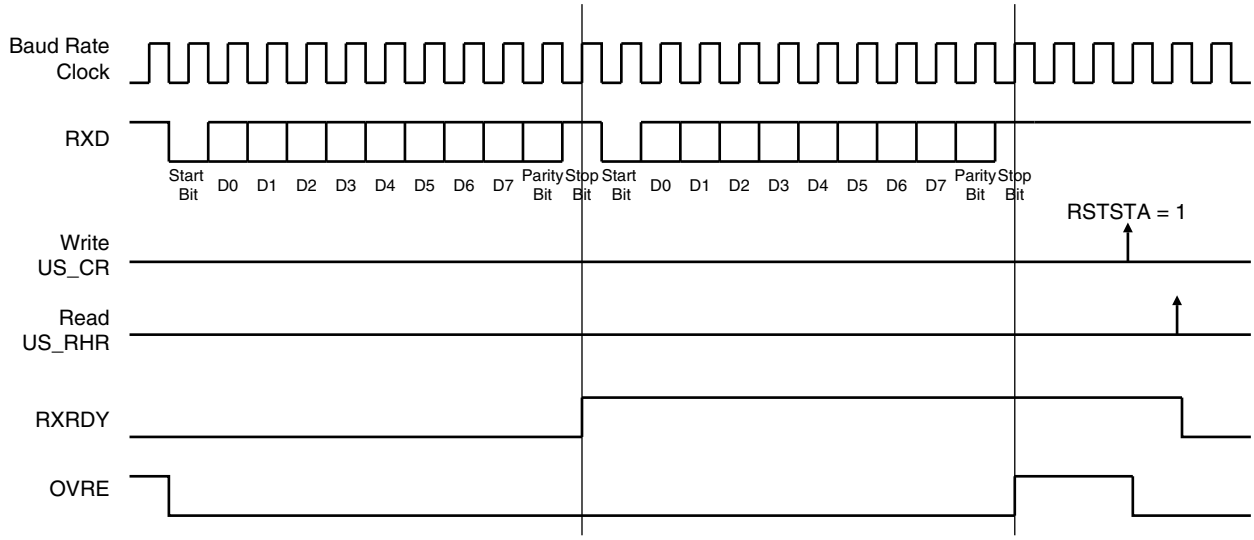
Figure 39-20.Synchronous Mode Character Reception



39.7.3.7 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit to 1.

Figure 39-21.Receiver Status



39.7.3.8 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see “Multidrop Mode” on page 809. 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 to 0 if a number of 1s in the character data bit is even, and to 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 to 1 if a number of 1s in the character data bit is even, and to 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 to 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 0. If the space parity is used, the parity generator of the transmitter drives the parity bit to 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

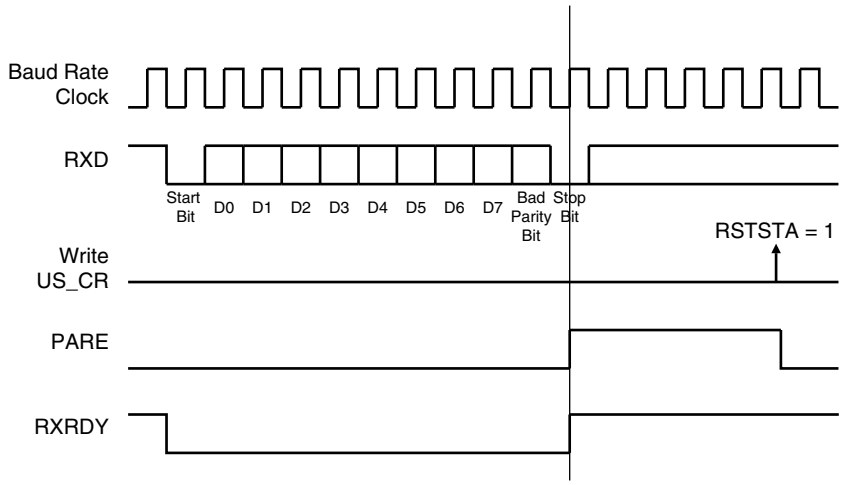
Table 39-9 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits to 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 39-9. 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 to 1. Figure 39-22 illustrates the parity bit status setting and clearing.

Figure 39-22.Parity Error



39.7.3.9 Multidrop Mode

If the PAR field in the Mode Register (US_MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit to 0 and addresses are transmitted with the parity bit to 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 to 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA to 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 to 0.

39.7.3.10 Transmitter Timeguard

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (US_TTGR). When this field is programmed to 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 39-23, 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 to 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 39-23.Timeguard Operations

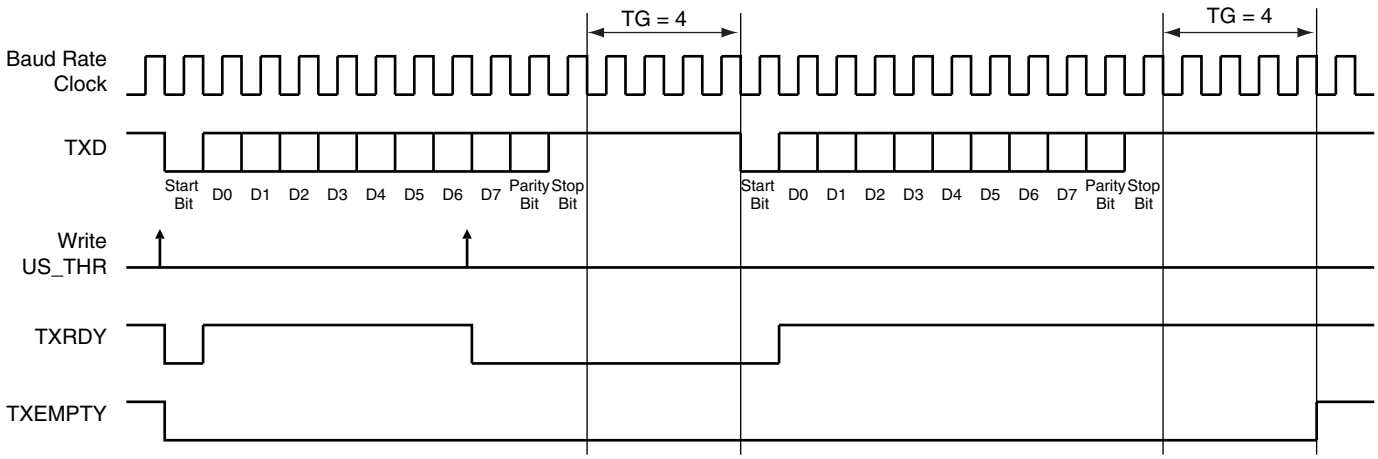


Table 39-10 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 39-10. 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

39.7.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US_RTOR). If the TO field is programmed to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains to 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 to 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 to 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 39-24 shows the block diagram of the Receiver Time-out feature.

Figure 39-24.Receiver Time-out Block Diagram

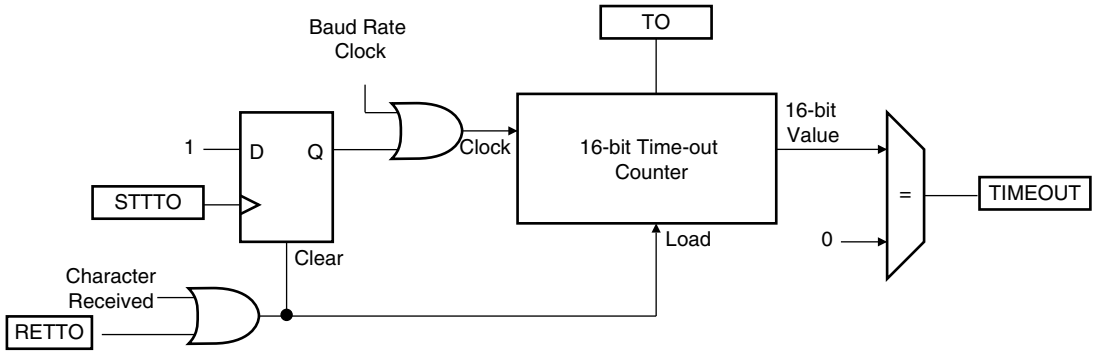


Table 39-11 gives the maximum time-out period for some standard baud rates.

Table 39-11. Maximum Time-out Period

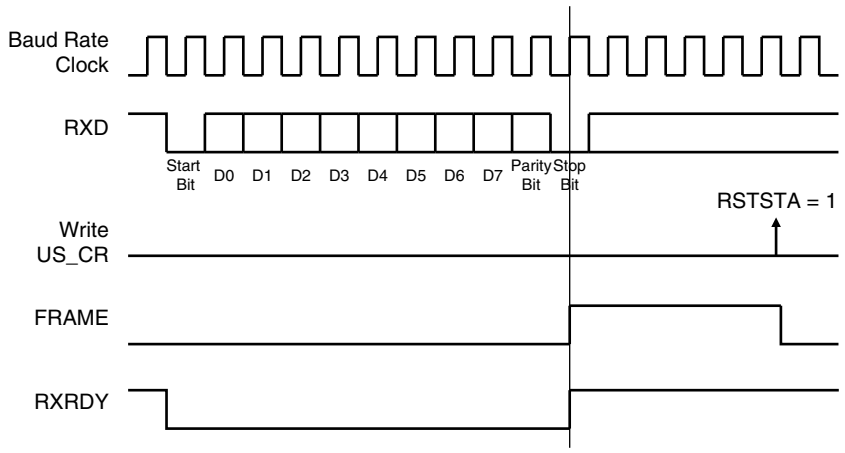
Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962
56000	18	1 170
57600	17	1 138
200000	5	328

39.7.3.12 Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (US_CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit to 1.

Figure 39-25.Framing Error Status



39.7.3.13 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits to 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 to 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 to 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 to 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

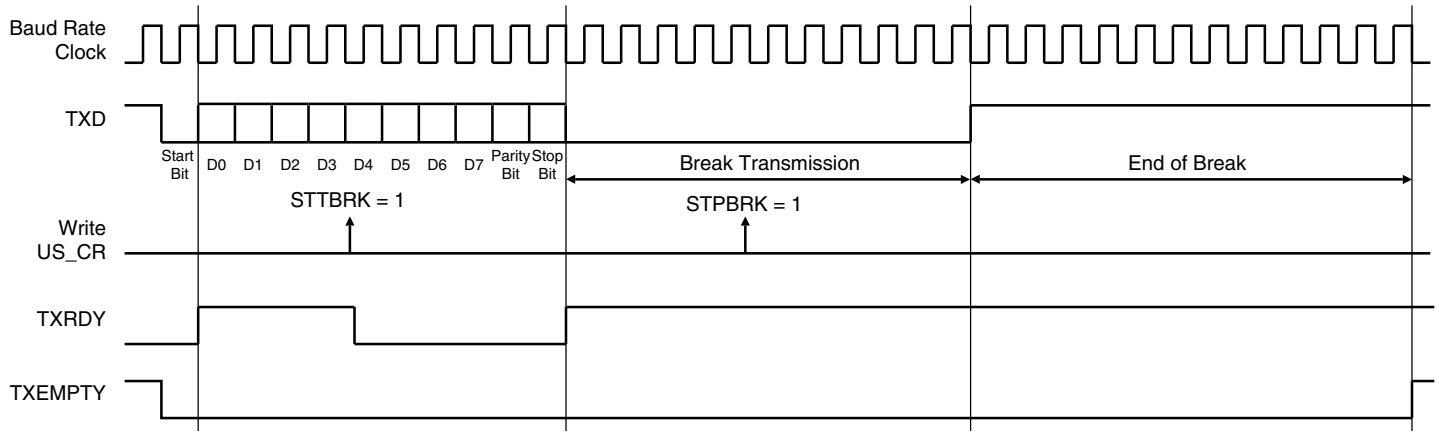
Writing US_CR with both STTBRK and STPBRK bits to 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 39-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 39-26.Break Transmission



39.7.3.14 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data to 0x00, but FRAME remains low.

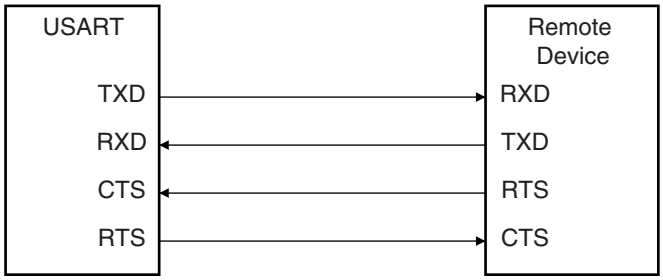
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA to 1.

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.

39.7.3.15 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 39-27.

Figure 39-27.Connection with a Remote Device for Hardware Handshaking

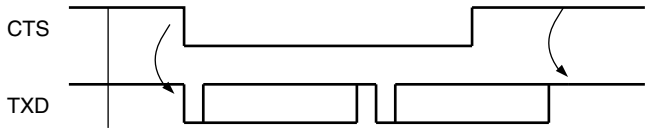


Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

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 DMAC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 39-28 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 39-28.Transmitter Behavior when Operating with Hardware Handshaking



39.7.4 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.

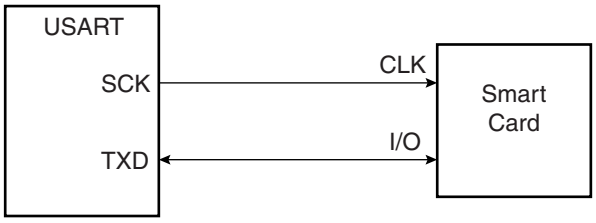
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

39.7.4.1 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 796).

The USART connects to a smart card as shown in Figure 39-29. 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 39-29.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 844 and “PAR: Parity Type” on page 845.

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.

39.7.4.2 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 to 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 39-30.

If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in Figure 39-31. 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 39-30.T = 0 Protocol without Parity Error

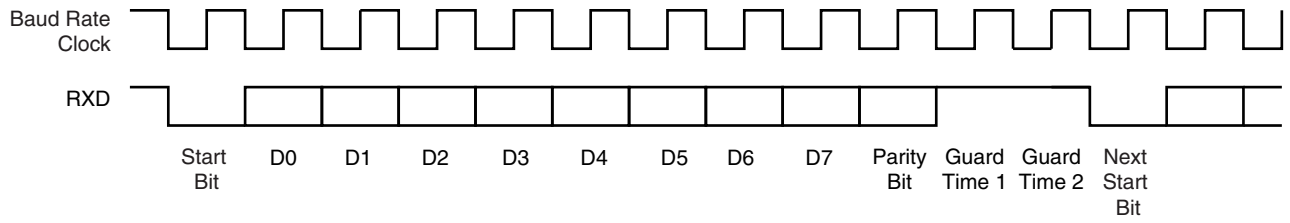
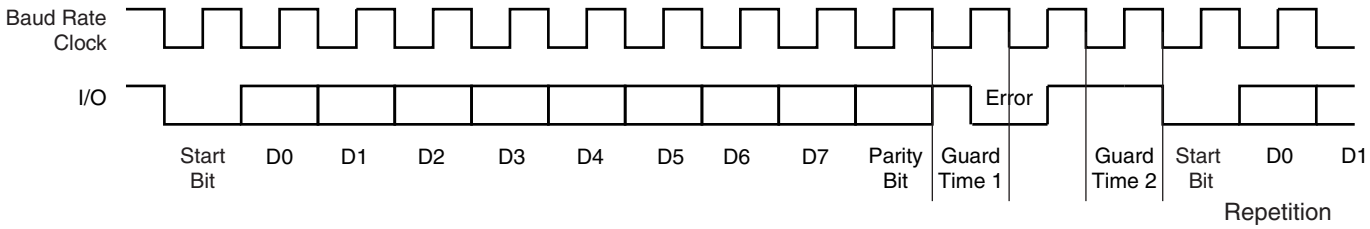


Figure 39-31.T = 0 Protocol with Parity Error



Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US_MR). If INACK is to 1, no error signal is driven on the I/O line even if a parity bit is detected.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred and the RXRDY bit does rise.

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 to 1.

Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US_MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

39.7.4.3 Protocol T = 1

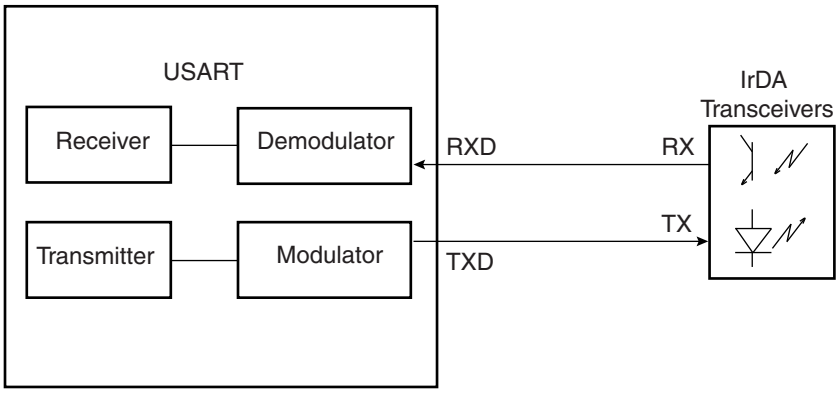
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

39.7.5 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 39-32. 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 39-32.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 to 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).
- Receive data

39.7.5.1 IrDA Modulation

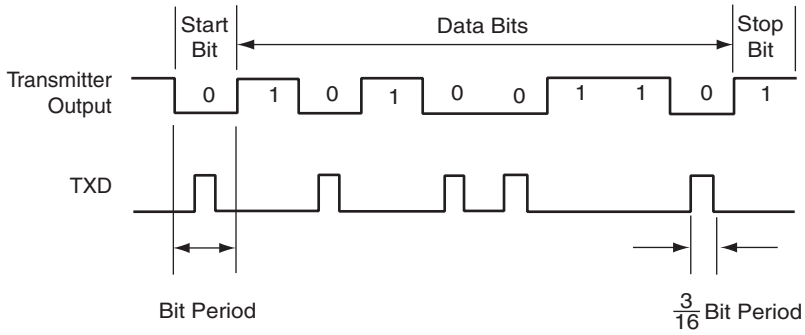
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 39-12.

Table 39-12. 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 39-33 shows an example of character transmission.

Figure 39-33.IrDA Modulation



39.7.5.2 IrDA Baud Rate

Table 39-13 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 39-13. IrDA Baud Rate Error

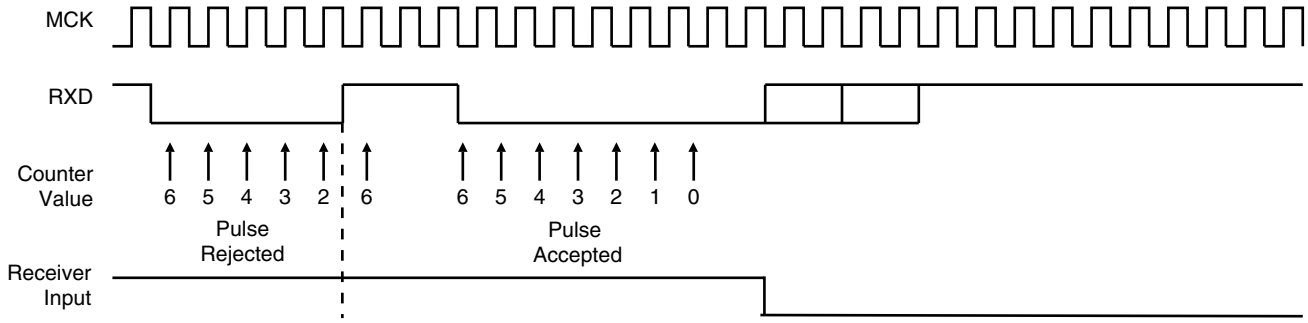
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
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

39.7.5.3 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 39-34 illustrates the operations of the IrDA demodulator.

Figure 39-34.IrDA Demodulator Operations

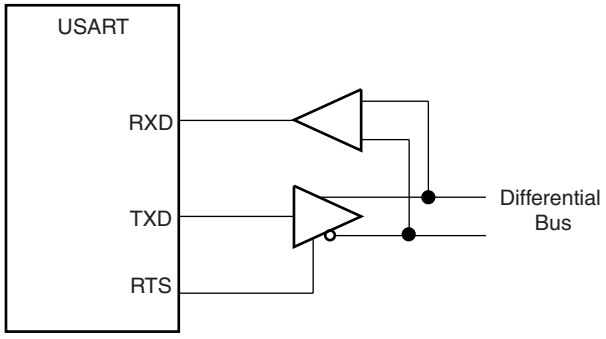


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

39.7.6 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 39-35.

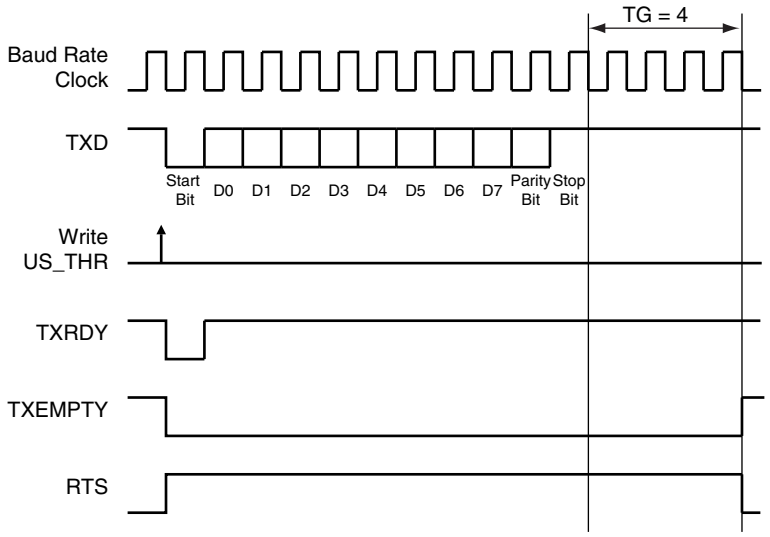
Figure 39-35.Typical Connection to a RS485 Bus



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 39-36 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 39-36.Example of RTS Drive with Timeguard



39.7.7 SPI Mode

The Serial Peripheral Interface (SPI) Mode 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 turns being masters and one master may simultaneously shift data into multiple slaves. (Multiple Master Protocol is the opposite of Single Master Protocol, where one CPU is always the master while all of the others are always 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 its NSS signal is asserted by the master. The USART in SPI Master mode can address only one SPI Slave because it can generate only one NSS signal.

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 of the slave.
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master.
- Serial Clock (SCK): 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 SCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows the master to select or deselect the slave.

39.7.7.1 Modes of Operation

The USART can operate in SPI Master Mode or in SPI Slave Mode.

Operation in SPI Master Mode is programmed by writing to 0xE the USART_MODE field in the Mode Register. In this case the SPI lines must be connected as described below:

- The MOSI line is driven by the output pin TXD
- The MISO line drives the input pin RXD
- The SCK line is driven by the output pin SCK
- The NSS line is driven by the output pin RTS

Operation in SPI Slave Mode is programmed by writing to 0xF the USART_MODE field in the Mode Register. In this case the SPI lines must be connected as described below:

- The MOSI line drives the input pin RXD
- The MISO line is driven by the output pin TXD
- The SCK line drives the input pin SCK
- The NSS line drives the input pin CTS

In order to avoid unpredicted behavior, any change of the SPI Mode must be followed by a software reset of the transmitter and of the receiver (except the initial configuration after a hardware reset). (See Section 39.7.8.3).

39.7.7.2 Baud Rate

In SPI Mode, the baudrate generator operates in the same way as in USART synchronous mode: See “Baud Rate in Synchronous Mode or SPI Mode” on page 798. However, there are some restrictions:

In SPI Master Mode:

- The external clock SCK must not be selected (USCLKS ≠ 0x3), and the bit CLKO must be set to “1” in the Mode Register (US_MR), in order to generate correctly the serial clock on the SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the value programmed in CD must be superior or equal to 6.
- If the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the SCK pin, this value can be odd if the internal clock is selected (MCK).

In SPI Slave Mode:

- The external clock (SCK) selection is forced regardless of the value of the USCLKS field in the Mode Register (US_MR). Likewise, the value written in US_BRGR has no effect, because the clock is provided directly by the signal on the USART SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the external clock (SCK) frequency must be at least 6 times lower than the system clock.

39.7.7.3 Data Transfer

Up to 9 data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (US_MR). The 9 bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI Mode (Master or Slave).

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Mode Register. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon 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 39-14. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 39-37.SPI Transfer Format (CPHA=1, 8 bits per transfer)

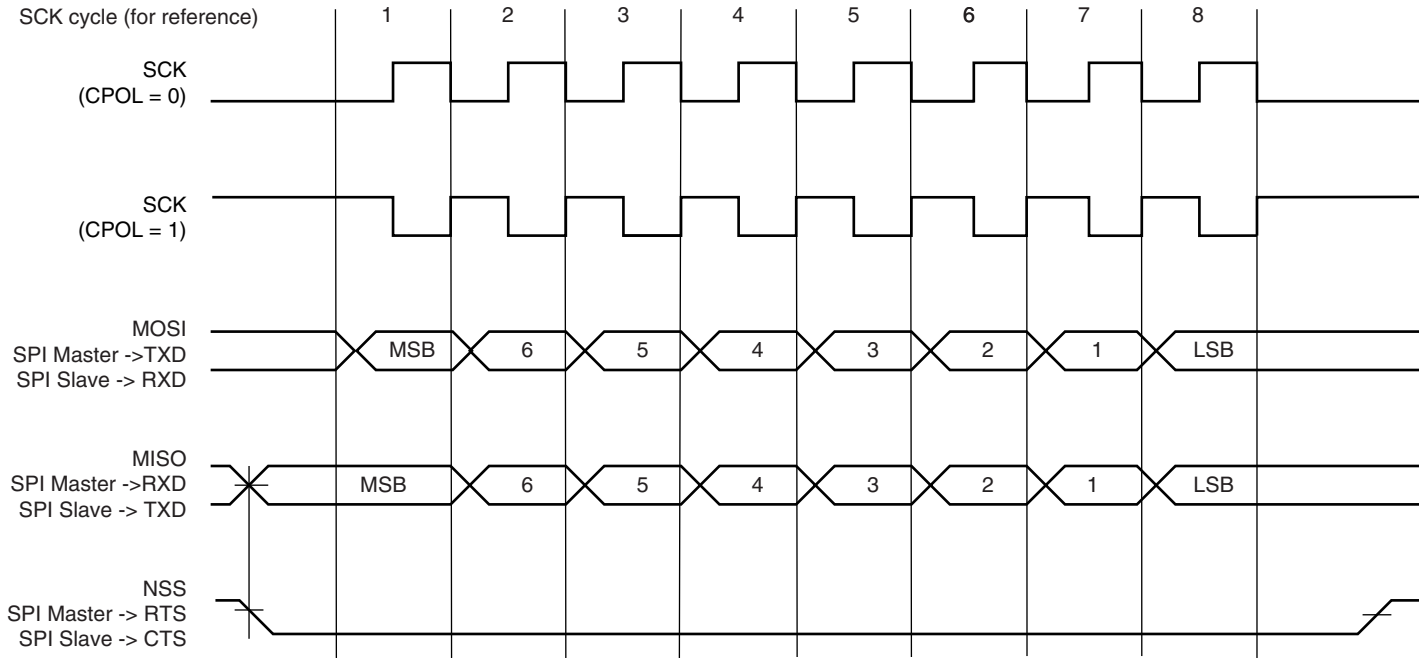
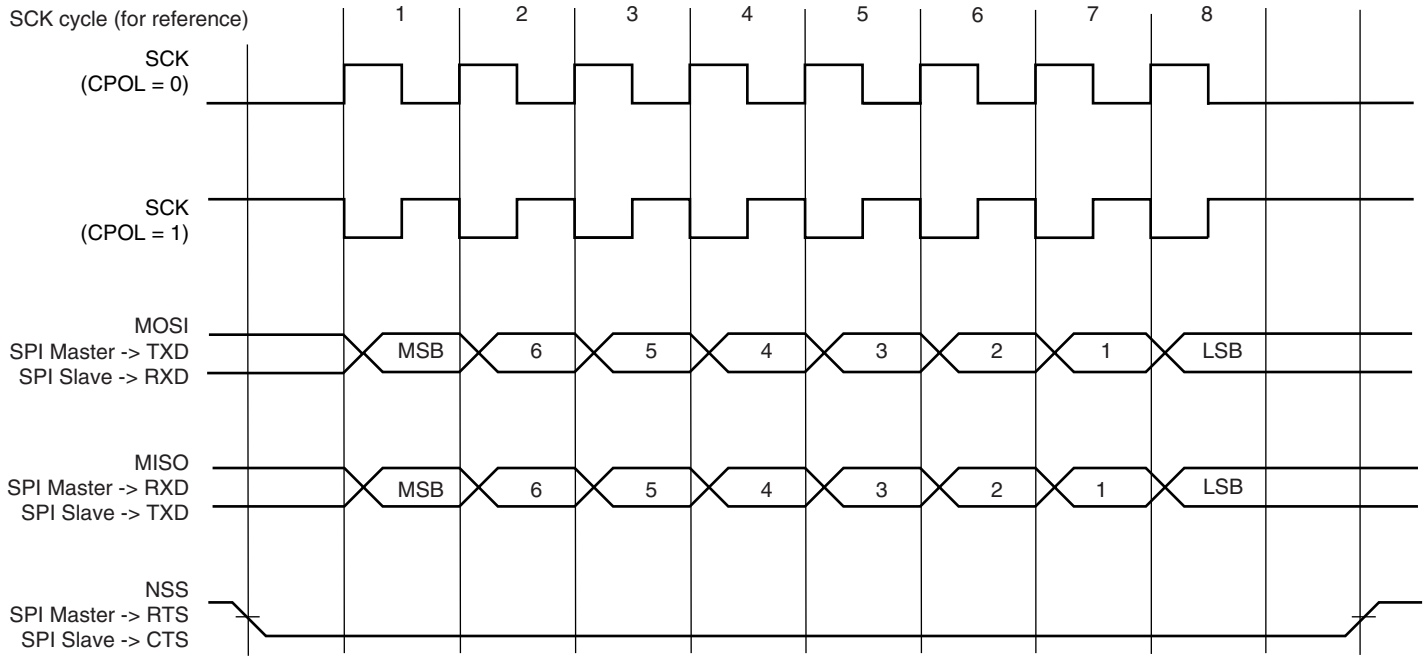


Figure 39-38.SPI Transfer Format (CPHA=0, 8 bits per transfer)



39.7.7.4 Receiver and Transmitter Control

See “Receiver and Transmitter Control” on page 800.

39.7.7.5 Character Transmission

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.

If the USART is in SPI Slave Mode and if a character must be sent while the Transmit Holding Register (US_THR) is empty, the UNRE (Underrun Error) bit is set. The TXD transmission line stays at high level during all this time. The UNRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit to 1.

In SPI Master Mode, the slave select line (NSS) is asserted at low level 1 Tbit (Time bit) before the transmission of the MSB bit and released at high level 1 Tbit after the transmission of the LSB bit. So, the slave select line (NSS) is always released between each character transmission and a minimum delay of 3 Tbits always inserted. However, in order to address slave devices supporting the CSAAT mode (Chip Select Active After Transfer), the slave select line (NSS) can be forced at low level by writing the Control Register (US_CR) with the RTSEN bit to 1. The slave select line (NSS) can be released at high level only by writing the Control Register (US_CR) with the RTSDIS bit to 1 (for example, when all data have been transferred to the slave device).

In SPI Slave Mode, the transmitter does not require a falling edge of the slave select line (NSS) to initiate a character transmission but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

39.7.7.6 Character Reception

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 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 to 1.

To ensure correct behavior of the receiver in SPI Slave Mode, the master device sending the frame must ensure a minimum delay of 1 Tbit between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

39.7.7.7 Receiver Timeout

Because the receiver baudrate clock is active only during data transfers in SPI Mode, a receiver timeout is impossible in this mode, whatever the Time-out value is (field TO) in the Time-out Register (US_RTOR).

39.7.8 LIN Mode

The LIN Mode provides Master node and Slave node connectivity on a LIN bus.

The LIN (Local Interconnect Network) is a serial communication protocol which efficiently supports the control of mechatronic nodes in distributed automotive applications.

The main properties of the LIN bus are:

- Single Master/Multiple Slaves concept
- Low cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as a pure state machine
- Self synchronization without quartz or ceramic resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20 kbit/s

LIN provides cost efficient bus communication where the bandwidth and versatility of CAN are not required.

The LIN Mode enables processing LIN frames with a minimum of action from the microprocessor.

39.7.8.1 Modes of Operation

The USART can act either as a LIN Master node or as a LIN Slave node.

The node configuration is chosen by setting the USART_MODE field in the USART Mode register (US_MR):

- LIN Master Node (USART_MODE=0xA)
- LIN Slave Node (USART_MODE=0xB)

In order to avoid unpredicted behavior, any change of the LIN node configuration must be followed by a software reset of the transmitter and of the receiver (except the initial node configuration after a hardware reset). (See Section 39.7.8.3)

39.7.8.2 Baud Rate Configuration

See “Baud Rate in Asynchronous Mode” on page 796.

The baud rate is configured in the Baud Rate Generator register (US_BRGR).

39.7.8.3 Receiver and Transmitter Control

See “Receiver and Transmitter Control” on page 800.

39.7.8.4 Character Transmission

See “Transmitter Operations” on page 800.

39.7.8.5 Character Reception

See “Receiver Operations” on page 807.

39.7.8.6 Header Transmission (Master Node Configuration)

All the LIN Frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

So in Master node configuration, the frame handling starts with the sending of the header.

The header is transmitted as soon as the identifier is written in the LIN Identifier register (US_LINIR). At this moment the flag TXRDY falls.

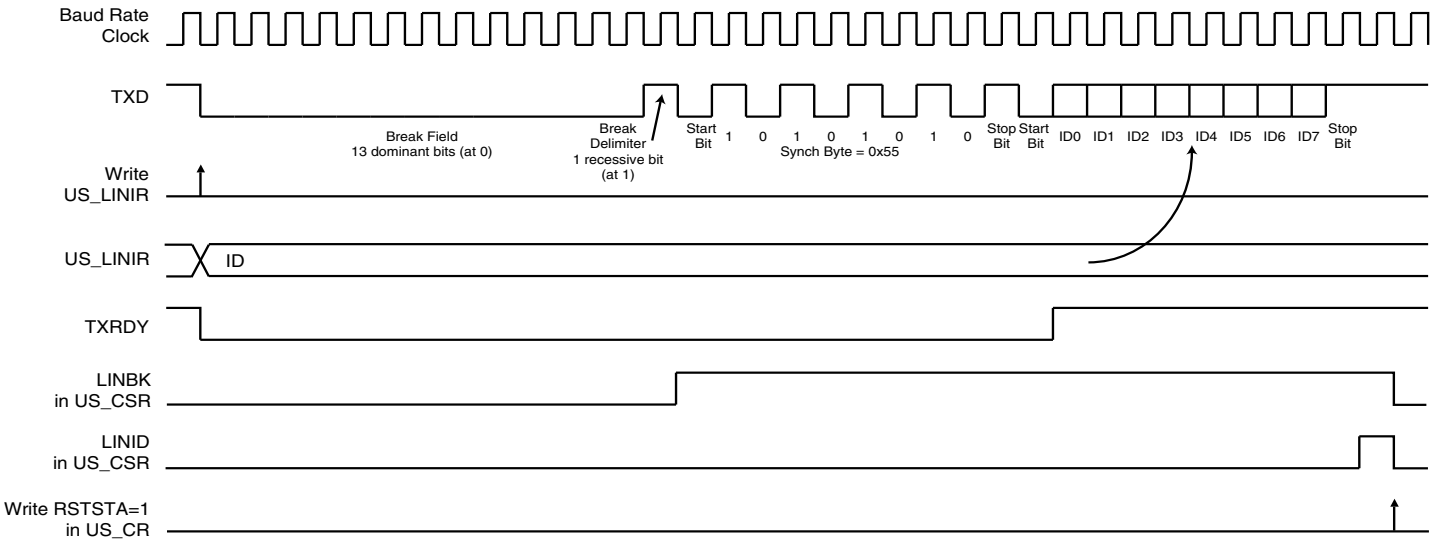
The Break Field, the Synch Field and the Identifier Field are sent automatically one after the other.

The Break Field consists of 13 dominant bits and 1 recessive bit, the Synch Field is the character 0x55 and the Identifier corresponds to the character written in the LIN Identifier Register (US_LINIR). The Identifier parity bits can be automatically computed and sent (see Section 39.7.8.9).

The flag TXRDY rises when the identifier character is transferred into the Shift Register of the transmitter.

As soon as the Synch Break Field is transmitted, the flag LINBK in the Channel Status register (US_CSR) is set to 1. Likewise, as soon as the Identifier Field is sent, the flag LINID in the Channel Status register (US_CSR) is set to 1. These flags are reset by writing the bit RSTSTA to 1 in the Control register (US_CR).

Figure 39-39.Header Transmission



39.7.8.7 Header Reception (Slave Node Configuration)

All the LIN Frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

In Slave node configuration, the frame handling starts with the reception of the header.

The USART uses a break detection threshold of 11 nominal bit times at the actual baud rate. At any time, if 11 consecutive recessive bits are detected on the bus, the USART detects a Break Field. As long as a Break Field has not been detected, the USART stays idle and the received data are not taken in account.

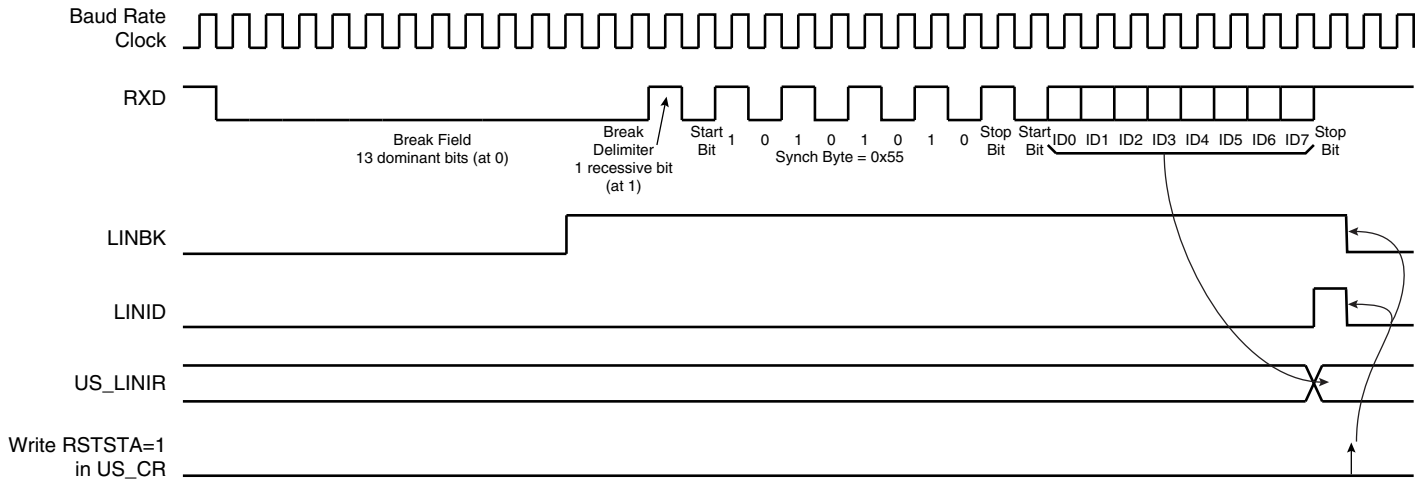
When a Break Field has been detected, the flag LINBK in the Channel Status register (US_CSR) is set to 1 and the USART expects the Synch Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized (see Section 39.7.8.8). If the received Synch character is not 0x55, an Inconsistent Synch Field error is generated (see Section 39.7.8.14).

After receiving the Synch Field, the USART expects to receive the Identifier Field.

When the Identifier Field has been received, the flag LINID in the Channel Status register (US_CSR) is set to 1. At this moment the field IDCHR in the LIN Identifier register (US_LINIR) is updated with the received character. The Identifier parity bits can be automatically computed and checked (see Section 39.7.8.9).

The flags LINID and LINBK are reset by writing the bit RSTSTA to 1 in the Control register (US_CR).

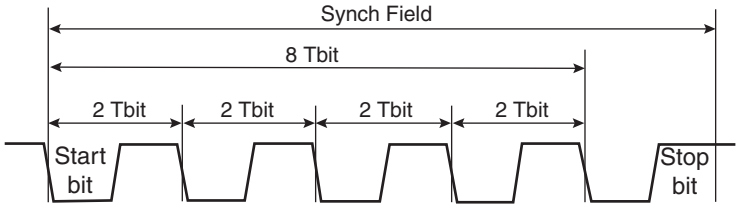
Figure 39-40.Header Reception



39.7.8.8 Slave Node Synchronization

The synchronization is done only in Slave node configuration. The procedure is based on time measurement between falling edges of the Synch Field. The falling edges are available in distances of 2, 4, 6 and 8 bit times.

Figure 39-41.Synch Field



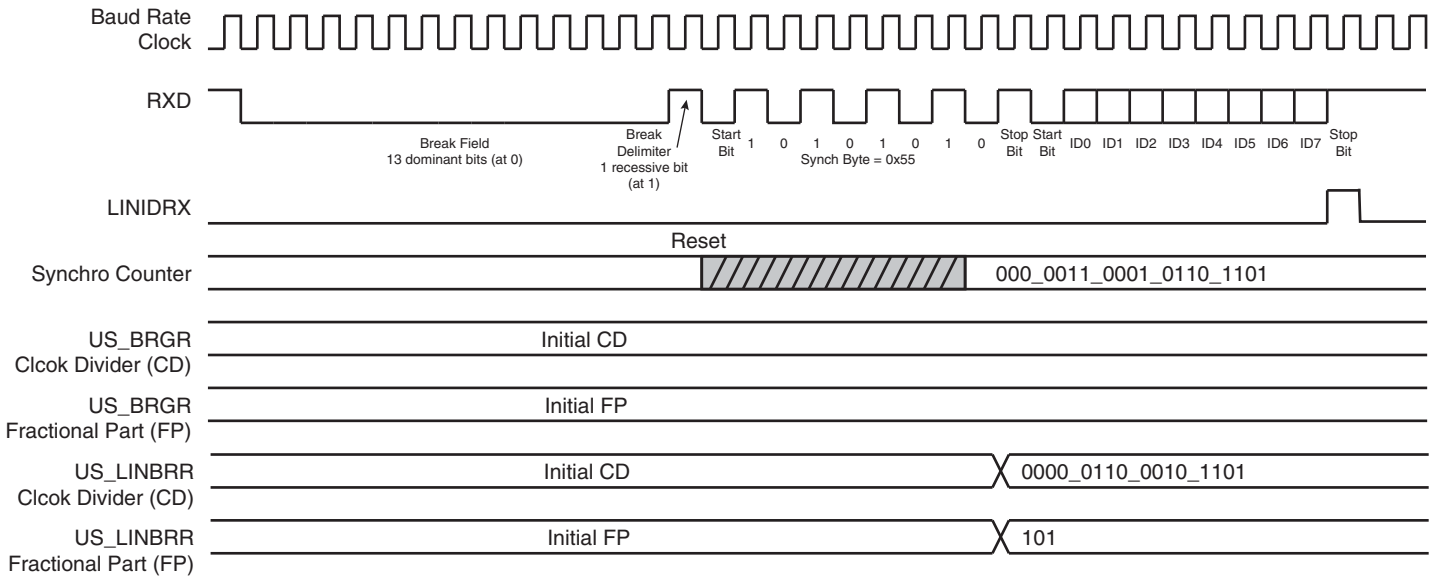
The time measurement is made by a 19-bit counter clocked by the sampling clock (see Section 39.7.1).

When the start bit of the Synch Field is detected, the counter is reset. Then during the next 8 Tbits of the Synch Field, the counter is incremented. At the end of these 8 Tbits, the counter is stopped. At this moment, the 16 most significant bits of the counter (value divided by 8) give the new clock divider (LINCD) and the 3 least significant bits of this value (the remainder) give the new fractional part (LINFP).

When the Synch Field has been received, the clock divider (CD) and the fractional part (FP) are updated in the Baud Rate Generator register (US_BRGR).

If it appears that the sampled Synch character is not equal to 0x55, then the error flag LINISFE in the Channel Status register (US_CSR) is set to 1. It is reset by writing bit RSTSTA to 1 in the Control register (US_CR).

Figure 39-42.Slave Node Synchronization



The accuracy of the synchronization depends on several parameters:

- The nominal clock frequency (F_{Nom}) (the theoretical slave node clock frequency)
- The Baud Rate
- The oversampling (Over=0 => 16X or Over=0 => 8X)

The following formula is used to compute the deviation of the slave bit rate relative to the master bit rate after synchronization (F_{SLAVE} is the real slave node clock frequency).

$$\text{Baudrate_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times F_{SLAVE}}\right)\%$$

$$\text{Baudrate_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times \left(\frac{F_{TOL_UNSYNCH}}{100}\right) \text{x} F_{Nom}}\right)\%$$

$$-0.5 \le \alpha \le +0.5 \quad -1 < \beta < +1$$

$F_{TOL_UNSYNCH}$ is the deviation of the real slave node clock from the nominal clock frequency. The LIN Standard imposes that it must not exceed ±15%. The LIN Standard imposes also that for communication between two nodes, their bit rate must not differ by more than ±2%. This means that the Baudrate_deviation must not exceed ±1%.

It follows from that, a minimum value for the nominal clock frequency:

$$F_{NOM}(\text{min}) = \left(100 \times \frac{[0.5 \times 8 \times (2 - \text{Over}) + 1] \times \text{Baudrate}}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%}\right)\text{Hz}$$

Examples:

- Baudrate = 20 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 2.64 MHz
- Baudrate = 20 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 1.47 MHz
- Baudrate = 1 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 132 kHz
- Baudrate = 1 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 74 kHz

39.7.8.9 Identifier Parity

A protected identifier consists of two sub-fields; the identifier and the identifier parity. Bits 0 to 5 are assigned to the identifier and bits 6 and 7 are assigned to the parity.

The USART interface can generate/check these parity bits, but this feature can also be disabled. The user can choose between two modes by the PARDIS bit of the LIN Mode register (US_LINMR):

- PARDIS = 0:

During header transmission, the parity bits are computed and sent with the 6 least significant bits of the IDCHR field of the LIN Identifier register (US_LINIR). The bits 6 and 7 of this register are discarded.

During header reception, the parity bits of the identifier are checked. If the parity bits are wrong, an Identifier Parity error occurs (see Section 39.7.3.8). Only the 6 least significant bits of the IDCHR field are updated with the received Identifier. The bits 6 and 7 are stuck to 0.

- PARDIS = 1:

During header transmission, all the bits of the IDCHR field of the LIN Identifier register (US_LINIR) are sent on the bus.

During header reception, all the bits of the IDCHR field are updated with the received Identifier.

39.7.8.10 Node Action

In function of the identifier, the node is concerned, or not, by the LIN response. Consequently, after sending or receiving the identifier, the USART must be configured. There are three possible configurations:

- PUBLISH: the node sends the response.
- SUBSCRIBE: the node receives the response.
- IGNORE: the node is not concerned by the response, it does not send and does not receive the response.

This configuration is made by the field, Node Action (NACT), in the US_LINMR register (see Section 39.8.26).

Example: a LIN cluster that contains a Master and two Slaves:

- Data transfer from the Master to the Slave 1 and to the Slave 2:

  NACT(Master)=PUBLISH

  NACT(Slave1)=SUBSCRIBE

  NACT(Slave2)=SUBSCRIBE
- Data transfer from the Master to the Slave 1 only:

  NACT(Master)=PUBLISH

  NACT(Slave1)=SUBSCRIBE

  NACT(Slave2)=IGNORE
- Data transfer from the Slave 1 to the Master:

  NACT(Master)=SUBSCRIBE

  NACT(Slave1)=PUBLISH

  NACT(Slave2)=IGNORE
- Data transfer from the Slave1 to the Slave2:

  NACT(Master)=IGNORE

  NACT(Slave1)=PUBLISH

  NACT(Slave2)=SUBSCRIBE
- Data transfer from the Slave2 to the Master and to the Slave1:

  NACT(Master)=SUBSCRIBE

  NACT(Slave1)=SUBSCRIBE

  NACT(Slave2)=PUBLISH

39.7.8.11 Response Data Length

The LIN response data length is the number of data fields (bytes) of the response excluding the checksum.

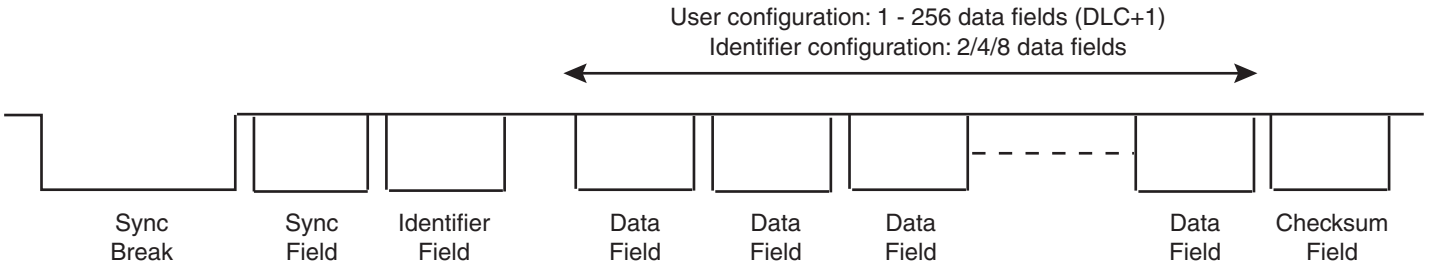
The response data length can either be configured by the user or be defined automatically by bits 4 and 5 of the Identifier (compatibility to LIN Specification 1.1). The user can choose between these two modes by the DLM bit of the LIN Mode register (US_LINMR):

- DLM = 0: the response data length is configured by the user via the DLC field of the LIN Mode register (US_LINMR). The response data length is equal to (DLC + 1) bytes. DLC can be programmed from 0 to 255, so the response can contain from 1 data byte up to 256 data bytes.
- DLM = 1: the response data length is defined by the Identifier (IDCHR in US_LINIR) according to the table below. The DLC field of the LIN Mode register (US_LINMR) is discarded. The response can contain 2 or 4 or 8 data bytes.

Table 39-15. Response Data Length if DLM = 1

IDCHR[5]	IDCHR[4]	Response Data Length [bytes]
0	0	2
0	1	2
1	0	4
1	1	8

Figure 39-43.Response Data Length



39.7.8.12 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8- bit sum with carry, over all data bytes or all data bytes and the protected identifier. Checksum calculation over the data bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and it is used for communication with LIN 2.0 slaves.

The USART can be configured to:

- Send/Check an Enhanced checksum automatically (CHKDIS = 0 & CHKTYP = 0)
- Send/Check a Classic checksum automatically (CHKDIS = 0 & CHKTYP = 1)
- Not send/check a checksum (CHKDIS = 1)

This configuration is made by the Checksum Type (CHKTYP) and Checksum Disable (CHKDIS) fields of the LIN Mode register (US_LINMR).

If the checksum feature is disabled, the user can send it manually all the same, by considering the checksum as a normal data byte and by adding 1 to the response data length (see Section 39.7.8.11).

39.7.8.13 Frame Slot Mode

This mode is useful only for Master nodes. It respects the following rule: each frame slot shall be longer than or equal to TFrame_Maximum.

If the Frame Slot Mode is enabled (FSDIS = 0) and a frame transfer has been completed, the TXRDY flag is set again only after TFrame_Maximum delay, from the start of frame. So the Master node cannot send a new header if the frame slot duration of the previous frame is inferior to TFrame_Maximum.

If the Frame Slot Mode is disabled (FSDIS = 1) and a frame transfer has been completed, the TXRDY flag is set again immediately.

The TFrame_Maximum is calculated as below:

If the Checksum is sent (CHKDIS = 0):

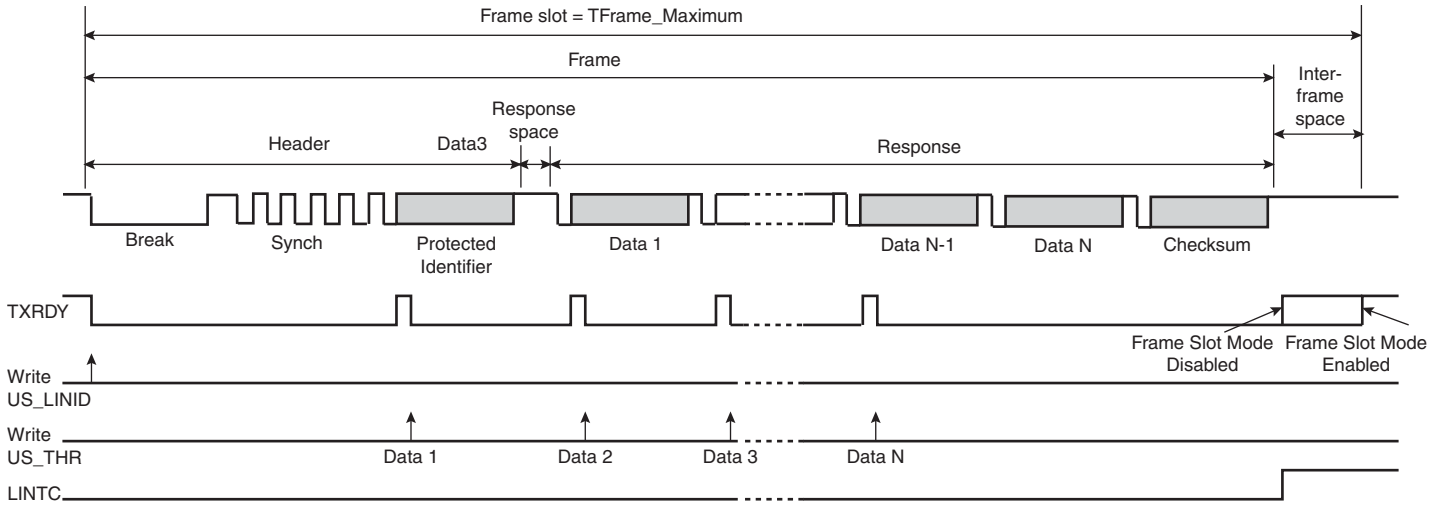
- THeader_Nominal = 34 x Tbit
- TResponse_Nominal = 10 x (NData + 1) x Tbit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1)(Note:)
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1 + 1) + 1) x Tbit
- TFrame_Maximum = (77 + 14 x DLC) x Tbit

If the Checksum is not sent (CHKDIS = 1):

- THeader_Nominal = 34 x Tbit
- TResponse_Nominal = 10 x NData x Tbit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1(Note:))
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1) + 1) x Tbit
- TFrame_Maximum = (63 + 14 x DLC) x Tbit

Note: The term “+1” leads to an integer result for TFrame_Max (LIN Specification 1.3).

Figure 39-44.Frame Slot Mode



39.7.8.14 LIN Errors

Bit Error

This error is generated in Master of Slave node configuration, when the USART is transmitting and if the transmitted value on the Tx line is different from the value sampled on the Rx line. If a bit error is detected, the transmission is aborted at the next byte border.

This error is reported by flag LINBE in the Channel Status Register (US_CSR).

Inconsistent Synch Field Error

This error is generated in Slave node configuration, if the Synch Field character received is other than 0x55.

This error is reported by flag LINISFE in the Channel Status Register (US_CSR).

Identifier Parity Error

This error is generated in Slave node configuration, if the parity of the identifier is wrong. This error can be generated only if the parity feature is enabled (PARDIS = 0).

This error is reported by flag LINIPE in the Channel Status Register (US_CSR).

Checksum Error

This error is generated in Master of Slave node configuration, if the received checksum is wrong. This flag can be set to “1” only if the checksum feature is enabled (CHKDIS = 0).

This error is reported by flag LINCE in the Channel Status Register (US_CSR).

Slave Not Responding Error

This error is generated in Master of Slave node configuration, when the USART expects a response from another node (NACT = SUBSCRIBE) but no valid message appears on the bus within the time given by the maximum length of the message frame, TFrame_Maximum (see Section 39.7.8.13). This error is disabled if the USART does not expect any message (NACT = PUBLISH or NACT = IGNORE).

This error is reported by flag LINSNRE in the Channel Status Register (US_CSR).

39.7.8.15 LIN Frame Handling

Master Node Configuration

- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the Master Node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM, FSDIS and DLC in US_LINMR to configure the frame transfer.
- Check that TXRDY in US_CSR is set to “1”
- Write IDCHR in US_LINIR to send the header

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in US_CSR rises
  - Write TCHR in US_THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in US_CSR rises
  - Read RCHR in US_RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors

Figure 39-45.Master Node Configuration, NACT = PUBLISH

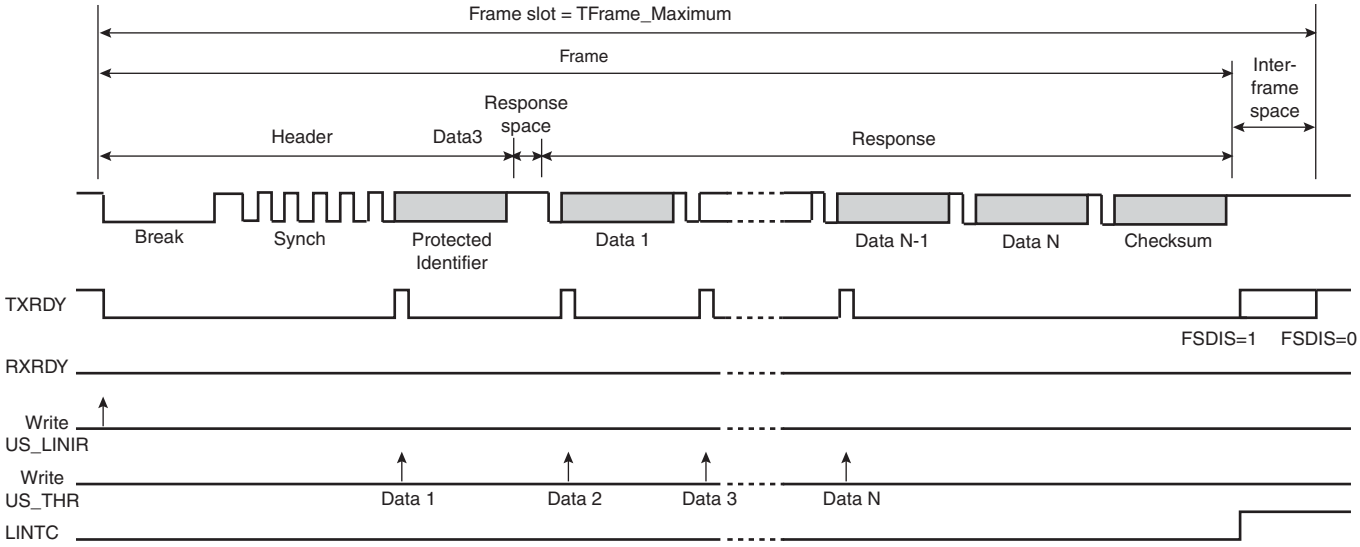


Figure 39-46.Master Node Configuration, NACT=SUBSCRIBE

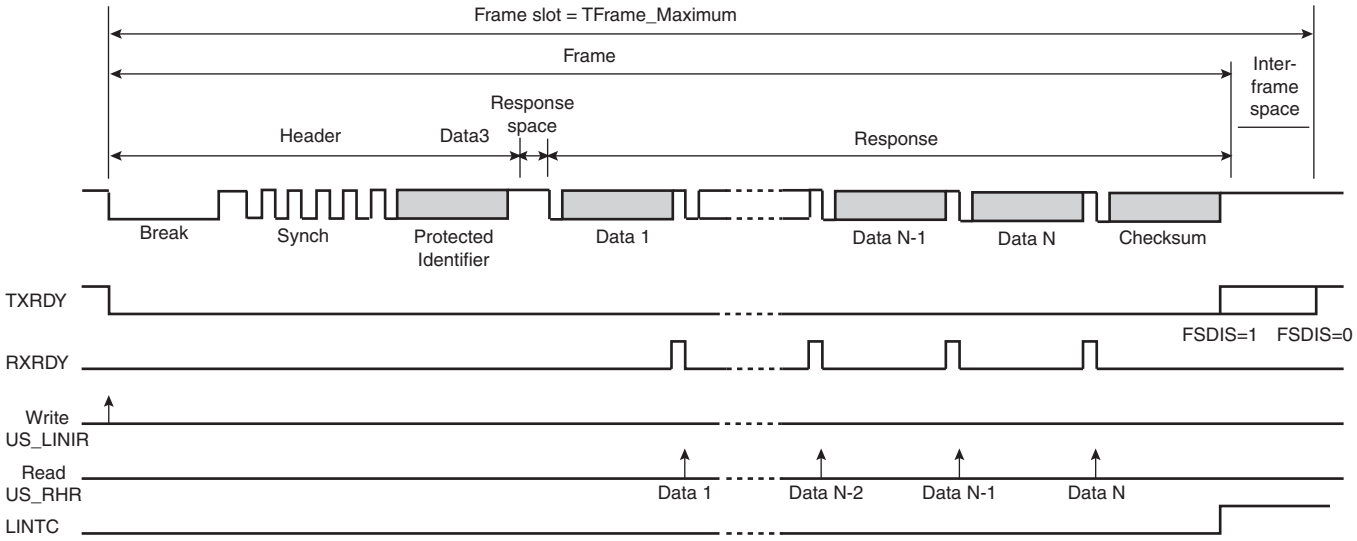
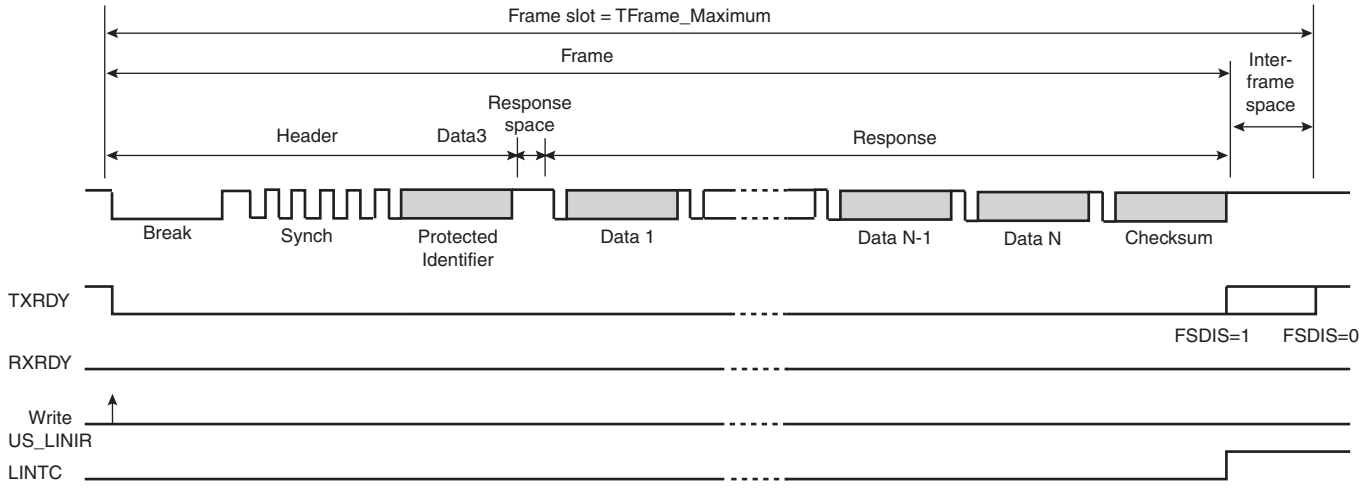


Figure 39-47.Master Node Configuration, NACT=IGNORE



Slave Node Configuration

- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the Slave Node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Wait until LINID in US_CSR rises
- Check LINISFE and LINPE errors
- Read IDCHR in US_RHR
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM and DLC in US_LINMR to configure the frame transfer.

IMPORTANT: If the NACT configuration for this frame is PUBLISH, the US_LINMR register, must be write with NACT = PUBLISH even if this field is already correctly configured, in order to set the TXREADY flag and the corresponding write transfer request.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the LIN controller sends the response
  - Wait until TXRDY in US_CSR rises
  - Write TCHR in US_THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in US_CSR rises
  - Read RCHR in US_RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors

Figure 39-48.Slave Node Configuration, NACT = PUBLISH

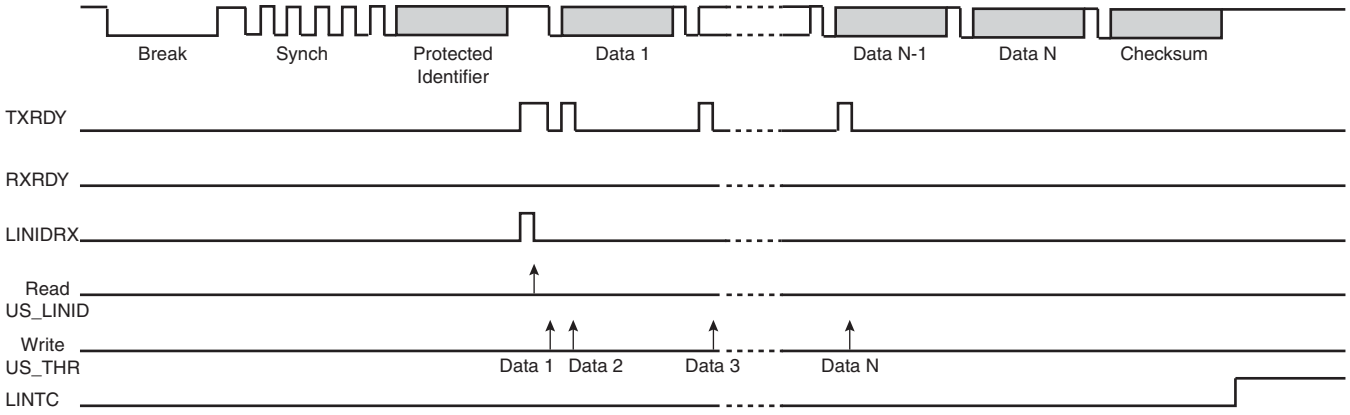


Figure 39-49.Slave Node Configuration, NACT = SUBSCRIBE

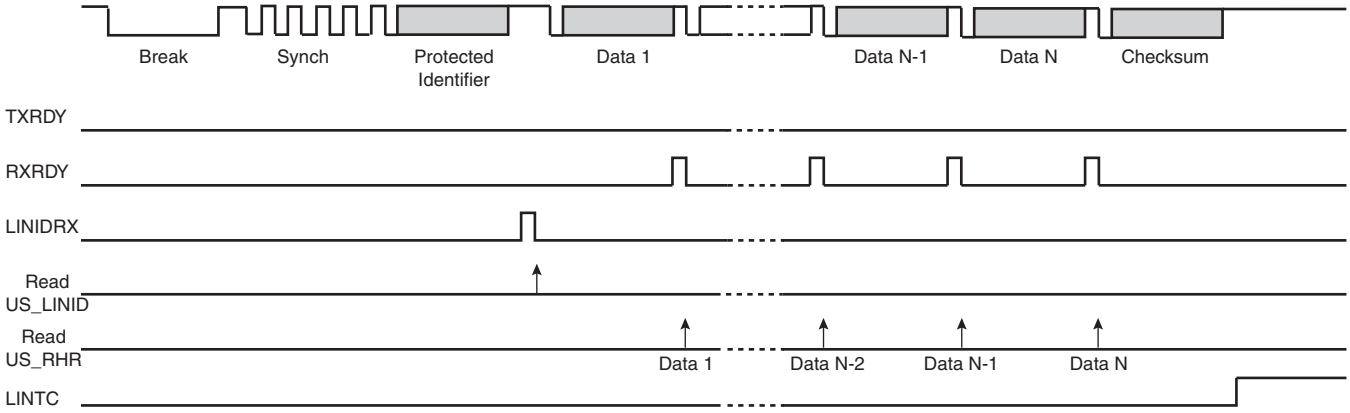
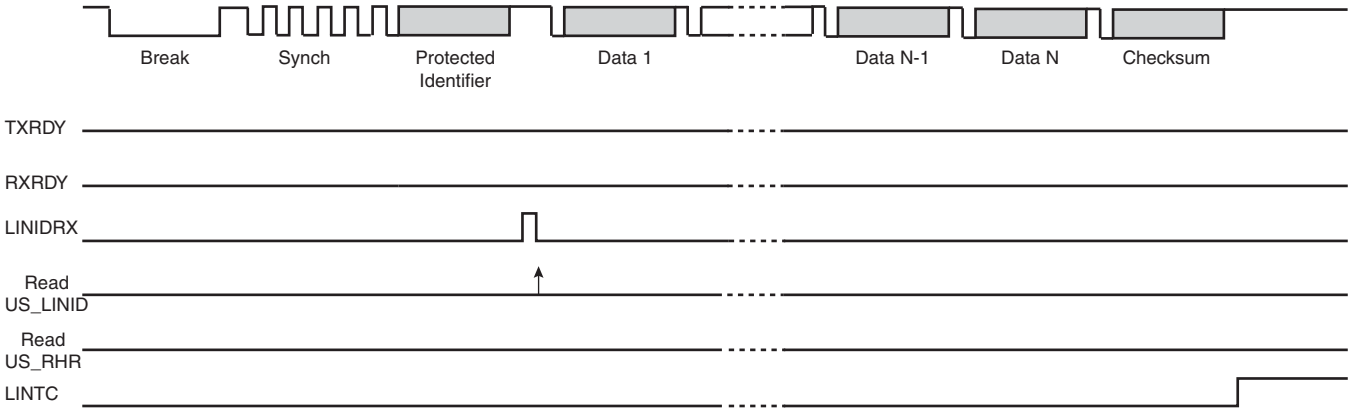


Figure 39-50.Slave Node Configuration, NACT = IGNORE



39.7.8.16 LIN Frame Handling With The DMAC

The USART can be used in association with the DMAC in order to transfer data directly into/from the on- and off-chip memories without any processor intervention.

The DMAC uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The DMAC always writes in the Transmit Holding register (US_THR) and it always reads in the Receive Holding register (US_RHR). The size of the data written or read by the DMAC in the USART is always a byte.

Master Node Configuration

The user can choose between two DMAC modes by the PDCM bit in the LIN Mode register (US_LINMR):

- PDCM = 1: the LIN configuration is stored in the WRITE buffer and it is written by the DMAC in the Transmit Holding register US_THR (instead of the LIN Mode register US_LINMR). Because the DMAC transfer size is limited to a byte, the transfer is split into two accesses. During the first access the bits, NACT, PARDIS, CHKDIS, CHKTYP, DLM and FSDIS are written. During the second access the 8-bit DLC field is written.
- PDCM = 0: the LIN configuration is not stored in the WRITE buffer and it must be written by the user in the LIN Mode register (US_LINMR).

The WRITE buffer also contains the Identifier and the DATA, if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Figure 39-51.Master Node with DMAC (PDCM = 1)

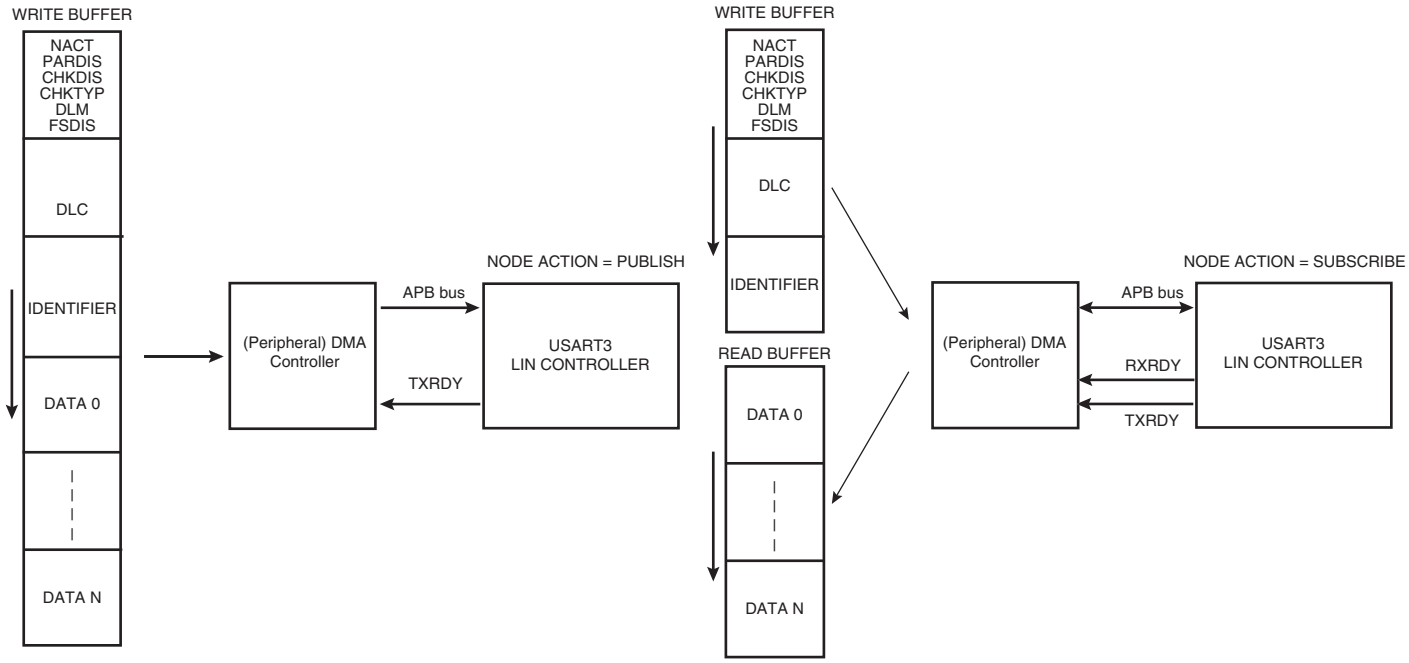
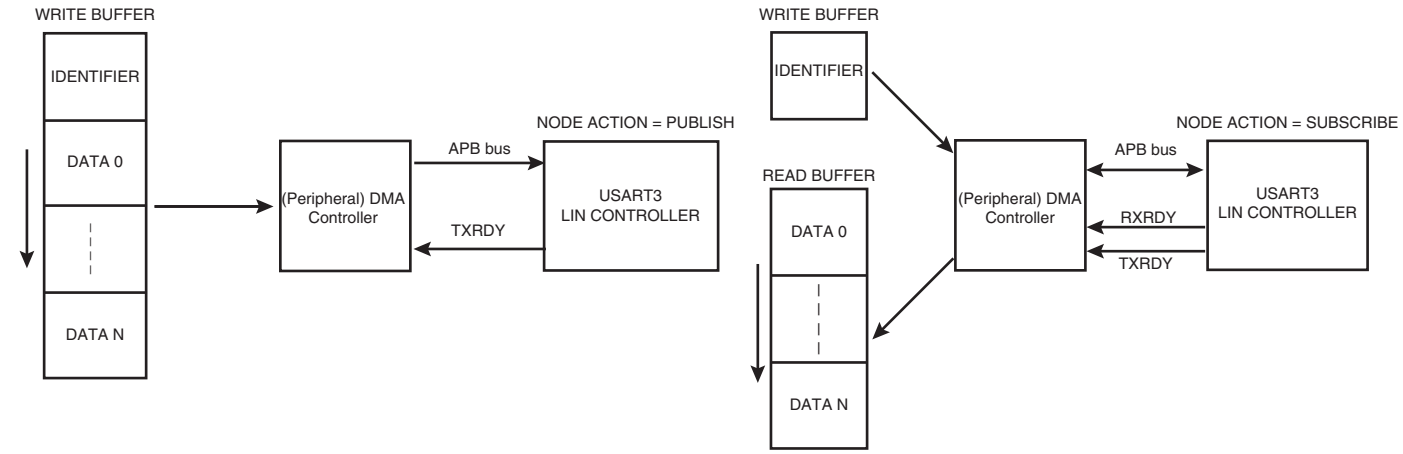


Figure 39-52.Master Node with DMAC (PDCM = 0)



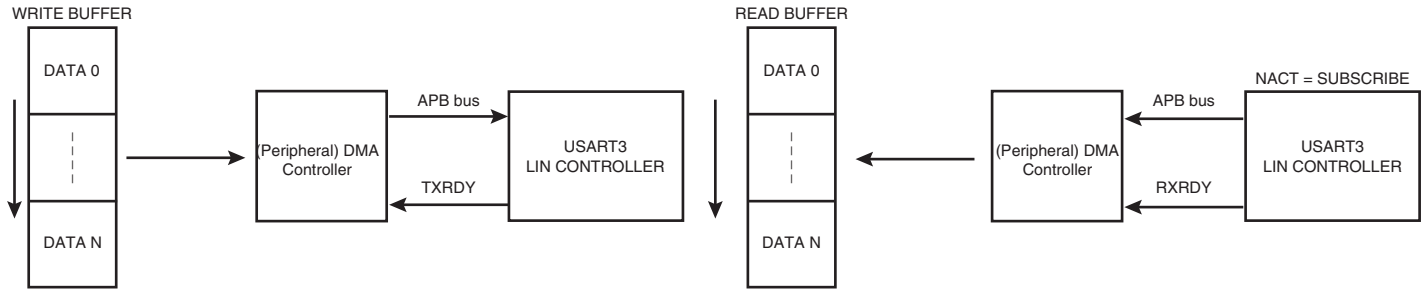
Slave Node Configuration

In this configuration, the DMAC transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (US_LINIR). The LIN mode must be written by the user in the LIN Mode register (US_LINMR).

The WRITE buffer contains the DATA if the USART sends the response (NACT=PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT=SUBSCRIBE).

Figure 39-53.Slave Node with DMAC



39.7.8.17 Wake-up Request

Any node in a sleeping LIN cluster may request a wake-up.

In the LIN 2.0 specification, the wakeup request is issued by forcing the bus to the dominant state from 250 µs to 5 ms. For this, it is necessary to send the character 0xF0 in order to impose 5 successive dominant bits. Whatever the baud rate is, this character respects the specified timings.

- Baud rate min = 1 kbit/s -> Tbit = 1ms -> 5 Tbits = 5 ms
- Baud rate max = 20 kbit/s -> Tbi t= 50 µs -> 5 Tbits = 250 µs

In the LIN 1.3 specification, the wakeup request should be generated with the character 0x80 in order to impose 8 successive dominant bits.

The user can choose by the WKUPTYP bit in the LIN Mode register (US_LINMR) either to send a LIN 2.0 wakeup request (WKUPTYP=0) or to send a LIN 1.3 wakeup request (WKUPTYP=1).

A wake-up request is transmitted by writing the Control Register (US_CR) with the LINWKUP bit to 1. Once the transfer is completed, the LINTC flag is asserted in the Status Register (US_SR). It is cleared by writing the Control Register (US_CR) with the RSTSTA bit to 1.

39.7.8.18 Bus Idle Time-out

If the LIN bus is inactive for a certain duration, the slave nodes shall automatically enter in sleep mode. In the LIN 2.0 specification, this time-out is fixed at 4 seconds. In the LIN 1.3 specification, it is fixed at 25000 Tbits.

In Slave Node configuration, the Receiver Time-out 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 to go into sleep mode.

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 to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains to 0. Otherwise, the receiver loads a 17-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.

If STTTO is performed, the counter clock is stopped until a first character is received.

If RETTO is performed, the counter starts counting down immediately from the value TO

.

Receiver Time-out programming

LIN Specification	Baud Rate	Time-out period	TO
2.0	1 000 bit/s	4s	4 000
	2 400 bit/s		9 600
	9 600 bit/s		38 400
	19 200 bit/s		76 800
	20 000 bit/s		80 000
1.3	-	25 000 Tbits	25 000

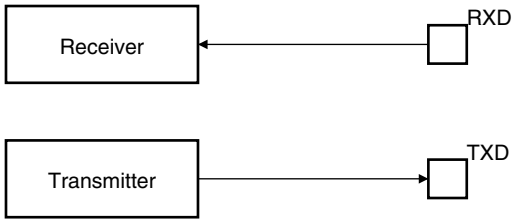
39.7.9 Test Modes

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.

39.7.9.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

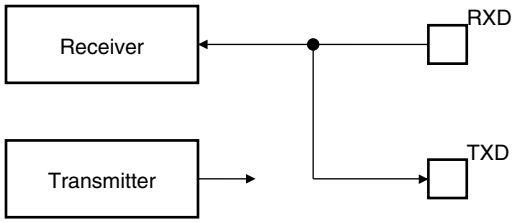
Figure 39-54.Normal Mode Configuration



39.7.9.2 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 39-55. 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.

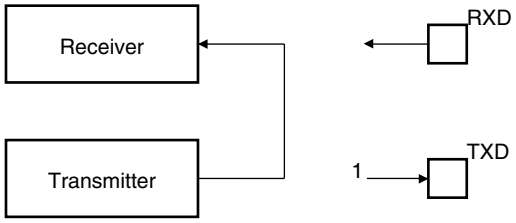
Figure 39-55.Automatic Echo Mode Configuration



39.7.9.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 39-56. 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.

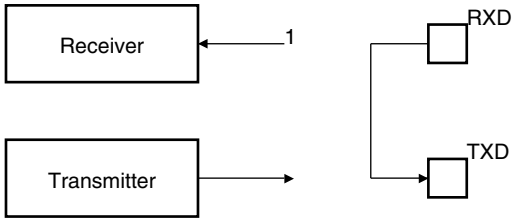
Figure 39-56.Local Loopback Mode Configuration



39.7.9.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 39-57. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 39-57.Remote Loopback Mode Configuration



39.7.10 Write Protection Registers

To prevent any single software error that may corrupt USART behavior, certain address spaces can be write-protected by setting the WPEN bit in the USART Write Protect Mode Register (US_WPMR).

If a write access to the protected registers is detected, then the WPVS flag in the USART Write Protect Status Register (US_WPSR) is set and the WPVSRC field indicates in which register the write access has been attempted.

The WPVS flag is automatically reset by reading the USART Write Protect Mode Register (US_WPMR) with the appropriate access key, WPKEY.

The protected registers are:

- "USART Mode Register"
- "USART Baud Rate Generator Register"
- "USART Receiver Time-out Register"
- "USART Transmitter Timeguard Register"
- "USART FI DI RATIO Register"
- "USART IrDA FILTER Register"
- "USART Manchester Configuration Register"

39.8 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 39-16. 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	0xB0011004
0x0054	LIN Mode Register	US_LINMR	Read-write	0x0
0x0058	LIN Identifier Register	US_LINIR	Read-write[1]	0x0
0x005C	LIN Baud Rate Register	US_LINBRR	Read-only	0x0
0xE4	Write Protect Mode Register	US_WPMR	Read-write	0x0
0xE8	Write Protect Status Register	US_WPSR	Read-only	0x0
0x5C - 0xFC	Reserved	–	–	–

Notes: 1. Write is possible only in LIN Master node configuration.

39.8.1 USART Control Register

Name: US_CR

Address: 0xF801C000 (0), 0xF8020000 (1), 0xF8024000 (2), 0xF8028000 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	LINWKUP	LINABT	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	–	–

For SPI control, see “USART Control Register (SPI_MODE)” on page 842.

- **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, LINBE, LINISFE, LINIPE, LINCE, LINSNRE, LINID, LINTC, LINBK 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.

- **LINABT: Abort LIN Transmission**

0: No effect.

1: Abort the current LIN transmission.

- **LINWKUP: Send LIN Wakeup Signal**

0: No effect:

1: Sends a wakeup signal on the LIN bus.

39.8.2 USART Control Register (SPI_MODE)

Name: US_CR (SPI_MODE)

Address: 0xF801C000 (0), 0xF8020000 (1), 0xF8024000 (2), 0xF8028000 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RCS	FCS	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

This configuration is relevant only if USART_MODE=0xE or 0xF in “USART Mode Register” on page 844.

- **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 OVRE, UNRE in US_CSR.

- **FCS: Force SPI Chip Select**
  - Applicable if USART operates in SPI Master Mode (USART_MODE = 0xE):

FCS = 0: No effect.

FCS = 1: Forces the Slave Select Line NSS (RTS pin) to 0, even if USART is no transmitting, in order to address SPI slave devices supporting the CSAAT Mode (Chip Select Active After Transfer).

- **RCS: Release SPI Chip Select**
  - Applicable if USART operates in SPI Master Mode (USART_MODE = 0xE):

RCS = 0: No effect.

RCS = 1: Releases the Slave Select Line NSS (RTS pin).

39.8.3 USART Mode Register

Name: US_MR

Address: 0xF801C004 (0), 0xF8020004 (1), 0xF8024004 (2), 0xF8028004 (3)

Access: Read-write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC	MAN	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

For SPI configuration, see “USART Mode Register (SPI_MODE)” on page 847.

- **USART_MODE: USART Mode of Operation**

Value	Name	Description
0x0	NORMAL	Normal mode
0x1	RS485	RS485
0x2	HW_HANDSHAKING	Hardware Handshaking
0x4	IS07816_T_0	IS07816 Protocol: T = 0
0x6	IS07816_T_1	IS07816 Protocol: T = 1
0x8	IRDA	IrDA
0xA	LIN_MASTER	LIN Master
0xB	LIN_SLAVE	LIN Slave
0xE	SPI_MASTER	SPI Master
0xF	SPI_SLAVE	SPI Slave

- **USCLKS: Clock Selection**

Value	Name	Description
0	MCK	Master Clock MCK is selected
1	DIV	Internal Clock Divided MCK/DIV (DIV=8) is selected
3	SCK	Serial Clock SLK is selected

- **CHRL: Character Length.**

Value	Name	Description
0	5_BIT	Character length is 5 bits
1	6_BIT	Character length is 6 bits
2	7_BIT	Character length is 7 bits
3	8_BIT	Character length is 8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

Value	Name	Description
0	EVEN	Even parity
1	ODD	Odd parity
2	SPACE	Parity forced to 0 (Space)
3	MARK	Parity forced to 1 (Mark)
4	NO	No parity
6	MULTIDROP	Multidrop mode

- **NBSTOP: Number of Stop Bits**

Value	Name	Description
0	1_BIT	1 stop bit
1	1_5_BIT	1.5 stop bit (SYNC = 0) or reserved (SYNC = 1)
2	2_BIT	2 stop bits

- **CHMODE: Channel Mode**

Value	Name	Description
0	NORMAL	Normal Mode
1	AUTOMATIC	Automatic Echo. Receiver input is connected to the TXD pin.
2	LOCAL_LOOPBACK	Local Loopback. Transmitter output is connected to the Receiver Input.
3	REMOTE_LOOPBACK	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 MODSYNC value.

1: The sync field is updated when a character is written into US_THR register.

- **MAX_ITERATION: Maximum Number of Automatic Iteration**

0 - 7: 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.

39.8.4 USART Mode Register (SPI_MODE)

Name: US_MR (SPI_MODE)

Address: 0xF801C004 (0), 0xF8020004 (1), 0xF8024004 (2), 0xF8028004 (3)

Access: Read-write

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

This configuration is relevant only if USART_MODE=0xE or 0xF in “USART Mode Register” on page 844.

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **USART_MODE: USART Mode of Operation**

Value	Name	Description
0xE	SPI_MASTER	SPI Master
0xF	SPI_SLAVE	SPI Slave

- **USCLKS: Clock Selection**

Value	Name	Description
0	MCK	Master Clock MCK is selected
1	DIV	Internal Clock Divided MCK/DIV (DIV=8) is selected
3	SCK	Serial Clock SLK is selected

- **CHRL: Character Length.**

Value	Name	Description
3	8_BIT	Character length is 8 bits

- **CPHA: SPI Clock Phase**
  - Applicable if USART operates in SPI Mode (USART_MODE = 0xE or 0xF):

CPHA = 0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

CPHA = 1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

CPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CHMODE: Channel Mode**

Value	Name	Description
0	NORMAL	Normal Mode
1	AUTOMATIC	Automatic Echo. Receiver input is connected to the TXD pin.
2	LOCAL_LOOPBACK	Local Loopback. Transmitter output is connected to the Receiver Input.
3	REMOTE_LOOPBACK	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **CPOL: SPI Clock Polarity**
  - Applicable if USART operates in SPI Mode (Slave or Master, USART_MODE = 0xE or 0xF):

CPOL = 0: The inactive state value of SPCK is logic level zero.

CPOL = 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 CPHA to produce the required clock/data relationship between master and slave devices.

- **WRDBT: Wait Read Data Before Transfer**

0: The character transmission starts as soon as a character is written into US_THR register (assuming TXRDY was set).

1: The character transmission starts when a character is written and only if RXRDY flag is cleared (Receiver Holding Register has been read).

39.8.5 USART Interrupt Enable Register

Name: US_IER

Address: 0xF801C008 (0), 0xF8020008 (1), 0xF8024008 (2), 0xF8028008 (3)

Access: Write-only

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

For SPI specific configuration, see “USART Interrupt Enable Register (SPI_MODE)” on page 850.

For LIN specific configuration, see “USART Interrupt Enable Register (LIN_MODE)” on page 851.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break 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: Max number of Repetitions Reached Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

39.8.6 USART Interrupt Enable Register (SPI_MODE)

Name: US_IER (SPI_MODE)

Address: 0xF801C008 (0), 0xF8020008 (1), 0xF8024008 (2), 0xF8028008 (3)

Access: Write-only

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

This configuration is relevant only if USART_MODE=0xE or 0xF in “USART Mode Register” on page 844.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **UNRE: SPI Underrun Error Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

39.8.7 USART Interrupt Enable Register (LIN_MODE)

Name: US_IER (LIN_MODE)

Address: 0xF801C008 (0), 0xF8020008 (1), 0xF8024008 (2), 0xF8028008 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE=0xA or 0xB in “USART Mode Register” on page 844.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY 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**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Enable**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Enable**
- **LINTC: LIN Transfer Completed Interrupt Enable**
- **LINBE: LIN Bus Error Interrupt Enable**
- **LINISFE: LIN Inconsistent Synch Field Error Interrupt Enable**
- **LINIPE: LIN Identifier Parity Interrupt Enable**
- **LINCE: LIN Checksum Error Interrupt Enable**
- **LINSNRE: LIN Slave Not Responding Error Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

39.8.8 USART Interrupt Disable Register

Name: US_IDR

Address: 0xF801C00C (0), 0xF802000C (1), 0xF802400C (2), 0xF802800C (3)

Access: Write-only

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

For SPI specific configuration, see “USART Interrupt Disable Register (SPI_MODE)” on page 853.

For LIN specific configuration, see “USART Interrupt Disable Register (LIN_MODE)” on page 854.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITER: Max Number of Repetitions Reached Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**
- **MANE: Manchester Error Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

39.8.9 USART Interrupt Disable Register (SPI_MODE)

Name: US_IDR (SPI_MODE)

Address: 0xF801C00C (0), 0xF802000C (1), 0xF802400C (2), 0xF802800C (3)

Access: Write-only

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

This configuration is relevant only if USART_MODE=0xE or 0xF in “USART Mode Register” on page 844.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **UNRE: SPI Underrun Error Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

39.8.10 USART Interrupt Disable Register (LIN_MODE)

Name: US_IDR (LIN_MODE)

Address: 0xF801C00C (0), 0xF802000C (1), 0xF802400C (2), 0xF802800C (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE=0xA or 0xB in "USART Mode Register" on page 844.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY 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**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Disable**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Disable**
- **LINTC: LIN Transfer Completed Interrupt Disable**
- **LINBE: LIN Bus Error Interrupt Disable**
- **LINISFE: LIN Inconsistent Synch Field Error Interrupt Disable**
- **LINIPE: LIN Identifier Parity Interrupt Disable**
- **LINCE: LIN Checksum Error Interrupt Disable**
- **LINSNRE: LIN Slave Not Responding Error Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

39.8.11 USART Interrupt Mask Register

Name: US_IMR

Address: 0xF801C010 (0), 0xF8020010 (1), 0xF8024010 (2), 0xF8028010 (3)

Access: Read-only

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

For SPI specific configuration, see “USART Interrupt Mask Register (SPI_MODE)” on page 856.

For LIN specific configuration, see “USART Interrupt Mask Register (LIN_MODE)” on page 857.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break 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: Max Number of Repetitions Reached Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**
- **MANE: Manchester Error Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

39.8.12 USART Interrupt Mask Register (SPI_MODE)

Name: US_IMR (SPI_MODE)

Address: 0xF801C010 (0), 0xF8020010 (1), 0xF8024010 (2), 0xF8028010 (3)

Access: Read-only

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

This configuration is relevant only if USART_MODE=0xE or 0xF in “USART Mode Register” on page 844.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **UNRE: SPI Underrun Error Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

39.8.13 USART Interrupt Mask Register (LIN_MODE)

Name: US_IMR (LIN_MODE)

Address: 0xF801C010 (0), 0xF8020010 (1), 0xF8024010 (2), 0xF8028010 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE=0xA or 0xB in “USART Mode Register” on page 844.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY 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**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Mask**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Mask**
- **LINTC: LIN Transfer Completed Interrupt Mask**
- **LINBE: LIN Bus Error Interrupt Mask**
- **LINISFE: LIN Inconsistent Synch Field Error Interrupt Mask**
- **LINIPE: LIN Identifier Parity Interrupt Mask**
- **LINCE: LIN Checksum Error Interrupt Mask**
- **LINSNRE: LIN Slave Not Responding Error Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

39.8.14 USART Channel Status Register

Name: US_CSR

Address: 0xF801C014 (0), 0xF8020014 (1), 0xF8024014 (2), 0xF8028014 (3)

Access: Read-only

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

For SPI specific configuration, see "USART Channel Status Register (SPI_MODE)" on page 860.

For LIN specific configuration, see "USART Channel Status Register (LIN_MODE)" on page 861.

- **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.

- **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: MaxNumber 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.

- **NACK: Non Acknowledge Interrupt**

0: 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 set to 0.

1: CTS is set to 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.

39.8.15 USART Channel Status Register (SPI_MODE)

Name: US_CSR (SPI_MODE)

Address: 0xF801C014 (0), 0xF8020014 (1), 0xF8024014 (2), 0xF8028014 (3)

Access: Read-only

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

This configuration is relevant only if USART_MODE=0xE or 0xF in “USART Mode Register” on page 844.

- **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 the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **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.

- **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.

- **UNRE: Underrun Error**

0: No SPI underrun error has occurred since the last RSTSTA.

1: At least one SPI underrun error has occurred since the last RSTSTA.

39.8.16 USART Channel Status Register (LIN_MODE)

Name: US_CSR (LIN_MODE)

Address: 0xF801C014 (0), 0xF8020014 (1), 0xF8024014 (2), 0xF8028014 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
LINBLS	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE=0xA or 0xB in "USART Mode Register" on page 844.

- **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 the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **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.

- **LINBK: LIN Break Sent or LIN Break Received**
  - Applicable if USART operates in LIN Master Mode (USART_MODE = 0xA):

0: No LIN Break has been sent since the last RSTSTA.

1:At least one LIN Break has been sent since the last RSTSTA

  - If USART operates in LIN Slave Mode (USART_MODE = 0xB):

0: No LIN Break has received sent since the last RSTSTA.

1:At least one LIN Break has been received since the last RSTSTA.

- **LINID: LIN Identifier Sent or LIN Identifier Received**
  - If USART operates in LIN Master Mode (USART_MODE = 0xA):

0: No LIN Identifier has been sent since the last RSTSTA.

1:At least one LIN Identifier has been sent since the last RSTSTA.

  - If USART operates in LIN Slave Mode (USART_MODE = 0xB):

0: No LIN Identifier has been received since the last RSTSTA.

1:At least one LIN Identifier has been received since the last RSTSTA

- **LINTC: LIN Transfer Completed**

0: The USART is idle or a LIN transfer is ongoing.

1: A LIN transfer has been completed since the last RSTSTA.

- **LINBLS: LIN Bus Line Status**

0: LIN Bus Line is set to 0.

1: LIN Bus Line is set to 1.

- **LINBE: LIN Bit Error**

0: No Bit Error has been detected since the last RSTSTA.

1: A Bit Error has been detected since the last RSTSTA.

- **LINISFE: LIN Inconsistent Synch Field Error**

0: No LIN Inconsistent Synch Field Error has been detected since the last RSTSTA

1: The USART is configured as a Slave node and a LIN Inconsistent Synch Field Error has been detected since the last RSTSTA.

- **LINIPE: LIN Identifier Parity Error**

0: No LIN Identifier Parity Error has been detected since the last RSTSTA.

1: A LIN Identifier Parity Error has been detected since the last RSTSTA.

- **LINCE: LIN Checksum Error**

0: No LIN Checksum Error has been detected since the last RSTSTA.

1: A LIN Checksum Error has been detected since the last RSTSTA.

- **LINSNRE: LIN Slave Not Responding Error**

0: No LIN Slave Not Responding Error has been detected since the last RSTSTA.

1: A LIN Slave Not Responding Error has been detected since the last RSTSTA.

39.8.17 USART Receive Holding Register

Name: US_RHR

Address: 0xF801C018 (0), 0xF8020018 (1), 0xF8024018 (2), 0xF8028018 (3)

Access: Read-only

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

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last Character received is a Data.

1: Last Character received is a Command.

39.8.18 USART Transmit Holding Register

Name: US_THR

Address: 0xF801C01C (0), 0xF802001C (1), 0xF802401C (2), 0xF802801C (3)

Access: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

- **TXSYNH: Sync Field to be Transmitted**

0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.

39.8.19 USART Baud Rate Generator Register

Name: US_BRGR

Address: 0xF801C020 (0), 0xF8020020 (1), 0xF8024020 (2), 0xF8028020 (3)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **CD: Clock Divider**

CD	USART_MODE ≠ ISO7816			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1 or USART_MODE = SPI (Master or Slave)	
	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/(FI_DI_RATIO*CD)

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baud rate resolution, defined by FP x 1/8.

39.8.20 USART Receiver Time-out Register

Name: US_RTOR

Address: 0xF801C024 (0), 0xF8020024 (1), 0xF8024024 (2), 0xF8028024 (3)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 131071: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

39.8.21 USART Transmitter Timeguard Register

Name: US_TTGR

Address: 0xF801C028 (0), 0xF8020028 (1), 0xF8024028 (2), 0xF8028028 (3)

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **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.

39.8.22 USART FI DI RATIO Register

Name: US_FIDI

Address: 0xF801C040 (0), 0xF8020040 (1), 0xF8024040 (2), 0xF8028040 (3)

Access: Read-write

Reset: 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							

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **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.

39.8.23 USART Number of Errors Register

Name: US_NER

Address: 0xF801C044 (0), 0xF8020044 (1), 0xF8024044 (2), 0xF8028044 (3)

Access: Read-only

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

This register is relevant only if USART_MODE=0x4 or 0x6 in “USART Mode Register” on page 844.

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

39.8.24 USART IrDA FILTER Register

Name: US_IF

Address: 0xF801C04C (0), 0xF802004C (1), 0xF802404C (2), 0xF802804C (3)

Access: Read-write

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

This register is relevant only if USART_MODE=0x8 in “USART Mode Register” on page 844.

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

39.8.25 USART Manchester Configuration Register

Name: US_MAN

Address: 0xF801C050 (0), 0xF8020050 (1), 0xF8024050 (2), 0xF8028050 (3)

Access: Read-write

31	30	29	28	27	26	25	24
–	DRIFT	ONE	RX_MPOL	–	–	RX_PP	
23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			
15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	
7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **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**

The following values assume that TX_MPOL field is not set:

Value	Name	Description
00	ALL_ONE	The preamble is composed of ‘1’s
01	ALL_ZERO	The preamble is composed of ‘0’s
10	ZERO_ONE	The preamble is composed of ‘01’s
11	ONE_ZERO	The preamble is composed of ‘10’s

- **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**

The following values assume that RX_MPOL field is not set:

Value	Name	Description
00	ALL_ONE	The preamble is composed of ‘1’s

01	ALL_ZERO	The preamble is composed of ‘0’s
10	ZERO_ONE	The preamble is composed of ‘01’s
11	ONE_ZERO	The preamble is composed of ‘10’s

- **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.

- **ONE: Must Be Set to 1**

Bit 29 must always be set to 1 when programming the US_MAN register.

- **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.

39.8.26 USART LIN Mode Register

Name: US_LINMR

Address: 0xF801C054 (0), 0xF8020054 (1), 0xF8024054 (2), 0xF8028054 (3)

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	PDCM
15	14	13	12	11	10	9	8
DLC							
7	6	5	4	3	2	1	0
WKUPTYP	FSDIS	DLM	CHKTYP	CHKDIS	PARDIS	NACT	

This register is relevant only if USART_MODE=0xA or 0xB in “USART Mode Register” on page 844.

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 877.

- **NACT: LIN Node Action**

Value	Name	Description
00	PUBLISH	The USART transmits the response.
01	SUBSCRIBE	The USART receives the response.
10	IGNORE	The USART does not transmit and does not receive the response.

Values which are not listed in the table must be considered as “reserved”.

- **PARDIS: Parity Disable**

0: In Master node configuration, the Identifier Parity is computed and sent automatically. In Master node and Slave node configuration, the parity is checked automatically.

1:Whatever the node configuration is, the Identifier parity is not computed/sent and it is not checked.

- **CHKDIS: Checksum Disable**

0: In Master node configuration, the checksum is computed and sent automatically. In Slave node configuration, the checksum is checked automatically.

1: Whatever the node configuration is, the checksum is not computed/sent and it is not checked.

- **CHKTYP: Checksum Type**

0: LIN 2.0 “Enhanced” Checksum

1: LIN 1.3 “Classic” Checksum

- **DLM: Data Length Mode**

0: The response data length is defined by the field DLC of this register.

1: The response data length is defined by the bits 5 and 6 of the Identifier (IDCHR in US_LINIR).

- **FSDIS: Frame Slot Mode Disable**

0: The Frame Slot Mode is enabled.

1: The Frame Slot Mode is disabled.

- **WKUPTYP: Wakeup Signal Type**

0: Setting the bit LINWKUP in the control register sends a LIN 2.0 wakeup signal.

1: Setting the bit LINWKUP in the control register sends a LIN 1.3 wakeup signal.

- **DLC: Data Length Control**

0 - 255: Defines the response data length if DLM=0,in that case the response data length is equal to DLC+1 bytes.

- **PDCM: DMAC Mode**

0: The LIN mode register US_LINMR is not written by the DMAC.

1: The LIN mode register US_LINMR (excepting that flag) is written by the DMAC.

39.8.27 USART LIN Identifier Register

Name: US_LINIR

Address: 0xF801C058 (0), 0xF8020058 (1), 0xF8024058 (2), 0xF8028058 (3)

Access: Read-write or 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
IDCHR							

This register is relevant only if USART_MODE=0xA or 0xB in “USART Mode Register” on page 844.

- **IDCHR: Identifier Character**

If USART_MODE=0xA (Master node configuration):

IDCHR is Read-write and its value is the Identifier character to be transmitted.

If USART_MODE=0xB (Slave node configuration):

IDCHR is Read-only and its value is the last Identifier character that has been received.

39.8.28 USART LIN Baud Rate Register

Name: US_LINBRR

Address: 0xF801C05C (0), 0xF802005C (1), 0xF802405C (2), 0xF802805C (3)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
LINCD							

7	6	5	4	3	2	1	0
LINCD							

This register is relevant only if USART_MODE=0xA or 0xB in “USART Mode Register” on page 844.

Returns the baud rate value after the synchronization process completion.

- **LINCD: Clock Divider after Synchronization**
- **LINFP: Fractional Part after Synchronization**

39.8.29 USART Write Protect Mode Register

Name: US_WPMR

Address: 0xF801C0E4 (0), 0xF80200E4 (1), 0xF80240E4 (2), 0xF80280E4 (3)

Access: Read-write

Reset: See Table 39-16

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x555341 (“USA” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x555341 (“USA” in ASCII).

Protects the registers:

- “USART Mode Register” on page 844
- “USART Baud Rate Generator Register” on page 865
- “USART Receiver Time-out Register” on page 866
- “USART Transmitter Timeguard Register” on page 867
- “USART FI DI RATIO Register” on page 868
- “USART IrDA FILTER Register” on page 870
- “USART Manchester Configuration Register” on page 871

- **WPKEY: Write Protect KEY**

Should be written at value 0x555341 (“USA” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

39.8.30 USART Write Protect Status Register

Name: US_WPSR

Address: 0xF801C0E8 (0), 0xF80200E8 (1), 0xF80240E8 (2), 0xF80280E8 (3)

Access: Read-only

Reset: See Table 39-16

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the US_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the US_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading US_WPSR automatically clears all fields.

40. Universal Asynchronous Receiver Transmitter (UART)

40.1 Description

The Universal Asynchronous Receiver Transmitter features a two-pin UART that can be used for communication and trace purposes and offers an ideal medium for in-situ programming solutions.

Moreover, the association with DMA controller permits packet handling for these tasks with processor time reduced to a minimum.

40.2 Embedded Characteristics

- Two-pin UART
  - 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
  - Interrupt Generation
  - Support for Two DMA Channels with Connection to Receiver and Transmitter

40.3 Block Diagram

Figure 40-1. UART Functional Block Diagram

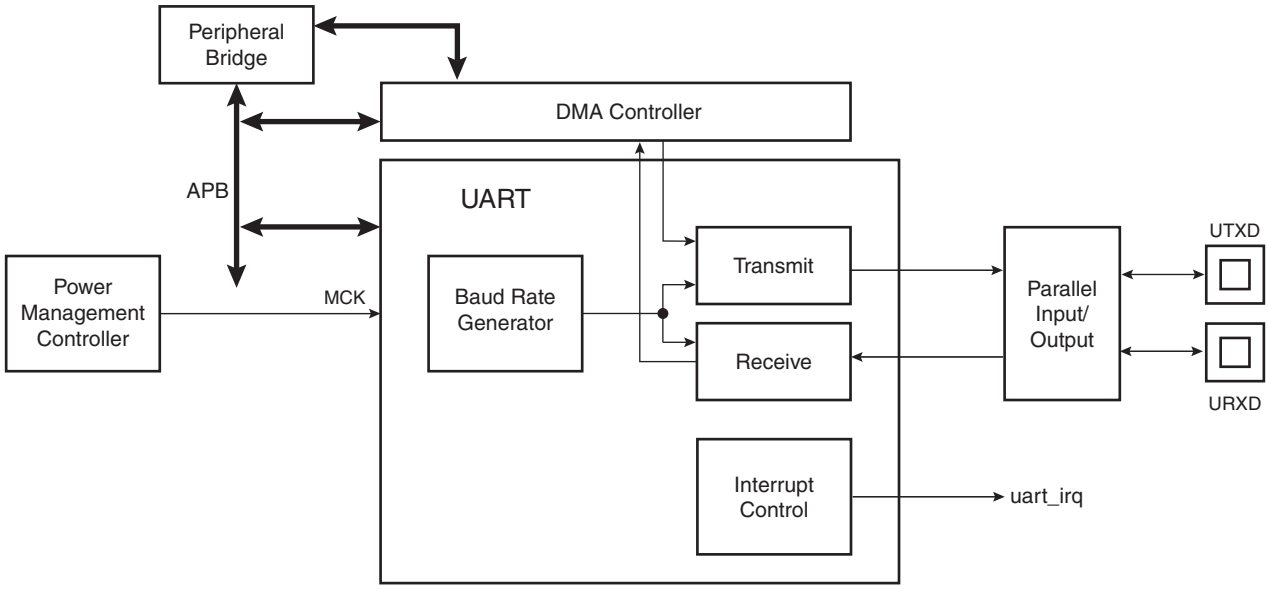


Table 40-1. UART Pin Description

Pin Name	Description	Type
URXD	UART Receive Data	Input
UTXD	UART Transmit Data	Output

40.4 Product Dependencies

40.4.1 I/O Lines

The UART pins are multiplexed with PIO lines. The programmer must first configure the corresponding PIO Controller to enable I/O line operations of the UART.

Table 40-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
UART0	URXD0	PC9	C
UART0	UTXD0	PC8	C
UART1	URXD1	PC17	C
UART1	UTXD1	PC16	C

40.4.2 Power Management

The UART clock is controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the UART clock. Usually, the peripheral identifier used for this purpose is 1.

40.4.3 Interrupt Source

The UART interrupt line is connected to one of the interrupt sources of the Interrupt Controller. Interrupt handling requires programming of the Interrupt Controller before configuring the UART.

40.5 UART Operations

The UART operates in asynchronous mode only and supports only 8-bit character handling (with parity). It has no clock pin.

The 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.

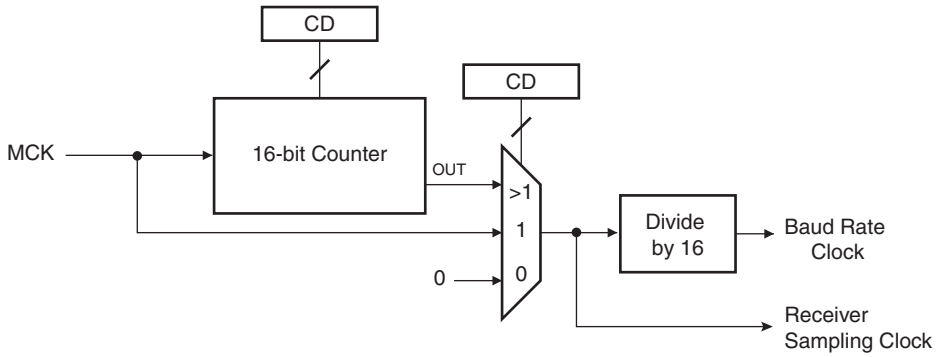
40.5.1 Baud Rate Generator

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 UART_BRGR (Baud Rate Generator Register). If UART_BRGR is set to 0, the baud rate clock is disabled and the 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 40-2. Baud Rate Generator



40.5.2 Receiver

40.5.2.1 Receiver Reset, Enable and Disable

After device reset, the UART receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register UART_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 UART_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 UART_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.

40.5.2.2 Start Detection and Data Sampling

The UART only supports asynchronous operations, and this affects only its receiver. The UART receiver detects the start of a received character by sampling the URXD signal until it detects a valid start bit. A low level (space) on URXD 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 URXD at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after the falling edge of the start bit was detected.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 40-3. Start Bit Detection

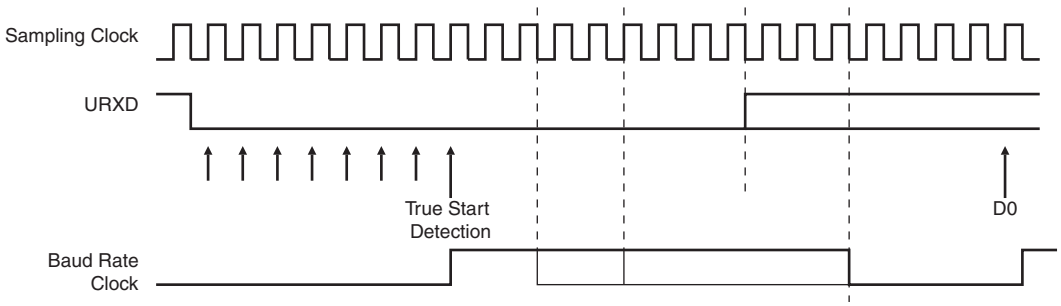
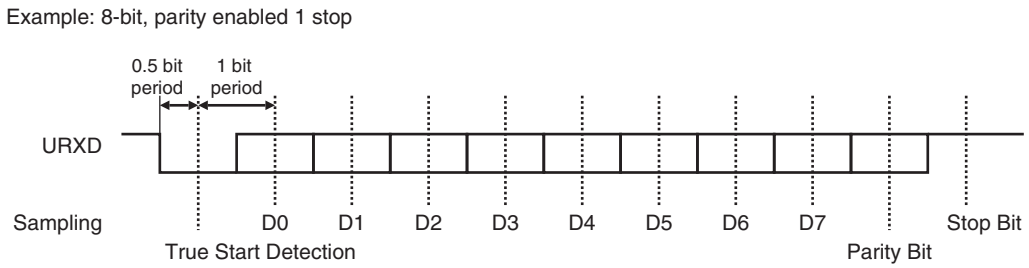


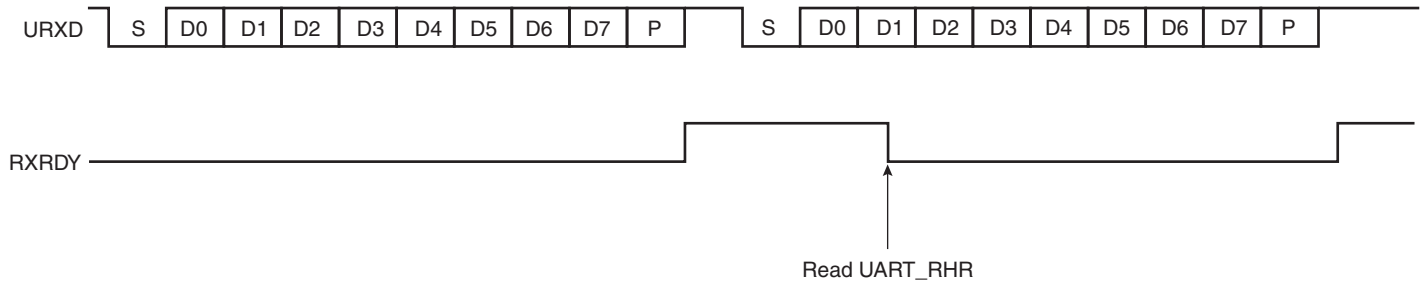
Figure 40-4. Character Reception



40.5.2.3 Receiver Ready

When a complete character is received, it is transferred to the UART_RHR and the RXRDY status bit in UART_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register UART_RHR is read.

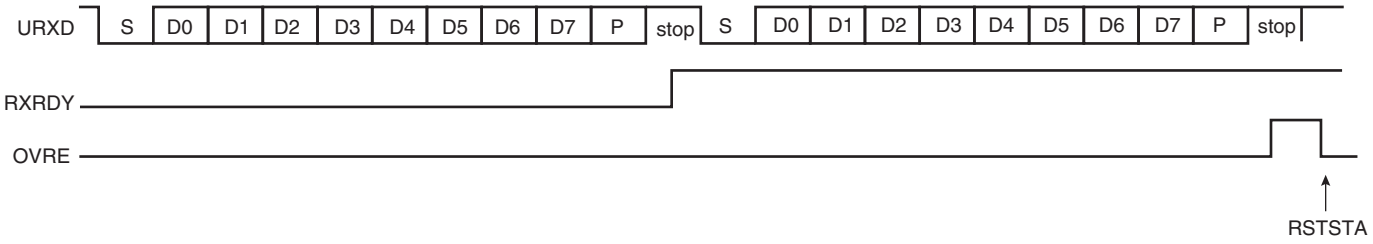
Figure 40-5. Receiver Ready



40.5.2.4 Receiver Overrun

If UART_RHR has not been read by the software (or the Peripheral Data Controller or DMA Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in UART_SR is set. OVRE is cleared when the software writes the control register UART_CR with the bit RSTSTA (Reset Status) at 1.

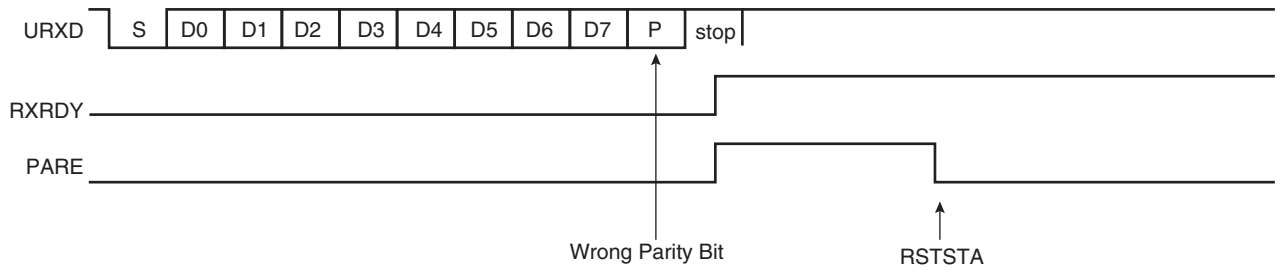
Figure 40-6. Receiver Overrun



40.5.2.5 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 UART_MR. It then compares the result with the received parity bit. If different, the parity error bit PARE in UART_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register UART_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.

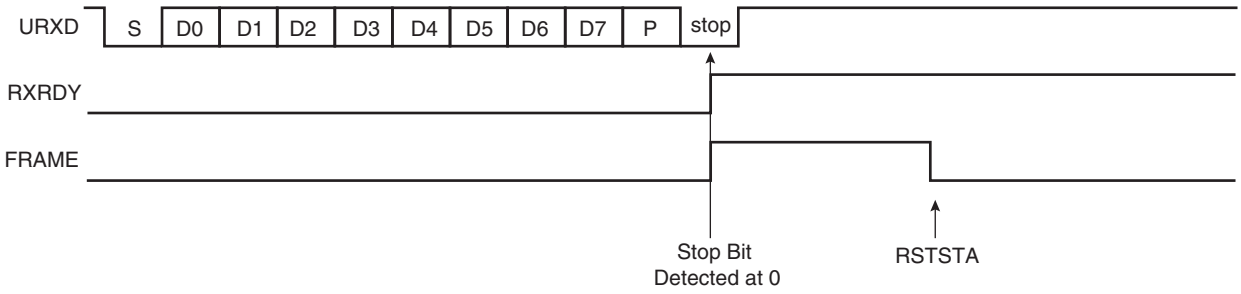
Figure 40-7. Parity Error



40.5.2.6 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 UART_SR is set at the same time the RXRDY bit is set. The FRAME bit remains high until the control register UART_CR is written with the bit RSTSTA at 1.

Figure 40-8. Receiver Framing Error



40.5.3 Transmitter

40.5.3.1 Transmitter Reset, Enable and Disable

After device reset, the UART transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register UART_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register (UART_THR) before actually starting the transmission.

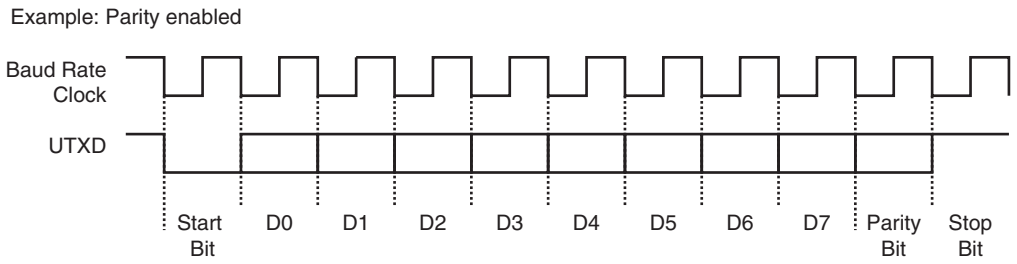
The programmer can disable the transmitter by writing UART_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

The programmer can also put the transmitter in its reset state by writing the UART_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

40.5.3.2 Transmit Format

The UART transmitter drives the pin UTXD 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 in the following figure. The field PARE in the mode register UART_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 40-9. Character Transmission

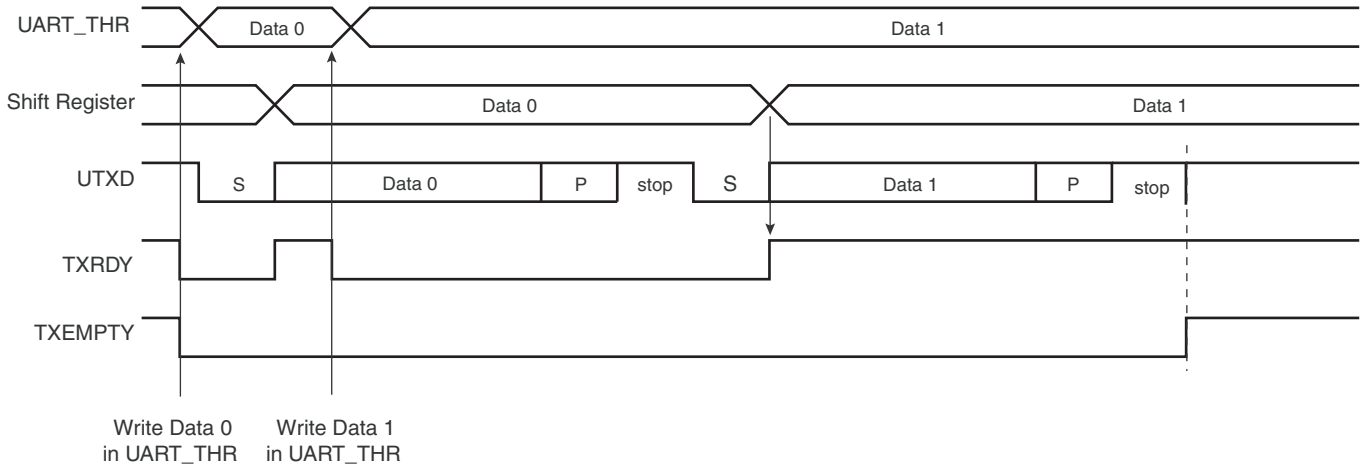


40.5.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register UART_SR. The transmission starts when the programmer writes in the Transmit Holding Register (UART_THR), and after the written character is transferred from UART_THR to the Shift Register. The TXRDY bit remains high until a second character is written in UART_THR. As soon as the first character is completed, the last character written in UART_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and UART_THR are empty, i.e., all the characters written in UART_THR have been processed, the TXEMPTY bit rises after the last stop bit has been completed.

Figure 40-10.Transmitter Control



40.5.4 DMA Support

Both the receiver and the transmitter of the UART are connected to a DMA Controller (DMAC) channel.

The DMA Controller channels are programmed via registers that are mapped within the DMAC user interface.

40.5.5 Test Modes

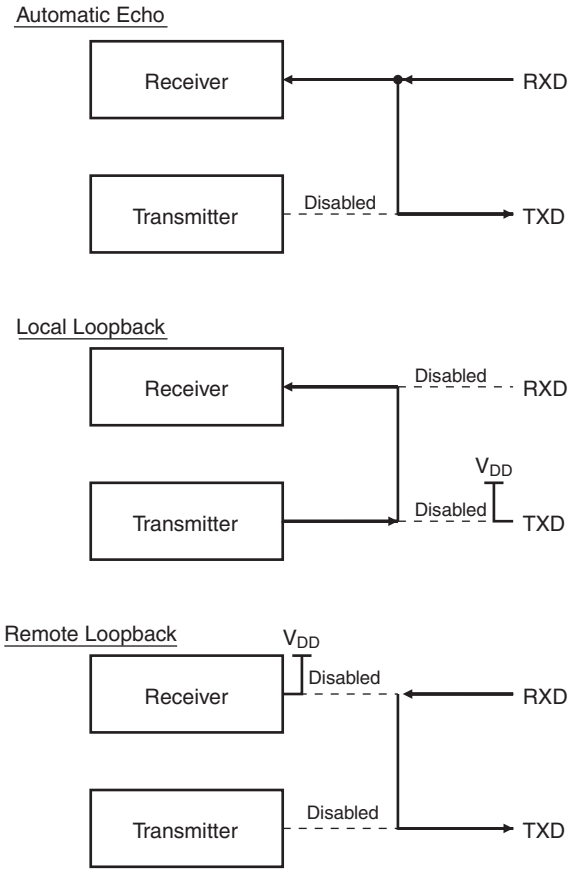
The UART supports three test modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register (UART_MR).

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the URXD line, it is sent to the UTXD line. The transmitter operates normally, but has no effect on the UTXD line.

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

The Remote Loopback mode directly connects the URXD pin to the UTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Figure 40-11.Test Modes



40.6 Universal Asynchronous Receiver Transmitter (UART) User Interface

Table 40-3. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	UART_CR	Write-only	–
0x0004	Mode Register	UART_MR	Read-write	0x0
0x0008	Interrupt Enable Register	UART_IER	Write-only	–
0x000C	Interrupt Disable Register	UART_IDR	Write-only	–
0x0010	Interrupt Mask Register	UART_IMR	Read-only	0x0
0x0014	Status Register	UART_SR	Read-only	–
0x0018	Receive Holding Register	UART_RHR	Read-only	0x0
0x001C	Transmit Holding Register	UART_THR	Write-only	–
0x0020	Baud Rate Generator Register	UART_BRGR	Read-write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x004C - 0x00FC	Reserved	–	–	–

40.6.1 UART Control Register

Name: UART_CR

Address: 0xF8040000 (0), 0xF8044000 (1)

Access: Write-only

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

40.6.2 UART Mode Register

Name: UART_MR

Address: 0xF8040004 (0), 0xF8044004 (1)

Access: Read-write

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

- **PAR: Parity Type**

Value	Name	Description
0	EVEN	Even Parity
1	ODD	Odd Parity
2	SPACE	Space: parity forced to 0
3	MARK	Mark: parity forced to 1
4	NO	No Parity

- **CHMODE: Channel Mode**

Value	Name	Description
0	NORMAL	Normal Mode
1	AUTOMATIC	Automatic Echo
2	LOCAL_LOOPBACK	Local Loopback
3	REMOTE_LOOPBACK	Remote Loopback

40.6.3 UART Interrupt Enable Register

Name: UART_IER

Address: 0xF8040008 (0), 0xF8044008 (1)

Access: Write-only

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

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

40.6.4 UART Interrupt Disable Register

Name: UART_IDR

Address: 0xF804000C (0), 0xF804400C (1)

Access: Write-only

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

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

40.6.5 UART Interrupt Mask Register

Name: UART_IMR

Address: 0xF8040010 (0), 0xF8044010 (1)

Access: Read-only

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

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

40.6.6 UART Status Register

Name: UART_SR

Address: 0xF8040014 (0), 0xF8044014 (1)

Access: Read-only

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

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the UART_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to UART_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to UART_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to UART_THR not yet transferred to the Shift Register.

- **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 UART_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in UART_THR and there are no characters being processed by the transmitter.

40.6.7 UART Receiver Holding Register

Name: UART_RHR

Address: 0xF8040018 (0), 0xF8044018 (1)

Access: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

40.6.8 UART Transmit Holding Register

Name: UART_THR

Address: 0xF804001C (0), 0xF804401C (1)

Access: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

40.6.9 UART Baud Rate Generator Register

Name: UART_BRGR

Address: 0xF8040020 (0), 0xF8044020 (1)

Access: Read-write

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

- **CD: Clock Divisor**

0 = Baud Rate Clock is disabled

1 to 65,535 = MCK / (CD x 16)

41. Controller Area Network (CAN) Programmer Datasheet

41.1 Description

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. 8 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.

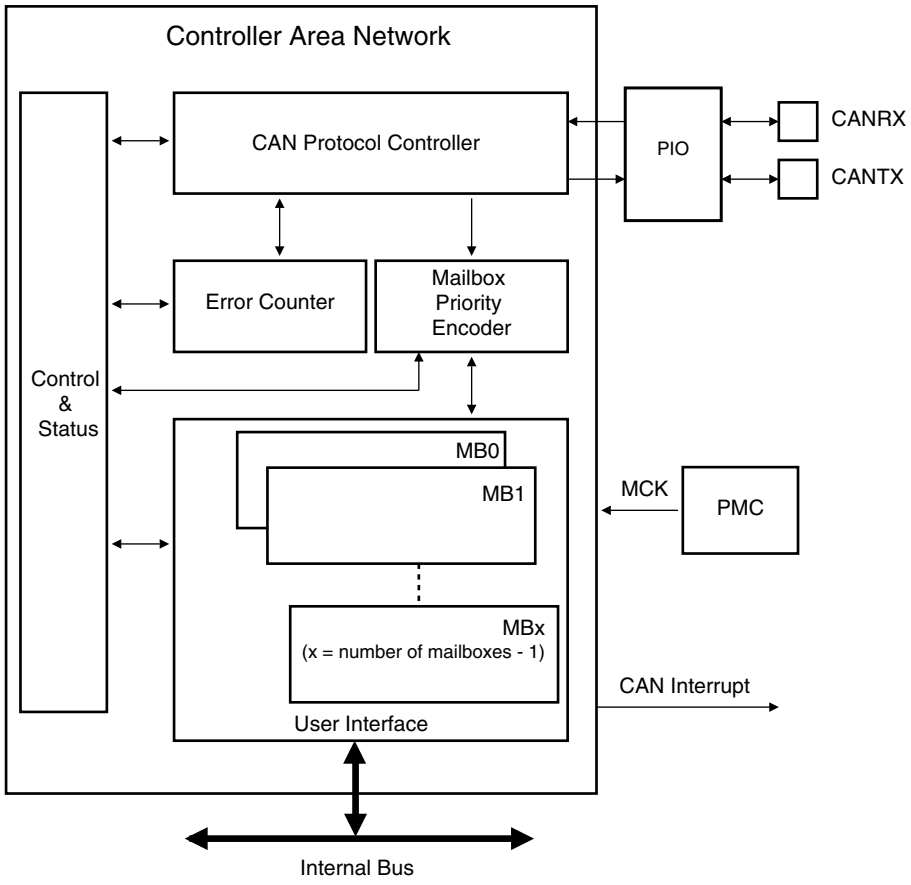
The CAN controller offers optimized features to support the Time Triggered Communication (TTC) protocol.

41.2 Embedded Characteristics

- Fully Compliant with CAN 2.0 Part A and 2.0 Part B
- Bit Rates up to 1Mbit/s
- 8 Object Oriented Mailboxes with the Following Properties:
  - CAN Specification 2.0 Part A or 2.0 Part B Programmable for Each Message
  - Object Configurable in Receive (with Overwrite or Not) or Transmit Modes
  - Independent 29-bit Identifier and Mask Defined for Each Mailbox
  - 32-bit Access to Data Registers for Each Mailbox Data Object
  - Uses a 16-bit Timestamp on Receive and Transmit Messages
  - Hardware Concatenation of ID Masked Bitfields To Speed Up Family ID Processing
- 16-bit Internal Timer for Timestamping and Network Synchronization
- Programmable Reception Buffer Length up to 8 Mailbox Objects
- 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
- Write Protected Registers

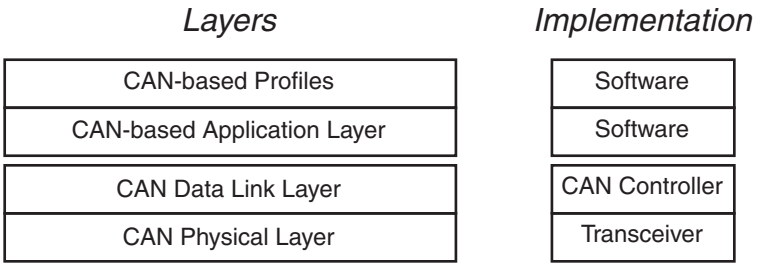
41.3 Block Diagram

Figure 41-1. CAN Block Diagram



41.4 Application Block Diagram

Figure 41-2. Application Block Diagram



41.5 I/O Lines Description

Table 41-1. I/O Lines Description

Name	Description	Type
CANRX	CAN Receive Serial Data	Input
CANTX	CAN Transmit Serial Data	Output

41.6 Product Dependencies

41.6.1 I/O Lines

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

Table 41-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
CAN0	CANRX0	PA9	B
CAN0	CANTX0	PA10	B
CAN1	CANRX1	PA6	B
CAN1	CANTX1	PA5	B

41.6.2 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.

41.6.3 Interrupt

The CAN interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the CAN interrupt requires the AIC to be programmed first. Note that it is not recommended to use the CAN interrupt line in edge-sensitive mode.

Table 41-3. Peripheral IDs

Instance	ID
CAN0	29
CAN1	30

41.7 CAN Controller Features

41.7.1 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.

41.7.2 Mailbox Organization

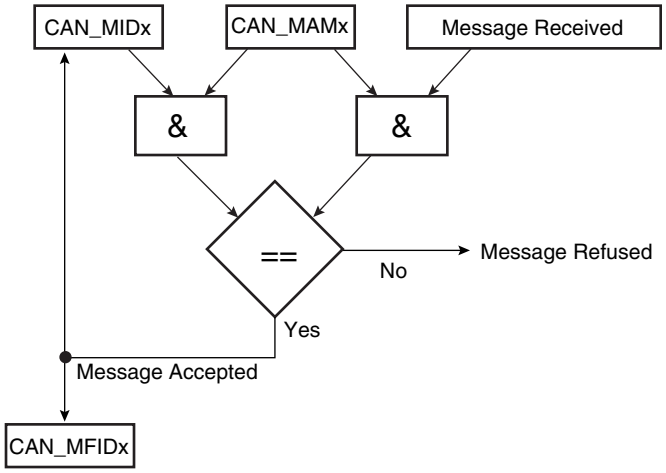
The CAN module has 8 buffers, also called channels or mailboxes. An identifier that corresponds to the CAN identifier is defined for each active mailbox. Message identifiers can match the standard frame identifier or the extended frame identifier. This identifier is defined for the first time during 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.

41.7.2.1 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 41-3. 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]();
```

41.7.2.2 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.

Table 41-4.

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.

41.7.2.3 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.

Table 41-5.

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 41.7.3). 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.

41.7.3 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.

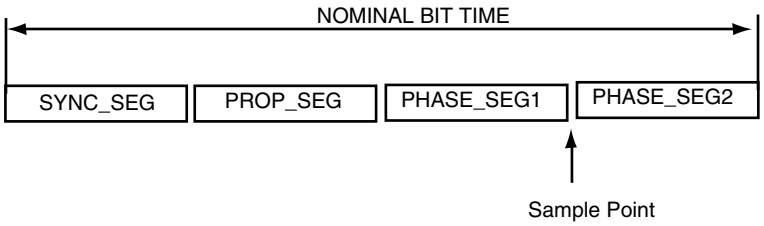
41.7.4 CAN 2.0 Standard Features

41.7.4.1 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 41-4. 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:

$$t_{CSC} = (BRP + 1)/MCK$$

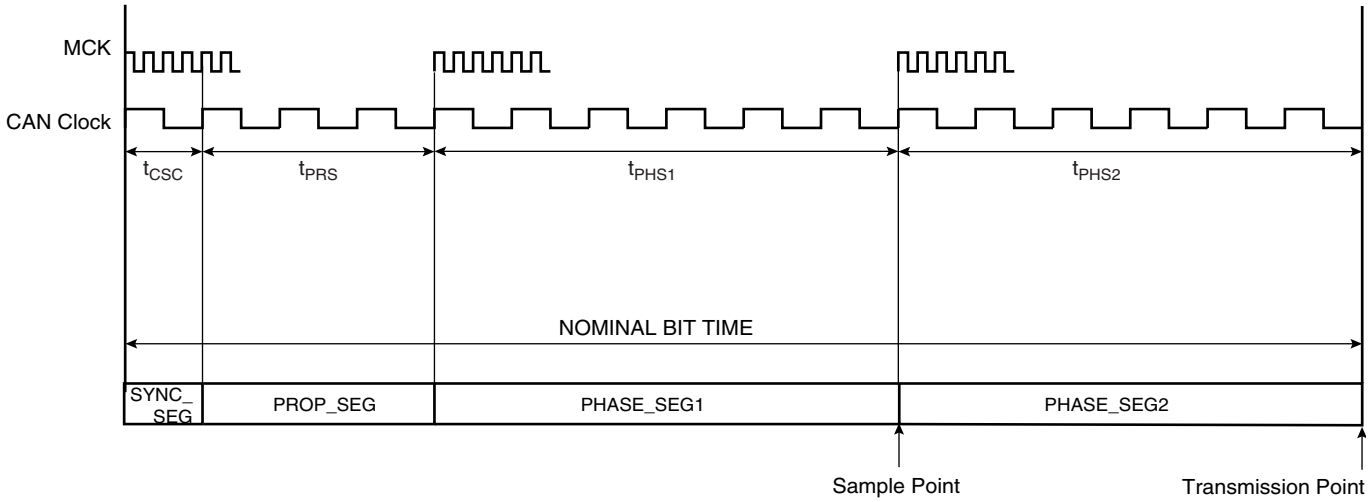
Note: The BRP field must be within the range [1, 0x7F], i.e., BRP = 0 is not authorized.

$$t_{PRS} = t_{CSC} \times (PROPAG + 1)$$
$$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$$
$$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$$

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 41-5. 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
```

CAN Bus Synchronization

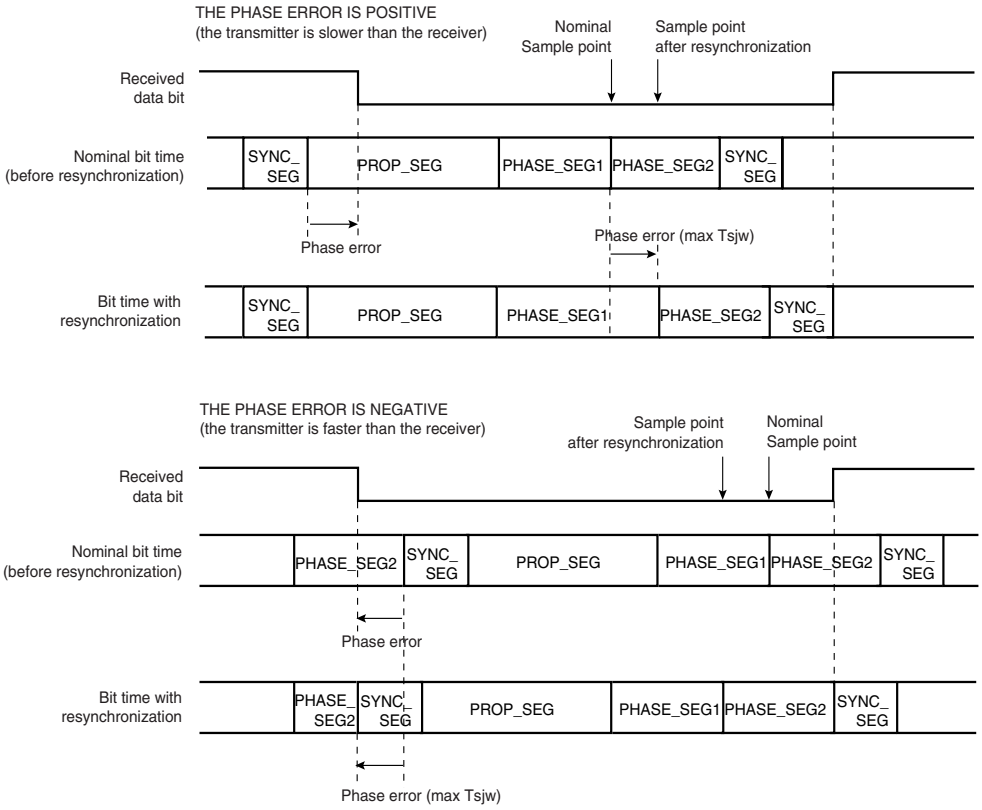
Two types of synchronization are distinguished: “hard synchronization” at the start of a frame and “resynchronization” inside a frame. After a hard synchronization, the bit time is restarted with the end of the SYNC_SEG segment, regardless of the phase error. Resynchronization causes a reduction or increase 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 41-6. CAN Resynchronization



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.

41.7.4.2 Error Detection

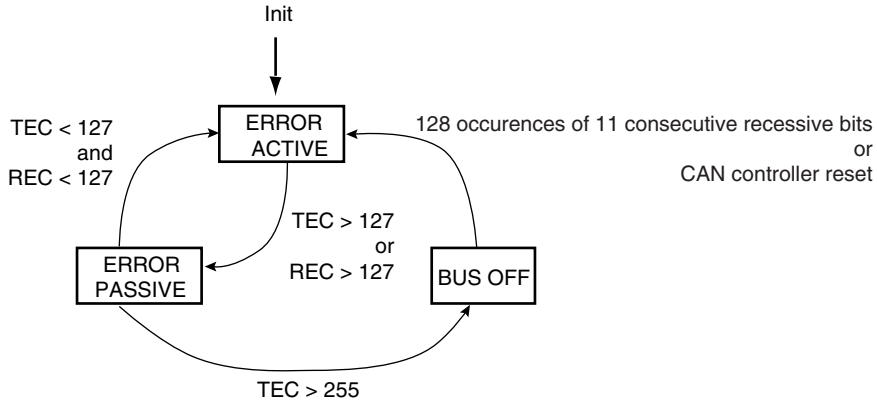
There are five different error types that are not mutually exclusive. Each error concerns only specific fields of the CAN data frame (refer to the Bosch CAN specification for their correspondence):

- 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.

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 41-7. 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 enters 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 enters 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 enters 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.

Error Interrupt Handler

ERRA, WARN, ERRP and BOFF (CAN_SR) store the key transitions of the CAN bus status as defined in Figure 41-7 on page 907. The transitions depend on the TEC and REC (CAN_ECR) values as described in Section “Fault Confinement” on page 907.

These flags are latched to keep from triggering a spurious interrupt in case these bits are used as the source of an interrupt. Thus, these flags may not reflect the current status of the CAN bus.

The current CAN bus state can be determined by reading the TEC and REC fields of CAN_ECR.

41.7.4.3 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.

41.7.5 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.

41.7.5.1 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.

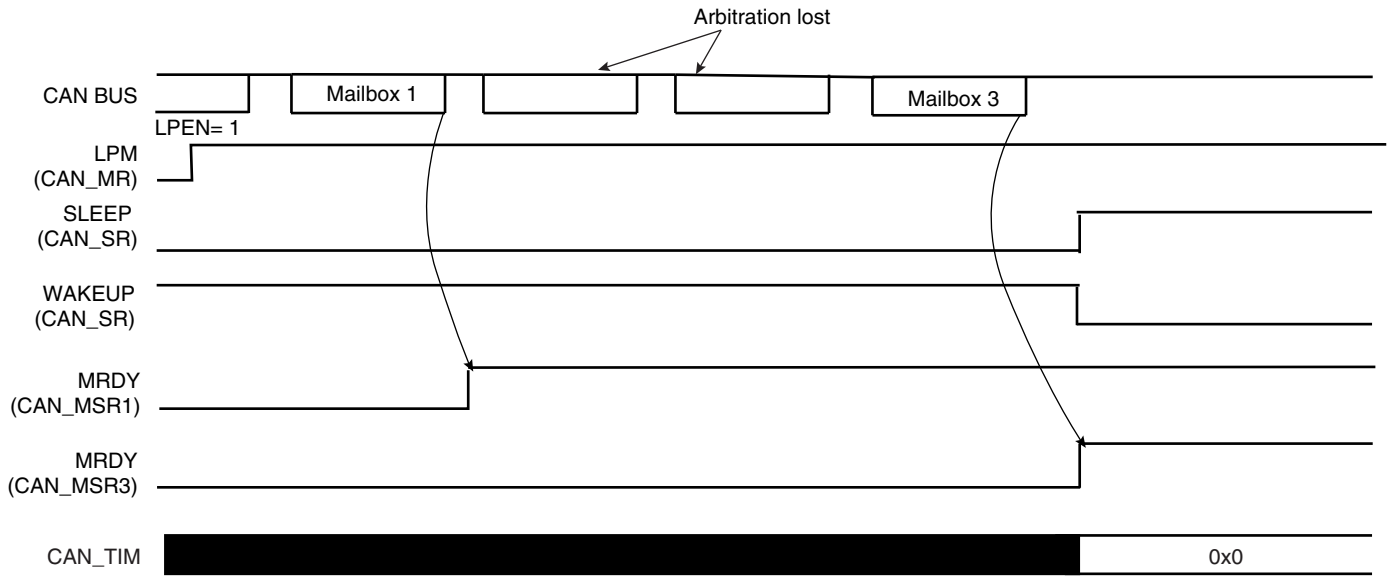
Error counters are disabled while the SLEEP signal is set to one.

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 41-8. Enabling Low-power Mode



41.7.5.2 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

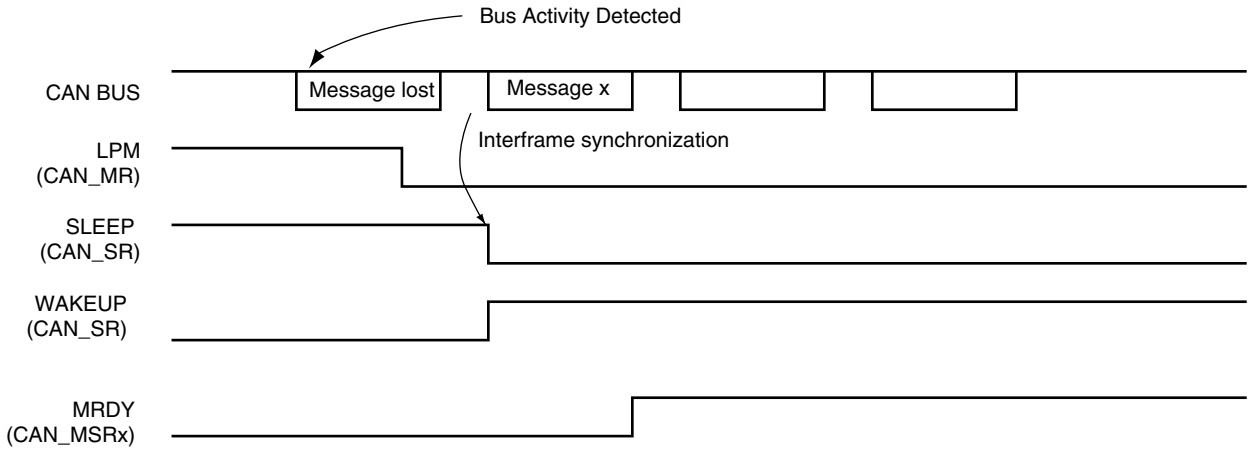
The CAN controller synchronizes itself with the bus activity by checking for eleven consecutive “recessive” bits. Once synchronized, the WAKEUP signal in the CAN_SR register is set.

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 41-9).

Figure 41-9. Disabling Low-power Mode



41.8 Functional Description

41.8.1 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).

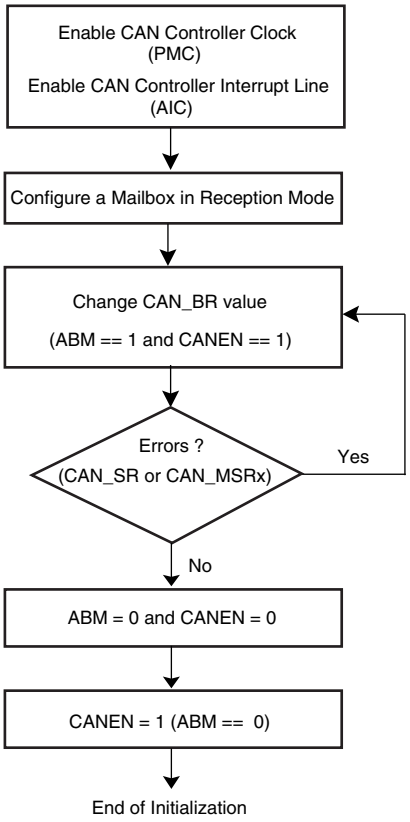
The CAN controller must be initialized with the CAN network parameters. The CAN_BR register defines the sampling point in the bit time period. CAN_BR must be set before the CAN controller is enabled by setting the CANEN field in the CAN_MR register.

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 41-10.Possible Initialization Procedure



41.8.2 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.

41.8.3 CAN Controller Message Handling

41.8.3.1 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.

Simple Receive Mailbox

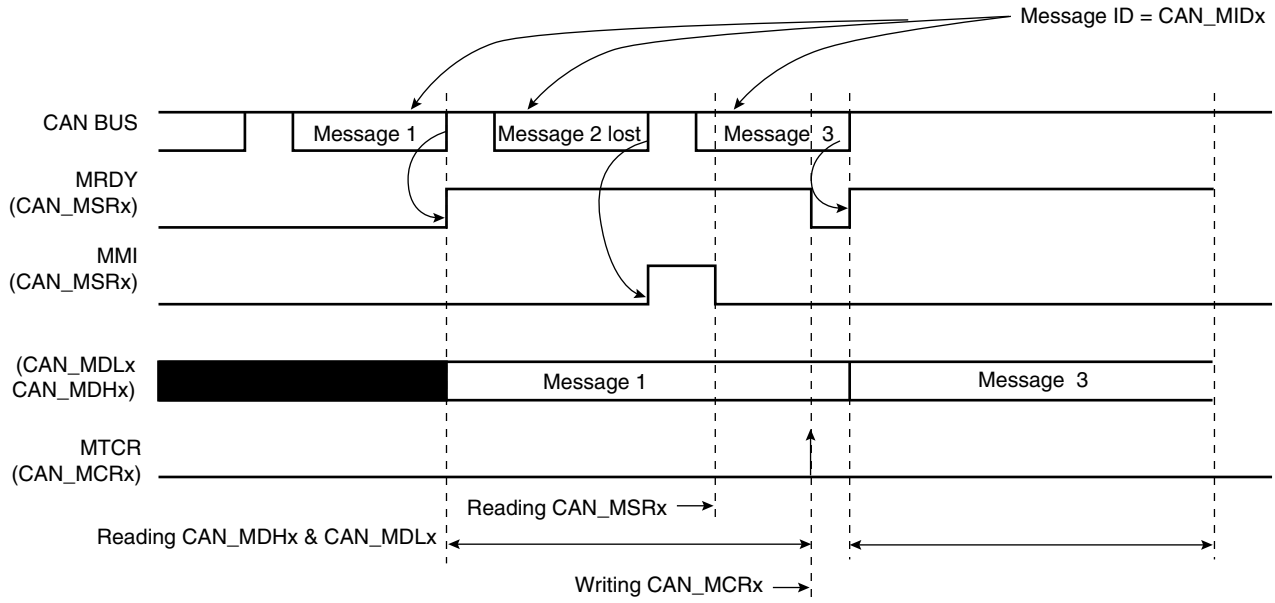
A mailbox is in Receive Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance Mask must be set before the Receive Mode is enabled.

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 41-11.

Figure 41-11.Receive Mailbox



Note: In the case of ARM architecture, CAN_MSRx, CAN_MDLx, CAN_MDHx can be read using an optimized ldm assembler instruction.

Receive with Overwrite Mailbox

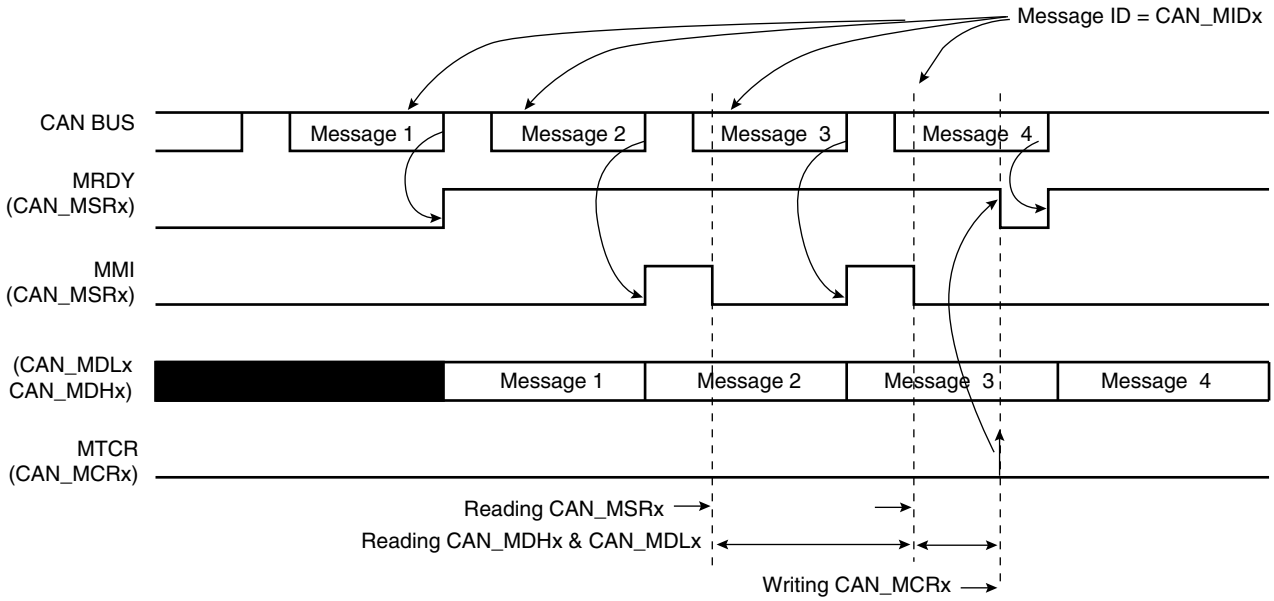
A mailbox is in Receive with Overwrite Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

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 41-12).

Figure 41-12.Receive with Overwrite Mailbox

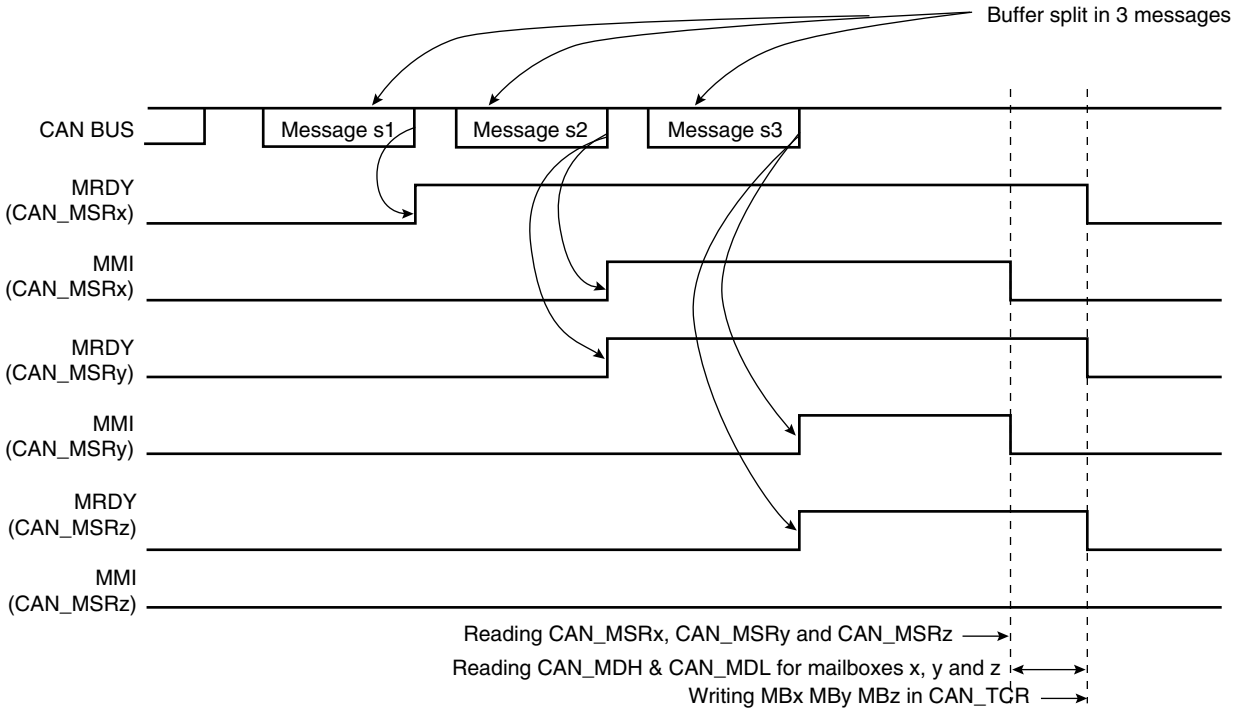


Chaining Mailboxes

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.

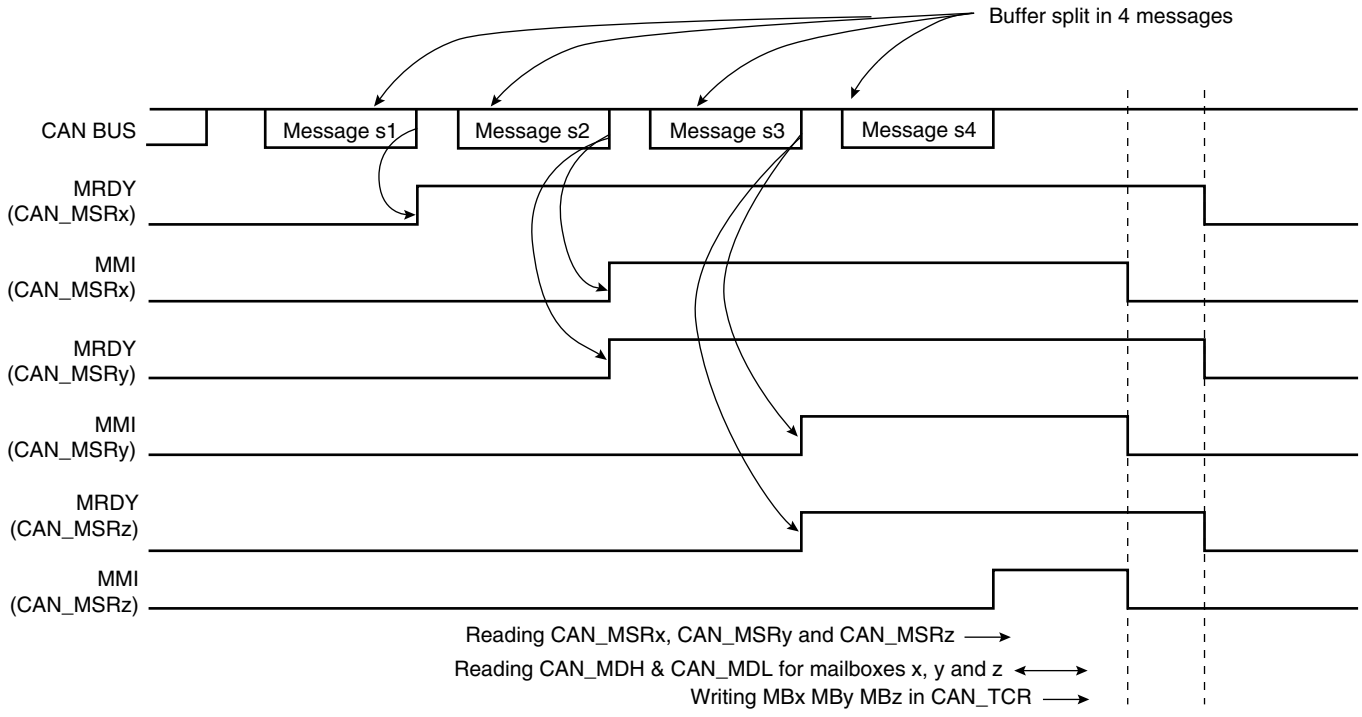
If several mailboxes are chained to receive a buffer split into several messages, all mailboxes except the last one (with the highest number) must be configured in Receive Mode. The first message received is handled by the first mailbox, the second one is refused by the first mailbox and accepted by the second mailbox, the last message is accepted by the last mailbox and refused by previous ones (see Figure 41-13).

Figure 41-13.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 41-14).

Figure 41-14.Chaining Three Mailboxes to Receive a Buffer Split into Four Messages



41.8.3.2 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 917.

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.

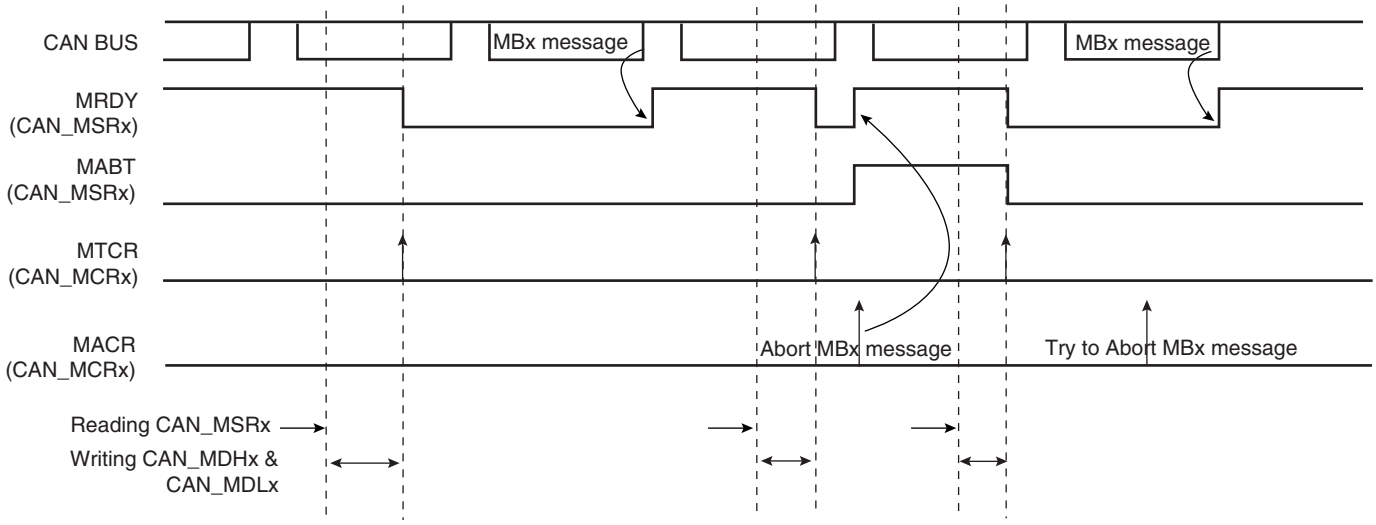
Setting the MACR bit in the CAN_MCRx register aborts the transmission. Transmission for several mailboxes can be aborted by writing MBx fields in the CAN_MACR register. If the message is being sent when the abort command is set, then the application is notified by the MRDY bit set and not the MABT in 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 41-15 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.

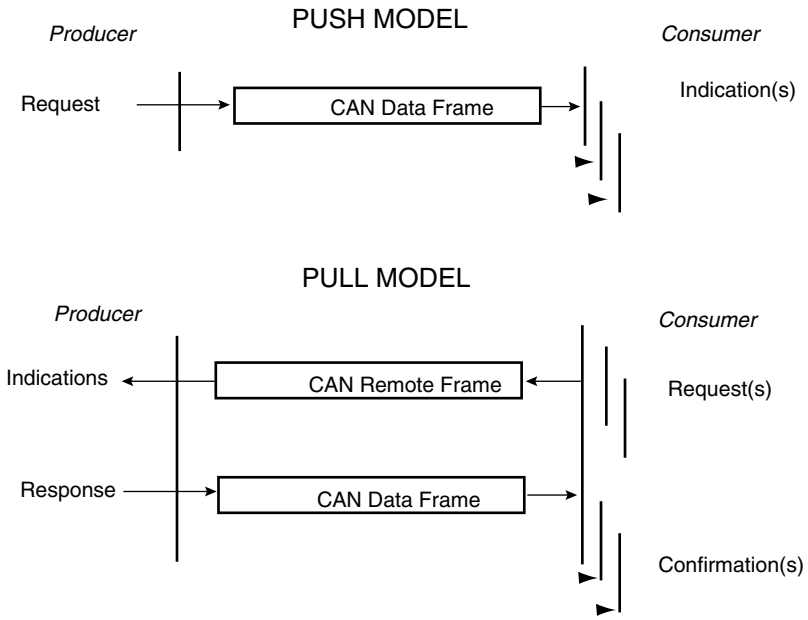
Figure 41-15.Transmitting Messages



41.8.3.3 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 41-16.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 8 mailboxes, the CAN controller can handle 8 independent producers/consumers.

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.

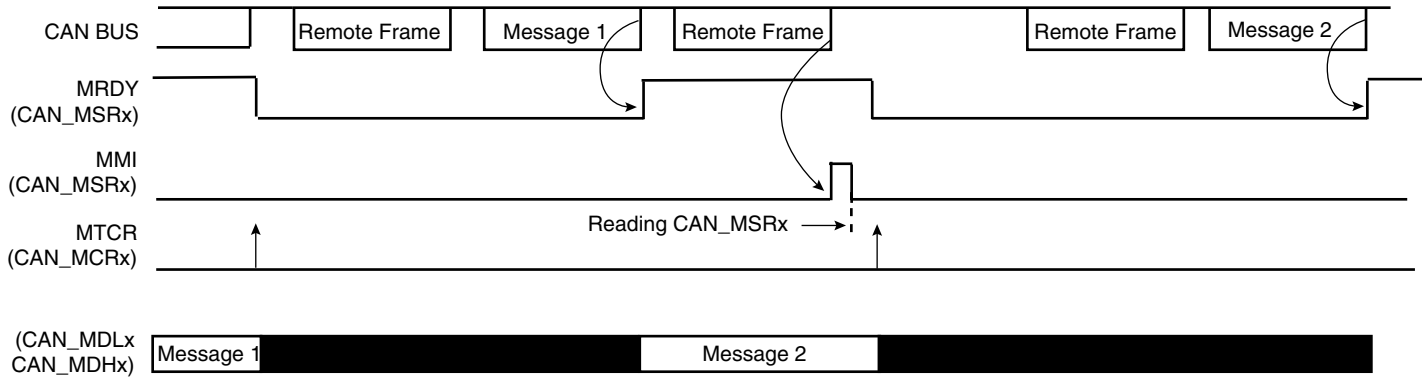
The MRDY flag remains at zero as long as the message has not been sent or aborted. No access to the mailbox data register can be done while MRDY flag is cleared. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked according to the mailbox flag in the CAN_IMR 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 916.

Figure 41-17.Producer Handling



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.

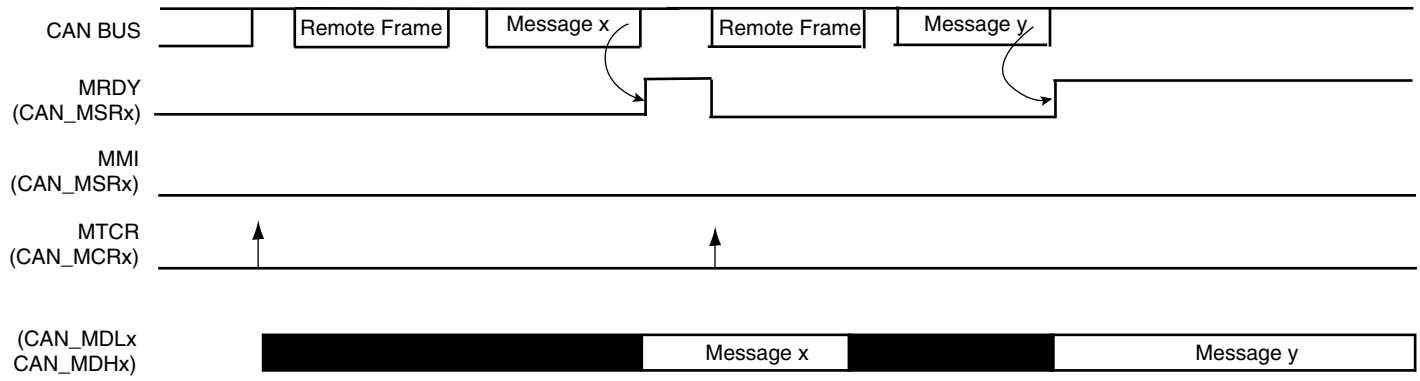
After Consumer Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first transfer request command. The software application sends a remote frame by setting the MTCR bit in the CAN_MCRx register or the MBx bit in the global CAN_TCR register. The application is notified 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 41-18.Consumer Handling



41.8.4 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:

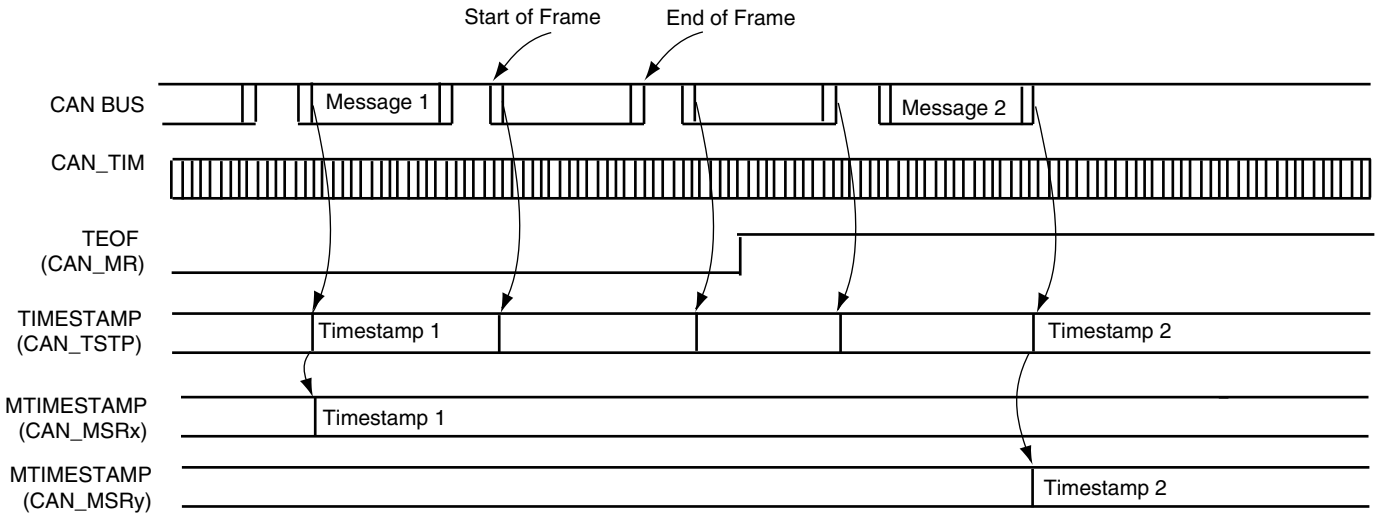
- Timestamping Mode: The value of the internal timer is captured at each Start Of Frame or each End Of Frame.
- Time Triggered Mode: The mailbox transfer operation is triggered when the internal timer reaches the mailbox trigger.

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.

41.8.4.1 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.

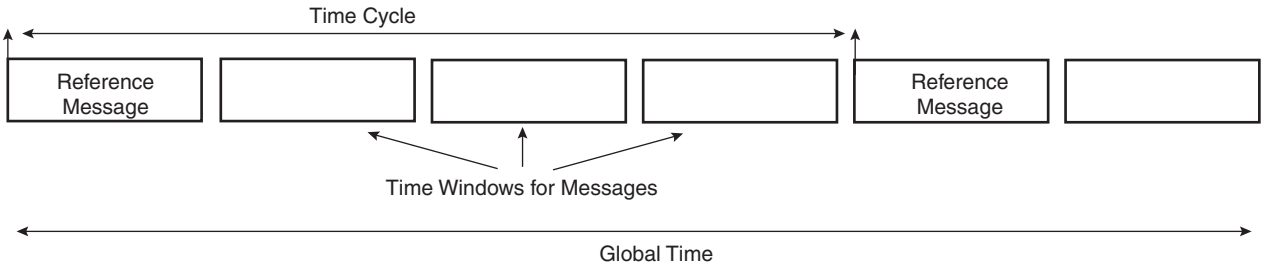
Figure 41-19.Mailbox Timestamp



41.8.4.2 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 41-20.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.

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.

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.

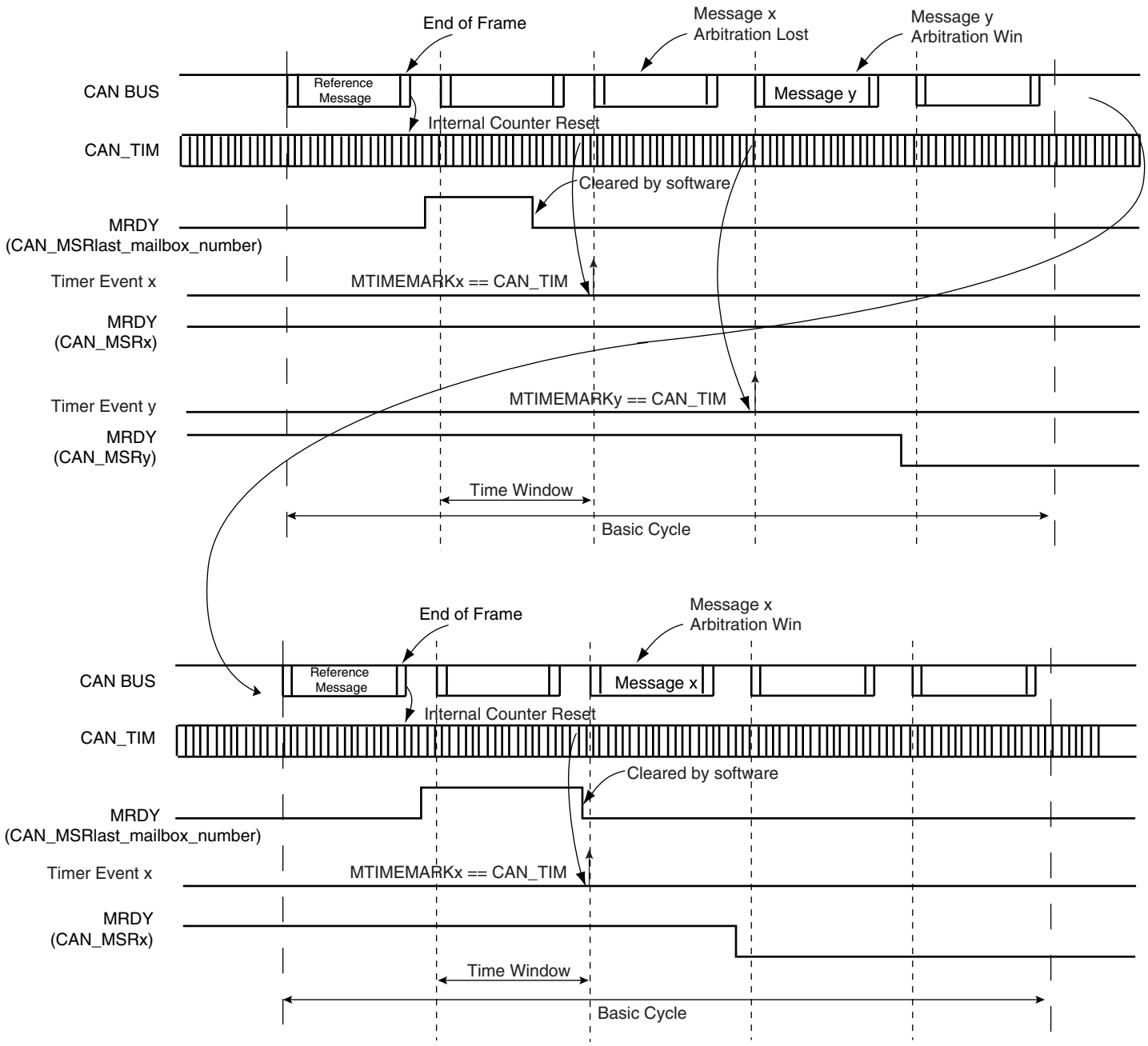
In Time Triggered Mode, transmit operations are delayed until the internal timer event for the mailbox. The application prepares a message to be sent by setting the MTCR in the CAN_MCRx register. The message is not sent until the CAN_TIM value is less than the MTIMEMARK value defined in the CAN_MMRx 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.

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 41-21.Time Triggered Operations



41.8.5 Write Protected Registers

To prevent any single software error that may corrupt CAN behavior, the registers listed below can be write-protected by setting the WPEN bit in the CAN Write Protection Mode Register (CAN_WPMR).

If a write access in a write-protected register is detected, then the WPVS flag in the CAN Write Protection Status Register (CAN_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically reset after reading the CAN Write Protection Status Register (CAN_WPSR).

List of the write-protected registers:

Section 41.9.1 “CAN Mode Register” on page 924

Section 41.9.6 “CAN Baudrate Register” on page 934

Section 41.9.14 “CAN Message Mode Register” on page 942

Section 41.9.15 “CAN Message Acceptance Mask Register” on page 943

Section 41.9.16 “CAN Message ID Register” on page 944

41.9 Controller Area Network (CAN) User Interface

Table 41-6. 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	-
0x002C - 0x00E0	Reserved	–	–	–
0x00E4	Write Protect Mode Register	CAN_WPMR	Read-write	0x0
0x00E8	Write Protect Status Register	CAN_WPSR	Read-only	0x0
0x00EC - 0x01FC	Reserved	–	–	–
0x0200 + MB * 0x20 + 0x00	Mailbox Mode Register(1)	CAN_MMR	Read-write	0x0
0x0200 + MB * 0x20 + 0x04	Mailbox Acceptance Mask Register	CAN_MAM	Read-write	0x0
0x0200 + MB * 0x20 + 0x08	Mailbox ID Register	CAN_MID	Read-write	0x0
0x0200 + MB * 0x20 + 0x0C	Mailbox Family ID Register	CAN_MFID	Read-only	0x0
0x0200 + MB * 0x20 + 0x10	Mailbox Status Register	CAN_MSR	Read-only	0x0
0x0200 + MB * 0x20 + 0x14	Mailbox Data Low Register	CAN_MDL	Read-write	0x0
0x0200 + MB * 0x20 + 0x18	Mailbox Data High Register	CAN_MDH	Read-write	0x0
0x0200 + MB * 0x20 + 0x1C	Mailbox Control Register	CAN_MCR	Write-only	-

Note: 1. Mailbox number ranges from 0 to 7.

41.9.1 CAN Mode Register

Name: CAN_MR

Address: 0xF8000000 (0), 0xF8004000 (1)

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–			
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
DRPT	TIMFRZ	TTM	TEOF	OVL	ABM	LPM	CANEN

This register can only be written if the WPEN bit is cleared in "CAN Write Protection Mode Register".

- **CANEN: CAN Controller Enable**

0: The CAN Controller is disabled.

1: The CAN Controller is enabled.

- **LPM: Disable/Enable Low Power Mode**

0: Disable Low Power Mode.

1: Enable Low Power Mode

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 920.

- **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.

41.9.2 CAN Interrupt Enable Register

Name: CAN_IER

Address: 0xF8000004 (0), 0xF8004004 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	BERR	FERR	AERR	SERR	CERR
23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
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.

41.9.3 CAN Interrupt Disable Register

Name: CAN_IDR

Address: 0xF8000008 (0), 0xF8004008 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	BERR	FERR	AERR	SERR	CERR
23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
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.

41.9.4 CAN Interrupt Mask Register

Name: CAN_IMR

Address: 0xF800000C (0), 0xF800400C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	BERR	FERR	AERR	SERR	CERR
23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
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.

41.9.5 CAN Status Register

Name: CAN_SR

Address: 0xF8000010 (0), 0xF8004010 (1)

Access: Read-only

31	30	29	28	27	26	25	24
OVLSY	TBSY	RBSY	BERR	FERR	AERR	SERR	CERR
23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
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 has not reached Error Active Mode since the last read of CAN_SR.

1: CAN controller has reached Error Active Mode since the last read of CAN_SR.

This flag is automatically cleared by reading CAN_SR register.

This flag is set depending on TEC and REC counter values. It is set when a node is neither in Error Passive Mode nor in Bus Off Mode.

- **WARN: Warning Limit**

0: CAN controller Warning Limit has not been reached since the last read of CAN_SR.

1: CAN controller Warning Limit has been reached since the last read of CAN_SR.

This flag is automatically cleared by reading CAN_SR register.

This flag is set depending on TEC and REC counter values. It is set when at least one of the counter values has reached a value greater or equal to 96.

- **ERRP: Error Passive Mode**

0: CAN controller has not reached Error Passive Mode since the last read of CAN_SR.

1: CAN controller has reached Error Passive Mode since the last read of CAN_SR.

This flag is set depending on TEC and REC counters values.

This flag is automatically cleared by reading CAN_SR register.

A node is in error passive state when TEC counter is greater or equal to 128 (decimal) or when the REC counter is greater or equal to 128 (decimal).

- **BOFF: Bus Off Mode**

0: CAN controller has not reached Bus Off Mode.

1: CAN controller has reached Bus Off Mode since the last read of CAN_SR.

This flag is automatically cleared by reading CAN_SR register.

This flag is set depending on TEC counter value. A node is in bus off state when TEC counter is greater or equal to 256 (decimal).

- **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.

41.9.6 CAN Baudrate Register

Name: CAN_BR

Address: 0xF8000014 (0), 0xF8004014 (1)

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	SMP
23	22	21	20	19	18	17	16
–	BRP						
15	14	13	12	11	10	9	8
–	–	SJW		–	PROPAG		
7	6	5	4	3	2	1	0
–	PHASE1			–	PHASE2		

This register can only be written if the WPEN bit is cleared in "CAN Write Protection Mode Register".

Any modification on one of the fields of the CAN_BR register must be done while CAN module is disabled.

To compute the different Bit Timings, please refer to the Section 41.7.4.1 "CAN Bit Timing Configuration" on page 903.

- **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 (ONCE): The incoming bit stream is sampled once at sample point.

1 (THREE): 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.

41.9.7 CAN Timer Register

Name: CAN_TIM

Address: 0xF8000018 (0), 0xF8004018 (1)

Access: Read-only

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

- **TIMER: Timer**

This field represents the internal CAN controller 16-bit timer value.

41.9.8 CAN Timestamp Register

Name: CAN_TIMESTP

Address: 0xF800001C (0), 0xF800401C (1)

Access: Read-only

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

- **MTIMESTAMP: Timestamp**

This field carries the value of the internal CAN controller 16-bit timer value at the start or end of frame.

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.

41.9.9 CAN Error Counter Register

Name: CAN_ECR

Address: 0xF8000020 (0), 0xF8004020 (1)

Access: Read-only

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

41.9.10 CAN Transfer Command Register

Name: CAN_TCR

Address: 0xF8000024 (0), 0xF8004024 (1)

Access: Write-only

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

41.9.11 CAN Abort Command Register

Name: CAN_ACR

Address: 0xF8000028 (0), 0xF8004028 (1)

Access: Write-only

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

41.9.12 CAN Write Protection Mode Register

Name: CAN_WPMR

Address: 0xF80000E4 (0), 0xF80040E4 (1)

Access: Read-write

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

- **WPEN: Write Protection Enable**

0: The Write Protection is Disabled

1: The Write Protection is Enabled

- **WPKEY: SPI Write Protection Key Password**

If a value is written in WPEN, the value is taken into account only if WPKEY is written with “CAN” (CAN written in ASCII Code, ie 0x43414E in hexadecimal).

Protects the registers:

Section 41.9.1 “CAN Mode Register” on page 924

Section 41.9.6 “CAN Baudrate Register” on page 934

Section 41.9.14 “CAN Message Mode Register” on page 942

Section 41.9.15 “CAN Message Acceptance Mask Register” on page 943

Section 41.9.16 “CAN Message ID Register” on page 944

41.9.13 CAN Write Protection Status Register

Name: CAN_WPSR

Address: 0xF80000E8 (0), 0xF80040E8 (1)

Access: Read-only

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

- **WPVS: Write Protection Violation Status**

0: No Write Protect Violation has occurred since the last read of the CAN_WPSR register.

1: A Write Protect Violation has occurred since the last read of the CAN_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

This Field indicates the offset of the register concerned by the violation

41.9.14 CAN Message Mode Register

Name: CAN_MMRx [x=0..7]

Address: 0xF8000200 (0)[0], 0xF8000220 (0)[1], 0xF8000240 (0)[2], 0xF8000260 (0)[3], 0xF8000280 (0)[4], 0xF80002A0 (0)[5], 0xF80002C0 (0)[6], 0xF80002E0 (0)[7], 0xF8004200 (1)[0], 0xF8004220 (1)[1], 0xF8004240 (1)[2], 0xF8004260 (1)[3], 0xF8004280 (1)[4], 0xF80042A0 (1)[5], 0xF80042C0 (1)[6], 0xF80042E0 (1)[7]

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in "CAN Write Protection Mode Register".

- **MTIMEMARK: 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 920.

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:

Value	Name	Description
0	MB_DISABLED	Mailbox is disabled. This prevents receiving or transmitting any messages with this mailbox.
1	MB_RX	Reception Mailbox. Mailbox is configured for reception. If a message is received while the mailbox data register is full, it is discarded.
2	MB_RX_OVERWRITE	Reception mailbox with overwrite. Mailbox is configured for reception. If a message is received while the mailbox is full, it overwrites the previous message.
3	MB_TX	Transmit mailbox. Mailbox is configured for transmission.
4	MB_CONSUMER	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.
5	MB_PRODUCER	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.
6	–	Reserved

41.9.15 CAN Message Acceptance Mask Register

Name: CAN_MAMx [x=0..7]

Address: 0xF8000204 (0)[0], 0xF8000224 (0)[1], 0xF8000244 (0)[2], 0xF8000264 (0)[3], 0xF8000284 (0)[4], 0xF80002A4 (0)[5], 0xF80002C4 (0)[6], 0xF80002E4 (0)[7], 0xF8004204 (1)[0], 0xF8004224 (1)[1], 0xF8004244 (1)[2], 0xF8004264 (1)[3], 0xF8004284 (1)[4], 0xF80042A4 (1)[5], 0xF80042C4 (1)[6], 0xF80042E4 (1)[7]

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in "CAN Write Protection Mode Register".

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.

41.9.16 CAN Message ID Register

Name: CAN_MIDx [x=0..7]

Address: 0xF8000208 (0)[0], 0xF8000228 (0)[1], 0xF8000248 (0)[2], 0xF8000268 (0)[3], 0xF8000288 (0)[4], 0xF80002A8 (0)[5], 0xF80002C8 (0)[6], 0xF80002E8 (0)[7], 0xF8004208 (1)[0], 0xF8004228 (1)[1], 0xF8004248 (1)[2], 0xF8004268 (1)[3], 0xF8004288 (1)[4], 0xF80042A8 (1)[5], 0xF80042C8 (1)[6], 0xF80042E8 (1)[7]

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in "CAN Write Protection Mode Register".

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**

41.9.17 CAN Message Family ID Register

Name: CAN_MFIDx [x=0..7]

Address: 0xF800020C (0)[0], 0xF800022C (0)[1], 0xF800024C (0)[2], 0xF800026C (0)[3], 0xF800028C (0)[4], 0xF80002AC (0)[5], 0xF80002CC (0)[6], 0xF80002EC (0)[7], 0xF800420C (1)[0], 0xF800422C (1)[1], 0xF800424C (1)[2], 0xF800426C (1)[3], 0xF800428C (1)[4], 0xF80042AC (1)[5], 0xF80042CC (1)[6], 0xF80042EC (1)[7]

Access: Read-only

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

- **MFID: Family ID**

This field contains the concatenation of CAN_MIDx register bits masked by the CAN_MAMx register. This field is useful to speed up message ID decoding. The message acceptance procedure is described below.

As an example:

```
CAN_MIDx = 0x305A4321
CAN_MAMx = 0x3FF0F0FF
CAN_MFIDx = 0x000000A3
```

41.9.18 CAN Message Status Register

Name: CAN_MSRx [x=0..7]

Address: 0xF8000210 (0)[0], 0xF8000230 (0)[1], 0xF8000250 (0)[2], 0xF8000270 (0)[3], 0xF8000290 (0)[4], 0xF80002B0 (0)[5], 0xF80002D0 (0)[6], 0xF80002F0 (0)[7], 0xF8004210 (1)[0], 0xF8004230 (1)[1], 0xF8004250 (1)[2], 0xF8004270 (1)[3], 0xF8004290 (1)[4], 0xF80042B0 (1)[5], 0xF80042D0 (1)[6], 0xF80042F0 (1)[7]

Access: Read-only

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

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.

- **MTIMESTAMP: 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.

41.9.19 CAN Message Data Low Register

Name: CAN_MDLx [x=0..7]

Address: 0xF8000214 (0)[0], 0xF8000234 (0)[1], 0xF8000254 (0)[2], 0xF8000274 (0)[3], 0xF8000294 (0)[4], 0xF80002B4 (0)[5], 0xF80002D4 (0)[6], 0xF80002F4 (0)[7], 0xF8004214 (1)[0], 0xF8004234 (1)[1], 0xF8004254 (1)[2], 0xF8004274 (1)[3], 0xF8004294 (1)[4], 0xF80042B4 (1)[5], 0xF80042D4 (1)[6], 0xF80042F4 (1)[7]

Access: Read-write

31	30	29	28	27	26	25	24
MDL							

23	22	21	20	19	18	17	16
MDL							

15	14	13	12	11	10	9	8
MDL							

7	6	5	4	3	2	1	0
MDL							

- **MDL: Message Data Low Value**

When MRDY field is set in the CAN_MSRx register, the lower 32 bits of a received message can be read or written by the software application. Otherwise, the MDL value is locked by the CAN controller to send/receive a new message.

In Receive with overwrite, the CAN controller may modify MDL value while the software application reads MDH and MDL registers. To check that MDH and MDL do not belong to different messages, the application has to check the MMI field in the CAN_MSRx register. In this mode, the software application must re-read CAN_MDH and CAN_MDL, while the MMI bit in the CAN_MSRx register is set.

Bytes are received/sent on the bus in the following order:

1. CAN_MDL[7:0]
2. CAN_MDL[15:8]
3. CAN_MDL[23:16]
4. CAN_MDL[31:24]
5. CAN_MDH[7:0]
6. CAN_MDH[15:8]
7. CAN_MDH[23:16]
8. CAN_MDH[31:24]

41.9.20 CAN Message Data High Register

Name: CAN_MDHx [x=0..7]

Address: 0xF8000218 (0)[0], 0xF8000238 (0)[1], 0xF8000258 (0)[2], 0xF8000278 (0)[3], 0xF8000298 (0)[4], 0xF80002B8 (0)[5], 0xF80002D8 (0)[6], 0xF80002F8 (0)[7], 0xF8004218 (1)[0], 0xF8004238 (1)[1], 0xF8004258 (1)[2], 0xF8004278 (1)[3], 0xF8004298 (1)[4], 0xF80042B8 (1)[5], 0xF80042D8 (1)[6], 0xF80042F8 (1)[7]

Access: Read-write

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

- **MDH: Message Data High Value**

When MRDY field is set in the CAN_MSRx register, the upper 32 bits of a received message are read or written by the software application. Otherwise, the MDH value is locked by the CAN controller to send/receive a new message.

In Receive with overwrite, the CAN controller may modify MDH value while the software application reads MDH and MDL registers. To check that MDH and MDL do not belong to different messages, the application has to check the MMI field in the CAN_MSRx register. In this mode, the software application must re-read CAN_MDH and CAN_MDL, while the MMI bit in the CAN_MSRx register is set.

Bytes are received/sent on the bus in the following order:

1. CAN_MDL[7:0]
2. CAN_MDL[15:8]
3. CAN_MDL[23:16]
4. CAN_MDL[31:24]
5. CAN_MDH[7:0]
6. CAN_MDH[15:8]
7. CAN_MDH[23:16]
8. CAN_MDH[31:24]

41.9.21 CAN Message Control Register

Name: CAN_MCRx [x=0..7]

Address: 0xF800021C (0)[0], 0xF800023C (0)[1], 0xF800025C (0)[2], 0xF800027C (0)[3], 0xF800029C (0)[4], 0xF80002BC (0)[5], 0xF80002DC (0)[6], 0xF80002FC (0)[7], 0xF800421C (1)[0], 0xF800423C (1)[1], 0xF800425C (1)[2], 0xF800427C (1)[3], 0xF800429C (1)[4], 0xF80042BC (1)[5], 0xF80042DC (1)[6], 0xF80042FC (1)[7]

Access: Write-only

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

23	22	21	20	19	18	17	16
MTCR	MACR	–	MRTR	MDLC			

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **MDLC: Mailbox Data Length Code**

Mailbox Object Type	Description
Receive	No action.
Receive with overwrite	No action.
Transmit	Length of the mailbox message.
Consumer	No action.
Producer	Length of the mailbox message to be sent after the remote frame reception.

- **MRTR: Mailbox Remote Transmission Request**

Mailbox Object Type	Description
Receive	No action
Receive with overwrite	No action
Transmit	Set the RTR bit in the sent frame
Consumer	No action, the RTR bit in the sent frame is set automatically
Producer	No action

Consumer situations can be handled automatically by setting the mailbox object type in Consumer. This requires only one mailbox.

It can also be handled using two mailboxes, one in reception, the other in transmission. The MRTR and the MTCR bits must be set in the same time.

- **MACR: Abort Request for Mailbox x**

Mailbox Object Type	Description
Receive	No action
Receive with overwrite	No action
Transmit	Cancels transfer request if the message has not been transmitted to the CAN transceiver.
Consumer	Cancels the current transfer before the remote frame has been sent.
Producer	Cancels the current transfer. The next remote frame will not be serviced.

It is possible to set MACR field for several mailboxes in the same time, setting several bits to the CAN_ACR register.

- **MTCR: Mailbox Transfer Command**

Mailbox Object Type	Description
Receive	Allows the reception of the next message.
Receive with overwrite	Triggers a new reception.
Transmit	Sends data prepared in the mailbox as soon as possible.
Consumer	Sends a remote transmission frame.
Producer	Sends data prepared in the mailbox after receiving a remote frame from a Consumer.

This flag clears the MRDY and MABT flags in the CAN_MSRx register.

When several mailboxes are requested to be transmitted simultaneously, they are transmitted in turn. The mailbox with the highest priority is serviced first. If several mailboxes have the same priority, the mailbox with the lowest number is serviced first (i.e., MBx0 will be serviced before MBx 15 if they have the same priority).

It is possible to set MTCR for several mailboxes at the same time by writing to the CAN_TCR register.

42. Analog-to-Digital Converter (ADC)

42.1 Description

The ADC is based on a 10-bit Analog-to-Digital Converter (ADC) managed by an ADC Controller. Refer to the Block Diagram: Figure 42-1. It also integrates a 12-to-1 analog multiplexer, making possible the analog-to-digital conversions of 12 analog lines. The conversions extend from 0V to ADVREF. The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register.

Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The comparison circuitry allows automatic detection of values below a threshold, higher than a threshold, in a given range or outside the range, thresholds and ranges being fully configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a DMA channel. These features reduce both power consumption and processor intervention.

A whole set of reference voltages is generated internally from a single external reference voltage node that may be equal to the analog supply voltage. An external decoupling capacitance is required for noise filtering.

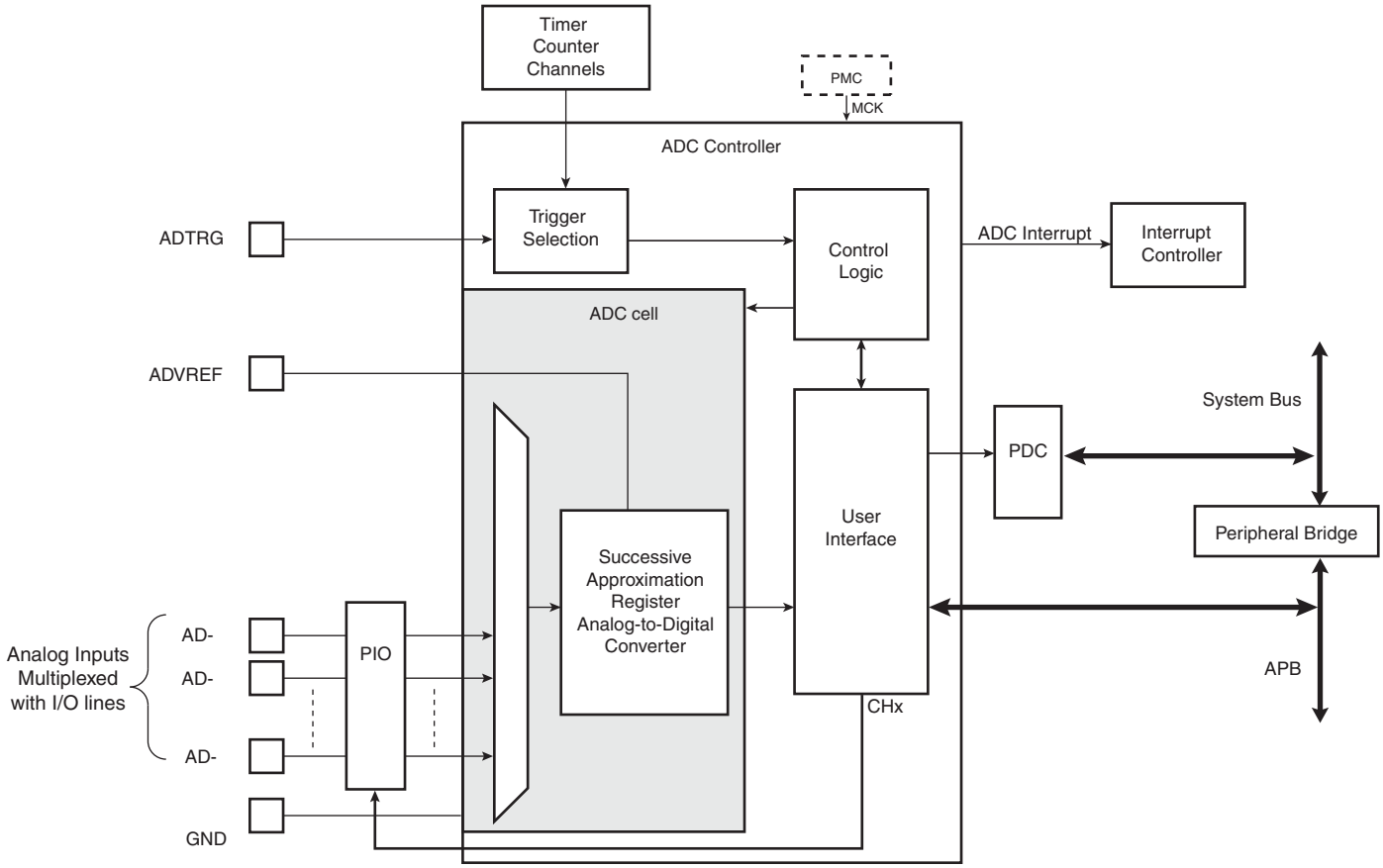
Finally, the user can configure ADC timings, such as Startup Time and Tracking Time.

42.2 Embedded Characteristics

- 10-bit Resolution
- 440 kHz Conversion Rate
- Wide Range Power Supply Operation
- Integrated Multiplexer Offering Up to 12 Independent Analog Inputs
- Individual Enable and Disable of Each Channel
- Hardware or Software Trigger
  - External Trigger Pin
  - Internal Trigger Counter
- DMA Support
- Possibility of ADC Timings Configuration
- Two Sleep Modes and Conversion Sequencer
  - Automatic Wakeup on Trigger and Back to Sleep Mode after Conversions of all Enabled Channels
  - Possibility of Customized Channel Sequence
- Standby Mode for Fast Wakeup Time Response
  - Power Down Capability
- Automatic Window Comparison of Converted Values
- Write Protect Registers

42.3 Block Diagram

Figure 42-1. Analog-to-Digital Converter Block Diagram



Note: DMA is sometimes referenced as PDC (Peripheral DMA Controller).

42.4 Signal Description

Table 42-1. ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
ADVREF	Reference voltage
AD0 - AD11	Analog input channels
ADTRG	External trigger

42.5 Product Dependencies

42.5.1 Power Management

The ADC Controller is not continuously clocked. The programmer must first enable the ADC Controller MCK in the Power Management Controller (PMC) before using the ADC Controller. However, if the application does not require ADC operations, the ADC Controller clock can be stopped when not needed and restarted when necessary. Configuring the ADC Controller does not require the ADC Controller clock to be enabled.

42.5.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the ADC interrupt requires the interrupt controller to be programmed first.

Table 42-2. Peripheral IDs

Instance	ID
ADC	19

42.5.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing the register ADC_CHER. By default, after reset, the PIO line is configured as input with its pull-up enabled and the ADC input is connected to the GND.

42.5.4 I/O Lines

The pin ADTRG may be shared with other peripheral functions through the PIO Controller. In this case, the PIO Controller should be set accordingly to assign the pin ADTRG to the ADC function.

Table 42-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
ADC	ADTRG	PB18	B
ADC	AD5	PB16	X1
ADC	AD6	PB17	X1
ADC	AD7	PB6	X1
ADC	AD8	PB7	X1
ADC	AD9	PB8	X1
ADC	AD10	PB9	X1
ADC	AD11	PB10	X1

42.5.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be unconnected.

42.5.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the product DC Characteristics section.

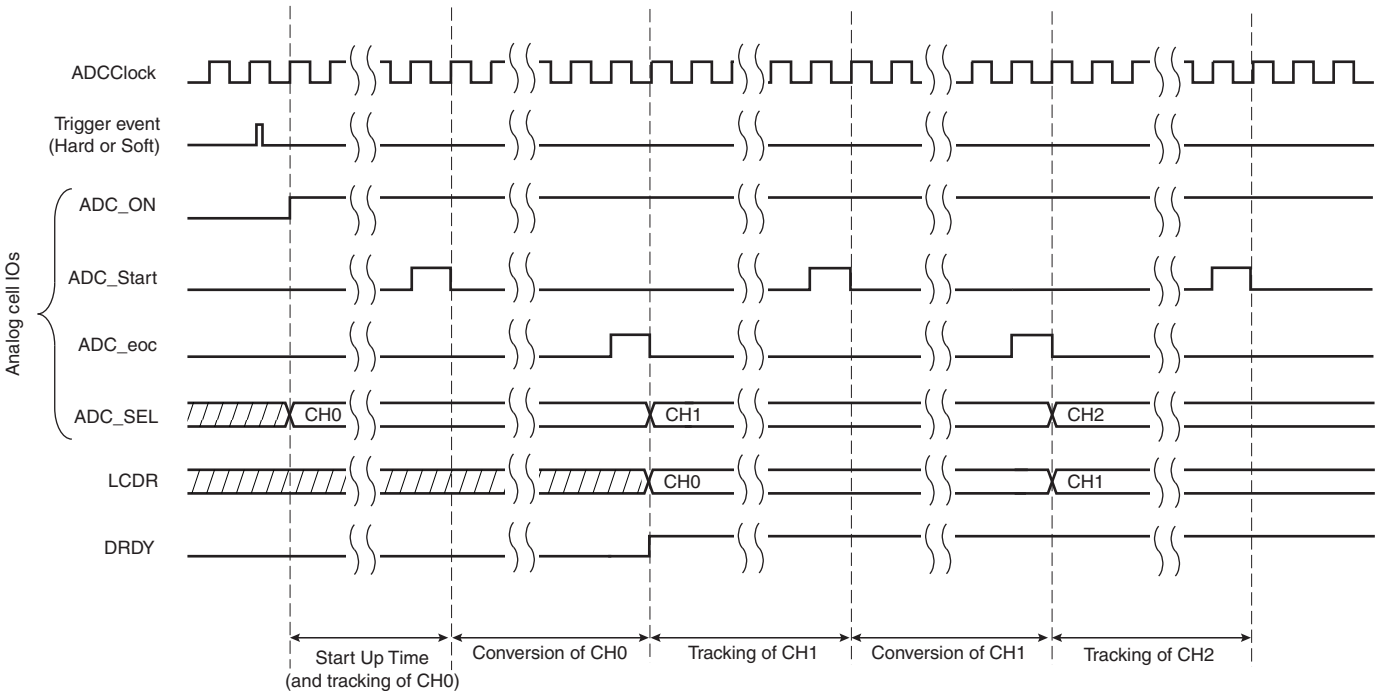
42.6 Functional Description

42.6.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires Tracking Clock cycles as defined in the field TRACKTIM of the “ADC Mode Register” on page 964 and Transfer Clock cycles as defined in the field TRANSFER of the same register. The ADC Clock frequency is selected in the PRESCAL field of the Mode Register (ADC_MR). The tracking phase starts during the conversion of the previous channel. If the tracking time is longer than the conversion time, the tracking phase is extended to the end of the previous conversion.

The ADC clock range is between MCK/2, if PRESCAL is 0, and MCK/512, if PRESCAL is set to 255 (0xFF). PRESCAL must be programmed in order to provide an ADC clock frequency according to the parameters given in the product Electrical Characteristics section.

Figure 42-2. Sequence of ADC conversions



42.6.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage pin ADVREF. Analog inputs between these voltages convert to values based on a linear conversion.

42.6.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the LOWRES bit in the ADC Mode Register (ADC_MR). By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the LOWRES bit, the ADC switches to the lowest resolution and the conversion results can be read in the lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding ADC_CDR register and of the LDATA field in the ADC_LCDR register read 0.

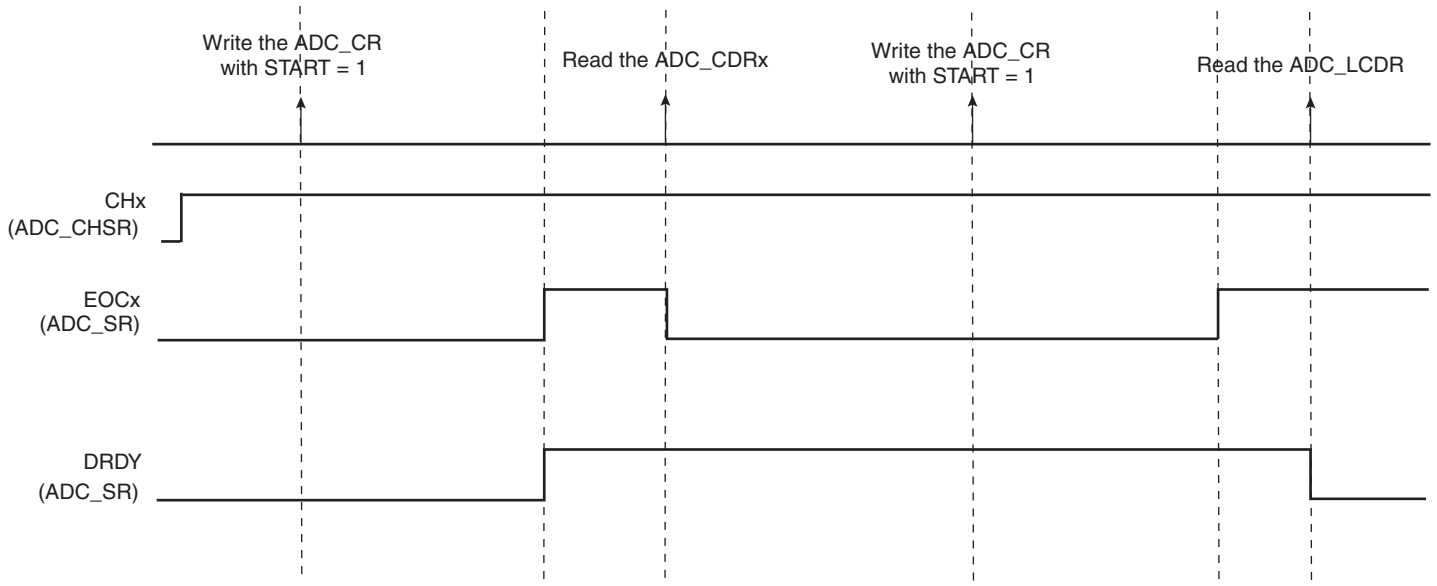
42.6.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the Channel Data Register (ADC_CDRx) of the current channel and in the ADC Last Converted Data Register (ADC_LCDR). By setting the TAG option in the ADC_EMR, the ADC_LCDR presents the channel number associated to the last converted data in the CHNB field.

The channel EOC bit in the Status Register (ADC_SR) is set and the DRDY is set. In the case of a connected DMA channel, DRDY rising triggers a data transfer request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDR registers clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit.

Figure 42-3. EOCx and DRDY Flag Behavior

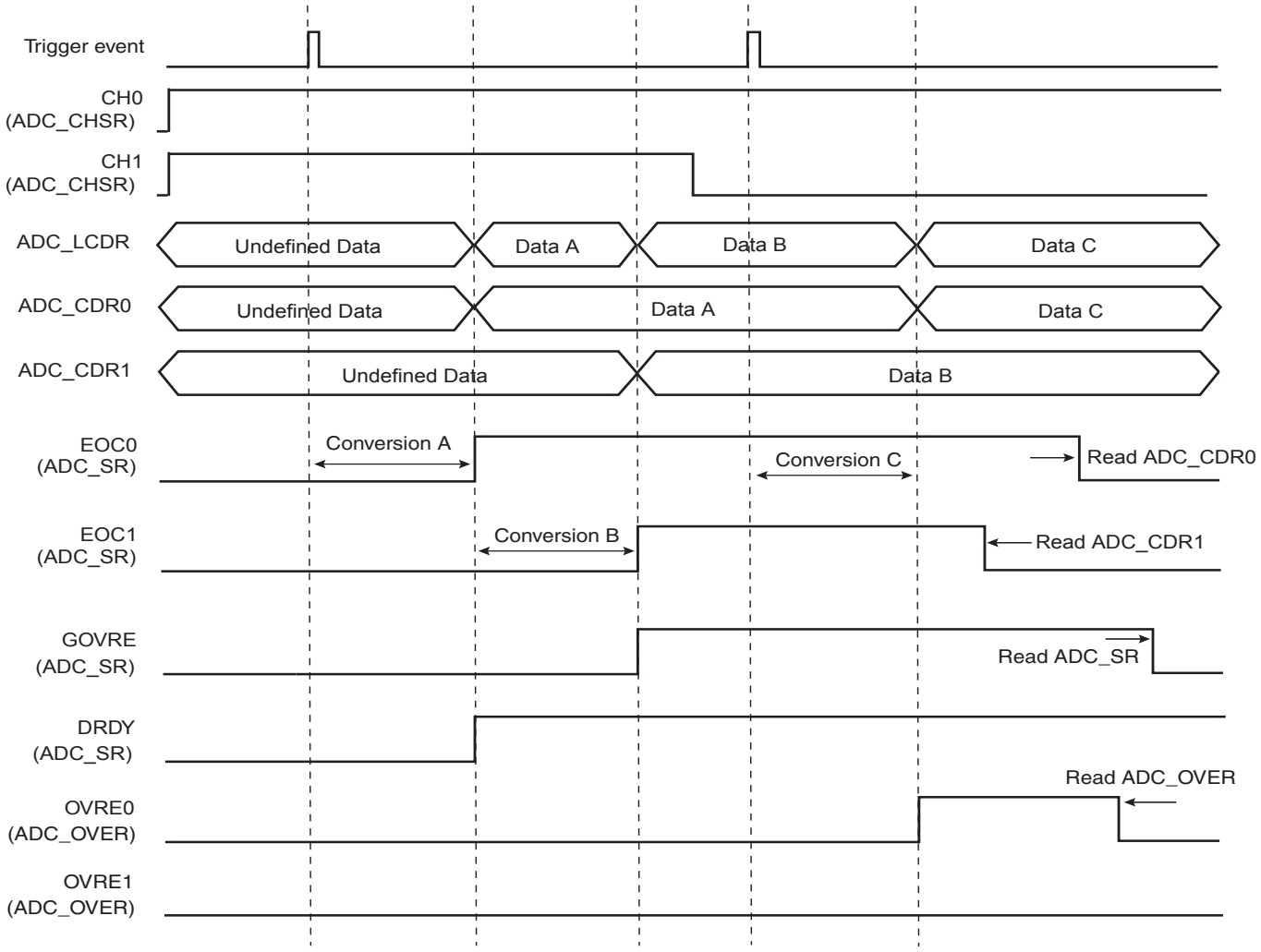


If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVREx) flag is set in the Overrun Status Register (ADC_OVER).

Likewise, new data converted when DRDY is high sets the GOVRE bit (General Overrun Error) in ADC_SR.

The OVREx flag is automatically cleared when ADC_OVER is read, and GOVRE flag is automatically cleared when ADC_SR is read.

Figure 42-4. GOVRE and OVREx Flag Behavior



Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

42.6.5 Conversion Triggers

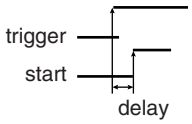
Conversions of the active analog channels are started with a software or hardware trigger. The software trigger is provided by writing the Control Register (ADC_CR) with the START bit at 1.

The hardware trigger can be selected by the TRGMOD field in the "ADC Trigger Register" between:

- Any edge, either rising or falling or both, detected on the external trigger pin, TSADTRG.
- A continuous trigger, meaning the ADC Controller restarts the next sequence as soon as it finishes the current one
- A periodic trigger, which is defined by programming the TRGPER field in the "ADC Trigger Register"

The minimum time between 2 consecutive trigger events must be strictly greater than the duration time of the longest conversion sequence according to configuration of registers ADC_MR, ADC_CHSR, ADC_SEQR1, ADC_SEQR2.

If a hardware trigger is selected, the start of a conversion is triggered after a delay starting at each rising edge of the selected signal. Due to asynchronous handling, the delay may vary in a range of 2 MCK clock periods to 1 ADC clock period.



Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers permit the analog channels to be enabled or disabled independently.

If the ADC is used with a DMA , only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

42.6.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by setting the SLEEP bit in the Mode Register ADC_MR.

The Sleep mode is automatically managed by a conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

This mode can be used when the minimum period of time between 2 successive trigger events is greater than the startup period of Analog-Digital converter (See the product ADC Characteristics section).

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically usingthe internal timer (ADC_TRGR register) . The periodic acquisition of several samples can be processed automatically without any intervention of the processor thanks to the DMA.

The sequence can be customized by programming the Sequence Channel Registers, ADC_SEQR1 and ADC_SEQR2 and setting to 1 the USEQ bit of the Mode Register (ADC_MR). The user can choose a specific order of channels and can program up to 12 conversions by sequence. The user is totally free to create a personal sequence, by writing channel numbers in ADC_SEQR1 and ADC_SEQR2. Not only can channel numbers be written in any sequence, channel numbers can be repeated several times. Only enabled sequence bitfields are converted, consequently to program a 15-conversion sequence, the user can simply put a disable in ADC_CHSR[15], thus disabling the 16THCH field of ADC_SEQR2.

If all ADC channels (i.e. 12) are used on an application board, there is no restriction of usage of the user sequence. But as soon as some ADC channels are not enabled for conversion but rather used as pure digital inputs, the respective indexes of these channels cannot be used in the user sequence fields (ADC_SEQR1, ADC_SEQR2 bitfields).

For example, if channel 4 is disabled (ADC_CSR[4] = 0), ADC_SEQR1, ADC_SEQR2 register bitfields USCH1 up to USCH12 must not contain the value 4. Thus the length of the user sequence may be limited by this behavior.

As an example, if only 4 channels over 12 (CH0 up to CH3) are selected for ADC conversions, the user sequence length cannot exceed 4 channels. Each trigger event may launch up to 4 successive conversions of any combination of channels 0 up to 3 but no more (i.e. in this case the sequence CH0, CH0, CH1, CH1, CH1 is impossible).

A sequence that repeats several times the same channel requires more enabled channels than channels actually used for conversion. For example, a sequence like CH0, CH0, CH1, CH1 requires 4 enabled channels (4 free channels on application boards) whereas only CH0, CH1 are really converted.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

42.6.7 Comparison Window

The ADC Controller features automatic comparison functions. It compares converted values to a low threshold or a high threshold or both, according to the CMPMODE function chosen in the Extended Mode Register (ADC_EMR). The comparison can be done on all channels or only on the channel specified in CMPSEL field of ADC_EMR. To compare all channels the CMP_ALL parameter of ADC_EMR should be set.

Moreover a filtering option can be set by writing the number of consecutive comparison errors needed to raise the flag. This number can be written and read in the CMPFILTER field of ADC_EMR.

The flag can be read on the COMPE bit of the Interrupt Status Register (ADC_ISR) and can trigger an interrupt.

The High Threshold and the Low Threshold can be read/write in the Comparison Window Register (ADC_CWR).

If the comparison window is to be used with LOWRES bit in ADC_MR set to 1, the thresholds do not need to be adjusted as adjustment will be done internally. Whether or not the LOWRES bit is set, thresholds must always be configured in consideration of the maximum ADC resolution.

42.6.8 ADC Timings

Each ADC has its own minimal Startup Time that is programmed through the field STARTUP in the Mode Register, ADC_MR.

A minimal Tracking Time is necessary for the ADC to guarantee the best converted final value between two channel selections. This time has to be programmed through the TRACKTIM bit field in the Mode Register, ADC_MR.

Warning: No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the TRACKTIM field. See the product ADC Characteristics section.

42.6.9 Buffer Structure

The DMA read channel is triggered each time a new data is stored in ADC_LCDR register. The same structure of data is repeatedly stored in ADC_LCDR register each time a trigger event occurs. Depending on user mode of operation (ADC_MR, ADC_CHSR, ADC_SEQR1, ADC_SEQR2) the structure differs. Each data transferred to DMA buffer, carried on a half-word (16-bit), consists of last converted data right aligned and when TAG is set in ADC_EMR register, the 4 most significant bits are carrying the channel number thus allowing an easier post-processing in the DMA buffer or better checking the DMA buffer integrity.

42.6.10 Write Protected Registers

To prevent any single software error that may corrupt ADC behavior, certain address spaces can be write-protected by setting the WPEN bit in the "ADC Write Protect Mode Register" (ADC_WPMR).

If a write access to the protected registers is detected, then the WPVS flag in the ADC Write Protect Status Register (ADC_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically reset by reading the ADC Write Protect Status Register (ADC_WPSR).

The protected registers are:

- "ADC Mode Register" on page 964
- "ADC Channel Sequence 1 Register" on page 966
- "ADC Channel Sequence 2 Register" on page 967
- "ADC Channel Enable Register" on page 968
- "ADC Channel Disable Register" on page 969
- "ADC Extended Mode Register" on page 977
- "ADC Compare Window Register" on page 978
- "ADC Trigger Register" on page 980

42.7 Analog-to-Digital Converter (ADC) User Interface

Any offset not listed in Table 42-4 must be considered as “reserved”.

Table 42-4. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	ADC_CR	Write-only	–
0x04	Mode Register	ADC_MR	Read-write	0x00000000
0x08	Channel Sequence Register 1	ADC_SEQR1	Read-write	0x00000000
0x0C	Channel Sequence Register 2	ADC_SEQR2	Read-write	0x00000000
0x10	Channel Enable Register	ADC_CHER	Write-only	–
0x14	Channel Disable Register	ADC_CHDR	Write-only	–
0x18	Channel Status Register	ADC_CHSR	Read-only	0x00000000
0x1C	Reserved	–	–	–
0x20	Last Converted Data Register	ADC_LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	ADC_IER	Write-only	–
0x28	Interrupt Disable Register	ADC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x30	Interrupt Status Register	ADC_ISR	Read-only	0x00000000
0x34	Reserved	–	–	–
0x38	Reserved	–	–	–
0x3C	Overrun Status Register	ADC_OVER	Read-only	0x00000000
0x40	Extended Mode Register	ADC_EMR	Read-write	0x00000000
0x44	Compare Window Register	ADC_CWR	Read-write	0x00000000
0x50	Channel Data Register 0	ADC_CDR0	Read-only	0x00000000
0x54	Channel Data Register 1	ADC_CDR1	Read-only	0x00000000
...	...	...	...	...
0x7C	Channel Data Register 11	ADC_CDR11	Read-only	0x00000000
0x80 - 0x90	Reserved	–	–	–
0x98 - 0xAC	Reserved	–	–	–
0xC0	Trigger Register	ADC_TRGR	Read-write	0x00000000
0xC4 - 0xE0	Reserved	–	–	–
0xE4	Write Protect Mode Register	ADC_WPMR	Read-write	0x00000000
0xE8	Write Protect Status Register	ADC_WPSR	Read-only	0x00000000
0xEC - 0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

42.7.1 ADC Control Register

Name: ADC_CR

Address: 0xF804C000

Access: Write-only

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

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

42.7.2 ADC Mode Register

Name: ADC_MR

Address: 0xF804C004

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **LOWRES: Resolution**

Value	Name	Description
0	BITS_10	10-bit resolution
1	BITS_8	8-bit resolution

- **SLEEP: Sleep Mode**

Value	Name	Description
0	NORMAL	Normal Mode: The ADC Core and reference voltage circuitry are kept ON between conversions
1	SLEEP	Sleep Mode: The ADC Core and reference voltage circuitry are OFF between conversions

- **FWUP: Fast Wake Up**

Value	Name	Description
0	OFF	Normal Sleep Mode: The sleep mode is defined by the SLEEP bit
1	ON	Fast Wake Up Sleep Mode: The Voltage reference is ON between conversions and ADC Core is OFF

- **PRESCAL: Prescaler Rate Selection**

  ADCClock = MCK / ((PRESCAL+1) * 2)

- **STARTUP: Start Up Time**

Value	Name	Description
0	SUT0	0 periods of ADCClock
1	SUT8	8 periods of ADCClock
2	SUT16	16 periods of ADCClock
3	SUT24	24 periods of ADCClock
4	SUT64	64 periods of ADCClock

Value	Name	Description
5	SUT80	80 periods of ADCClock
6	SUT96	96 periods of ADCClock
7	SUT112	112 periods of ADCClock
8	SUT512	512 periods of ADCClock
9	SUT576	576 periods of ADCClock
10	SUT640	640 periods of ADCClock
11	SUT704	704 periods of ADCClock
12	SUT768	768 periods of ADCClock
13	SUT832	832 periods of ADCClock
14	SUT896	896 periods of ADCClock
15	SUT960	960 periods of ADCClock

- **TRACKTIM: Tracking Time**
  Tracking Time = (TRACKTIM + 1) * ADCClock periods.

- **USEQ: Use Sequence Enable**

Value	Name	Description
0	NUM_ORDER	Normal Mode: The controller converts channels in a simple numeric order depending only on the channel index.
1	REG_ORDER	User Sequence Mode: The sequence respects what is defined in ADC_SEQR1 and ADC_SEQR2 registers and can be used to convert several times the same channel.

42.7.3 ADC Channel Sequence 1 Register

Name: ADC_SEQR1

Address: 0xF804C008

Access: Read-write

31	30	29	28	27	26	25	24
USCH8				USCH7			
23	22	21	20	19	18	17	16
USCH6				USCH5			
15	14	13	12	11	10	9	8
USCH4				USCH3			
7	6	5	4	3	2	1	0
USCH2				USCH1			

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **USCHx: User Sequence Number x**

The sequence number x (USCHx) can be programmed by the Channel number CHy where y is the value written in this field. The allowed range is 0 up to 11. So it is only possible to use the sequencer from CH0 to CH11.

This register activates only if ADC_MR(USEQ) field is set to ‘1’.

Any USCHx field is taken into account only if ADC_CHSR(CHx) register field reads logical ‘1’ else any value written in USCHx does not add the corresponding channel in the conversion sequence.

Configuring the same value in different fields leads to multiple samples of the same channel during the conversion sequence. This can be done consecutively, or not, according to user needs.

42.7.4 ADC Channel Sequence 2 Register

Name: ADC_SEQR2

Address: 0xF804C00C

Access: Read-write

31	30	29	28	27	26	25	24
USCH16				USCH15			
23	22	21	20	19	18	17	16
USCH14				USCH13			
15	14	13	12	11	10	9	8
USCH12				USCH11			
7	6	5	4	3	2	1	0
USCH10				USCH9			

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **USCHx: User Sequence Number x**

The sequence number x (USCHx) can be programmed by the Channel number CHy where y is the value written in this field. The allowed range is 0 up to 11. So it is only possible to use the sequencer from CH0 to CH11.

This register activates only if ADC_MR(USEQ) field is set to ‘1’.

Any USCHx field is taken into account only if ADC_CHSR(CHx) register field reads logical ‘1’ else any value written in USCHx does not add the corresponding channel in the conversion sequence.

Configuring the same value in different fields leads to multiple samples of the same channel during the conversion sequence. This can be done consecutively, or not, according to user needs.

42.7.5 ADC Channel Enable Register

Name: ADC_CHER

Address: 0xF804C010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

Note: If USEQ = 1 in ADC_MR register, CHx corresponds to the xth channel of the sequence described in ADC_SEQR1 and ADC_SEQR2.

42.7.6 ADC Channel Disable Register

Name: ADC_CHDR

Address: 0xF804C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

42.7.7 ADC Channel Status Register

Name: ADC_CHSR

Address: 0xF804C018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

42.7.8 ADC Last Converted Data Register

Name: ADC_LCDR

Address: 0xF804C020

Access: Read-only

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

- **LDATA: Last Data Converted**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

- **CHNB: Channel Number**

Indicates the last converted channel when the TAG option is set to 1 in the ADC_EMR register. If the TAG option is not set, CHNB = 0.

42.7.9 ADC Interrupt Enable Register

Name: ADC_IER

Address: 0xF804C024

Access: Write-only

31	30	29	28	27	26	25	24
–			–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–				–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Enable x**
- **DRDY: Data Ready Interrupt Enable**
- **GOVRE: General Overrun Error Interrupt Enable**
- **COMPE: Comparison Event Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

42.7.10 ADC Interrupt Disable Register

Name: ADC_IDR

Address: 0xF804C028

Access: Write-only

31	30	29	28	27	26	25	24
–			–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–				–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **COMPE: Comparison Event Interrupt Disable**

0 = No effect.

1 = Enables the corresponding interrupt.

42.7.11 ADC Interrupt Mask Register

Name: ADC_IMR

Address: 0xF804C02C

Access: Read-only

31	30	29	28	27	26	25	24
–			–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–				–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **COMPE: Comparison Event Interrupt Mask**

0 = No effect.

1 = Enables the corresponding interrupt.

42.7.12 ADC Interrupt Status Register

Name: ADC_ISR

Address: 0xF804C030

Access: Read-only

31	30	29	28	27	26	25	24
			–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–				–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished. This flag is cleared when reading the corresponding ADC_CDRx registers.

1 = Corresponding analog channel is enabled and conversion is complete.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_ISR.

1 = At least one General Overrun Error has occurred since the last read of ADC_ISR.

- **COMPE: Comparison Error**

0 = No Comparison Error since the last read of ADC_ISR.

1 = At least one Comparison Error (defined in the ADC_EMR and ADC_CWR registers) has occurred since the last read of ADC_ISR.

42.7.13 ADC Overrun Status Register

Name: ADC_OVER

Address: 0xF804C03C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	OVRE11	OVRE10	OVRE9	OVRE8
7	6	5	4	3	2	1	0
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0

- **OVREx: Overrun Error x**

0 = No overrun error on the corresponding channel since the last read of ADC_OVER.

1 = There has been an overrun error on the corresponding channel since the last read of ADC_OVER.

42.7.14 ADC Extended Mode Register

Name: ADC_EMR

Address: 0xF804C040

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **CMPMODE: Comparison Mode**

Value	Name	Description
0	LOW	Generates an event when the converted data is lower than the low threshold of the window.
1	HIGH	Generates an event when the converted data is higher than the high threshold of the window.
2	IN	Generates an event when the converted data is in the comparison window.
3	OUT	Generates an event when the converted data is out of the comparison window.

- **CMPSEL: Comparison Selected Channel**

If CMPALL = 0: CMPSEL indicates which channel has to be compared.

If CMPALL = 1: No effect.

- **CMPALL: Compare All Channels**

0 = Only channel indicated in CMPSEL field is compared.

1 = All channels are compared.

- **CMPFILTER: Compare Event Filtering**

Number of consecutive compare events necessary to raise the flag = CMPFILTER+1

When programmed to 0, the flag rises as soon as an event occurs.

- **TAG: TAG of the ADC_LDCR register**

0 = Sets CHNB to zero in ADC_LDCR.

1 = Appends the channel number to the conversion result in ADC_LDCR register.

42.7.15 ADC Compare Window Register

Name: ADC_CWR

Address: 0xF804C044

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 981.

- **LOWTHRES: Low Threshold**

Low threshold associated to compare settings of the ADC_EMR register.

If LOWRES is set in ADC_MR, only the 10 LSB of LOWTHRES must be programmed. The 2 LSB will be automatically discarded to match the value carried on ADC_CDR (8-bit).

- **HIGHTHRES: High Threshold**

High threshold associated to compare settings of the ADC_EMR register.

If LOWRES is set in ADC_MR, only the 10 LSB of HIGHTHRES must be programmed. The 2 LSB will be automatically discarded to match the value carried on ADC_CDR (8-bit).

42.7.16 ADC Channel Data Register

Name: ADC_CDRx [x=0..11]

Address: 0xF804C050

Access: Read-write

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

- **DATA: Converted Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. The Convert Data Register (CDR) is only loaded if the corresponding analog channel is enabled.

42.7.17 ADC Trigger Register

Name: ADC_TRGR

Address: 0xF804C0C0

Access: Read-write

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

- **TRGMOD: Trigger Mode**

Value	Name	Description
0	NO_TRIGGER	No trigger, only software trigger can start conversions
1	EXT_TRIG_RISE	External Trigger Rising Edge
2	EXT_TRIG_FALL	External Trigger Falling Edge
3	EXT_TRIG_ANY	External Trigger Any Edge
4	–	Reserved
5	PERIOD_TRIG	Periodic Trigger (TRGPER shall be initiated appropriately)
6	CONTINUOUS	Continuous Mode
7	–	Reserved

- **TRGPER: Trigger Period**

Effective only if TRGMOD defines a Periodic Trigger.

Defines the periodic trigger period, with the following equation:

Trigger Period = (TRGPER+1) /ADCCLK

The minimum time between 2 consecutive trigger events must be strictly greater than the duration time of the longest conversion sequence according to configuration of registers ADC_MR, ADC_CHSR, ADC_SEQR1, ADC_SEQR2 .

42.7.18 ADC Write Protect Mode Register

Name: ADC_WPMR

Address: 0xF804C0E4

Access: Read-write

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x414443 (“ADC” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x414443 (“ADC” in ASCII).

Protects the registers:

“ADC Mode Register” on page 964

“ADC Channel Sequence 1 Register” on page 966

“ADC Channel Sequence 2 Register” on page 967

“ADC Channel Enable Register” on page 968

“ADC Channel Disable Register” on page 969

“ADC Extended Mode Register” on page 977

“ADC Compare Window Register” on page 978

“ADC Trigger Register” on page 980

- **WPKEY: Write Protect KEY**

Should be written at value 0x414443 (“ADC” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

42.7.19 ADC Write Protect Status Register

Name: ADC_WPSR

Address: 0xF804C0E8

Access: Read-only

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the ADC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the ADC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Reading ADC_WPSR automatically clears all fields.

43. Software Modem Device (SMD)

43.1 Description

The Software Modem Device (SMD) is a block for communication via a modem's Digital Isolation Barrier (DIB) with a complementary Line Side Device (HLSD).

SMD and HLSD are two parts of the "Transformer only" solution. The transformer is the only component connecting SMD and HLSD. The transformer is used for power, clock and data transfers. Power and clock are supplied by the SMD and consumed by the HLSD. The data flow is bidirectional. The data transfer is based on pulse width modulation for transmission from the SMD to the HLSD, and for receiving from the HLSD.

There are two channels embedded into the protocol of the DIB link:

- Data channel
- Control channel

Each channel is bidirectional.

The data channel is used to transfer digitized signal samples at a constant rate of 16 bits at 16 kHz.

The control channel is used to communicate with control registers of the HLSD at a maximum rate of 8 bits at 16 kHz.

The SMD performs all protocol-related data conversion for transmission and received data interpretation in both data and control channels of the link.

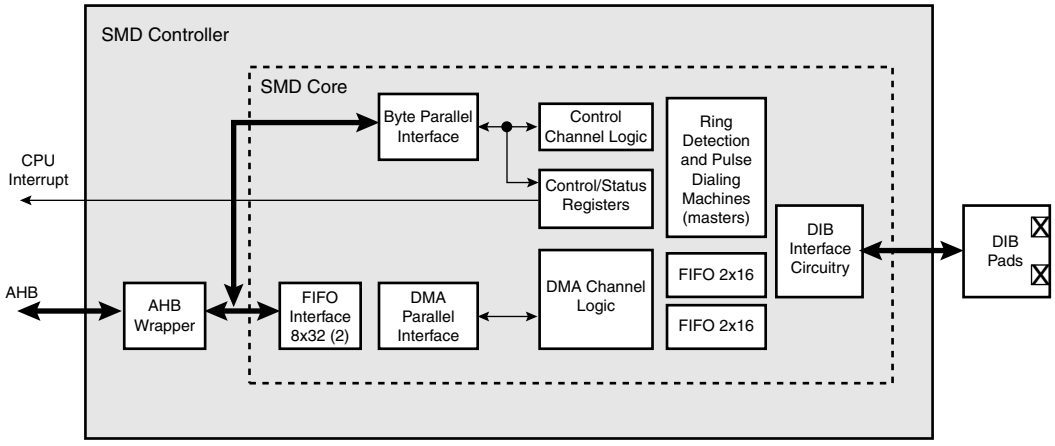
The SMD incorporates both RX and TX FIFOs, available through the DMAC interface. Each FIFO is able to hold eight 32-bit words (equivalent to 16 modem data samples).

43.2 Embedded Characteristics

- Modulations and protocols
  - V.90
  - V.34
  - V.32bis, V.32, V.22bis, V.22, V.23, V.21
  - V.23 reverse, V.23 half-duplex
  - Bell 212A/Bell 103
  - V.29 FastPOS
  - V.22bis fast connect
  - V.80 Synchronous Access Mode
- Data compression and error correction
  - V.44 data compression (V.92 model)
  - V.42bis and MNP 5 data compression
  - V.42 LAPM and MNP 2-4 error correction
  - EIA/TIA 578 Class 1 and T.31 Class 1.0
- Call Waiting (CW) detection and Type II Caller ID decoding during data mode
- Type I Caller ID (CID) decoding
- Sixty-three embedded and upgradable country profiles
- Embedded AT commands
- SmartDAA
  - Extension pick-up detection
  - Digital line protection
  - Line reversal detection
  - Line-in-use detection
  - Remote hang-up detection
  - Worldwide compliance

43.3 Block Diagram

Figure 43-1. Software Modem Device Block Diagram



44. Synchronous Serial Controller (SSC)

44.1 Description

The 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 high-level of programmability and its use of DMA permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to the DMA, the SSC permits interfacing with low processor overhead to the following:

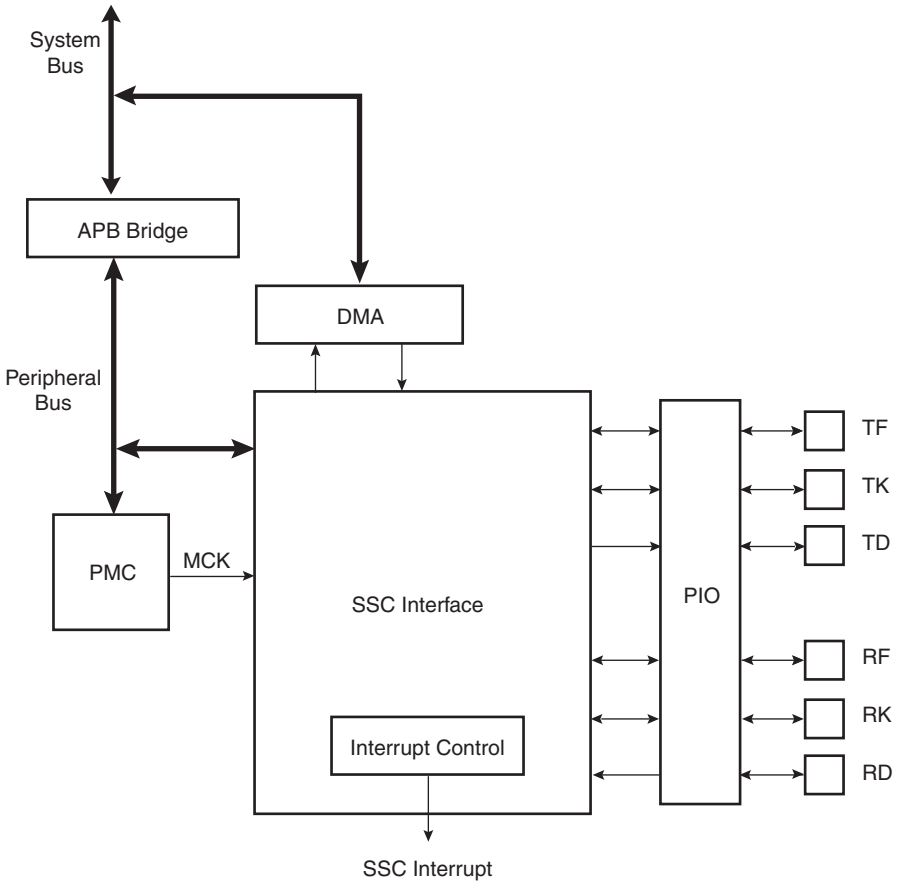
- CODEC's in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

44.2 Embedded Characteristics

- Provides Serial Synchronous Communication Links Used in Audio and Telecom Applications
- Contains an Independent Receiver and Transmitter and a Common Clock Divider
- Interfaced with the DMA Controller (DMAC) to Reduce Processor Overhead
- Offers a Configurable Frame Sync and Data Length
- Receiver and Transmitter can be Programmed to Start Automatically or on Detection of Different Events on the Frame Sync Signal
- Receiver and Transmitter Include a Data Signal, a Clock Signal and a Frame Synchronization Signal

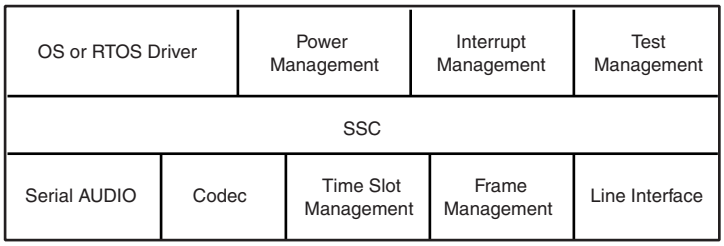
44.3 Block Diagram

Figure 44-1. Block Diagram



44.4 Application Block Diagram

Figure 44-2. Application Block Diagram



44.5 Pin Name List

Table 44-1. 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

44.6 Product Dependencies

44.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

Table 44-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SSC	RD	PA27	B
SSC	RF	PA29	B
SSC	RK	PA28	B
SSC	TD	PA26	B
SSC	TF	PA25	B
SSC	TK	PA24	B

44.6.2 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.

44.6.3 Interrupt

The SSC interface has an interrupt line connected to the interrupt controller. Handling interrupts requires programming the interrupt controller before configuring the SSC.

All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

Table 44-3. Peripheral IDs

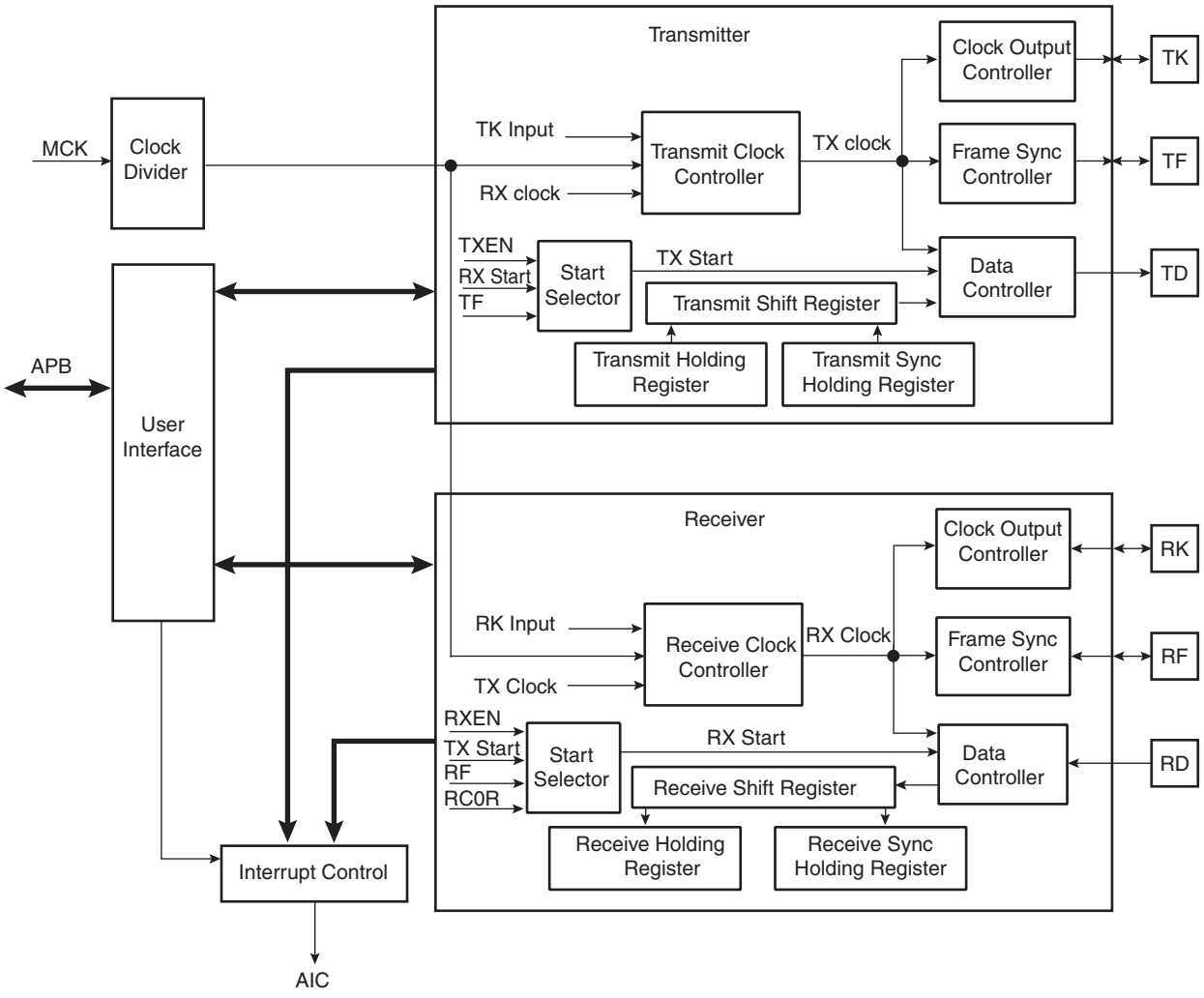
Instance	ID
SSC	28

44.7 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 44-3. SSC Functional Block Diagram



44.7.1 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:

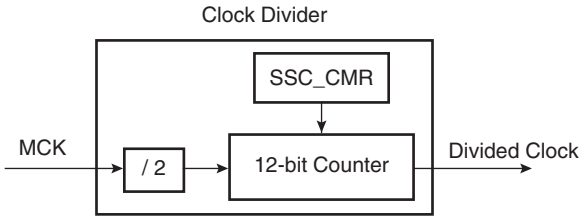
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

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.

44.7.1.1 Clock Divider

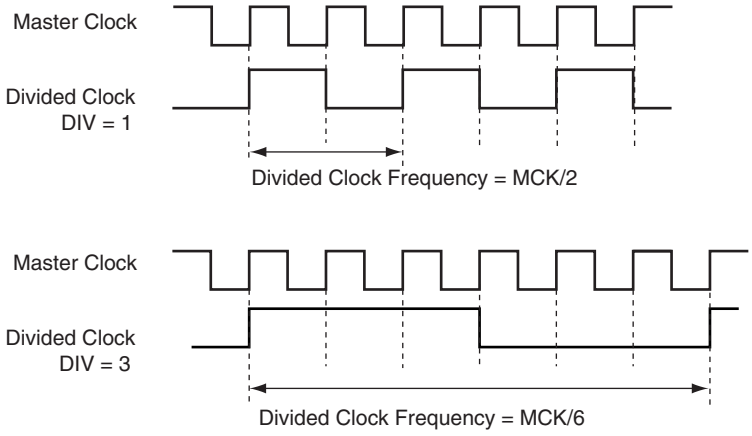
Figure 44-4. Divided Clock Block Diagram



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 44-5. Divided Clock Generation

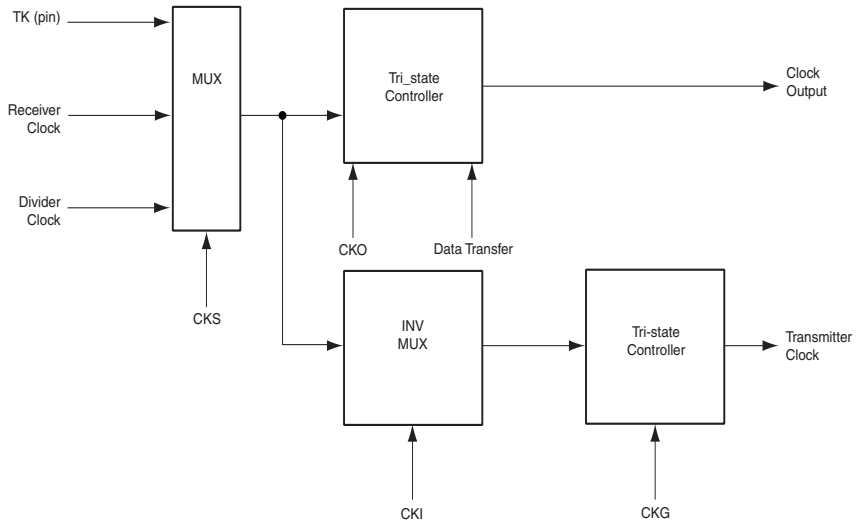


44.7.1.2 Transmitter Clock Management

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 44-6. Transmitter Clock Management

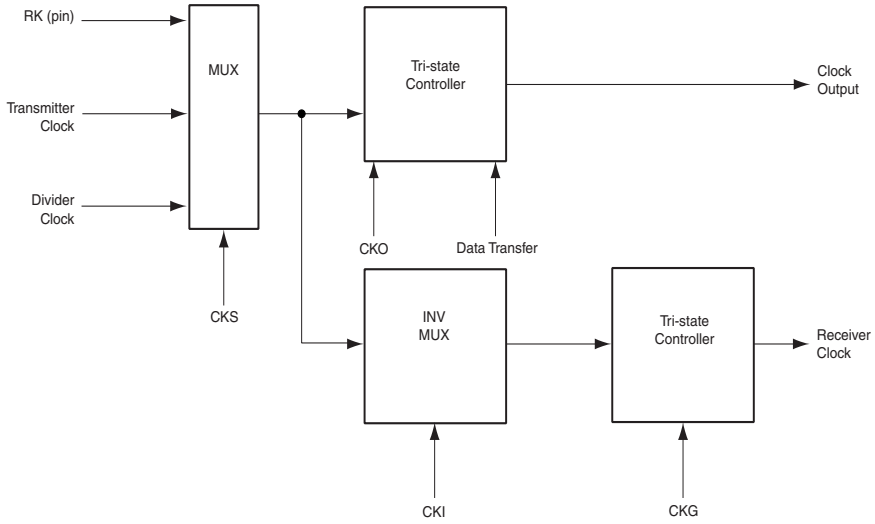


44.7.1.3 Receiver Clock Management

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 44-7. Receiver Clock Management



44.7.1.4 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

44.7.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

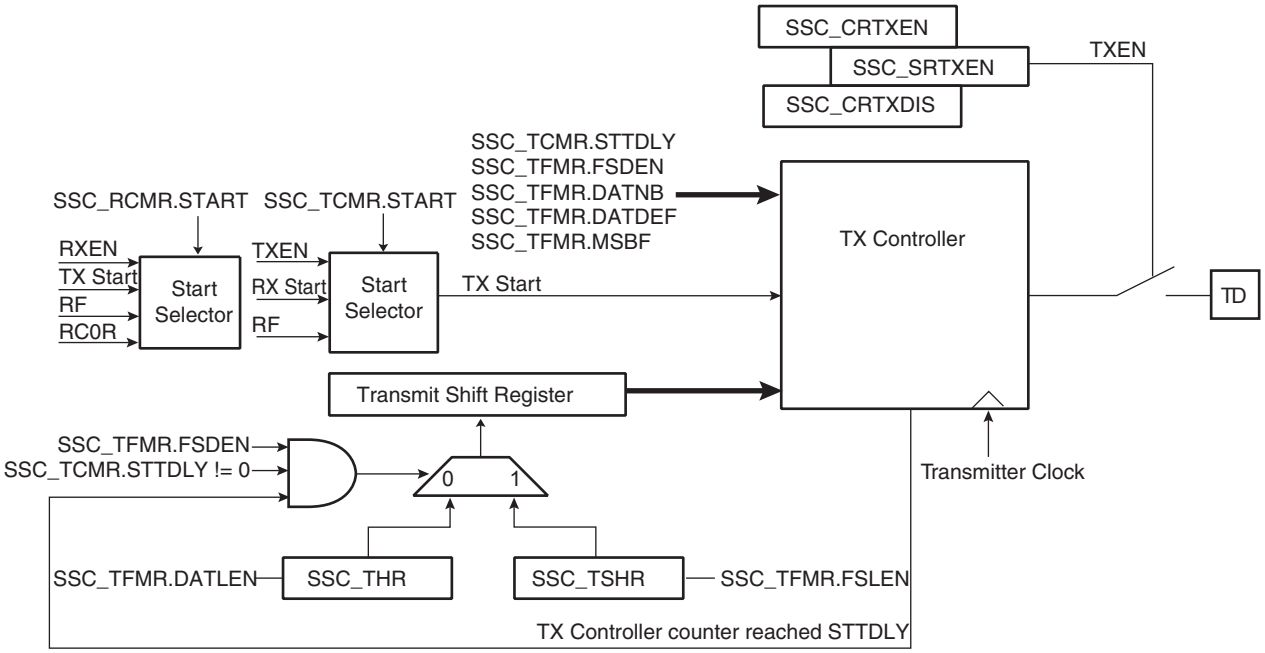
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See “Start” on page 993.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See “Frame Sync” on page 995.

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 44-8. Transmitter Block Diagram



44.7.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

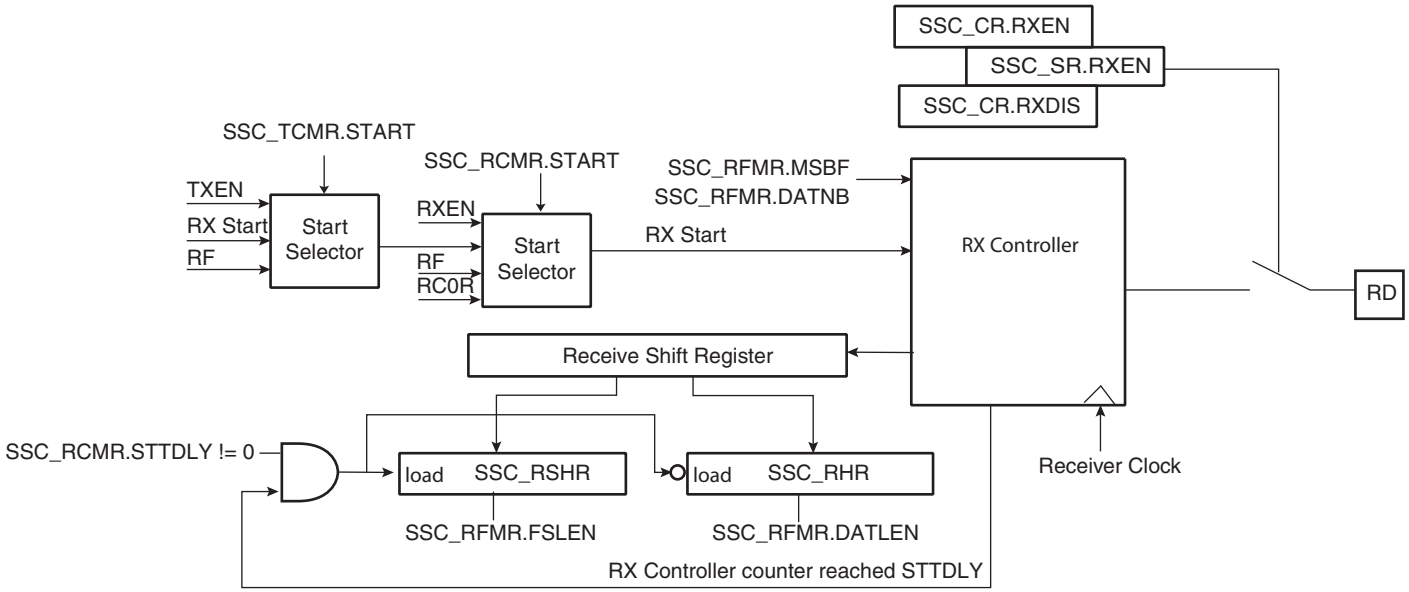
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See “Start” on page 993.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See “Frame Sync” on page 995.

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 44-9. Receiver Block Diagram



44.7.4 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 44-10.Transmit Start Mode

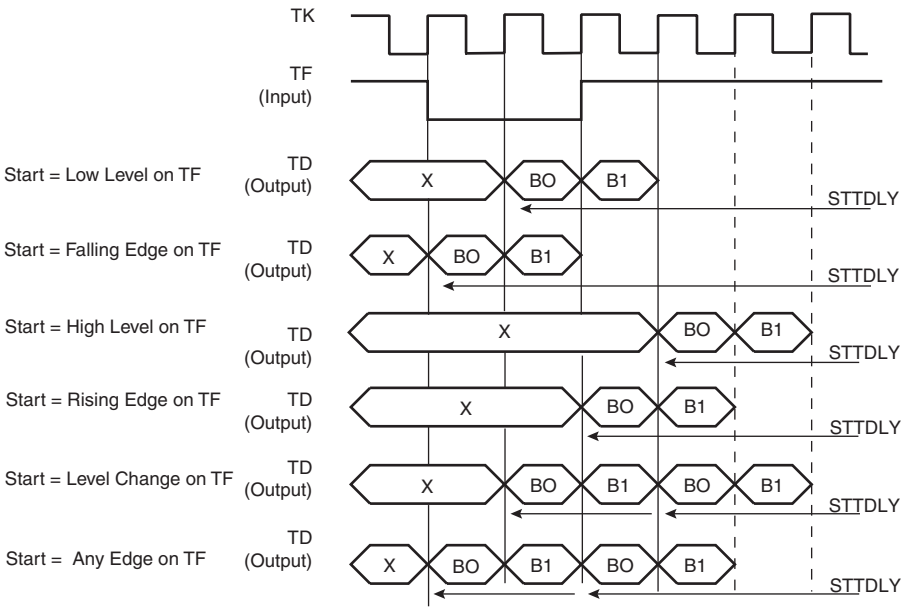
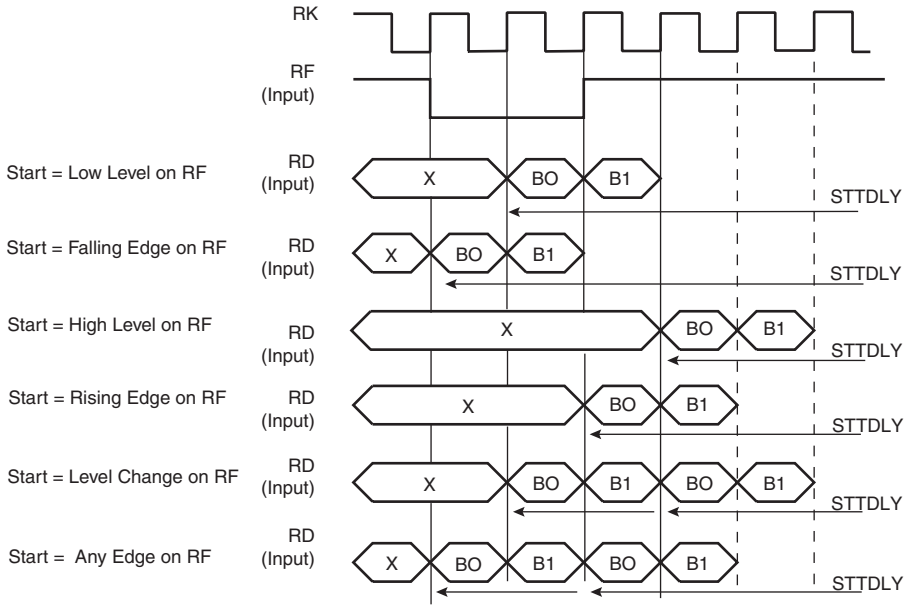


Figure 44-11.Receive Pulse/Edge Start Modes



44.7.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 256 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

44.7.5.1 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.

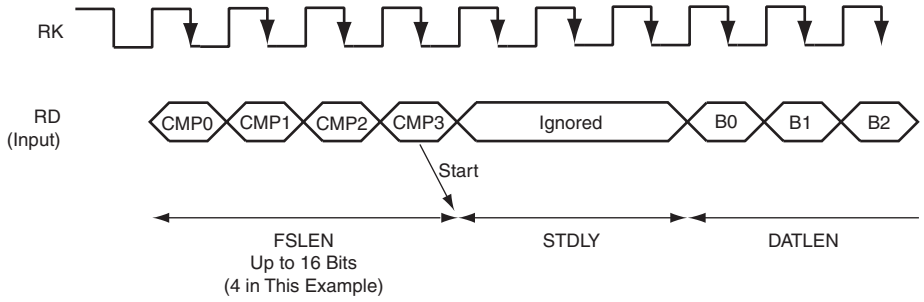
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

44.7.5.2 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).

44.7.6 Receive Compare Modes

Figure 44-12.Receive Compare Modes



44.7.6.1 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.

44.7.7 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:

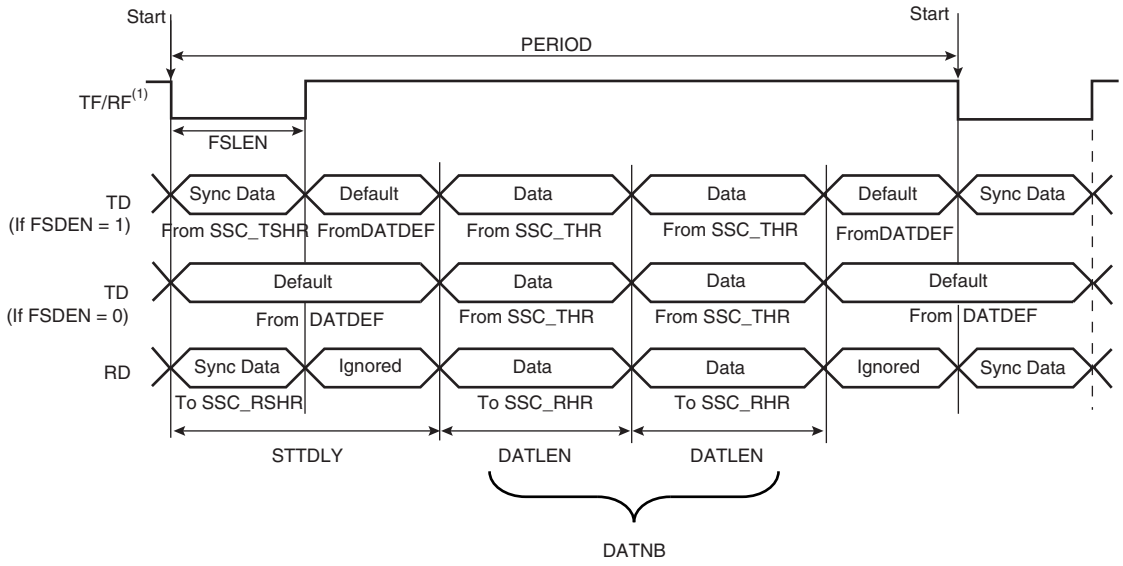
- The event that starts the data transfer (START)
- The delay in number of bit periods between the start event and the first data bit (STTDLY)
- The length of the data (DATLEN)
- The number of data to be transferred for each start event (DATNB).
- The length of synchronization transferred for each start 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 44-4. Data Frame Registers

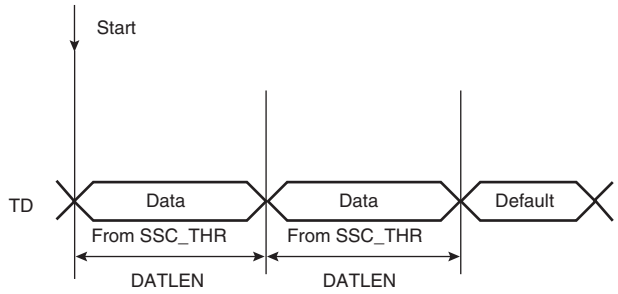
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data 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 44-13.Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

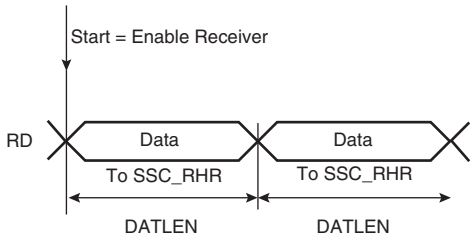
Figure 44-14.Transmit Frame Format in Continuous Mode



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 44-15.Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

44.7.8 Loop Mode

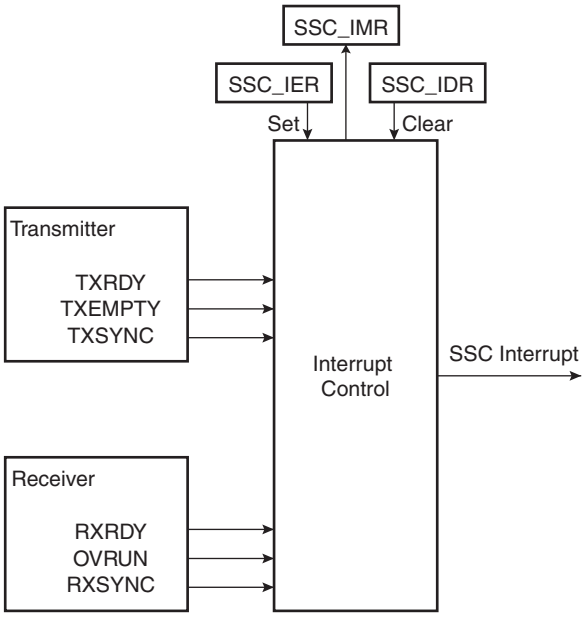
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

44.7.9 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 interrupt controller.

Figure 44-16.Interrupt Block Diagram



44.8 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 44-17.Audio Application Block Diagram

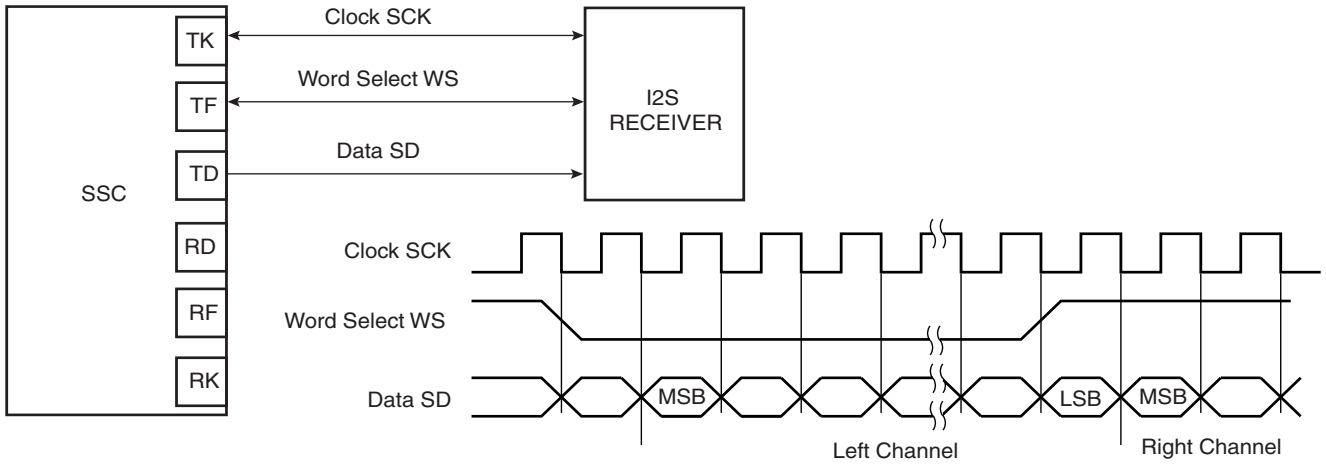


Figure 44-18.Codec Application Block Diagram

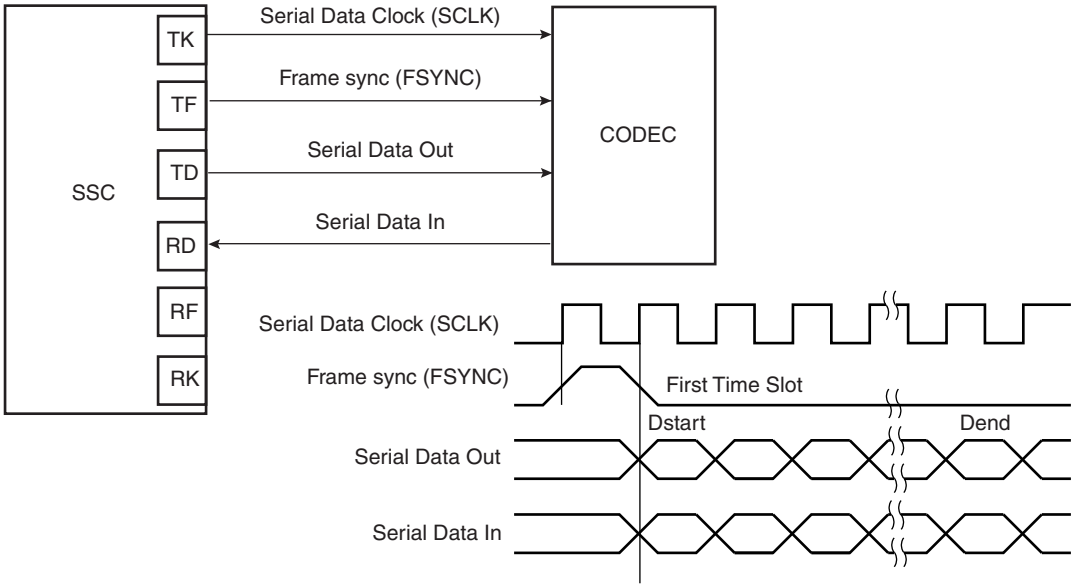
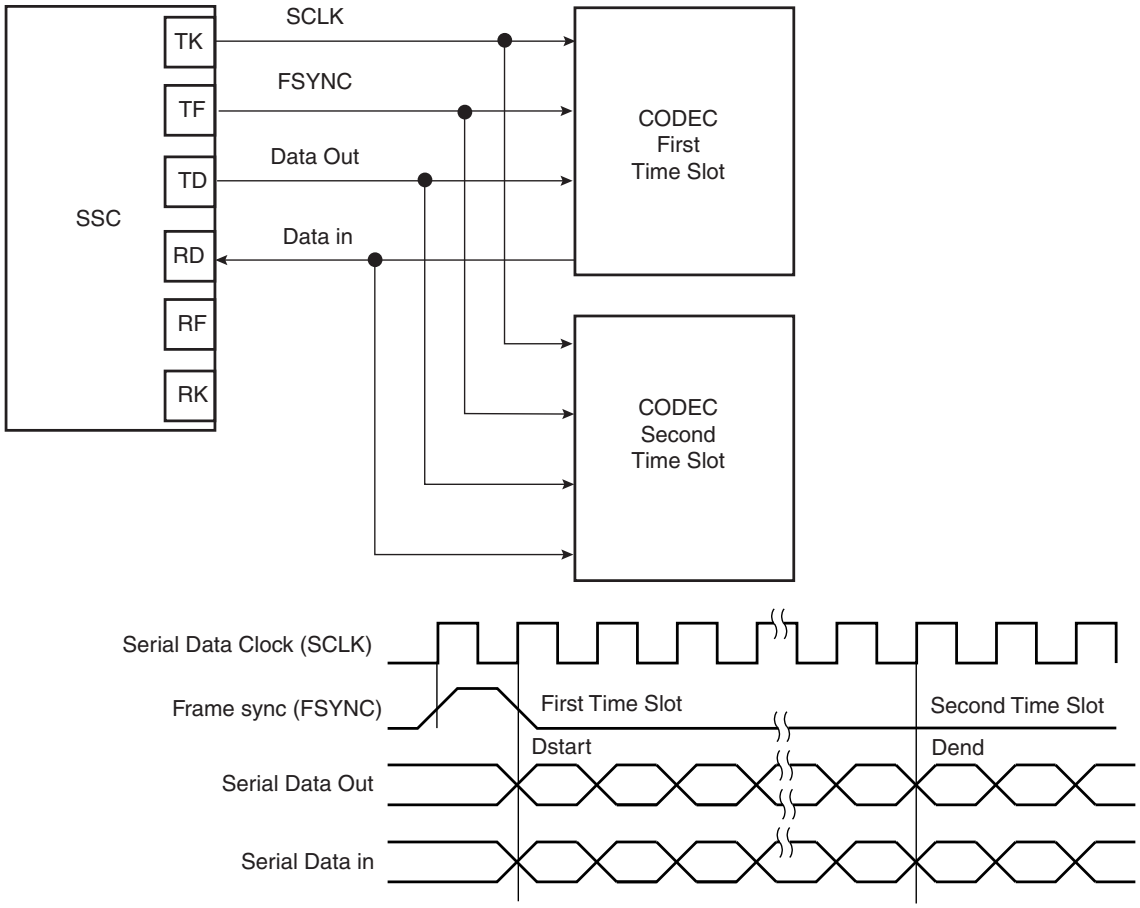


Figure 44-19.Time Slot Application Block Diagram



44.8.1 Write Protection Registers

To prevent any single software error that may corrupt SSC behavior, certain address spaces can be write-protected by setting the WPEN bit in the "SSC Write Protect Mode Register" (SSC_WPMR).

If a write access to the protected registers is detected, then the WPVS flag in the SSC Write Protect Status Register (US_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the SSC Write Protect Mode Register (SSC_WPMR) with the appropriate access key, WPKEY.

The protected registers are:

- "SSC Clock Mode Register" on page 1004
- "SSC Receive Clock Mode Register" on page 1005
- "SSC Receive Frame Mode Register" on page 1007
- "SSC Transmit Clock Mode Register" on page 1009
- "SSC Transmit Frame Mode Register" on page 1011
- "SSC Receive Compare 0 Register" on page 1015
- "SSC Receive Compare 1 Register" on page 1015

44.9 Synchronous Serial Controller (SSC) User Interface

Table 44-5. 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
0xE4	Write Protect Mode Register	SSC_WPMR	Read-write	0x0
0xE8	Write Protect Status Register	SSC_WPSR	Read-only	0x0
0x50-0xFC	Reserved	–	–	–
0x100-0x124	Reserved	–	–	–

44.9.1 SSC Control Register

Name: SSC_CR:

Address: 0xF0010000

Access: Write-only

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

- **RXEN: Receive Enable**

0 = No effect.

1 = Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0 = No effect.

1 = Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0 = No effect.

1 = Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0 = No effect.

1 = Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0 = No effect.

1 = Performs a software reset. Has priority on any other bit in SSC_CR.

44.9.2 SSC Clock Mode Register

Name: SSC_CMR

Address: 0xF0010004

Access: Read-write

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

This register can only be written if the WPEN bit is cleared in “SSC Write Protect Mode Register” .

- **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.

44.9.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Address: 0xF0010010

Access: Read-write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	STOP	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

This register can only be written if the WPEN bit is cleared in “SSC Write Protect Mode Register” .

- **CKS: Receive Clock Selection**

Value	Name	Description
0	MCK	Divided Clock
1	TK	TK Clock signal
2	RK	RK pin

- **CKO: Receive Clock Output Mode Selection**

Value	Name	Description
0	NONE	None, RK pin is an input
1	CONTINUOUS	Continuous Receive Clock, RK pin is an output
2	TRANSFER	Receive Clock only during data transfers, RK pin is an output

- **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**

Value	Name	Description
0	CONTINUOUS	None
1	EN_RF_LOW	Receive Clock enabled only if RF Pin is Low
2	EN_RF_HIGH	Receive Clock enabled only if RF Pin is High

- **START: Receive Start Selection**

Value	Name	Description
0	CONTINUOUS	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
1	TRANSMIT	Transmit start
2	RF_LOW	Detection of a low level on RF signal
3	RF_HIGH	Detection of a high level on RF signal
4	RF_FALLING	Detection of a falling edge on RF signal
5	RF_RISING	Detection of a rising edge on RF signal
6	RF_LEVEL	Detection of any level change on RF signal
7	RF_EDGE	Detection of any edge on RF signal
8	CMP_0	Compare 0

- **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.

44.9.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Address: 0xF0010014

Access: Read-write

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

23	22	21	20	19	18	17	16
–	FSOS			FSLEN			

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

7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

This register can only be written if the WPEN bit is cleared in “SSC Write Protect Mode Register” .

- **DATLEN: Data Length**

0 = Forbidden value (1-bit data length not supported).
Any other value: The bit stream contains DATLEN + 1 data bits.

- **LOOP: Loop Mode**

0 = Normal operating mode.

1 = RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0 = The lowest significant bit of the data register is sampled first in the bit stream.

1 = The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Receive Frame Sync Length**

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT * 16) + 1 Receive Clock periods.

- **FSOS: Receive Frame Sync Output Selection**

Value	Name	Description
0	NONE	None, RF pin is an input
1	NEGATIVE	Negative Pulse, RF pin is an output
2	POSITIVE	Positive Pulse, RF pin is an output
3	LOW	Driven Low during data transfer, RF pin is an output
4	HIGH	Driven High during data transfer, RF pin is an output
5	TOGGLING	Toggling at each start of data transfer, RF pin is an output

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

- **FSLEN_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on page 1007.

44.9.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Address: 0xF0010018

Access: Read-write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

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

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

This register can only be written if the WPEN bit is cleared in "SSC Write Protect Mode Register" .

- **CKS: Transmit Clock Selection**

Value	Name	Description
0	MCK	Divided Clock
1	RK	RK Clock signal
2	TK	TK pin

- **CKO: Transmit Clock Output Mode Selection**

Value	Name	Description
0	NONE	None, TK pin is an input
1	CONTINUOUS	Continuous Transmit Clock, TK pin is an output
2	TRANSFER	Transmit Clock only during data transfers, TK pin is an output

- **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**

Value	Name	Description
0	CONTINUOUS	None
1	EN_TF_LOW	Transmit Clock enabled only if TF pin is Low
2	EN_TF_HIGH	Transmit Clock enabled only if TF pin is High

- **START: Transmit Start Selection**

Value	Name	Description
0	CONTINUOUS	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.
1	RECEIVE	Receive start
2	TF_LOW	Detection of a low level on TF signal
3	TF_HIGH	Detection of a high level on TF signal
4	TF_FALLING	Detection of a falling edge on TF signal
5	TF_RISING	Detection of a rising edge on TF signal
6	TF_LEVEL	Detection of any level change on TF signal
7	TF_EDGE	Detection of any edge on TF signal

- **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.

44.9.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Address: 0xF001001C

Access: Read-write

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

23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			

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

7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

This register can only be written if the WPEN bit is cleared in “SSC Write Protect Mode Register” .

- **DATLEN: Data Length**

0 = Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. .

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0 = The lowest significant bit of the data register is shifted out first in the bit stream.

1 = The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per frame**

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB +1).

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

This field is used with FSLEN_EXT to determine the pulse length of the Transmit Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT * 16) + 1 Transmit Clock period.

- **FSOS: Transmit Frame Sync Output Selection**

Value	Name	Description
0	NONE	None, TF pin is an input
1	NEGATIVE	Negative Pulse, TF pin is an output
2	POSITIVE	Positive Pulse,TF pin is an output

Value	Name	Description
3	LOW	TF pin Driven Low during data transfer
4	HIGH	TF pin Driven High during data transfer
5	TOGGLING	TF pin Toggles at each start of data transfer

- **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).

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

- **FSLEN_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on page 1011.

44.9.7 SSC Receive Holding Register

Name: SSC_RHR

Address: 0xF0010020

Access: Read-only

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

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

44.9.8 SSC Transmit Holding Register

Name: SSC_THR

Address: 0xF0010024

Access: Write-only

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

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

44.9.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

Address: 0xF0010030

Access: Read-only

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

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

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

44.9.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

Address: 0xF0010034

Access: Read-write

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

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

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: Transmit Synchronization Data**

44.9.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Address: 0xF0010038

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CP0							
7	6	5	4	3	2	1	0
CP0							

This register can only be written if the WPEN bit is cleared in "SSC Write Protect Mode Register" .

- **CP0: Receive Compare Data 0**

44.9.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Address: 0xF001003C

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CP1							
7	6	5	4	3	2	1	0
CP1							

This register can only be written if the WPEN bit is cleared in "SSC Write Protect Mode Register" .

- **CP1: Receive Compare Data 1**

44.9.13 SSC Status Register

Name: SSC_SR

Address: 0xF0010040

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	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.

- **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.

- **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.

44.9.14 SSC Interrupt Enable Register

Name: SSC_IER

Address: 0xF0010044

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0 = No effect.

1 = Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0 = No effect.

1 = Enables the Transmit Empty Interrupt.

- **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.

- **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.

44.9.15 SSC Interrupt Disable Register

Name: SSC_IDR

Address: 0xF0010048

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	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.

- **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.

- **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.

44.9.16 SSC Interrupt Mask Register

Name: SSC_IMR

Address: 0xF001004C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	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.

- **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.

- **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.

44.9.17 SSC Write Protect Mode Register

Name: SSC_WPMR

Address: 0xF00100E4

Access: Read-write

Reset: See Table 44-5

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

Protects the registers:

- “SSC Clock Mode Register” on page 1004
- “SSC Receive Clock Mode Register” on page 1005
- “SSC Receive Frame Mode Register” on page 1007
- “SSC Transmit Clock Mode Register” on page 1009
- “SSC Transmit Frame Mode Register” on page 1011
- “SSC Receive Compare 0 Register” on page 1015
- “SSC Receive Compare 1 Register” on page 1015

- **WPKEY: Write Protect KEY**

Should be written at value 0x535343 (“SSC” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

44.9.18 SSC Write Protect Status Register

Name: SSC_WPSR

Address: 0xF00100E8

Access: Read-only

Reset: See Table 44-5

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the SSC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the SSC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading SSC_WPSR automatically clears all fields.

45. Ethernet MAC 10/100 (EMAC)

45.1 Description

The EMAC module implements a 10/100 Ethernet MAC compatible with the IEEE 802.3 standard using an address checker, statistics and control registers, receive and transmit blocks, and a DMA interface.

The address checker recognizes four specific 48-bit addresses and contains a 64-bit hash register for matching multicast and unicast addresses. It can recognize the broadcast address of all ones, copy all frames, and act on an external address match signal.

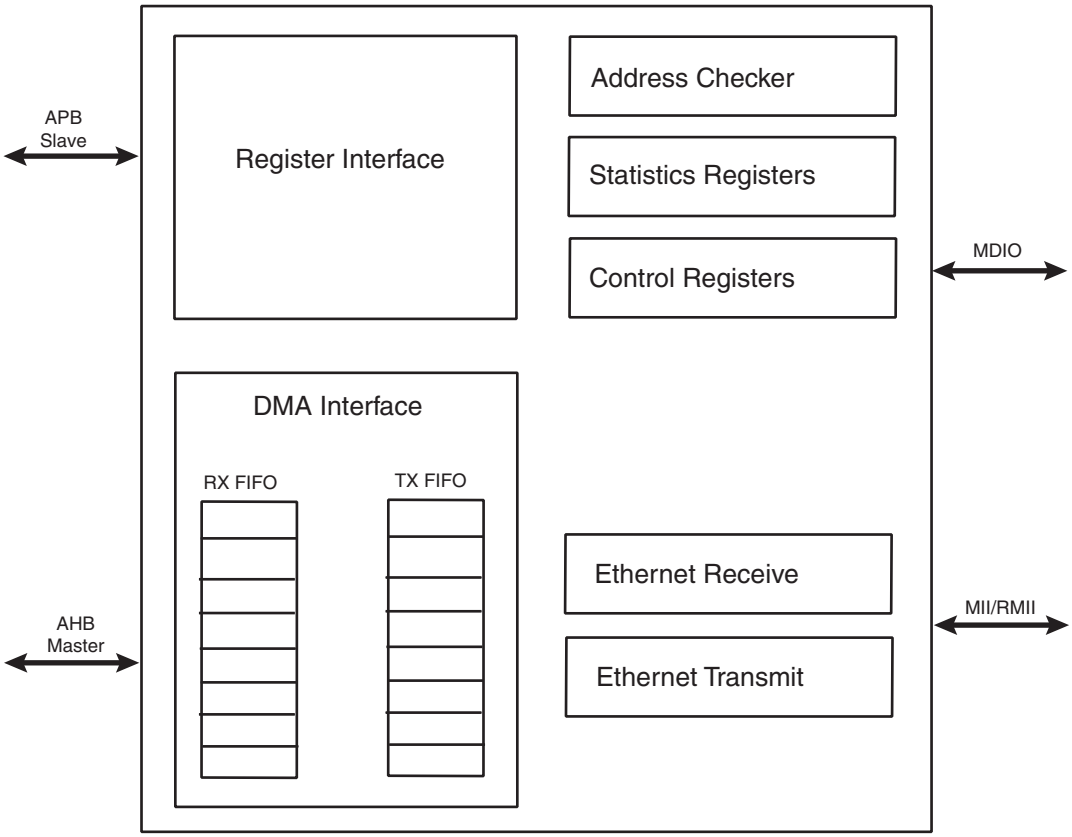
The statistics register block contains registers for counting various types of event associated with transmit and receive operations. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3.

45.2 Embedded Characteristics

- EMAC0 supports MII and RMII interfaces to the physical layer (EMAC1 supports RMII only)
- Compatible with IEEE Standard 802.3
- 10 and 100 Mbit/s Operation
- Full-duplex and Half-duplex Operation
- Statistics Counter Registers
- Interrupt Generation to Signal Receive and Transmit Completion
- DMA Master on Receive and Transmit Channels
- Transmit and Receive FIFOs
- Automatic Pad and CRC Generation on Transmitted Frames
- Automatic Discard of Frames Received with Errors
- Address Checking Logic Supports Up to Four Specific 48-bit Addresses
- Supports Promiscuous Mode Where All Valid Received Frames are Copied to Memory
- Hash Matching of Unicast and Multicast Destination Addresses
- Physical Layer Management through MDIO Interface
- Half-duplex Flow Control by Forcing Collisions on Incoming Frames
- Full-duplex Flow Control with Recognition of Incoming Pause Frames
- Support for 802.1Q VLAN Tagging with Recognition of Incoming VLAN and Priority Tagged Frames
- Multiple Buffers per Receive and Transmit Frame
- Jumbo Frames Up to 10240 bytes Supported

45.3 Block Diagram

Figure 45-1. EMAC Block Diagram



45.4 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 block

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 45-1 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.

45.4.1 Clock

Synchronization module in the EMAC requires that the bus clock (MCK) 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.

45.4.2 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

45.4.2.1 FIFO

The FIFO depths are 128 bytes for receive and 128 bytes for transmit and are a function of the system clock speed, memory latency and network speed.

Data is typically transferred into and out of the FIFOs in bursts of four words. For receive, a bus request is asserted when the FIFO contains four words and has space for 28 more. For transmit, a bus request is generated when there is space for four words, or when there is space for 27 words if the next transfer is to be only one or two words.

Thus the bus latency must be less than the time it takes to load the FIFO and transmit or receive three words (112 bytes) of data.

At 100 Mbit/s, it takes 8960 ns to transmit or receive 112 bytes of data. In addition, six master clock cycles should be allowed for data to be loaded from the bus and to propagate through the FIFOs. For a 133 MHz master clock this takes 45 ns, making the bus latency requirement 8915 ns.

45.4.2.2 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 45-1 for details of the receive buffer descriptor list.

Table 45-1. Receive Buffer Descriptor Entry

Bit	Function
Word 0	
31:2	Address of beginning of buffer
1	Wrap - marks last descriptor in receive buffer descriptor list.
0	Ownership - needs to be zero for the EMAC to write data to the receive buffer. The EMAC sets this to one once it has successfully written a frame to memory. Software has to clear this bit before the buffer can be used again.
Word 1	
31	Global all ones broadcast address detected
30	Multicast hash match
29	Unicast hash match
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

Table 45-1. Receive Buffer Descriptor Entry (Continued)

Bit	Function
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 register may be any word-aligned address, provided that there are at least 2048 word locations available between the pointer and the top of the memory.

Section 3.6 of the AMBA 2.0 specification states that bursts should not cross 1K boundaries. As receive buffer manager writes are bursts of two words, to ensure that this does not occur, it is best to write the pointer register with the least three significant bits set to zero. As receive buffers are used, the receive buffer manager sets bit zero of the first word of the descriptor to indicate *used*. If a receive error is detected the receive buffer currently being written is recovered. Previous buffers are not recovered. Software should search through the *used* bits in the buffer descriptors to find out how many frames have been received. It should be checking the start-of-frame and end-of-frame bits, and not rely on the value returned by the receive buffer queue pointer register which changes continuously as more buffers are used.

For CRC errored frames, excessive length frames or length field mismatched frames, all of which are counted in the statistics registers, it is possible that a frame fragment might be stored in a sequence of receive buffers. Software can detect this by looking for start of frame bit set in a buffer following a buffer with no end of frame bit set.

For a properly working Ethernet system, there should be no excessively long frames or frames greater than 128 bytes with CRC/FCS errors. Collision fragments are less than 128 bytes long. Therefore, it is a rare occurrence to find a frame fragment in a receive buffer.

If bit zero is set when the receive buffer manager reads the location of the receive buffer, then the buffer has already been used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the DMA block sets the buffer not available bit in the receive status register and triggers an interrupt.

If bit zero is set when the receive buffer manager reads the location of the receive buffer and a frame is being received, the frame is discarded and the receive resource error statistics register is incremented.

A receive overrun condition occurs when bus was not granted in time or because HRESP was not OK (bus error). In a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame received with an address that is recognized reuses the buffer.

If bit 17 of the network configuration register is set, the FCS of received frames shall not be copied to memory. The frame length indicated in the receive status field shall be reduced by four bytes in this case.

45.4.2.3 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 45-2 on page 1029 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 45-2. Transmit Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	**Byte Address of buffer**
Word 1	
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

45.4.3 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.

45.4.4 Pause Frame Support

The start of an 802.3 pause frame is as follows:

Table 45-3. 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).

45.4.5 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.

45.4.6 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

45.4.7 Broadcast Address

The broadcast address of 0xFFFFFFFFFFFF is recognized if the ‘no broadcast’ bit in the network configuration register is zero.

45.4.8 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.

45.4.9 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.

45.4.10 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.

45.4.11 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 45-4. 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

45.4.12 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 1039.

45.4.13 Physical Interface

Depending on products, the Ethernet MAC is capable of interfacing to RMII or MII Interface. 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 interfaces are capable of both 10 Mb/s and 100 Mb/s data rates as described in the IEEE 802.3u standard. The signals used by the MII and RMII interfaces are described in Table 45-5.

Table 45-5. 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 two 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 100 Mb/s data rate.

45.4.13.1 RMII Transmit and Receive Operation

The same signals are used internally for both the RMII and the MII operations. The RMII maps the 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.

45.5 Programming Interface

45.5.1 Initialization

45.5.1.1 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:

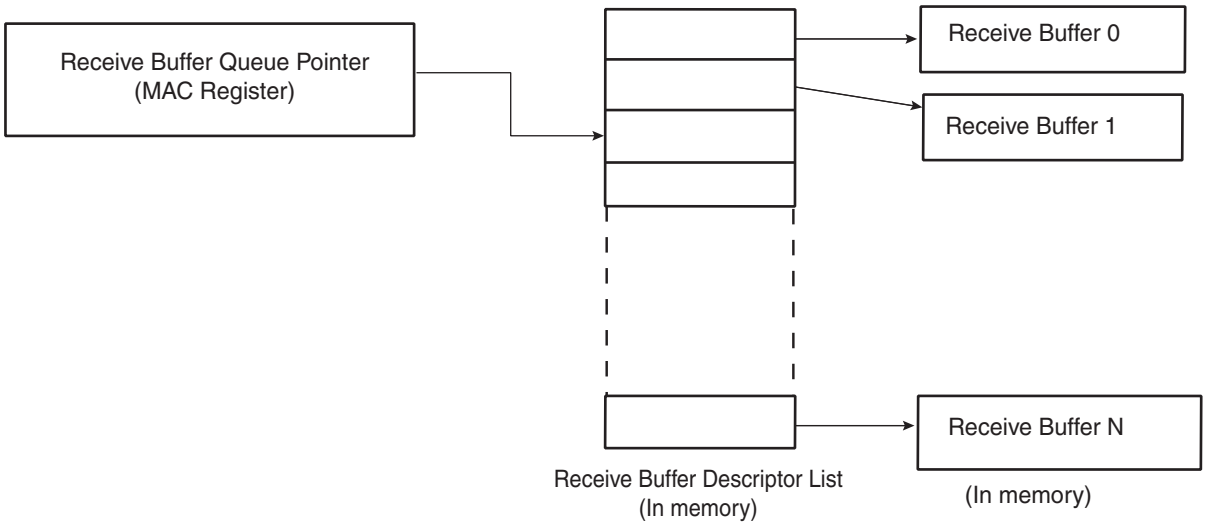
1. Write to network control register to disable transmit and receive circuits.
2. Write to network control register to change loop-back mode.
3. Write to network control register to re-enable transmit or receive circuits.

Note: These writes to network control register cannot be combined in any way.

45.5.1.2 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 1026. It points to this data structure.

Figure 45-2. 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.

45.5.1.3 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 45-2 on page 1029) 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.

45.5.1.4 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 1030. for details of address matching. Each register-pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

45.5.1.5 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 Interrupt Controller). 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.

45.5.1.6 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.

45.5.1.7 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 45-1 on page 1026) 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.

45.6 Ethernet MAC 10/100 (EMAC) User Interface

Table 45-6. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Network Control Register	EMAC_NCR	Read-write	0
0x04	Network Configuration Register	EMAC_NCFGR	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 45-6. 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	–	–	–

45.6.1 Network Control Register

Name: EMAC_NCR

Address: 0xF802C000

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	THALT	TSTART	BP
7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TE	RE	LLB	LB

- **LB: LoopBack**

Asserts the loopback signal to the PHY.

- **LLB: Loopback Local**

Connects `txd` to `rxd`, `tx_en` to `rx_dv`, forces full duplex and drives `rx_clk` and `tx_clk` with MCK 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.

45.6.2 Network Configuration Register

Name: EMAC_NCFGR

Address: 0xF802C004

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	IRXFCS	EFRHD	DRFCS	RLCE
15	14	13	12	11	10	9	8
RBOF		PAE	RTY	CLK		–	BIG
7	6	5	4	3	2	1	0
UNI	MTI	NBC	CAF	JFRAME	–	FD	SPD

- **SPD: Speed**

Set to 1 to indicate 100 Mbit/s operation, 0 for 10 Mbit/s. The value of this pin is reflected on the `speed` pin.

- **FD: Full Duplex**

If set to 1, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting. Also controls the `half_duplex` pin.

- **CAF: Copy All Frames**

When set to 1, all valid frames are received.

- **JFRAME: Jumbo Frames**

Set to one to enable jumbo frames of up to 10240 bytes to be accepted.

- **NBC: No Broadcast**

When set to 1, frames addressed to the broadcast address of all ones are not received.

- **MTI: Multicast Hash Enable**

When set, multicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.

- **UNI: Unicast Hash Enable**

When set, unicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.

- **BIG: Receive 1536 bytes Frames**

Setting this bit means the EMAC receives frames up to 1536 bytes in length. Normally, the EMAC would reject any frame above 1518 bytes.

- **CLK: MDC Clock Divider**

Set according to system clock speed. This determines by what number system clock is divided to generate MDC. For conformance with 802.3, MDC must not exceed 2.5 MHz (MDC is only active during MDIO read and write operations).

Value	Name	Description
0	MCK_8	MCK divided by 8 (MCK up to 20 MHz).
1	MCK_16	MCK divided by 16 (MCK up to 40 MHz).
2	MCK_32	MCK divided by 32 (MCK up to 80 MHz).
3	MCK_64	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.

Value	Name	Description
0	OFFSET_0	No offset from start of receive buffer.
1	OFFSET_1	One-byte offset from start of receive buffer.
2	OFFSET_2	Two-byte offset from start of receive buffer.
3	OFFSET_3	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 counted as length errors.

- **DRFCS: Discard Receive FCS**

When set, the FCS field of received frames is not 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.

45.6.3 Network Status Register

Name: EMAC_NSR

Address: 0xF802C008

Access: Read-only

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

- **MDIO**

Returns status of the mdio_in pin. Use the PHY maintenance register for reading managed frames rather than this bit.

- **IDLE**

0 = The PHY logic is running.

1 = The PHY management logic is idle (i.e., has completed).

45.6.4 Transmit Status Register

Name: EMAC_TSR

Address: 0xF802C014

Access: Read-write

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

- **RLES: 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.

45.6.5 Receive Buffer Queue Pointer Register

Name: EMAC_RBQP

Address: 0xF802C018

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR						–	–

This register points to the entry in the receive buffer queue (descriptor list) currently being used. It is written with the start location of the receive buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set.

Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits.

Receive buffer writes also comprise bursts of two words and, as with transmit buffer reads, it is recommended that bit 2 is always written with zero to prevent a burst crossing a 1K boundary, in violation of section 3.6 of the AMBA specification.

- **ADDR: Receive Buffer Queue Pointer Address**

Written with the address of the start of the receive queue, reads as a pointer to the current buffer being used.

45.6.6 Transmit Buffer Queue Pointer Register

Name: EMAC_TBQP

Address: 0xF802C01C

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR						–	–

This register points to the entry in the transmit buffer queue (descriptor list) currently being used. It is written with the start location of the transmit buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set. This register can only be written when bit 3 in the transmit status register is low.

As transmit buffer reads consist of bursts of two words, it is recommended that bit 2 is always written with zero to prevent a burst crossing a 1K boundary, in violation of section 3.6 of the AMBA specification.

- **ADDR: Transmit Buffer Queue Pointer Address**

Written with the address of the start of the transmit queue, reads as a pointer to the first buffer of the frame being transmitted or about to be transmitted.

45.6.7 Receive Status Register

Name: EMAC_RSR

Address: 0xF802C020

Access: Read-write

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

This register, when read, provides details of the status of a receive. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

- **BNA: Buffer Not Available**

An attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA rereads the pointer each time a new frame starts until a valid pointer is found. This bit is set at each attempt that fails even if it has not had a successful pointer read since it has been cleared.

Cleared by writing a one to this bit.

- **REC: Frame Received**

One or more frames have been received and placed in memory. Cleared by writing a one to this bit.

- **OVR: Receive Overrun**

The DMA block was unable to store the receive frame to memory, either because the bus was not granted in time or because a not OK `hresp(bus error)` was returned. The buffer is recovered if this happens.

Cleared by writing a one to this bit.

45.6.8 Interrupt Status Register

Name: EMAC_ISR

Address: 0xF802C024

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	PTZ	PFRE	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TXERR	RLEX	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.

- **RLEX: 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.

- **PFRE: 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.

45.6.9 Interrupt Enable Register

Name: EMAC_IER

Address: 0xF802C028

Access: Write-only

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

45.6.10 Interrupt Disable Register

Name: EMAC_IDR

Address: 0xF802C02C

Access: Write-only

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

45.6.11 Interrupt Mask Register

Name: EMAC_IMR

Address: 0xF802C030

Access: Read-only

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

45.6.12 PHY Maintenance Register

Name: EMAC_MAN

Address: 0xF802C034

Access: Read-write

31	30	29	28	27	26	25	24
SOF		RW		PHYA			

23	22	21	20	19	18	17	16
PHYA		REGA				CODE	

15	14	13	12	11	10	9	8
DATA							

7	6	5	4	3	2	1	0
DATA							

- **DATA**

For a write operation this is written with the data to be written to the PHY.

After a read operation this contains the data read from the PHY.

- **CODE:**

Must be written to 10. Reads as written.

- **REGA: Register Address**

Specifies the register in the PHY to access.

- **PHYA: PHY Address**

- **RW: Read-write**

10 is read; 01 is write. Any other value is an invalid PHY management frame

- **SOF: Start of frame**

Must be written 01 for a valid frame.

45.6.13 Pause Time Register

Name: EMAC_PTR

Address: 0xF802C038

Access: Read-write

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

- **PTIME: Pause Time**

Stores the current value of the pause time register which is decremented every 512 bit times.

45.6.14 Hash Register Bottom

Name: EMAC_HRB

Address: 0xF802C090

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR:**

Bits 31:0 of the hash address register. See “Hash Addressing” on page 1032.

45.6.15 Hash Register Top

Name: EMAC_HRT

Address: 0xF802C094

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR:**

Bits 63:32 of the hash address register. See “Hash Addressing” on page 1032.

45.6.16 Specific Address 1 Bottom Register

Name: EMAC_SA1B

Address: 0xF802C098

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

45.6.17 Specific Address 1 Top Register

Name: EMAC_SA1T

Address: 0xF802C09C

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

45.6.18 Specific Address 2 Bottom Register

Name: EMAC_SA2B

Address: 0xF802C0A0

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

45.6.19 Specific Address 2 Top Register

Name: EMAC_SA2T

Address: 0xF802C0A4

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

45.6.20 Specific Address 3 Bottom Register

Name: EMAC_SA3B

Address: 0xF802C0A8

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

45.6.21 Specific Address 3 Top Register

Name: EMAC_SA3T

Address: 0xF802C0AC

Access: Read-write

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

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

45.6.22 Specific Address 4 Bottom Register

Name: EMAC_SA4B

Address: 0xF802C0B0

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

45.6.23 Specific Address 4 Top Register

Name: EMAC_SA4T

Address: 0xF802C0B4

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

45.6.24 Type ID Checking Register

Name: EMAC_TID

Address: 0xF802C0B8

Access: Read-write

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

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

15	14	13	12	11	10	9	8
TID							

7	6	5	4	3	2	1	0
TID							

- **TID: Type ID checking**

For use in comparisons with received frames TypeID/Length field.

45.6.25 User Input/Output Register

Name: EMAC_USRIO

Address: 0xF802C0C0

Access: Read-write

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

- **RMII: Reduced MII**

When set, this bit enables the RMII operation mode. When reset, it selects the MII mode.

- **CLKEN: Clock Enable**

When set, this bit enables the transceiver input clock.

Setting this bit to 0 reduces power consumption when the treasurer is not used.

45.6.26 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.

45.6.26.1 Pause Frames Received Register

Name: EMAC_PFR

Address: 0xF802C03C

Access: Read-write

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

- **FROK: Pause Frames received OK**

A 16-bit register counting the number of good pause frames received. A good frame has a length of 64 to 1518 (1536 if bit 8 set in network configuration register) and has no FCS, alignment or receive symbol errors.

45.6.26.2 Frames Transmitted OK Register

Name: EMAC_FTO

Address: 0xF802C040

Access: Read-write

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

- **FTOK: Frames Transmitted OK**

A 24-bit register counting the number of frames successfully transmitted, i.e., no underrun and not too many retries.

45.6.26.3 Single Collision Frames Register

Name: EMAC_SCF

Address: 0xF802C044

Access: Read-write

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

- **SCF: Single Collision Frames**

A 16-bit register counting the number of frames experiencing a single collision before being successfully transmitted, i.e., no underrun.

45.6.26.4 Multicollision Frames Register

Name: EMAC_MCF

Address: 0xF802C048

Access: Read-write

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

- **MCF: Multicollision Frames**

A 16-bit register counting the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

45.6.26.5 Frames Received OK Register

Name: EMAC_FRO

Address: 0xF802C04C

Access: Read-write

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

23	22	21	20	19	18	17	16
FROK							

15	14	13	12	11	10	9	8
FROK							

7	6	5	4	3	2	1	0
FROK							

- **FROK: Frames Received OK**

A 24-bit register counting the number of good frames received, i.e., address recognized and successfully copied to memory. A good frame is of length 64 to 1518 bytes (1536 if bit 8 set in network configuration register) and has no FCS, alignment or receive symbol errors.

45.6.26.6 Frames Check Sequence Errors Register

Name: EMAC_FCSE

Address: 0xF802C050

Access: Read-write

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

- **FCSE: Frame Check Sequence Errors**

An 8-bit register counting frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes.

45.6.26.7 Alignment Errors Register

Name: EMAC_ALE

Address: 0xF802C054

Access: Read-write

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

- **ALE: Alignment Errors**

An 8-bit register counting frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes.

45.6.26.8 Deferred Transmission Frames Register

Name: EMAC_DTF

Address: 0xF802C058

Access: Read-write

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

- **DTF: Deferred Transmission Frames**

A 16-bit register counting the number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit underrun.

45.6.26.9 Late Collisions Register

Name: EMAC_LCOL

Address: 0xF802C05C

Access: Read-write

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

- **LCOL: Late Collisions**

An 8-bit register counting the number of frames that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice; i.e., both as a collision and a late collision.

45.6.26.10 Excessive Collisions Register

Name: EMAC_ECOL

Address: 0xF802C060

Access: Read-write

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

- **EXCOL: Excessive Collisions**

An 8-bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions.

45.6.26.11 Transmit Underrun Errors Register

Name: EMAC_TUND

Address: 0xF802C064

Access: Read-write

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

- **TUND: Transmit Underruns**

An 8-bit register counting the number of frames not transmitted due to a transmit DMA underrun. If this register is incremented, then no other statistics register is incremented.

45.6.26.12 Carrier Sense Errors Register

Name: EMAC_CSE

Address: 0xF802C068

Access: Read-write

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

- **CSE: Carrier Sense Errors**

An 8-bit register counting the number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no underrun). Only incremented in half-duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other statistics registers is unaffected by the detection of a carrier sense error.

45.6.26.13 Receive Resource Errors Register

Name: EMAC_RRE

Address: 0xF802C06C

Access: Read-write

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

- **RRE: Receive Resource Errors**

A 16-bit register counting the number of frames that were address matched but could not be copied to memory because no receive buffer was available.

45.6.26.14 Receive Overrun Errors Register

Name: EMAC_ROV

Address: 0xF802C070

Access: Read-write

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

- **ROVR: Receive Overrun**

An 8-bit register counting the number of frames that are address recognized but were not copied to memory due to a receive DMA overrun.

45.6.26.15 Receive Symbol Errors Register

Name: EMAC_RSE

Address: 0xF802C074

Access: Read-write

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

- **RSE: Receive Symbol Errors**

An 8-bit register counting the number of frames that had `rx_er` asserted during reception. Receive symbol errors are also counted as an FCS or alignment error if the frame is between 64 and 1518 bytes in length (1536 if bit 8 is set in the network configuration register). If the frame is larger, it is recorded as a jabber error.

45.6.26.16 Excessive Length Errors Register

Name: EMAC_ELE

Address: 0xF802C078

Access: Read-write

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

- **EXL: Excessive Length Errors**

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length but do not have either a CRC error, an alignment error nor a receive symbol error.

45.6.26.17 Receive Jabbers Register

Name: EMAC_RJA

Address: 0xF802C07C

Access: Read-write

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

- **RJB: Receive Jabbers**

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length and have either a CRC error, an alignment error or a receive symbol error.

45.6.26.18 Undersize Frames Register

Name: EMAC_USF

Address: 0xF802C080

Access: Read-write

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

- **USF: Undersize Frames**

An 8-bit register counting the number of frames received less than 64 bytes in length but do not have either a CRC error, an alignment error or a receive symbol error.

45.6.26.19 SQE Test Errors Register

Name: EMAC_STE

Address: 0xF802C084

Access: Read-write

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

- **SQER: SQE Test Errors**

An 8-bit register counting the number of frames where `col` was not asserted within 96 bit times (an interframe gap) of `tx_en` being deasserted in half duplex mode.

45.6.26.20 Received Length Field Mismatch Register

Name: EMAC_RLE

Address: 0xF802C088

Access: Read-write

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

- **RLFM: Receive Length Field Mismatch**

An 8-bit register counting the number of frames received that have a measured length shorter than that extracted from its length field. Checking is enabled through bit 16 of the network configuration register. Frames containing a type ID in bytes 13 and 14 (i.e., length/type ID = 0x0600) are not counted as length field errors, neither are excessive length frames.

46. Electrical Characteristics

46.1 Absolute Maximum Ratings

Table 46-1. Absolute Maximum Ratings*

Operating Temperature (Industrial).....-40°C to + 85°C
Junction Temperature.....125°C
Storage Temperature.....-60°C to + 150°C
Voltage on Input Pins with Respect to Ground.....-0.3V to VDDIO+0.3V(+ 4V max)
Maximum Operating Voltage (VDDCORE, VDDPLLA, VDDUTMIC).....1.2V
(VDDIOM0).....2.0V
(VDDIOM1, VDDIOPx, VDDUTMII, VDDOSC, VDDANA and VDDBU).....4.0V
Total DC Output Current on all I/O lines.....350 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.

46.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to +85°C, unless otherwise specified.

Table 46-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDCORE}	DC Supply Core		0.9	1.0	1.1	V
$V_{DDCORErip}$	VDDCORE ripple				20	mVrms
$V_{DDUTMIC}$	DC Supply UDPHS and UHPHS UTMI+ Core		0.9	1.0	1.1	V
$V_{DDUTMII}$	DC Supply UDPHS and UHPHS UTMI+ Interface		3.0	3.3	3.6	V
V_{DDBU}	DC Supply Backup		1.8		3.6	V
$V_{DDBUrip}$	VDDBU ripple				30	mVrms
V_{DDPLLA}	DC Supply PLLA		0.9	1.0	1.1	V
$V_{DDPLLArip}$	VDDPLLA ripple				10	mVrms
V_{DDOSC}	DC Supply Oscillator		1.65		3.6	V
$V_{DDOSCrip}$	VDDOSC ripple				30	mVrms
V_{DDIOM}	DC Supply EBI I/Os		1.65/3.0	1.8/3.3	1.95/3.6	V
V_{DDNF}	DC Supply NAND Flash I/Os		1.65/3.0	1.8/3.3	1.95/3.6	V
V_{DDIOP0}	DC Supply Peripheral I/Os		1.65		3.6	V
V_{DDIOP1}	DC Supply Peripheral I/Os		1.65		3.6	V
V_{DDANA}	DC Supply Analog		3.0	3.3	3.6	V
V_{IL}	Input Low-level Voltage	V_{DDIO} from 3.0V to 3.6V	-0.3		0.8	V
		V_{DDIO} from 1.65V to 1.95V	-0.3		$0.3 \times V_{DDIO}$	V
V_{IH}	Input High-level Voltage	V_{DDIO} from 3.0V to 3.6V	2		V_{DDIO} + 0.3	V
		V_{DDIO} from 1.65V to 1.95V	$0.7 \times V_{DDIO}$		V_{DDIO} + 0.3	V
V_{OL}	Output Low-level Voltage	I_O Max, V_{DDIO} from 3.0V to 3.6V			0.4	V
		CMOS (I_O < 0.3 mA), V_{DDIO} from 1.65V to 1.95V			0.1	V
		TTL (I_O Max), V_{DDIO} from 1.65V to 1.95V			0.4	V
V_{OH}	Output High-level Voltage	I_O Max, V_{DDIO} from 3.0V to 3.6V	V_{DDIO} - 0.4			V
		CMOS (I_O < 0.3 mA), V_{DDIO} from 1.65V to 1.95V	V_{DDIO} - 0.1			V
		TTL (I_O Max), V_{DDIO} from 1.65V to 1.95V	V_{DDIO} - 0.4			V
V_{T-}	Schmitt trigger Negative going threshold Voltage	I_O Max, V_{DDIO} from 3.0V to 3.6V	0.8	1.1		V
		TTL (I_O Max), V_{DDIO} from 1.65V to 1.95V			$0.3 \times V_{DDIO}$	V
V_{T+}	Schmitt trigger Positive going threshold Voltage	I_O Max, V_{DDIO} from 3.0V to 3.6V		1.6	2.0	V
		TTL (I_O Max), V_{DDIO} from 1.65V to 1.95V	$0.3 \times V_{DDIO}$			V

Table 46-2. DC Characteristics (Continued)

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{HYS}	Schmitt trigger Hysteresis	V_{DDIO} from 3.0V to 3.6V		0.5		0.75	V
		V_{DDIO} from 1.65V to 1.95V		0.28		0.6	V
R_{PULLUP}	Pull-up/Pull-down Resistance	PA0–PA31 PB0–PB31 PC0–PC31 NTRST and NRST		40	75	190	kΩ
		PD0–PD21 V_{DDIOM} in 1.8V range		80		300	
		PD0–PD21 V_{DDIOM1} in 3.3V range		120		350	
I_O	Output Current	PA0–PA31 PB0–PB31 PD0–PD31 PE0–PE31				8	mA
		PC0–PC31 V_{DDIOM1} in 1.8V range				2	
		PC0–PC31 V_{DDIOM1} in 3.3V range				4	
I_{SC}	Static Current	On V_{DDCORE} = 1.0V, MCK = 0 Hz, excluding POR	T_A = 25°C		14		mA
		All inputs driven TMS, TDI, TCK, NRST = 1	T_A = 85°C			46	
		On V_{DDBU} = 3.3V, Logic cells consumption, excluding POR	T_A = 25°C		8		µA
		All inputs driven WKUP = 0	T_A = 85°C			18	
Z_{IN}	Input impedance	V_{DDIO} = 3.3V			3.3		MΩ
		V_{DDIO} = 1.8V			1.8		MΩ

46.3 Power Consumption

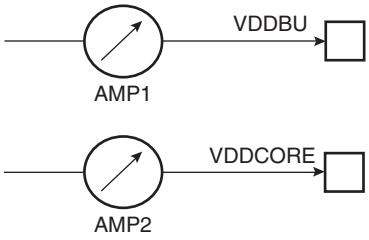
- Typical power consumption of PLLs, Slow Clock and Main Oscillator.
- Power consumption of power supply in four different modes: Active, Idle, Ultra Low-power and Backup.
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

46.3.1 Power Consumption versus Modes

The values in Table 46-3 and Table 46-4 are estimated values of the power consumption with operating conditions as follows:

- V_{DDIOM} = 1.8V
- V_{DDIOP0} and V_{DDIOP1} = 3.3V
- V_{DDPLLA} = 1.0V
- V_{DDCORE} = 1.0V
- V_{DDBU} = 3.3V
- T_A = 25°C
- There is no consumption on the I/Os of the device

Figure 46-1. Measures Schematics



These figures represent the power consumption estimated on the power supplies.

Table 46-3. Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Active	ARM Core clock is 400 MHz. MCK is 133 MHz. All peripheral clocks activated. onto AMP2	109	mA
Idle	Idle state, waiting an interrupt. All peripheral clocks de-activated. onto AMP2	38	mA
Ultra low power	ARM Core clock is 500 Hz. All peripheral clocks de-activated. onto AMP2	8	mA
Backup	Device only V_{DDBU} powered onto AMP1	8	µA

Table 46-4. Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
PIO Controller	1	µA/MHz (1)
USART	6	
UHPHS	60	
UDPHS	22	
ADC	5	
TWI	2	
SPI	3	
PWM	6	
HSMCI	28	
SSC	5	
Timer Counter Channels	12	
DMA	1	
SMD	14	
CAN	17	
EMAC	39	

Note: 1. Reference frequency is peripheral frequency. It can be a division (1,2,4,8) of MCK. Refer to PMC section for more details.

46.4 Clock Characteristics

46.4.1 Processor Clock Characteristics

Table 46-5. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	V_{DDCORE} = 0.9V, T_A = 85°C	125 (1)	400	MHz

Note: 1. For DDR2 usage only, there are no limitations to LP-DDR, SDRAM and mobile SDRAM.

46.4.2 Master Clock Characteristics

The master clock is the maximum clock at which the system is able to run. It is given by the smallest value of the internal bus clock and EBI clock.

Table 46-6. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	Master Clock Frequency	V_{DDCORE} = 0.9V, T_A = 85°C	125 (1)	133	MHz

Note: 1. For DDR2 usage only, there are no limitations to LP-DDR, SDRAM and mobile SDRAM.

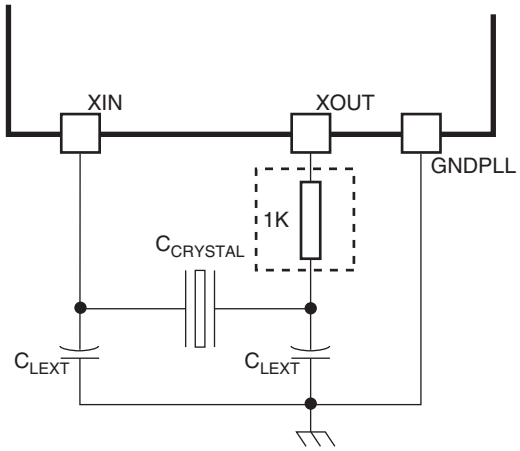
46.5 Main Oscillator Characteristics

Table 46-7. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		12		16	MHz
$C_{CRYSTAL}$ (1)	Crystal Load Capacitance		15		20	pF
C_{LEXT}	External Load Capacitance	$C_{CRYSTAL}$ = 15 pF(1)		27		pF
		$C_{CRYSTAL}$ = 20 pF(1)		32		pF
	Duty Cycle		40		60	%
t_{ST}	Startup Time				2	ms
I_{DDST}	Standby Current Consumption	Standby mode			1	µA
P_{ON}	Drive Level				150	µW
$I_{DD\ ON}$	Current Dissipation	@ 12 MHz		0.52	0.55	mA
		@ 16 MHz		0.7	1.1	mA

Note: 1. The $C_{CRYSTAL}$ value is specified by the crystal manufacturer. In our case, $C_{CRYSTAL}$ must be between 15 pF and 20 pF. All parasitic capacitance, package and board, **must be calculated** in order to reach 15 pF (minimum targeted load for the oscillator) by taking into account the internal load C_{INT}. So, to target the minimum oscillator load of 15 pF, external capacitance must be: 15 pF - 4 pF = 11 pF which means that 22 pF is the target value (22 pF from XIN to GND and 22 pF from XOUT to GND). If 20 pF load is targeted, the sum of pad, package, board and external capacitances must be 20 pF - 4 pF = 16 pF which means 32 pF (32 pF from XIN to GND and 32 pF from XOUT to GND).

Figure 46-2. Main Oscillator Schematics



Note: A 1K resistor must be added on XOUT pin for crystals with frequencies lower than 8 MHz.

46.5.1 Crystal Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 46-8. Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	@ 16 MHz @ 12 MHz $C_{CRYSTAL}$ Max @ 12 MHz $C_{CRYSTAL}$ Min			80 90 110	Ω
C_M	Motional Capacitance		5		9	fF
C_S	Shunt Capacitance				7	pF

46.5.2 XIN Clock Characteristics

Table 46-9. XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPXIN})$	XIN Clock Frequency			50	MHz
t_{CPXIN}	XIN Clock Period		20		ns
t_{CHXIN}	XIN Clock High Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	ns
t_{CLXIN}	XIN Clock Low Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	ns
C_{IN}	XIN Input Capacitance	(1)		25	pF
R_{IN}	XIN Pulldown Resistor	(1)		500	kΩ
V_{IN}	XIN Voltage	(1)	V_{DDOSC}	V_{DDOSC}	V

Note: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1) in the CKGR_MOR. See “PMC Clock Generator Main Oscillator Register” in the PMC section.

46.6 12 MHz RC Oscillator Characteristics

Table 46-10. 12 MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F0	Nominal Frequency		8.4	12	15.6	MHz
Duty	Duty Cycle		45	50	55	%
$I_{DD\ ON}$	Power Consumption Oscillation	Without trimming	86		140	µA
		After trimming sequence	86		125	
t_{ST}	Startup Time		6		10	µs
$I_{DD\ STDBY}$	Standby Consumption				22	µA

46.7 32 kHz Oscillator Characteristics

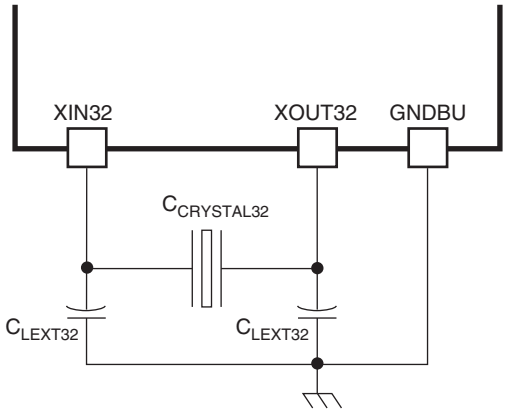
Table 46-11. 32 kHz Oscillator Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency				32.768		kHz
$C_{CRYSTAL32}$	Load Capacitance	Crystal @ 32.768 kHz		6		12.5	pF
C_{LEXT32} (2)	External Load Capacitance	$C_{CRYSTAL32}$ = 6 pF			6		pF
		$C_{CRYSTAL32}$ = 12.5 pF			19		pF
	Duty Cycle			40	50	60	%
t_{ST}	Startup Time	R_S = 50 kΩ (1)	$C_{CRYSTAL32}$ = 6 pF			400	ms
			$C_{CRYSTAL32}$ = 12.5 pF			900	ms
		R_S = 100 kΩ (1)	$C_{CRYSTAL32}$ = 6 pF			600	ms
			$C_{CRYSTAL32}$ = 12.5 pF			1200	ms

Notes: 1. R_S is the equivalent series resistance.

2. C_{LEXT32} is determined by taking into account internal, parasitic and package load capacitance.

Figure 46-3. 32 kHz Oscillator Schematics



46.7.1 32 kHz Crystal Characteristics

Table 46-12. 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
$I_{DD\ ON}$	Current Dissipation	R_S = 50 kΩ (1) $C_{CRYSTAL32}$ = 6 pF		0.55	1.3	µA
		R_S = 50 kΩ (1) $C_{CRYSTAL32}$ = 12.5pF		0.85	1.6	µA
		R_S = 100 kΩ (1) $C_{CRYSTAL32}$ = 6 pF		0.7	2.0	µA
		R_S = 100 kΩ (1) $C_{CRYSTAL32}$ = 12.5 pF		1.1	2.2	µA
$I_{DD\ STDBY}$	Standby Consumption				0.3	µA

46.7.2 XIN32 Clock Characteristics

Table 46-13. XIN32 Clock Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPXIN32})$	XIN32 Clock Frequency			44	kHz
$t_{CPXIN32}$	XIN32 Clock Period		22		µs
$t_{CHXIN32}$	XIN32 Clock High Half-period		11		µs
$t_{CLXIN32}$	XIN32 Clock Low Half-period		11		µs
t_{CLCH32}	XIN32 Clock Rise time		400		ns
t_{CLCL32}	XIN32 Clock Fall time		400		ns
C_{IN32}	XIN32 Input Capacitance	(1)		6	pF
R_{IN32}	XIN32 Pulldown Resistor	(1)		4	MΩ
V_{IN32}	XIN32 Voltage	(1)	V_{DDBU}	V_{DDBU}	V
V_{INIL32}	XIN32 Input Low-Level Voltage	(1)	-0.3	$0.3 \times V_{DDBU}$	V
V_{INIH32}	XIN32 Input High-Level Voltage	(1)	$0.7 \times V_{DDBU}$	V_{DDBU} + 0.3	V

Note: 1. These characteristics apply only when the 32.768 kHz Oscillator is in bypass mode (i.e., when RCEN = 0, OSC32EN = 0, OSCSEL = 1 and OSC32BYP = 1) in the Slow Clock Controller Configuration Register (SCKC_CR). See “Slow Clock Selection” in the PMC section.

46.8 32 kHz RC Oscillator Characteristics

Table 46-14. 32 kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPRCz})$	Crystal Oscillator Frequency		20	32	44	kHz
	Duty Cycle		45		55	%
t_{ST}	Startup Time				75	µs
$I_{DD\ ON}$	Power Consumption Oscillation	After startup time		1.1	2.1	µA
$I_{DD\ STDBY}$	Standby Consumption				0.4	µA

46.9 PLL Characteristics

Table 46-15. PLLA Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output Frequency	Refer to following table	400		800	MHz
f_{IN}	Input Frequency		2		32	MHz
I_{PLL}	Current Consumption	Active mode		7	9	mA
		Standby mode			1	µA
t_{ST}	Startup Time				50	µs

The following configuration of bit PMC_PLLICPR.ICPLLA and field CKGR_PLLAR.OUTA must be done for each PLLA frequency range.

Table 46-16. PLLA Frequency Regarding ICPLLA and OUTA

PLL Frequency Range (MHz)	PMC_PLLICPR.ICPLLA Value	CKGR_PLLAR.OUTA Value
745–800	0	00
695–750	0	01
645–700	0	10
595–650	0	11
545–600	1	00
495–550	1	01
445–500	1	10
400–450	1	11

46.9.1 UTMI PLL Characteristics

Table 46-17. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{IN}	Input Frequency		4	12	32	MHz
f_{OUT}	Output Frequency		450	480	600	MHz
I_{PLL}	Current Consumption	Active mode		5	8	mA
		Standby mode			1.5	µA
t_{ST}	Startup Time				50	µs

46.10 I/Os

Criteria used to define the maximum frequency of the I/Os:

- Output duty cycle (40%–60%)
- Minimum output swing: 100 mV to V_{DDIO} - 100 mV
- Addition of rising and falling time inferior to 75% of the period

Table 46-18. I/O Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
f_{max}	VDDIOP powered pins frequency	3.3V domain (1)	100 (Low Drive)	200 (High Drive)	MHz
		1.8V domain (2)	50 (Low Drive)	166 (High Drive)	MHz

Notes: 1. 3.3V domain: V_{DDIOP} from 3.0V to 3.6V, maximum external capacitor = 20 pF
2. 1.8V domain: V_{DDIOP} from 1.65V to 1.95V, maximum external capacitor = 20 pF

46.11 USB HS Characteristics

Table 46-19. USB HS Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
R_{PUI}	Bus Pull-up Resistor on Upstream Port (idle bus)	In LS or FS Mode		1.5		kΩ
R_{PUA}	Bus Pull-up Resistor on Upstream Port (upstream port receiving)	In LS or FS Mode		15		kΩ
Settling time						
t_{BIAS}	Bias settling time				20	µs
t_{OSC}	Oscillator settling time	With Crystal 12 MHz			2	ms
$t_{SETTLING}$	Settling time	f_{IN} = 12 MHz		0.3	0.5	ms

Table 46-20. USB HS Static Power Consumption

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{BIAS}	Bias current consumption on VBG				1	µA
$I_{VDDUTMII}$	HS Transceiver and I/O current consumption				8	µA
	LS / FS Transceiver and I/O current consumption	No connection (1)			3	µA
$I_{VDDUTMIC}$	Core, PLL, and Oscillator current consumption				2	µA

Note: 1. If cable is connected add 200 µA (Typical) due to Pull-up/Pull-down current consumption.

Table 46-21. USB HS Dynamic Power Consumption

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{BIAS}	Bias current consumption on VBG			0.7	0.8	mA
$I_{VDDUTMII}$	HS Transceiver current consumption	HS transmission		47	60	mA
	HS Transceiver current consumption	HS reception		18	27	mA
	LS / FS Transceiver current consumption	FS transmission 0m cable (1)		4	6	mA
	LS / FS Transceiver current consumption	FS transmission 5m cable (1)		26	30	mA
	LS / FS Transceiver current consumption	FS reception (1)		3	4.5	mA
$I_{VDDUTMIC}$	PLL, Core and Oscillator current consumption			5.5	9	mA

Note: 1. Including 1 mA due to Pull-up/Pull-down current consumption.

46.12 USB Transceiver Characteristics

Table 46-22. USB Transceiver 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_{lkg}	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 46-4	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	kΩ
R_{PUA}	Bus Pull-up Resistor on Upstream Port (upstream port receiving)		1.425		3.090	kΩ
R_{PD}	Bus Pull-down resistor		14.25		24.8	kΩ

Figure 46-4. USB Data Signal Rise and Fall Times

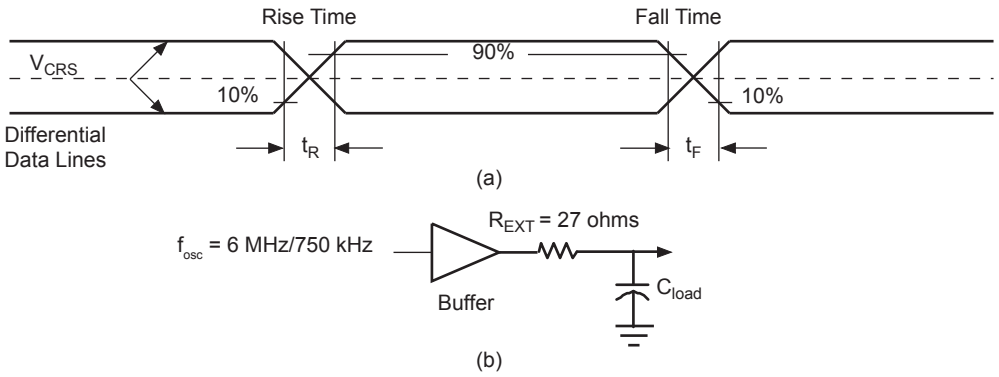


Table 46-23. In Full Speed

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	%

46.13 Analog-to-Digital Converter (ADC)

Table 46-24. Channel Conversion Time and ADC Clock

Parameter	Conditions	Min	Typ	Max	Unit
ADC Clock Frequency	10-bit resolution mode			13.2	MHz
Startup Time	Return from Idle Mode			40	µs
Track and Hold Acquisition Time (TTH)	ADC Clock = 13.2 MHz (1)	0.5			µs
Conversion Time (TCT)	ADC Clock = 13.2 MHz (1) ADC Clock = 5 MHz (1)			1.74 4.6	µs
Throughput Rate	ADC Clock = 13.2 MHz (1) ADC Clock = 5 MHz (1)			440 192	kSPS

Note: 1. The Track-and-Hold Acquisition Time is given by: TTH (ns) = 500 + $(0.12 \times Z_{IN})(\Omega)$

The ADC internal clock is divided by 2 in order to generate a clock with a duty cycle of 75%. So the maximum conversion time is given by:

$$TCT(\mu s) = \frac{23}{f_{clk}}(MHz)$$

The full speed is obtained for an input source impedance of < 50 Ω maximum, or TTH = 500 ns.
In order to make the TSADC work properly, the SHTIM field in TSADCC Mode Register is to be calculated according to this Track and Hold Acquisition Time, also called Sampled and Hold Time.

Table 46-25. External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Unit
ADVREF Input Voltage Range		2.4		VDDANA	V
ADVREF Average Current				600	µA
Current Consumption on VDDANA				600	µA

Table 46-26. Analog Inputs

Parameter	Conditions	Min	Typ	Max	Unit
Input Voltage Range		0		ADVREF	V
Input Peak Current				2.5	mA
Input Capacitance			7	10	pF
Input Impedance			50		Ω

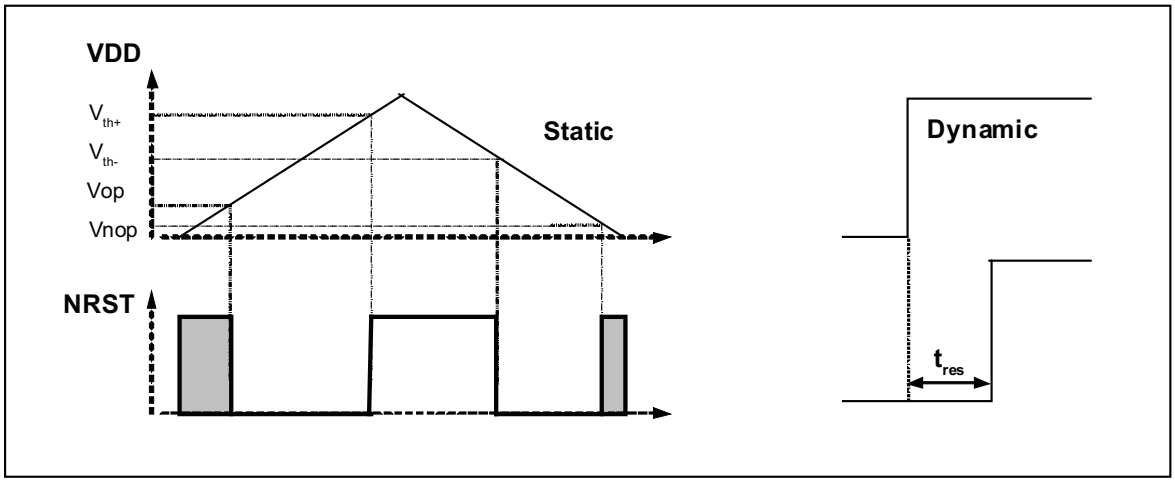
Table 46-27. Transfer Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
Resolution			10		bit
Integral Non-linearity				±2	LSB
Differential Non-linearity	ADC Clock = 13.2 MHz			±2	LSB
	ADC Clock = 5 MHz			±0.9	
Offset Error				±10	mV
Gain Error	ADC Clock = 13.2 MHz			±3	LSB
	ADC Clock = 5 MHz			±2	

46.14 POR Characteristics

A general presentation of Power-On-Reset (POR) characteristics is provided in Figure 46-5.

Figure 46-5. General Presentation of POR Behavior



When a very slow (versus t_{RES}) supply rising slope is applied on POR VDD pin, the reset time becomes negligible and the reset signal is released when V_{DD} rises higher than V_{th+}.

When a very fast (versus t_{RES}) supply rising slope is applied on POR VDD pin, the voltage threshold becomes negligible and the reset signal is released after t_{RES} time. It is the smallest possible reset time.

46.14.1 Core Power Supply POR Characteristics

Table 46-28. Core Power Supply POR Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{th+}	Threshold Voltage Rising	Minimum Slope of +2.0V/30ms	0.5	0.7	0.89	V
V_{th-}	Threshold Voltage Falling	Minimum Slope of +2.0V/30ms	0.4	0.6	0.85	V
t_{RES}	Reset Time		30	70	130	µs
I_{DD}	Current consumption	After t_{RES}		3	7	µA

46.14.2 Backup Power Supply POR Characteristics

Table 46-29. Backup Power Supply POR Characteristics

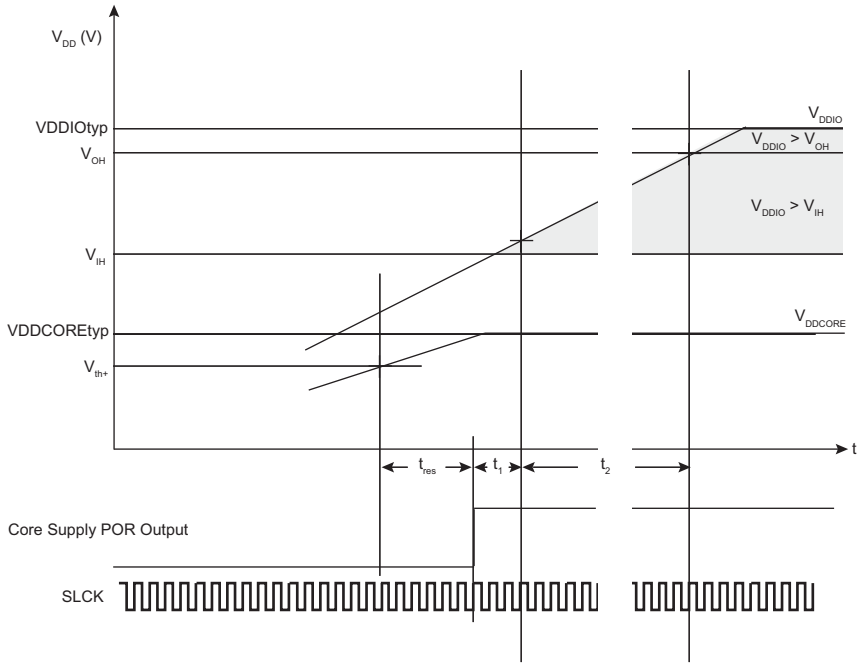
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{th+}	Threshold Voltage Rising	Minimum Slope of +2.0V/30ms	1.42	4.52	1.62	V
V_{th-}	Threshold Voltage Falling	Minimum Slope of +2.0V/30ms	1.35	1.45	1.55	V
t_{RES}	Reset Time	V_{DDBU} is 3.3V	30	80	220	µs
		V_{DDBU} is 1.8V	40	100	330	
I_{DD}	Current consumption	After t_{RES}		6	8.5	µA

46.15 Power Sequence Requirements

The AT91 board design must comply with the power-up guidelines below to guarantee reliable operation of the device. Any deviation from these sequences may prevent the device from booting.

46.15.1 Power-Up Sequence

Figure 46-6. V_{DDCORE} and V_{DDIO} Constraints at Startup



V_{DDCORE} and V_{DDBU} are controlled by internal POR (Power-On-Reset) to guarantee that these power sources reach their target values prior to the release of POR.

- V_{DDIOP} must be ≥ V_{IH} (refer to DC characteristics, Table 46-2, for more details), (t_{RES} + t_1) at the latest, after V_{DDCORE} has reached V_{th+}.
- V_{DDIOM} must reach V_{OH} (refer to DC characteristics, Table 46-2, for more details), (t_{res} + t_1 + t_2) at the latest, after V_{DDCORE} has reached V_{th+}.
  - t_{RES} is a POR characteristic
  - $t_1 = 3 \times t_{SLCK}$
  - $t_2 = 16 \times t_{SLCK}$

The t_{SLCK} min (22 µs) is obtained for the maximum frequency of the internal RC oscillator (44KHz).

  - t_{RES} = 30 µs
  - t_1 = 66 µs
  - t_2 = 352 µs

- V_{DDPLL} is to be established prior to V_{DDCORE} to ensure the PLL is powered once enabled into the ROM code.

As a conclusion, establish V_{DDIOP} and V_{DDIOM} first, then V_{DDPLL}, and V_{DDCORE} last, to ensure a reliable operation of the device.

46.16 SMC Timings

46.16.1 Timing Conditions

SMC Timings are given for MAX corners.

Timings are given assuming a capacitance load on data, control and address pads.

Table 46-30. Capacitance Load

Supply	Corner	
	Max	Min
3.3V	50pF	5 pF
1.8V	30 pF	5 pF

In the following tables, t_{CPMCK} is MCK period.

46.16.2 Timing Extraction

46.16.2.1 Zero Hold Mode Restrictions

Table 46-31. Zero Hold Mode Use Maximum System Clock Frequency (MCK)

Symbol	Parameter	Max		Unit
		VDDIOM supply 1.8V	VDDIOM supply 3.3V	
f_{max}	MCK frequency	66	66	MHz

46.16.2.2 Read Timings

Table 46-32. SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	Parameter	Min		Unit
		VDDIOM supply 1.8V	VDDIOM supply 3.3V	
NO HOLD SETTINGS (nrd hold = 0)				
SMC_1	Data Setup before NRD High	13.6	11.7	ns
SMC_2	Data Hold after NRD High	0	0	ns
HOLD SETTINGS (nrd hold ≠ 0)				
SMC_3	Data Setup before NRD High	10.9	9.0	ns
SMC_4	Data Hold after NRD High	0	0	ns
HOLD or NO HOLD SETTINGS (nrd hold ≠ 0, nrd hold =0)				
SMC_5	NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25 valid before NRD High	(nrd setup + nrd pulse) × t_{CPMCK} - 4.7	(nrd setup + nrd pulse) × t_{CPMCK} - 4.7	ns
SMC_6	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) × t_{CPMCK} - 4.3	(nrd setup + nrd pulse - ncs rd setup) × t_{CPMCK} - 4.4	ns
SMC_7	NRD Pulse Width	nrd pulse × t_{CPMCK} - 3.2	nrd pulse × t_{CPMCK} - 3.3	ns

Table 46-33. SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	Parameter	Min		Unit
		VDDIOM supply 1.8V	VDDIOM supply 3.3V	
NO HOLD SETTINGS (ncs rd hold = 0)				
SMC_8	Data Setup before NCS High	26.9	25.0	ns
SMC_9	Data Hold after NCS High	0	0	ns
HOLD SETTINGS (ncs rd hold ≠ 0)				
SMC_{10}	Data Setup before NCS High	12.3	10.4	ns
SMC_{11}	Data Hold after NCS High	0	0	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} - 18.4	(ncs rd setup + ncs rd pulse) × t_{CPMCK} - 18.4	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup) × t_{CPMCK} - 2.0	(ncs rd setup + ncs rd pulse - nrd setup) × t_{CPMCK} - 2.1	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length × t_{CPMCK} - 4.0	ncs rd pulse length × t_{CPMCK} - 4.0	ns

46.16.2.3 Write Timings

Table 46-34. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	Min		Max		Unit
		1.8V Supply	3.3V Supply	1.8 V Supply	3.3 V Supply	
HOLD or NO HOLD SETTINGS (nwe hold ≠ 0, nwe hold = 0)						
SMC_{15}	Data Out Valid before NWE High	nwe pulse × t_{CPMCK} - 3.9	nwe pulse × t_{CPMCK} - 3.9			ns
SMC_{16}	NWE Pulse Width	nwe pulse × t_{CPMCK} - 3.2	nwe pulse × t_{CPMCK} - 3.2			ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NWE low	nwe setup × t_{CPMCK} - 4.2	nwe setup × t_{CPMCK} - 4.0			ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) × t_{CPMCK} - 4.2	(nwe setup - ncs rd setup + nwe pulse) × t_{CPMCK} - 4.2			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} - 4.8	nwe hold × t_{CPMCK} - 4.0			ns
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold) × t_{CPMCK} - 4.0	(nwe hold - ncs wr hold) × t_{CPMCK} - 3.5			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)	1.9	1.5			ns
SMC_{21b}	Min Period/Max Frequency with No Hold settings	11.4	9.7	87	103	ns/ MHz

Note: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “NWE hold length”.

Table 46-35. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min		Unit
		1.8V Supply	3.3V Supply	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse × t_{CPMCK} - 2.9	ncs wr pulse × t_{CPMCK} - 3.0	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse × t_{CPMCK} - 4.0	ncs wr pulse × t_{CPMCK} - 4.0	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NCS low	ncs wr setup × t_{CPMCK} - 3.6	ncs wr setup × t_{CPMCK} - 3.5	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse) × t_{CPMCK} - 4.6	(ncs wr setup - nwe setup + ncs pulse) × t_{CPMCK} - 4.6	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25, change	ncs wr hold × t_{CPMCK} - 5.4	ncs wr hold × t_{CPMCK} - 4.5	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold) × t_{CPMCK} - 4.2	(ncs wr hold - nwe hold) × t_{CPMCK} - 3.8	ns

Figure 46-7. SMC Timings - NCS Controlled Read and Write

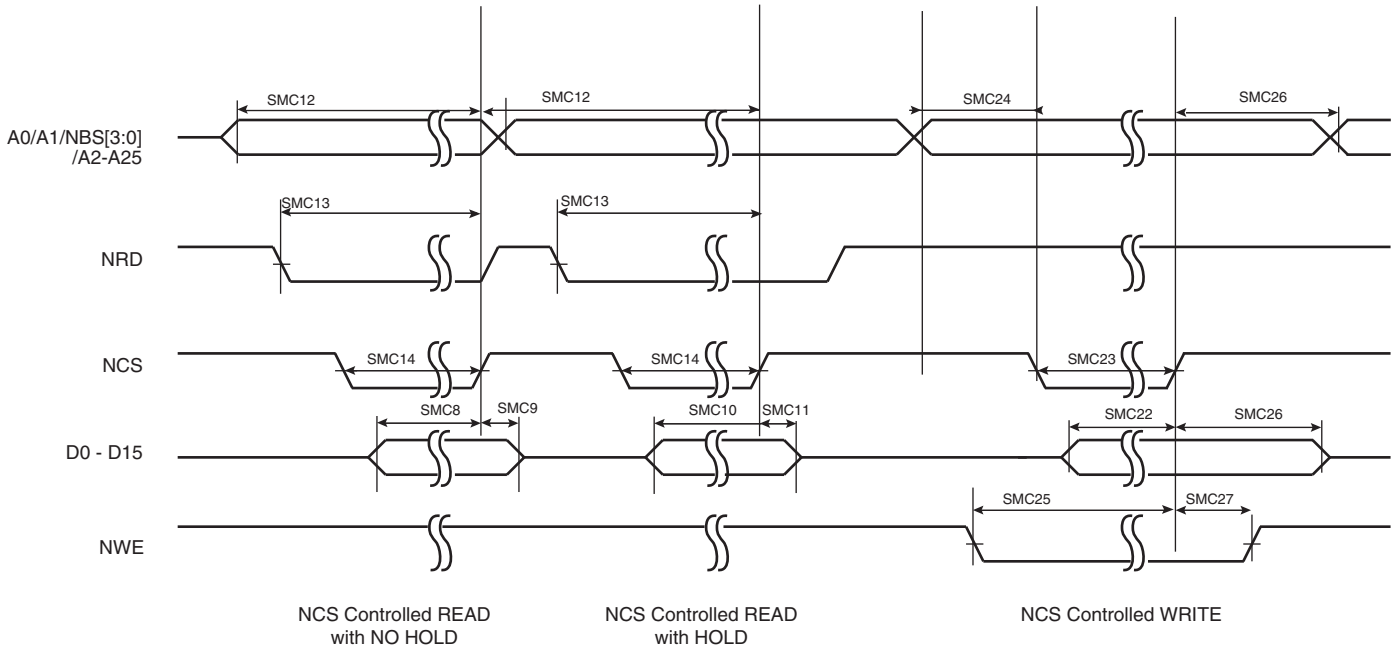
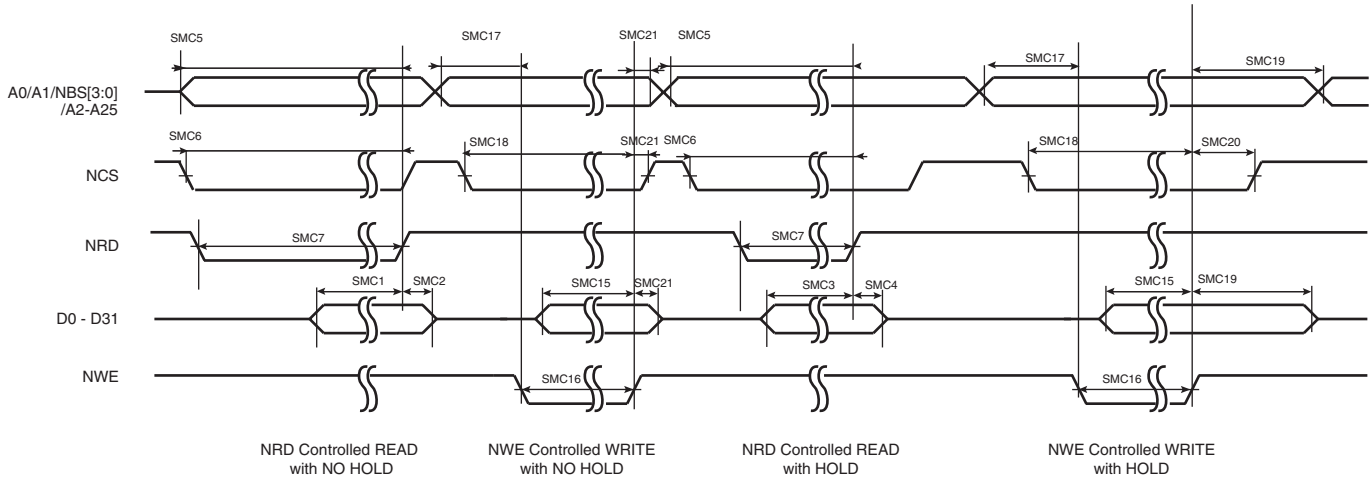


Figure 46-8. SMC Timings - NRD Controlled Read and NWE Controlled Write



46.17 DDRSDRC Timings

The DDRSDRC controller satisfies the timings of standard DDR2, LP-DDR, SDR and LP-SDR modules.

DDR2, LP-DDR and SDR timings are specified by the JEDEC standard.

Supported speed grade limitations:

- DDR2-400 limited at 133 MHz clock frequency (1.8V, 30 pF on data/control, 10 pF on CK/CK#)
- LP-DDR limited at 133 MHz clock frequency (1.8V, 30 pF on data/control, 10 pF on CK)
- SDR-100 (3.3V, 50 pF on data/control, 10 pF on CK)
- SDR-133 (3.3V, 50 pF on data/control, 10 pF on CK)
- LP-SDR-133 (1.8V, 30 pF on data/control, 10 pF on CK)

46.18 Peripheral Timings

46.18.1 SPI

46.18.1.1 Maximum SPI Frequency

The following formulas give maximum SPI frequency in Master read and write modes and in Slave read and write modes.

Master Write Mode

The SPI is only sending data to a slave device such as an LCD, for example. The limit is given by SPI_2 (or SPI_5) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 46.10 “I/Os”), the maximum SPI frequency is the one from the pad.

Master Read Mode

$$f_{SPCK}Max = \frac{1}{SPI_0(\text{or } SPI_3) + t_{valid}}$$

t_{valid} is the slave time response to output data after deleting an SPCK edge. For Atmel SPI DataFlash (AT45DB642D), t_{valid} (or t_v) is 12 ns Max.

This gives $f_{SPCK}Max$ = 39 MHz @ V_{DDIO} = 3.3V.

Slave Read Mode

In slave mode, SPCK is the input clock for the SPI. The max SPCK frequency is given by setup and hold timings SPI_7/SPI_8 (or SPI_{10}/SPI_{11}). Since this gives a frequency well above the pad limit, the limit in slave read mode is given by SPCK pad.

Slave Write Mode

$$f_{SPCK}Max = \frac{1}{SPI_6(\text{or } SPI_9) + t_{setup}}$$

t_{setup} is the setup time from the master before sampling data (12 ns).

This gives $f_{SPCK}Max$ = 39 MHz @ V_{DDIO} = 3.3V.

46.18.1.2 Timing Conditions

Timings are given assuming a capacitance load on MISO, SPCK and MOSI.

Table 46-36. Capacitance Load for MISO, SPCK and MOSI (product dependent)

Supply	Corner	
	Max	Min
3.3V	40 pF	5 pF
1.8V	20 pF	5 pF

46.18.1.3 Timing Extraction

Figure 46-9. SPI Master mode 1 and 2

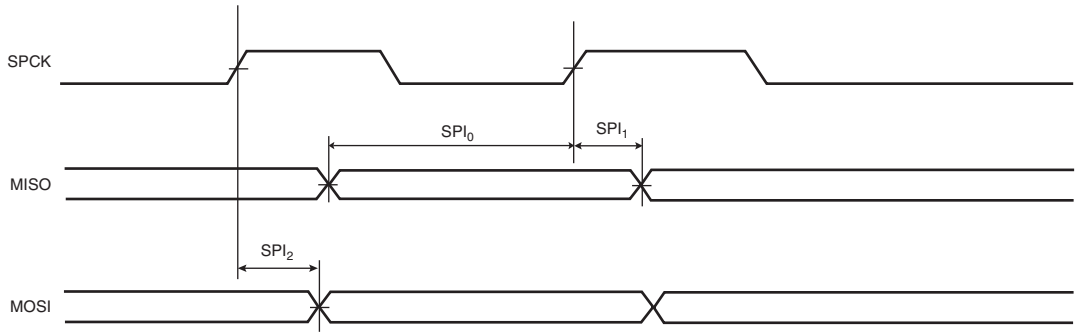


Figure 46-10.SPI Master mode 0 and 3

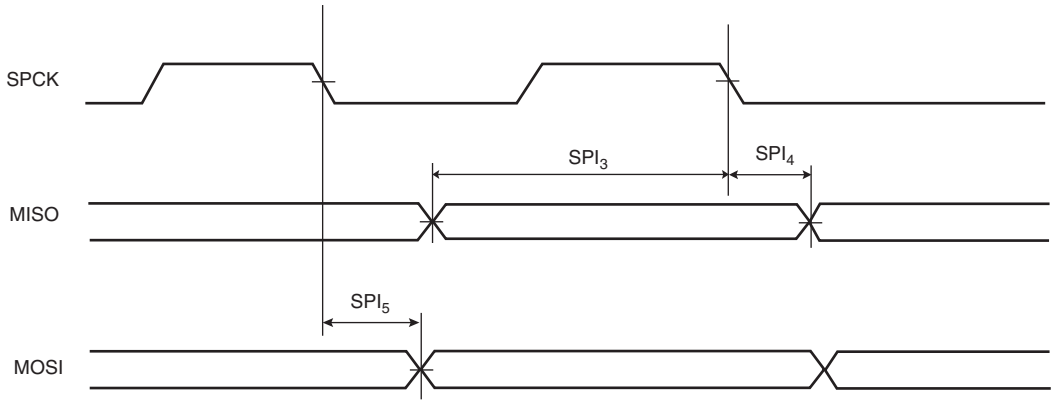


Figure 46-11.SPI Slave mode 0 and 3

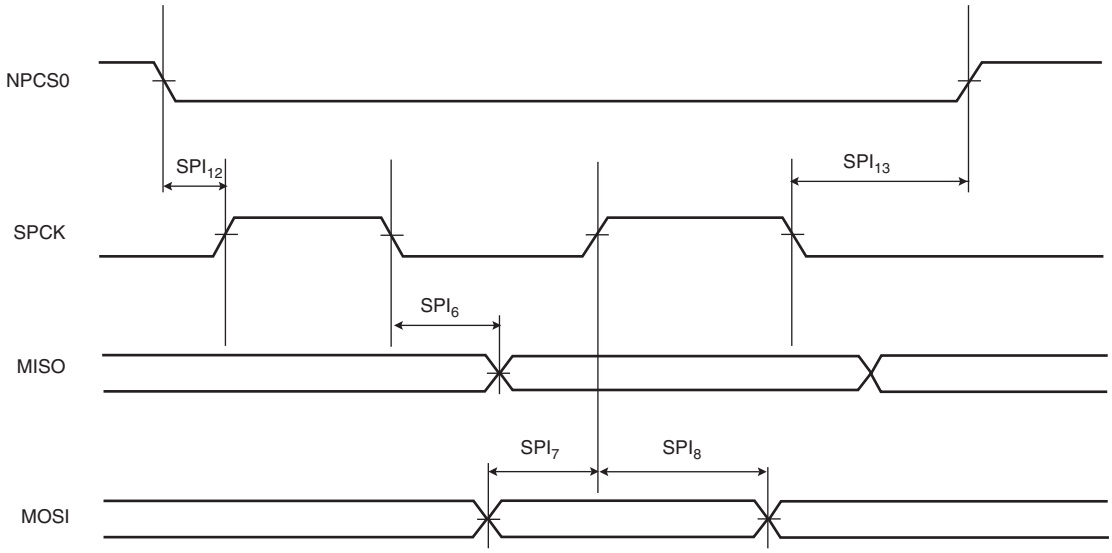


Figure 46-12.SPI Slave mode 1 and 2

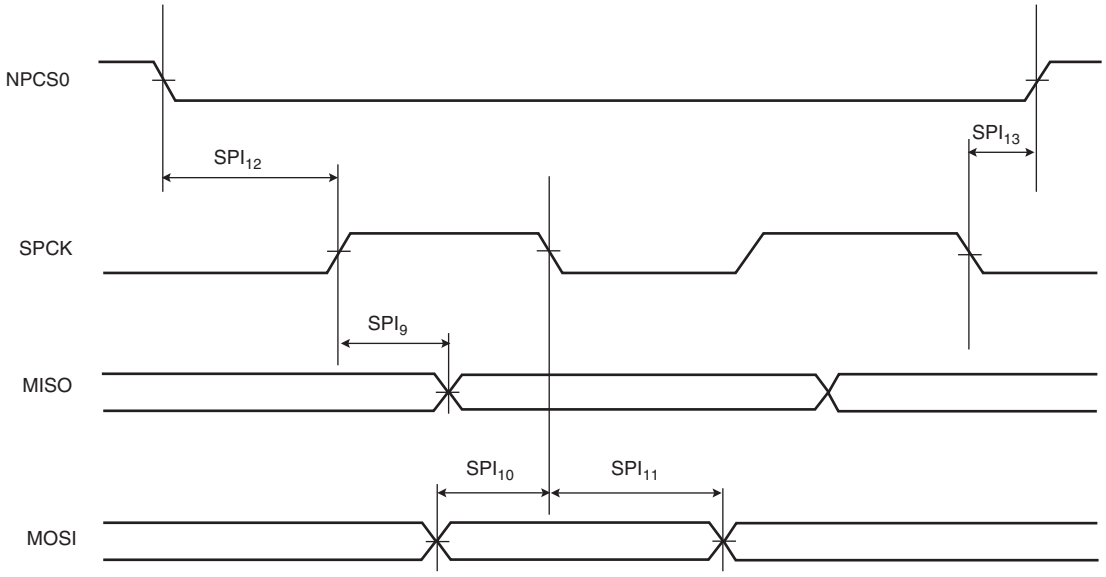


Figure 46-13.SPI Slave mode - NPCS timings

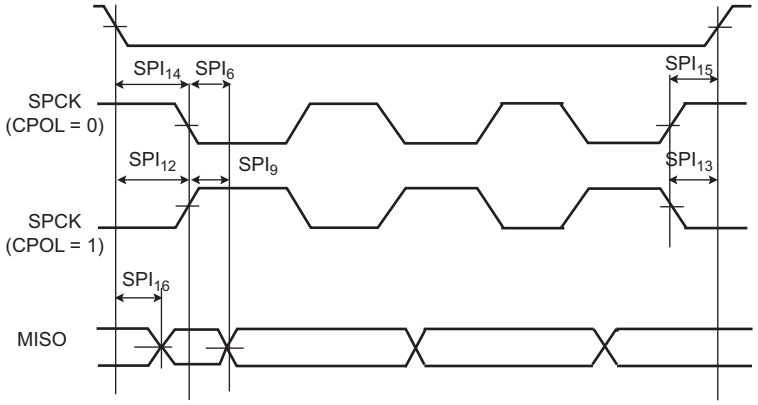


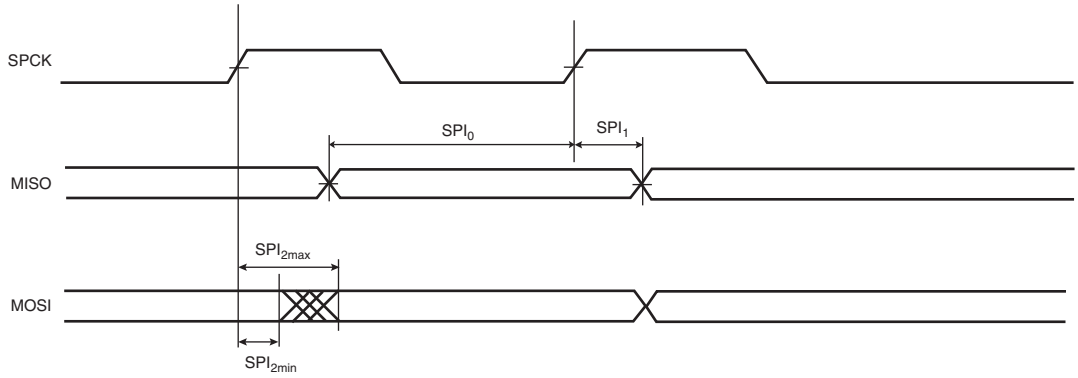
Table 46-37. SPI Timings with 3.3V Peripheral Supply

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_{SPCK}	SPI Clock	Master mode		66	MHz
SPI_0	MISO Setup time before SPCK rises		13.3		ns
SPI_1	MISO Hold time after SPCK rises		0		ns
SPI_2	SPCK rising to MOSI		0	7.4	ns
SPI_3	MISO Setup time before SPCK falls		12.8		ns
SPI_4	MISO Hold time after SPCK falls		0		ns
SPI_5	SPCK falling to MOSI		0	7.6	ns
SPI_6	SPCK falling to MISO	Slave mode	2.9	12.7	ns
SPI_7	MOSI Setup time before SPCK rises		2.0		ns
SPI_8	MOSI Hold time after SPCK rises		0		ns
SPI_9	SPCK rising to MISO		2.7	13.3	ns
SPI_{10}	MOSI Setup time before SPCK falls		1.7		ns
SPI_{11}	MOSI Hold time after SPCK falls		0		ns
SPI_{12}	NPCS0 setup to SPCK rising		3.8		ns
SPI_{13}	NPCS0 hold after SPCK falling		0		ns
SPI_{14}	NPCS0 setup to SPCK falling		3.5		ns
SPI_{15}	NPCS0 hold after SPCK rising		0		ns
SPI_{16}	NPCS0 falling to MISO valid			15.4	ns

Table 46-38. SPI Timings with 1.8V Peripheral Supply

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_{SPCK}	SPI Clock	Master Mode		66	MHz
SPI_0	MISO Setup time before SPCK rises		15.9		ns
SPI_1	MISO Hold time after SPCK rises		0		ns
SPI_2	SPCK rising to MOSI		0	6.7	ns
SPI_3	MISO Setup time before SPCK falls		14.8		ns
SPI_4	MISO Hold time after SPCK falls		0		ns
SPI_5	SPCK falling to MOSI		0	6.8	ns
SPI_6	SPCK falling to MISO	Slave Mode	3.8	16.0	ns
SPI_7	MOSI Setup time before SPCK rises		2.2		ns
SPI_8	MOSI Hold time after SPCK rises		0		ns
SPI_9	SPCK rising to MISO		3.5	15.8	ns
SPI_{10}	MOSI Setup time before SPCK falls		1.8		ns
SPI_{11}	MOSI Hold time after SPCK falls		0.2		ns
SPI_{12}	NPCS0 setup to SPCK rising		4.0		ns
SPI_{13}	NPCS0 hold after SPCK falling		0		ns
SPI_{14}	NPCS0 setup to SPCK falling		3.6		ns
SPI_{15}	NPCS0 hold after SPCK rising		0		ns
SPI_{16}	NPCS0 falling to MISO valid			17.9	ns

Figure 46-14.Min and Max access time for SPI output signal



46.18.2 SSC

46.18.2.1 Timing conditions

Timings are given assuming a capacitance load as defined in Table 46-39.

Table 46-39. Capacitance Load

Supply	Corner	
	Max	Min
3.3V	30 pF	5 pF
1.8V	20 pF	5 pF

46.18.2.2 Timing Extraction

Figure 46-15.SSC Transmitter, TK and TF in Output

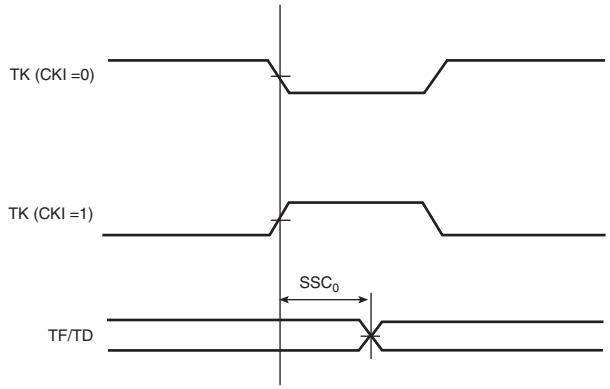


Figure 46-16.SSC Transmitter, TK in Input and TF in Output

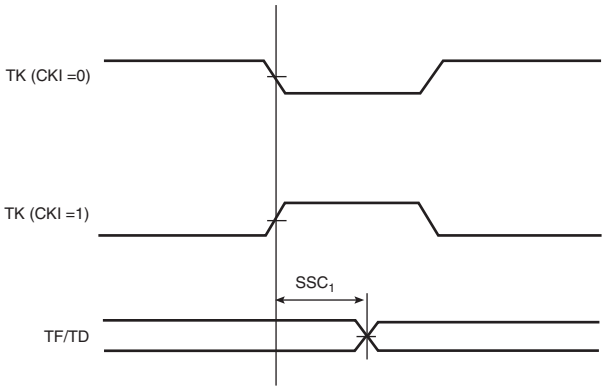


Figure 46-17.SSC Transmitter, TK in Output and TF in Input

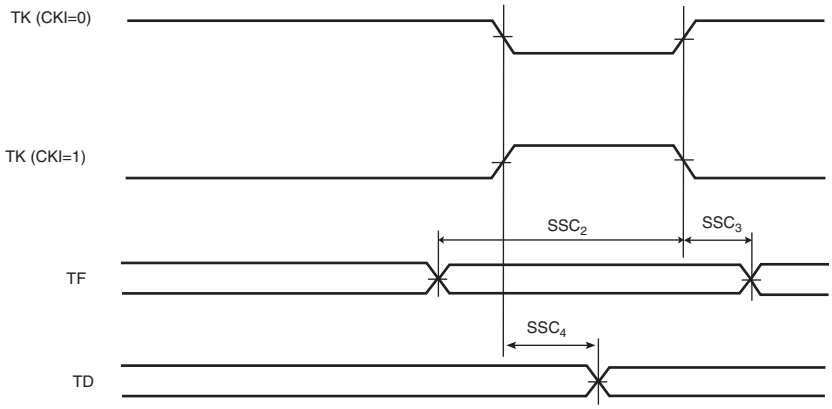


Figure 46-18.SSC Transmitter, TK and TF in Input

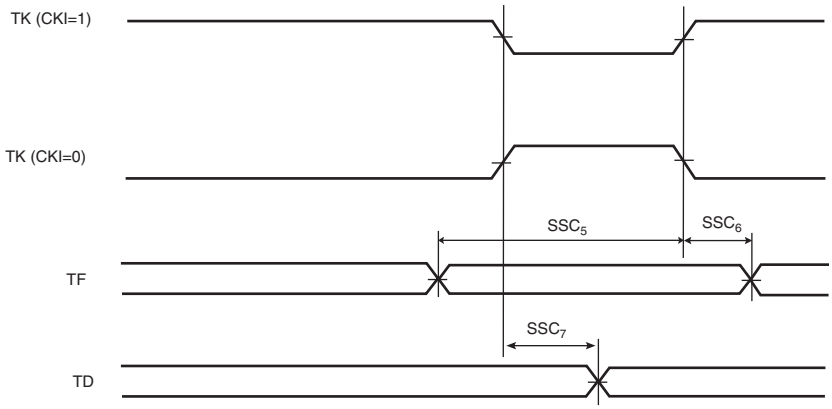


Figure 46-19.SSC Receiver RK and RF in Input

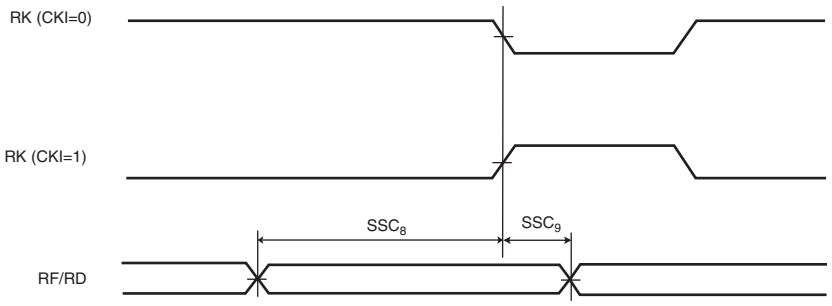


Figure 46-20.SSC Receiver, RK in Input and RF in Output

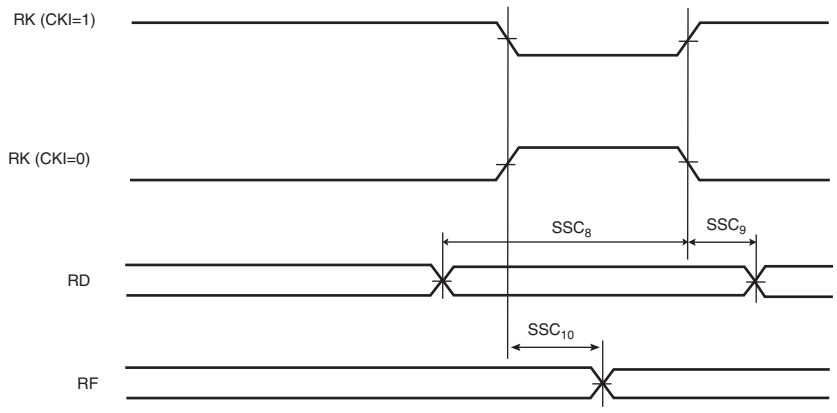


Figure 46-21.SSC Receiver, RK and RF in Output

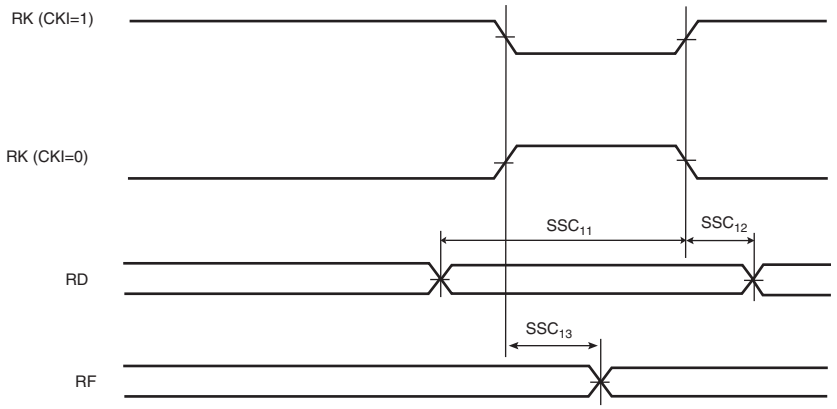


Figure 46-22.SSC Receiver, RK in Ouput and RF in Input

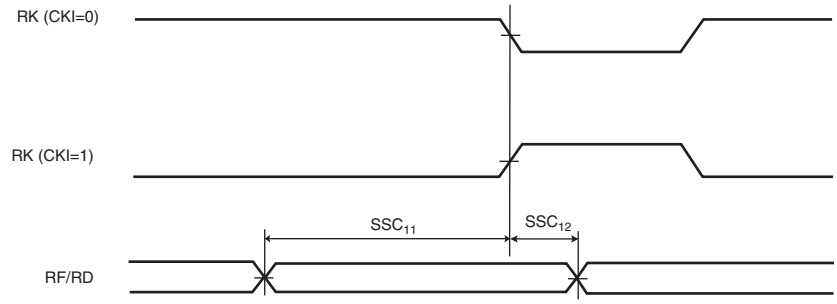
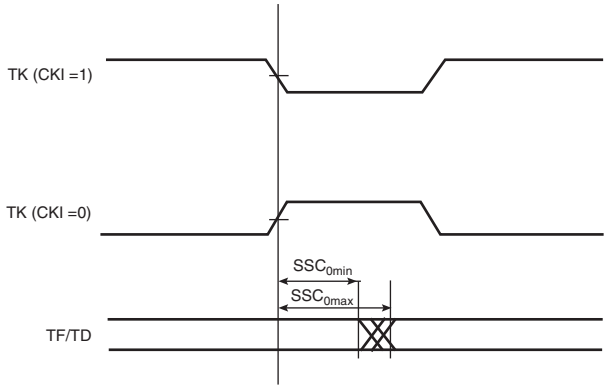


Table 46-40. SSC Timings

Symbol	Parameter	Conditions	Min	Max	Unit
		Transmitter			
SSC_0	TK edge to TF/TD (TK output, TF output)	1.8V domain[3] 3.3V domain[4]	-5.6 -4.6	5.8 4.9	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	1.8V domain[3] 3.3V domain[4]	3.0 2.3	15.7 11.4	ns
SSC_2	TF setup time before TK edge (TK output)	1.8V domain[3] 3.3V domain[4]	14.0 9.9		ns
SSC_3	TF hold time after TK edge (TK output)	1.8V domain[3] 3.3V domain[4]	0 0		ns
SSC_4[1]	TK edge to TD (TK output, TF input)	1.8V domain[3] 3.3V domain[4]	-5.6 ($+2 \times t_{CPMCK}$)[1][4] -4.6 ($+2 \times t_{CPMCK}$)[1][4]	5.7 ($+2 \times t_{CPMCK}$)[1][4] 4.7 ($+2 \times t_{CPMCK}$)[1][4]	ns
SSC_5	TF setup time before TK edge (TK input)	1.8V domain[3] 3.3V domain[4]	0		ns
SSC_6	TF hold time after TK edge (TK input)	1.8V domain[3] 3.3V domain[4]	t_{CPMCK}		ns
SSC_7[1]	TK edge to TD (TK input, TF input)	1.8V domain[3] 3.3V domain[4]	3.0 ($+3 \times t_{CPMCK}$)[1][4] 2.3 ($+3 \times t_{CPMCK}$)[1][4]	15.5($+3 \times t_{CPMCK}$)[1][4] 11.1($+3 \times t_{CPMCK}$)[1][4]	ns
		Receiver			
SSC_8	RF/RD setup time before RK edge (RK input)	1.8V domain[3] 3.3V domain[4]	0		ns
SSC_9	RF/RD hold time after RK edge (RK input)	1.8V domain[3] 3.3V domain[4]	t_{CPMCK}		ns
SSC_{10}	RK edge to RF (RK input)	1.8V domain[3] 3.3V domain[4]	2.6 2.0	15.2 10.9	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	1.8V domain[3] 3.3V domain[4]	14.1 - t_{CPMCK} 10.0 - t_{CPMCK}		ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	1.8V domain[3] 3.3V domain[4]	t_{CPMCK} - 2.5 t_{CPMCK} - 1.8		ns
SSC_{13}	RK edge to RF (RK output)	1.8V domain[3] 3.3V domain[4]	-5.9 -4.9	5.2 4.3	ns

Notes:
1. Timings SSC4 and SSC7 depend on the start condition. When STTDLY = 0 (Receive start delay) and START = 4, or 5 or 7 (Receive Start Selection), two periods of the MCK must be added to timings.
2. For output signals (TF, TD, RF), minimum and maximum access times are defined. The minimum access time is the time between the TK (or RK) edge and the signal change. The maximum access time is the time between the TK edge and the signal stabilization. Figure 46-23 illustrates minimum and maximum accesses for SSC0. The same applies to SSC1, SSC4, and SSC7, SSC10 and SSC13.
3. 1.8V domain: V_{DDIO} from 1.65V to 1.95V, maximum external capacitor = 20 pF.
4. 3.3V domain: V_{DDIO} from 3.0V to 3.6V, maximum external capacitor = 30 pF.

Figure 46-23.Min and Max access time of output signals



46.18.3 HSMCI

The High Speed MultiMedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

46.18.4 EMAC

46.18.4.1 Timing conditions

Table 46-41. Capacitance Load on Data, Clock Pads

Supply	Corner		
	Max	Typical Voltage High Temperature	Min
3.3V	20 pF	20 pF	0 pF
1.8V	20 pF	20 pF	0 pF

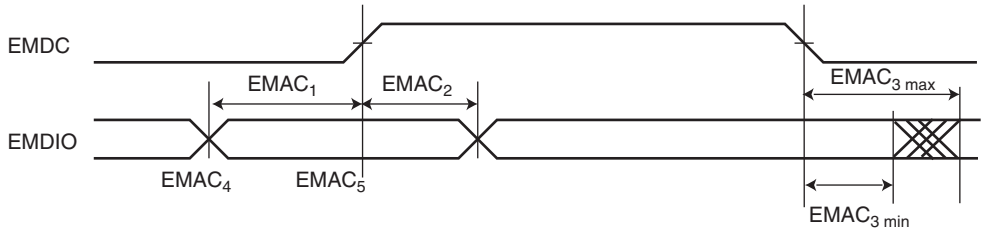
46.18.4.2 Timing constraints

Table 46-42. EMAC Signals Relative to EMDC

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_1$	Setup for EMDIO from EMDC rising	10	
$EMAC_2$	Hold for EMDIO from EMDC rising	10	
$EMAC_3$	EMDIO toggling from EMDC rising	0 (1)	300 (1)

Note: 1. For EMAC output signals, minimum and maximum access time are defined. The minimum access time is the time between the EDMC rising edge and the signal change. The maximum access timing is the time between the EDMC rising edge and the signal stabilizes.. Figure 46-24 illustrates minimum and maximum accesses for EMAC3.

Figure 46-24.Min and Max access time of EMAC output signals



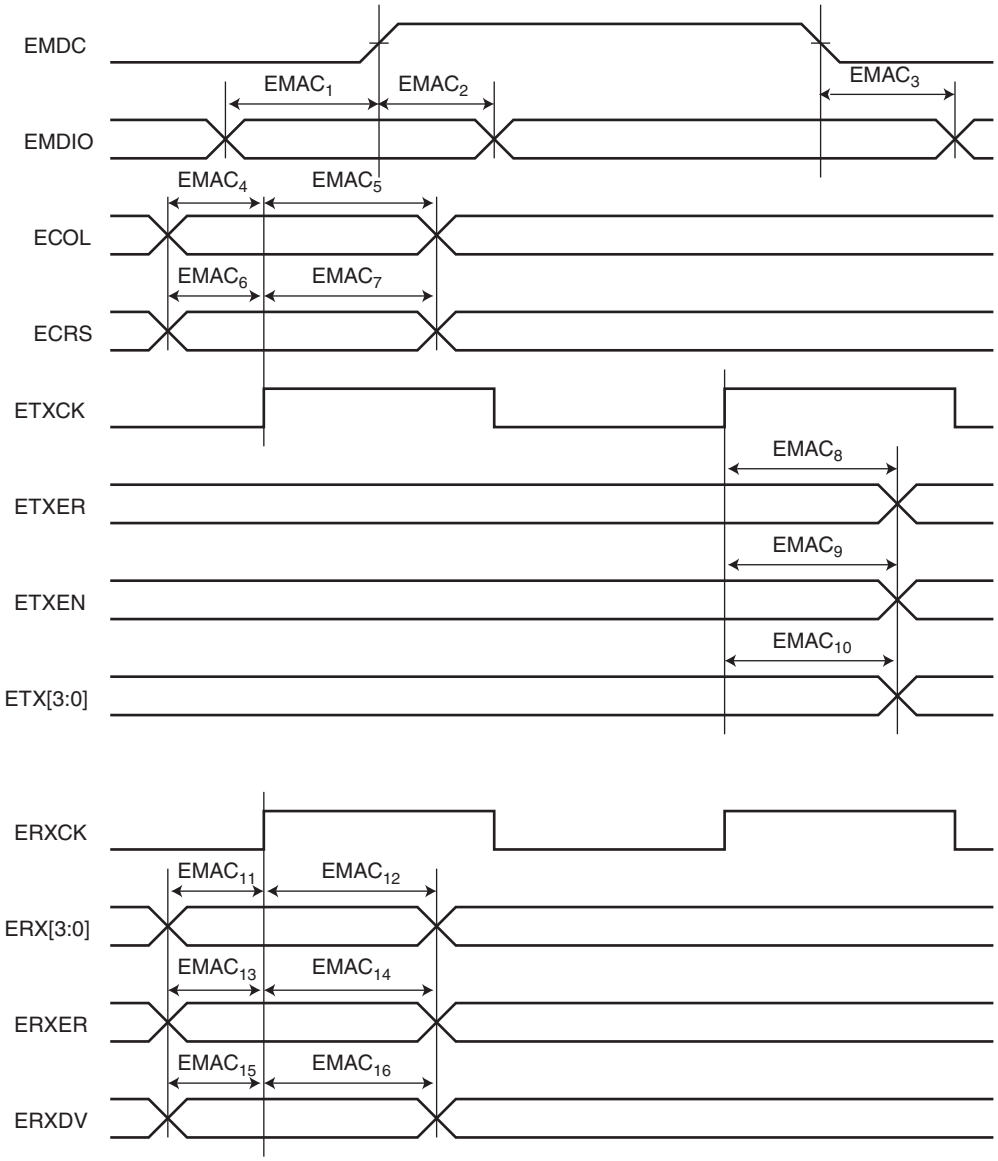
46.18.4.3 MII Mode

Table 46-43. EMAC MII Timings

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_4$	Setup for ECOL from ETXCK rising	10	
$EMAC_5$	Hold for ECOL from ETXCK rising	10	
$EMAC_6$	Setup for ECRS from ETXCK rising	10	
$EMAC_7$	Hold for ECRS from ETXCK rising	10	
$EMAC_8$	ETXER toggling from ETXCK rising	10(2)	25(2)
$EMAC_9$	ETXEN toggling from ETXCK rising	10(2)	25(2)
$EMAC_{10}$	ETX toggling from ETXCK rising	10(2)	25(2)
$EMAC_{11}$	Setup for ERX from ERXCK	10	
$EMAC_{12}$	Hold for ERX from ERXCK	10	
$EMAC_{13}$	Setup for ERXER from ERXCK	10	
$EMAC_{14}$	Hold for ERXER from ERXCK	10	
$EMAC_{15}$	Setup for ERXDV from ERXCK	10	
$EMAC_{16}$	Hold for ERXDV from ERXCK	10	

Notes: 1. VDDIO from 3.0V to 3.6V, maximum external capacitor = 20 pF
2. See Note (4) of Table 46-42.

Figure 46-25.EMAC MII Mode Signals



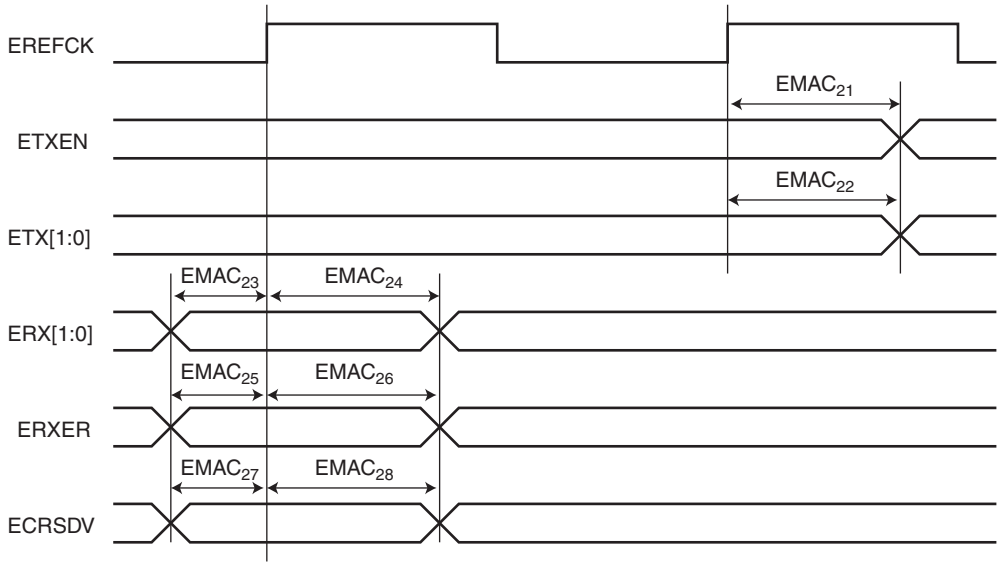
46.18.4.4 RMII Mode

Table 46-44. EMAC RMII Timings

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_{21}$	ETXEN toggling from EREFCK rising	2 (1)	16 (1)
$EMAC_{22}$	ETX toggling from EREFCK rising	2 (1)	16 (1)
$EMAC_{23}$	Setup for ERX from EREFCK rising	4	
$EMAC_{24}$	Hold for ERX from EREFCK rising	2	
$EMAC_{25}$	Setup for ERXER from EREFCK rising	4	
$EMAC_{26}$	Hold for ERXER from EREFCK rising	2	
$EMAC_{27}$	Setup for ECRSDV from EREFCK rising	4	
$EMAC_{28}$	Hold for ECRSDV from EREFCK rising	2	

Notes: 1. See Note (4) of Table 46-42.

Figure 46-26.EMAC RMII Mode Signals



46.18.5 USART in SPI Mode Timings

46.18.5.1 Timing conditions

Timings are given assuming a capacitance load as defined in Table 46-39.

Table 46-45. Capacitance Load

Supply	Corner	
	Max	Min
3.3V	40 pF	5 pF
1.8V	20 pF	5 pF

46.18.5.2 Timing extraction

Figure 46-27.USART SPI Master Mode

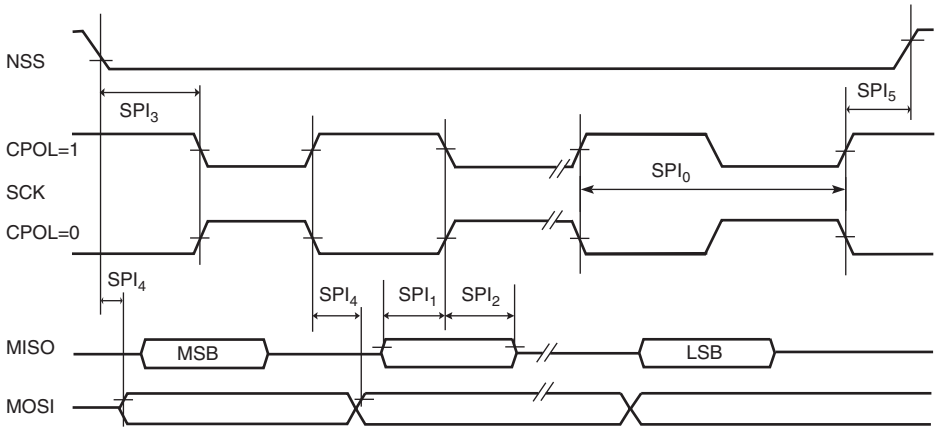


Figure 46-28.USART SPI Slave mode: (Mode 1 or 2)

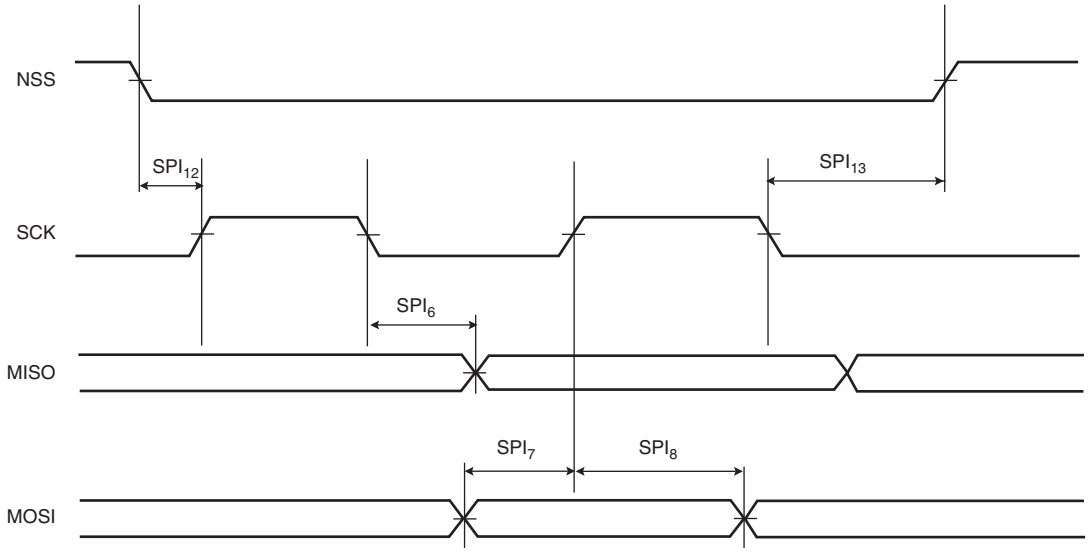


Figure 46-29.USART SPI Slave mode: (Mode 0 or 3)

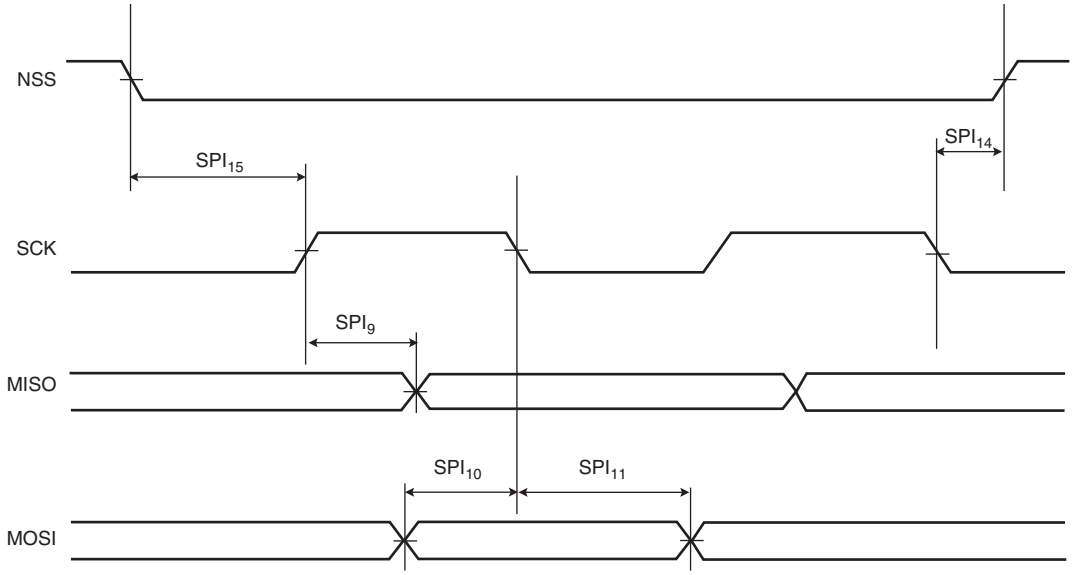


Table 46-46. USART SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	SCK Period	1.8V domain[1] 3.3V domain[2]	MCK/6		ns
SPI_1	Input Data Setup Time	1.8V domain[1] 3.3V domain[2]	0.5 × MCK + 4.1 0.5 × MCK + 3.8		ns
SPI_2	Input Data Hold Time	1.8V domain[1] 3.3V domain[2]	1.5 × MCK + 0.9 1.5 × MCK + 1.1		ns
SPI_3	Chip Select Active to Serial Clock	1.8V domain[1] 3.3V domain[2]	1.5 × SCK - 2.0 1.5 × SCK - 2.6		ns
SPI_4	Output Data Setup Time	1.8V domain[1] 3.3V domain[2]	0 0	7.6 8.0	ns
SPI_5	Serial Clock to Chip Select Inactive	1.8V domain[1] 3.3V domain[2]	1 × SCK - 6.7 1 × SCK - 7.5		ns
Slave Mode					
SPI_6	SCK falling to MISO	1.8V domain[1] 3.3V domain[2]	3.7 2.9	19.9 16.9	ns
SPI_7	MOSI Setup time before SCK rises	1.8V domain[1] 3.3V domain[2]	2 × MCK + 3.4 2 × MCK + 3.1		ns
SPI_8	MOSI Hold time after SCK rises	1.8V domain[1] 3.3V domain[2]	1.6 1.4		ns
SPI_9	SCK rising to MISO	1.8V domain[1] 3.3V domain[2]	3.4 2.7	19.4 16.5	ns
SPI_{10}	MOSI Setup time before SCK falls	1.8V domain[1] 3.3V domain[2]	2 × MCK + 2.9 2 × MCK + 2.8		ns
SPI_{11}	MOSI Hold time after SCK falls	1.8V domain[1] 3.3V domain[2]	2.1 1.8		ns
SPI_{12}	NPCS0 setup to SCK rising	1.8V domain[1] 3.3V domain[2]	2.5 × MCK + 1.4 2.5 × MCK + 1.2		ns
SPI_{13}	NPCS0 hold after SCK falling	1.8V domain[1] 3.3V domain[2]	1.5 × MCK + 2.5 1.5 × MCK + 2.2		ns
SPI_{14}	NPCS0 setup to SCK falling	1.8V domain[1] 3.3V domain[2]	2.5 × MCK + 0.9 2.5 × MCK + 0.8		ns
SPI_{15}	NPCS0 hold after SCK rising	1.8V domain[1] 3.3V domain[2]	1.5 × MCK + 2.1 1.5 × MCK + 1.9		ns

Notes: 1. 1.8V domain: V_{DDIO} from 1.65V to 1.95V, maximum external capacitor = 20pF
2. 3.3V domain: V_{DDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF.

46.19 Two-wire Interface Characteristics

Table 46-47 describes the requirements for devices connected to the Two-wire Serial Bus.

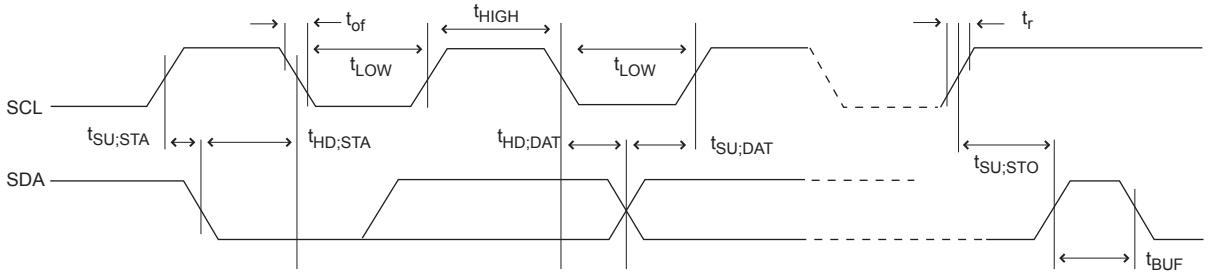
For timing symbols, please refer to Figure 46-30.

Table 46-47. Two-wire Serial Bus Requirements

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IL}	Input Low-voltage	—	-0.3	$0.3 \times V_{DDIO}$	V
V_{IH}	Input High-voltage	—	$0.7 \times V_{DDIO}$	V_{CC} + 0.3	V
V_{HYS}	Hysteresis of Schmitt Trigger Inputs	—	0.150	–	V
V_{OL}	Output Low-voltage	3 mA sink current	—	0.4	V
t_R	Rise Time for both TWD and TWCK		$20 + 0.1C_b$ (1)(2)	300	ns
t_{OF}	Output Fall Time from V_{IHmin} to V_{ILmax}	10 pF < C_b < 400 pF (Figure 46-30)	$20 + 0.1C_b$ (1)(2)	250	ns
C_i (1)	Capacitance for each I/O pin	—	—	10	pF
f_{TWCK}	TWCK Clock Frequency	—	0	400	kHz
Rp	Value of Pull-up Resistor	$f_{TWCK} \le 100$ kHz	$(V_{DDIO} - 0.4V) \div 3mA$	$1000ns \div C_b$	Ω
		$f_{TWCK} > 100$ kHz	$(V_{DDIO} - 0.4V) \div 3mA$	$300ns \div C_b$	Ω
t_{LOW}	Low Period of the TWCK Clock	$f_{TWCK} \le 100$ kHz	(3)	—	µs
		$f_{TWCK} > 100$ kHz	(3)	—	µs
t_{HIGH}	High Period of the TWCK Clock	$f_{TWCK} \le 100$ kHz	(4)	—	µs
		$f_{TWCK} > 100$ kHz	(4)	—	µs
$t_{HD;STA}$	Hold Time (repeated) START condition	$f_{TWCK} \le 100$ kHz	t_{HIGH}	—	µs
		$f_{TWCK} > 100$ kHz	t_{HIGH}	—	µs
$t_{SU;STA}$	Set-up Time for a repeated START condition	$f_{TWCK} \le 100$ kHz	t_{HIGH}	—	µs
		$f_{TWCK} > 100$ kHz	t_{HIGH}	—	µs
$t_{HD;DAT}$	Data Hold Time	$f_{TWCK} \le 100$ kHz	0	$3 \times t_{CP_MCK}$ (5)	µs
		$f_{TWCK} > 100$ kHz	0	$3 \times t_{CP_MCK}$ (5)	µs
$t_{SU;DAT}$	Data Setup Time	$f_{TWCK} \le 100$ kHz	$t_{LOW} - 3 \times t_{CP_MCK}$ (5)	—	ns
		$f_{TWCK} > 100$ kHz	$t_{LOW} - 3 \times t_{CP_MCK}$ (5)	—	ns
$t_{SU;STO}$	Setup Time for STOP condition	$f_{TWCK} \le 100$ kHz	t_{HIGH}	—	µs
		$f_{TWCK} > 100$ kHz	t_{HIGH}	—	µs
$t_{HD;STA}$	Bus free time between a STOP and START condition	$f_{TWCK} \le 100$ kHz	t_{HIGH}	—	µs
		$f_{TWCK} > 100$ kHz	t_{HIGH}	—	µs

Notes:
1. Required only for $f_{TWCK} > 100$ kHz.
2. C_b = capacitance of one bus line in pF. Per I2C Standard, C_b Max = 400 pF
3. The TWCK low period is defined as follows: $t_{low} = ((CLDIV \times 2^{CKDIV}) + 4) \times t_{MCK}$
4. The TWCK high period is defined as follows: $t_{high} = ((CHDIV \times 2^{CKDIV}) + 4 \times t_{MCK}$
5. t_{CP_MCK} = MCK bus period.

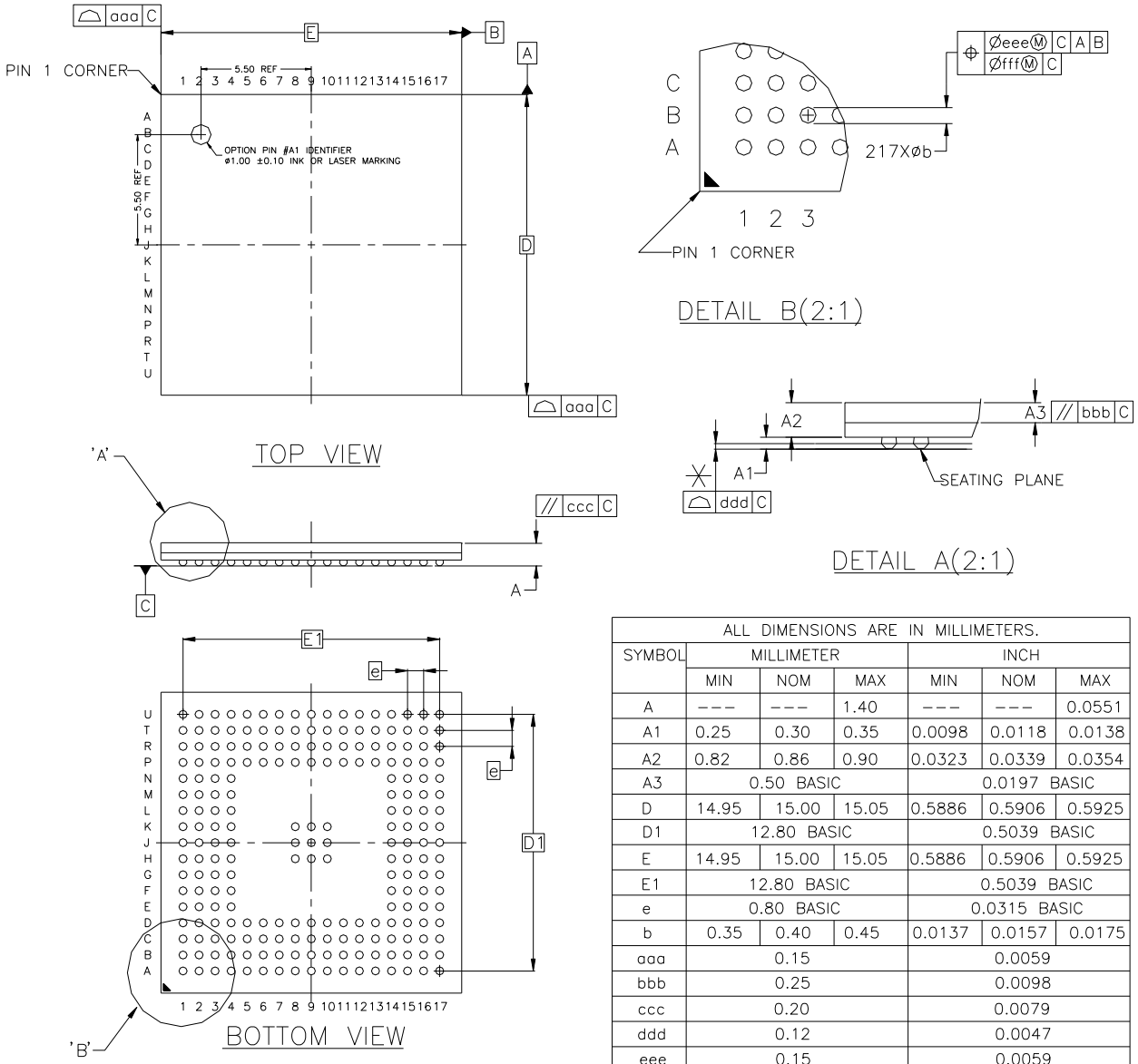
Figure 46-30.Two-wire Serial Bus Timing



47. Mechanical Overview

47.1 217-ball BGA Package

Figure 47-1. 217-ball BGA Package Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.						
SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.40	---	---	0.0551
A1	0.25	0.30	0.35	0.0098	0.0118	0.0138
A2	0.82	0.86	0.90	0.0323	0.0339	0.0354
A3	0.50 BASIC			0.0197 BASIC		
D	14.95	15.00	15.05	0.5886	0.5906	0.5925
D1	12.80 BASIC			0.5039 BASIC		
E	14.95	15.00	15.05	0.5886	0.5906	0.5925
E1	12.80 BASIC			0.5039 BASIC		
e	0.80 BASIC			0.0315 BASIC		
b	0.35	0.40	0.45	0.0137	0.0157	0.0175
aaa	0.15			0.0059		
bbb	0.25			0.0098		
ccc	0.20			0.0079		
ddd	0.12			0.0047		
eee	0.15			0.0059		
fff	0.08			0.0031		

Table 47-2. 217-ball BGA Package Characteristics

Moisture Sensitivity Level	3

Table 47-3. Package Reference

JEDEC Drawing Reference	MO-205
JESD97 Classification	e1

Table 47-1. Device and 217-ball BGA Package Maximum Weight

450	mg

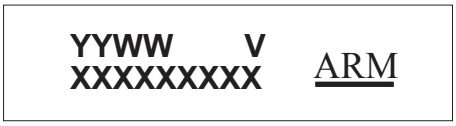
Table 47-4. Package Information

Ball Land	0.43 mm ± 0.05
Solder Mask Opening	0.30 mm ± 0.05

47.2 Marking

All devices are marked with the Atmel logo and the ordering code.

Additional marking may be in one of the following formats:



where

- “YY”: manufactory year
- “WW”: manufactory week
- “V”: revision
- “XXXXXXXXX”: lot number

48. SAM9X25 Ordering Information

Table 48-1. SAM9X25 Ordering Information

Ordering Code	Package	Package Type	Temperature Operating Range
AT91SAM9X25-CU	BGA217	Green	Industrial -40°C to 85°C

49. SAM9X25 Errata

49.1 External Bus Interface (EBI)

49.1.1 EBI: Data lines are Hi-Z after reset

Data lines are Hi-Z after reset. This does not affect boot capabilities neither on NOR nor on NAND memories.

Problem Fix/Workaround

None.

49.2 Reset Controller (RSTC)

49.2.1 RSTC: Reset during SDRAM Accesses

When a Reset occurs (user reset, software reset) the SDRAM clock is turned off. Inopportunately, if this occurs at the same time as a SDRAM read access, the SDRAM maintains the data until the restart of the SDRAM clock.

This leads to a data bus conflict and affects adversely the boot memories connected on the EBI:

- NAND Flash boot functionality, if the system boots out of the internal ROM.
- NOR Flash boot, if the system boots on an external memory connected on the EBI CS0.

Problem Fix/Workaround

Two workarounds are available:

1. Boot from Serial Flash or Data Flash on SPI.
2. Connect the NAND Flash on D16-D23 and set NFD0_ON_D16 to 1 in the CCFG_EBICSA register.

Warning! Due to databus sharing, workaround 2 prohibits connecting another device on the EBI, even if VDDNF equals VDDIOM.

49.2.2 Static Memory Controller (SMC)

49.2.3 SMC: SMC DELAY I/O Registers are write-only

Contrary to what is stated in the datasheet, the SMC DELAY I/O Registers are Write-only.

Problem Fix/Workaround

None.

49.3 USB High Speed Host Port (UHPHS) and Device Port (UDPHS)

49.3.1 UHPHS/UDPHS: Bad Lock of the USB High speed transceiver DLL

The DLL used to oversample the incoming bitstream may not lock in the correct phase, leading to a bad reception of the incoming packets.

This issue may occur after the USB device resumes from the Suspend mode.

The DLL is used only in the High Speed mode, meaning the Full Speed mode is not impacted by this issue.

This issue may occur on the USB device after a reset leading to a SAM-BA connection issue.

Problem Fix/Workaround:

To prevent a SAM-BA execution issue, the USB device must be connected via a USB Full Speed hub to the PC.

At application level, the DLL can be re-initialized in the correct state by toggling the BIASEN bit (high -> low -> high) when resuming from the Suspend mode.

The BIASEN bit is located in the CKGR_UCKR register in PMC user interface.

The function below can be used to generate the pulse on the bias signal.

```
void generate_pulse_bias(void)
{
unsigned int * pckgr_uckr = (unsigned int *) 0xFFFFFC1C;
* pckgr_uckr &= ~AT91_PMC_BIASEN;
* pckgr_uckr |= AT91_PMC_BIASEN;
}
```

In the USB device driver, the generate_pulse_bias function must be implemented in the “USB end of reset” and “USB end of resume” interrupts.

49.4 Timer Counter (TC)

49.4.1 TC: The TIOA5 signal is not well connected

The TIOA5 enable signal is not well connected internally, it is shared with the TIOB5 enable signal.

TIOB5 is working normally.

TIOA5 is working normally in Capture Mode.

Waveform Mode is not available for TIOA5 if the TC_CMR.ETRGEDG bit is set to 1, 2 or 3.

Problem Fix/Workaround

None.

49.5 Boot Strategy

49.5.1 NAND Flash Boot Detection using ONFI parameters does not work

During NAND Flash initialization, the ONFI parameters detection may not work correctly.

This can lead to an incorrect configuration of ECC settings, reading wrong data from the NAND Flash memory, and the unability to boot from this memory.

Problem Fix/Workaround

When programming the bootable program in the NAND Flash, always use the header method, with any NAND Flash memory, ONFI compliant or not.

49.6 Real Time Clock (RTC)

49.6.1 RTC: Interrupt Mask Register cannot be used

Interrupt Mask Register read always returns 0.

Problem Fix/Workaround

None.

Revision History

In the tables that follow, the most recent version of the document appears first.

Doc. Rev. 11054E	Comments
	General editorial and formatting changes throughout document Updated first line of document title on page 1 (was “AT91SAM ARM-based Embedded MPU”; is “ARM-based Embedded MPU”)
	Section 2. “Block Diagram” Figure 2-1 “SAM9X25 Block Diagram”: flipped diagram right for ease of viewing
	Section 4. “Package and Pinout” Table 4-2 “SAM9X25 I/O Type Assignment and Frequency”: - GPIO: replaced “All PIO lines except the following” with “All PIO lines except GPIO_CLK, GPIO_CLK2, and GPIO_ANA” - EBI: replaced “All Data lines (Input/output) except the following” with “All data lines (Input/output)” - EBI_O: replaced “All Address and control lines (output only) except the following” with “All address and control lines (output only) except EBI_CLK” Table 4-3 “Pin Description BGA217”: removed “PU” reset state for SHDN signal
	Section 5. “Power Considerations” Table 5-1 “SAM9X25 Power Supplies”: in VDDNF “Powers” description, replaced instance of “D16-D32” with “D16–D31”
	Section 6. “Memories” Figure 6-1 “SAM9X25 Memory Mapping”: replaced “SCKCR” with “SCKC_CR”; replaced “BSCR” with “BSC_CR”
	Section 7. “System Controller” Figure 7-1 “SAM9X25 System Controller Block Diagram”: replaced “SCKCR” with “SCKC_CR”; replaced “BSCR” with “BSC_CR” Section 7.2 “Backup Section”: replaced bullet “Slow Clock Control Register (SCKCR)” with “Slow Clock Controller Configuration Register (SCKC_CR)”; corrected instance of “BSCR” to “BSC_CR”

Doc. Rev. 11054E	Comments
	Section 25. “Bus Matrix (MATRIX)”
	Updated Table 25-1 “List of Bus Matrix Masters”:
	- Master 9 (was ISI DMA; is Reserved)
	- Master 10 (was EMAC DMA; is EMAC0 DMA)
	- added Master 11 (EMAC1 DMA)
	Section 25.2.2 “Matrix Slaves”: in first sentence, replaced “manages 9 slaves” with “manages 10 slaves”
	Updated Table 25-3 “Master to Slave Access”:
	- Master 9 (was ISI DMA; is Reserved)
	- Master 10 (was EMAC DMA; is EMAC0 DMA)
	- modified Master 11 (was Reserved; is EMAC1 DMA)
	Section 25.6 “Register Write Protection”: changed title (was “Write Protect Registers”) and revised contents
	Deleted section “Chip Configuration User Interface” (register CCFG_EBICSA is now found in Section 25.7 “Bus Matrix (MATRIX) User Interface”
	Table 25-4 “Register Mapping”:
	- defined offset 0x0024 as reserved
	- defined offsets 0x0104–0x011C as reserved
	- at offset 0x0120, inserted register CCFG_EBICSA
	- defined offsets 0x0124–0x01FC as reserved
	Section 25.7.1 “Bus Matrix Master Configuration Registers”:
	- updated register range in Name (was MATRIX_MCFG0...MATRIX_MCFG10; is MATRIX_MCFG0...MATRIX_MCFG11) and removed address 0xFFFFDE24 of non-implemented MATRIX_MCFG9
	- inserted sentence about write protection
	Section 25.7.2 “Bus Matrix Slave Configuration Registers”: inserted sentence about write protection
	Section 25.7.3 “Bus Matrix Priority Registers A For Slaves”:
	- updated register range in Name (was MATRIX_PRAS0...MATRIX_PRAS8; is MATRIX_PRAS0...MATRIX_PRAS9)
	- inserted sentence about write protection
	Section 25.7.4 “Bus Matrix Priority Registers B For Slaves”:
	- updated register range in Name (was MATRIX_PRBS0...MATRIX_PRBS8; is MATRIX_PRBS0...MATRIX_PRBS9)
	- added field M11PR in register bits 13:12
	- removed field M9PR from register bits 5:4
	- inserted sentence about write protection
	Section 25.7.5 “Bus Matrix Master Remap Control Register”:
	- added bit RCB11 to register bit 11
	- removed bit RCB9 from register bit 9
	- inserted sentence about write protection
	Section 25.7.6 “EBI Chip Select Assignment Register”: changed reset value from 0x00000000 to 0x00000200; updated NFD0_ON_D16 and DDR_MP_EN bit descriptions
	Updated Section 25.7.7 “Write Protection Mode Register”
	Updated Section 25.7.8 “Write Protection Status Register”

Doc. Rev. 11054E	Comments
	Section 26. “External Bus Interface (EBI)” Section 26.2 “Embedded Characteristics”: replaced bullet “MLC Nand Flash ECC Controller” with “8-bit NAND Flash ECC Controller” Table 26-4 “EBI Pins and External Device Connections”: in footnote, replaced instance of “D16-D24” with “D16–D23” Section 26.5.3.4 “Power supplies”: in second paragraph, replaced instance of “D16-D32” with “D16–D31”
	Section 45. “Ethernet MAC 10/100 (EMAC)” Section 45.2 “Embedded Characteristics”: changed first bullet from “Supports MII Interface to the physical layer” to “EMAC0 supports MII and RMII interfaces to the physical layer (EMAC1 supports RMII only)” Table 45-5 “Pin Configuration”: added “RMII” column
	Section 46. “Electrical Characteristics” Table 46-2 “DC Characteristics”: added input impedance characteristics Table 46-5 “Processor Clock Waveform Parameters”: added footnote “For DDR2 usage only, there are no limitations to LP-DDR, SDRAM and mobile SDRAM” Figure 46-2 “Main Oscillator Schematics”: added note “A 1K resistor must be added on XOUT pin for crystals with frequencies lower than 8 MHz” below figure Table 46-10 “12 MHz RC Oscillator Characteristics”: added conditions to parameter “Power Consumption Oscillation” Table 46-18 “I/O Characteristics”: added values; replaced “40 pF” with “20 pF” in footnote defining 3.3V domain Revised Section 46.14 “POR Characteristics” to add Figure 46-5 “General Presentation of POR Behavior” and Section 46.14.2 “Backup Power Supply POR Characteristics” Table 46-28 “Core Power Supply POR Characteristics”: added conditions to parameter “Threshold Voltage Falling” Promoted Section 46.15 “Power Sequence Requirements” to heading level 2 (was level 3) Table 46-31 “Zero Hold Mode Use Maximum System Clock Frequency (MCK)”: in values columns, changed header “Min” to “Max” Added Section 46.19 “Two-wire Interface Characteristics”
	Section 49. “SAM9X25 Errata” Updated Section 49.2.1 “RSTC: Reset during SDRAM Accesses” Added Section 49.5 “Boot Strategy” Added Section 49.6 “Real Time Clock (RTC)”

Doc. Rev. 11054D	Comments	Change Request Ref. (1)
	Introduction: Section 1. “Features”, added DBGU in the Peripherals list. Section 8.2 “Peripheral Identifiers”, added data on System Controller Interrupt in Table 8-1 “Peripheral Identifiers”.	 rfo 8516
	MATRIX: Section 25.7.6.1 “EBI Chip Select Assignment Register”, updated the description of a warning note in “DDR_MP_EN: DDR Multi-port Enable” .	 8532
	DMAC: Added Section 31.2.1 “DMA Controller 0” and Section 31.2.2 “DMA Controller 1”.	 8526
	SPI: Added references on SPKC in Section 35.2 “Embedded Characteristics”.	8541

Doc. Rev. 11054D	Comments	Change Request Ref. (1)
	Errata:	
	Added Section 49.4 “Timer Counter (TC)”.	8517

Doc. Rev. 11054C	Comments	Change Request Ref. (1)
	Introduction:	
	Section 6.3.3 “DDR2SDR Controller”, replaced LPDDR2 with LPDDR.	8146
	Added “Write Protected Registers” in the peripherals list in Section 1. “Features”.	8213
	Added “4-bank” references to the DDR2 characteristics in Section 1. “Features”, Section 2. “Block Diagram” and Section 6.3.3 “DDR2SDR Controller”.	8282
	Section 5.1 “Power Supplies”, added PLLUTMI cell as a power to the VDDPLLA line in Table 5-1 “SAM9X25 Power Supplies”.	8368
	Section 1. “Features”, replaced “MLC/SLC NAND Controller“ with “MLC/SLC 8-bit NAND Controller” in Memories list.	8403
	Section 6.3.2 “Static Memory Controller”, replaced “8- or 16-bit Data Bus” with “8-, 16-, or 32-bit Data Bus”.	8420
	Replaced TSADVREF with ADVREF in Figure 2-1 “SAM9X25 Block Diagram”.	8454
	Boot Startegies:	
	Section 11.3 “Chip Setup”, added Table 11-1 “External Clock and Crystal Frequencies allowed for Boot Sequence (in MHz)” and the corresponding text below the table.	8269
	Section 11.4.1 “NVM Boot Sequence”, replaced “Boot Sequence Register (BSCR)” with “Boot Sequence Configuration Register (BSC_CR)” and updated the acronym of this register in the entire section. Added a reference to the “Boot Sequence Controller (BSC)” section. Replaced “BSCR value” with “BOOT Value” in the heading line in Table 11-2 “Boot Sequence Configuration Register Values”.	rfo
	BSC:	
	Section 12.4.1 “Boot Sequence Configuration Register”:	7996
	- updated the BSC_CR register table	8184
	- added a reference to the “NVM Boot Sequence” section in “BOOT: Boot Media Sequence” .	rfo
	Section 12.2 “Embedded Characteristics”, removed “Product-dependent order” line.	
	Added Section 12.3 “Product Dependencies”.	
	Updated the acronym of Boot Sequence Configuration Register from “BSCR” to “BSC_CR”.	
	AIC:	
	Section 13.10.2 “AIC Source Mode Register”, removed the PRIOR bitfield table as values 0 to 7 can be used and updated the description of this bitfield in “PRIOR: Priority Level” .	8017
	RSTC:	
	Section 14.5.1 “Reset Controller Control Register”, updated description of the EXTRST bitfield for the RSTC_CR register in “EXTRST: External Reset” .	8271
	RTC:	
	Section 15.6 “Real-time Clock (RTC) User Interface”, updated the peripheral name from “Real Time Clock” to “Real-time Clock” and replaced the Reserved Register line “0x30-0xF8” with two lines “0x30–0xC4” and “0xC8–0xF8” (Reserved Register) in Table 15-1 “Register Mapping”.	8280 rfo

Doc. Rev. 11054C	Comments	Change Request Ref. (1)
	WDT:	
	Added the 4th paragraph “If the watchdog is restarted...” in Section 17.4 “Functional Description”.	8128:
	Section 17.5.3 “Watchdog Timer Status Register”, added a note in “WDERR: Watchdog Error” .	
	Updated Section 17.2 “Embedded Characteristics”.	8218
	SHDWC:	
	Removed AMBA references from Section 18.2 “Embedded Characteristics”.	rfo
	Section 18.3 “Block Diagram”, removed redundant Figure 18-2. Sutdown Controller Block Diagram.	8454
	GPBR:	
	Section 19.3.1 “General Purpose Backup Register x”, removed ‘x’ from the bitfield names in the SYS_GPBRx register table and in the description below.	7990
	SCKC:	
	Section 20.3 “Block Diagram”, updated the first paragraph: the RCEN, OSC32EN, OSCSEL and OSC32BYP bits are located not in Slow Clock Control Register (SCKCR) but in Slow Clock Configuration Register (SCKC_CR).	8322
	Fixed Figure 20-1 “Block Diagram” for better representation.	rfo
	CKGR:	
	Section 21.6.2 “Switch from Internal 12 MHz RC Oscillator to the 12 MHz Crystal”, fixed a typo in the sequence order: MAINRDY --> MOSCXTS.	8327
	Section 21.7 “Divider and PLLA Block”, added the PLLADIV2 block between the PLLA block and the PLLACK reference in Figure 21-6 “Divider and PLLA Block Diagram”.	8401
	Updated Crystal Oscillator range from “3 to 20 MHz” to “12 to 16 MHz” in Section 21.2 “Embedded Characteristics”, Section 21.5 “Main Clock”, Figure 21-3 “Main Clock Block Diagram”, Section 21.6.6 “12 to 16 MHz Crystal Oscillator”, Section 21.6.7 “Main Clock Oscillator Selection”, and Section 21.6.8 “Main Clock Frequency Counter”.	8413
	Section 21.3 “CKGR Block Diagram”, updated the UPLL block connections in Figure 21-1 “Clock Generator Block Diagram”.	rfo
	PMC:	
	Section 22.7 “LP-DDR/DDR2 Clock”, removed phrases with references to SysClk.	7974
	Section 22.4 “Block Diagram”, removed the “/1, /2” divider block in Figure 22-2 “General Clock Block Diagram”.	8401
	Section 22.13 “Power Management Controller (PMC) User Interface”, updated the CKGR_MOR reset value (0x0100_0008 --> 0x0000_0008) in Table 22-3 “Register Mapping”.	8447
	PIO:	
	Section 23.4.4 “Interrupt Generation”, updated the 1st paragraph.	8324
	Section 23.5.10 “Input Edge/Level Interrupt”, replaced “...to the Advanced Interrupt Controller (AIC)” with “...to the interrupt controller” in the last phrase of the paragraph “When an input Edge or Level is detected...”.	
	EBI:	
	Section 26.5.1 “Hardware Interface”, fixed typos in Table 26-4 “EBI Pins and External Device Connections”: the power supply of A20, A23, A24, A25, NCS2, NCS4 and NCS5 is VDDNF and not VDDIOM.	8179
	Updated EBIx pin data in Table 26-2 “EBI Pins and Memory Controllers I/O Lines Connections” and added A13 as SDRAMC pin in the A15 line in Table 26-4 “EBI Pins and External Device Connections”.	rfo

Doc. Rev. 11054C	Comments	Change Request Ref. [1]
	PMECC:	
	Figure 27-2 “Software/Hardware Multibit Error Correction Dataflow”, “READ PAGE” and “PROGRAM PAGE” positions swapped in the flow chart.	7495
	Figure 27-5 “Read Operation with Spare Decoding”, configuration revised as ”...SPAREEN set to One and AUTO set to Zero.”	
	Section 27.2 “Embedded Characteristics”, added a line about supporting 8-bit Nand Flash data bus.	8403
	Section 27.6.11 “PMECC Interrupt Status Register”, replaced duplicate bits 31 - 24 with missing 7 - 0 in the PMECC_ISR register table.	rfo
	PMERRLOC:	
	Section 28.5.10 “Error Location SIGMAx Register”, “SIGMAN” bitfield name replaced with “SIGMAx” in the PMERRLOC_SIGMAx [x=0..24] register table.	8339
	SMC:	
	Replaced “...turned out...” with “...switched to output mode...” in the first paragraphs in Section 29.9.4.1 “Write is Controlled by NWE (WRITE_MODE = 1)” and Section 29.9.4.2 “Write is Controlled by NCS (WRITE_MODE = 0)”.	7925
	DDRSDRC:	
	Section 30.2 “Embedded Characteristics”, removed duplicate reference to DDR2-SDRAM.	8146
	DMAC:	
	Section 31.4.5.1 “Programming Examples”, value ‘1’ --> ‘0’ for a masked BTC (DMAC_EBCIMR.BTCx = ‘0’) in “Multi-buffer Transfer with Source Address Auto-reloaded and Destination Address Auto-reloaded (Row 10)” .	7393
	Updated names:	
	- ‘Buffer Complete Interrupt’ --> ‘Buffer Transfer Completed Interrupt’	
	- ‘Chained Buffer Interrupt’ --> ‘Chained Buffer Transfer Completed Interrupt’	
	- ‘Transfer Complete Interrupt’ --> ‘Chained Buffer Transfer Completed Interrupt’	
	- KEEPON[n] --> KEEPx, STALLED[n] --> STALx, ENABLE[n] --> ENAx, SUSPEND[n] --> SUSPx, RESUME[n] --> RESx, EMPTY[n] --> EMPTx.	
	- Read the Channel Enable register --> Read the Channel Handler Status register.	
	Detailed bitfield acronyms when missing.	
	Updated Section 31.2 “Embedded Characteristics”:	rfo
	- updated the list of embedded characteristics	
	- removed Section 31.2.1 DMA Controller 0 and Section 31.2.1 DMA Controller 1.	
	Section 31.7.16 “DMAC Channel x [x = 0..7] Control A Register”, updated SCSIZE and DCSIZE bitfield tables.	8143
	Section 31.7.21 “DMAC Write Protect Mode Register”, updated the descriptions of WPEN and WPKEY bitfields: replaced the wrong values 0x444D4143 and 0x50494F with 0x444D41, and replaced ‘(“DMAC” in ASCII)’ with ‘(“DMA” in ASCII)’.	8404
	Section 31.7.2 “DMAC Enable Register”, Section 31.7.15 “DMAC Channel x [x = 0..7] Descriptor Address Register”, Section 31.7.16 “DMAC Channel x [x = 0..7] Control A Register”, and Section 31.7.17 “DMAC Channel x [x = 0..7] Control B Register”, added respectively descriptions of the following bitfields:	rfo
	- “ENABLE: General Enable of DMA”	
	- “DSCR_IF: Descriptor Interface Selection”	
	- “DONE: Current Descriptor Stop Command and Transfer Completed Memory Indicator”	
	- “IEN: Interrupt Enable Not”	
	Updated the last paragraph in Section 31.4.4.3 “Ending Multi-buffer Transfers”.	8441

Doc. Rev. 11054C	Comments	Change Request Ref. (1)
	UDPHS:	
	Section 32.4 “Typical Connection”, completed a note below Figure 32-2 “Board Schematic”.	7986
	Section 32.7 “USB High Speed Device Port (UDPHS) User Interface”, removed duplicated names in fields and created separated view for UDPHS Control and Status Registers in:	8396
	- Section 32.7.9 “UDPHS Endpoint Control Enable Register (Control, Bulk, Interrupt Endpoints)”	
	- Section 32.7.10 “UDPHS Endpoint Control Enable Register (Isochronous Endpoints)”	
	- Section 32.7.11 “UDPHS Endpoint Control Disable Register (Control, Bulk, Interrupt Endpoints)”	
	- Section 32.7.12 “UDPHS Endpoint Control Disable Register (Isochronous Endpoint)”	
	- Section 32.7.13 “UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)”	
	- Section 32.7.14 “UDPHS Endpoint Control Register (Isochronous Endpoint)”	
	- Section 32.7.15 “UDPHS Endpoint Set Status Register (Control, Bulk, Interrupt Endpoints)”	
	- Section 32.7.16 “UDPHS Endpoint Set Status Register (Isochronous Endpoint)”	
	- Section 32.7.17 “UDPHS Endpoint Clear Status Register (Control, Bulk, Interrupt Endpoints)”	
	- Section 32.7.18 “UDPHS Endpoint Clear Status Register (Isochronous Endpoint)”	
	- Section 32.7.19 “UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)”	
	- Section 32.7.20 “UDPHS Endpoint Status Register (Isochronous Endpoint)”	
	Renamed ER_CRC_NTR bitfield to ERR_CRC_NTR.	8405
	Added ISOENDPT right-hand side qualifier to alternate register definitions in Section 32.7.10, Section 32.7.12, Section 32.7.14, Section 32.7.16, Section 32.7.18, and Section 32.7.20. Fixed typos.	
	Section 32.2 “Embedded Characteristics”: removed Figure 32-1. USB Selection and Table 32-1. UDPHS Endpoint Description (see Section 32.6.1 and Section 32.6.4 instead).	rfo
	Added Section 32.6.1 “UTMI Transceivers Sharing” (extracted from Section 32.2 “Embedded Characteristics”).	
	Updated Section 32.6.4 “USB Transfer Event Definitions”: added Table 32-4 “UDPHS Endpoint Description” with notes and the text below (extracted from Section 32.2 “Embedded Characteristics”).	
	UHPHS:	
	Section 33.2 “Embedded Characteristics”: removed Figure 33-1 USB Selection, Section 33.2.1 EHCI and Section 33.2.2 OHCI including Figure 33-2 Board Schematics to Interface UHP Device Controller.	8104, 8236
	Added Section 33.4 “Typical Connection” and Section 33.6 “Functional Description” (extracted from Section 33.2 “Embedded Characteristics”).	
	Section 33.4 “Typical Connection”, replaced the typical connection figure with a new Figure 33-2 “Board Schematic to Interface UHP High-speed Host Controller”.	
	HSMCI:	
	Section 34.14.12 “HSMCI Status Register”, removed the first phrase in the “NOTBUSY: HSMCI Not Busy” bitfield description (not only for Write operations now).	8394
	Section 34.6.3 “Interrupt”, replaced references to NVIC/AIC with “interrupt controller”.	8431
	Section 34.14.7 “HSMCI Block Register”, replaced BCNT bitfield table with the corresponding description and updated Warning note in “BCNT: MMC/SDIO Block Count - SDIO Byte Count” .	
	Section 34.14.16 “HSMCI DMA Configuration Register”, updated CHKSIZE bitfield in the register table (bits 6, 5 and 4 now), and updated the description of this bitfield in “CHKSIZE: DMA Channel Read and Write Chunk Size” .	

Doc. Rev. 11054C	Comments	Change Request Ref. (1)
	SPI:	
	Replaced references to “Advanced Interrupt Controller” with “Interrupt Controller”.	7513
	Section 35.8.9 “SPI Chip Select Register”, added a phrase specifying when this register can be written and updated the table in “BITS: Bits Per Transfer” : reserved bits are from 9 to 15.	7931
	Section 35.7.3.5 “Peripheral Selection”, corrected a cross-reference for the footnote.	8025
	Section 35.8.10 “SPI Write Protection Mode Register”, replaced “SPIWPKEY” with “WPKEY” and “SPIWPEN” with “WPEN” and added a list of write-protected registers.	8136
	Section 35.8.11 “SPI Write Protection Status Register”, replaced “SPIWPVSRC” with “WPVSRC” and “SPIWPVS” with “WPVS” and updated the description of “WPVS: Write Protection Violation Status” .	
	Section 35.2 “Embedded Characteristics”, removed redundant text line and updated the line “Programmable Transfer Delay Between Consecutive ...”.	8210
	Section 35.8.1 “SPI Control Register”, removed the last phrase in “SWRST: SPI Software Reset” .	8362
	TC:	
	The number of identical 32-bit Timer Counter channels is not three anymore but six.	8648
	Section 36.2 “Embedded Characteristics”, updated the line on input/output signals.	
	Section 36.7 “Timer Counter (TC) User Interface”, added a row for reserved registers (offsets ‘0xC8 - 0xD4’) in Table 36-5 “Register Mapping”.	
	Updated the order of register description sections to match the order in Table 36-5 “Register Mapping”.	rfo
	PWM:	
	Section 37.5.2 “Power Management”, updated the second paragraph.	8105
	Section 37.2 “Embedded characteristics”, updated the last line of the list.	rfo
	TWI:	
	Section 38.1 “Description”, fixed a typo: removed “20” at the end of the 1st paragraph.	7921
	Added three paragraphs in Section 38.8.5 “Master Receiver Mode”.	8426
	Added Figure 38-11 “Master Read Clock Stretching with Multiple Data Bytes”.	
	Added Section 38.11 “Write Protection System”.	
	Added Section 38.8.7.1 “Data Transmit with the DMA” and Section 38.8.7.2 “Data Receive with the DMA”.	
	Updated Section 38.12 “Two-wire Interface (TWI) User Interface”:	
	- Table 38-6 “Register Mapping”, added rows for Protection Mode Register (0xE4) and Protection Status Register	
	- added Section 38.12.12 “TWI Write Protection Mode Register” and Section 38.12.13 “TWI Write Protection Status Register”	
	- added a phrase specifying when the TWI_SMR and TWI_CWGR registers can be written in Section 38.12.3 “TWI Slave Mode Register” and Section 38.12.5 “TWI Clock Waveform Generator Register”.	

Doc. Rev. 11054C	Comments	Change Request Ref. [1]
	USART:	
	Section 39.7.3.4 “Manchester Decoder”, added a paragraph “In order to increase the compatibility...”.	8012
	Section 39.8 “Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface”:	
	-updated the reset value of the US_MAN register from ‘0x30011004’ to ‘0xB0011004’ in Table 39-16 “Register Mapping”	
	- updated descriptions of US_CR, US_MR, US_IER, US_IDR, US_IMR, and US_CSR registers in:	
	Section 39.8.1 “USART Control Register”	
	Section 39.8.3 “USART Mode Register”	
	Section 39.8.5 “USART Interrupt Enable Register”	
	Section 39.8.8 “USART Interrupt Disable Register”	
	Section 39.8.11 “USART Interrupt Mask Register”	
	Section 39.8.14 “USART Channel Status Register”	
	- added sections:	
	Section 39.8.2 “USART Control Register (SPI_MODE)”	
	Section 39.8.4 “USART Mode Register (SPI_MODE)”	
	Section 39.8.6 “USART Interrupt Enable Register (SPI_MODE)”	
	Section 39.8.7 “USART Interrupt Enable Register (LIN_MODE)”	
	Section 39.8.9 “USART Interrupt Disable Register (SPI_MODE)”	
	Section 39.8.10 “USART Interrupt Disable Register (LIN_MODE)”	
	Section 39.8.12 “USART Interrupt Mask Register (SPI_MODE)”	
	Section 39.8.13 “USART Interrupt Mask Register (LIN_MODE)”	
	Section 39.8.15 “USART Channel Status Register (SPI_MODE)”	
	Section 39.8.16 “USART Channel Status Register (LIN_MODE)”	
	Section 39.7.4.1 “ISO7816 Mode Overview”, removed the last phrase about missing ISO7816 inverted mode support.	8097
	Section 39.7.10 “Write Protection Registers”, updated the WPVS flag reset description in the 3d paragraph.	8212
	Section 39.8.3 “USART Mode Register”, updated the MAX_ITERATION field description.	
	Section 39.8.25 “USART Manchester Configuration Register”, changed the definition of the bitfield 29 from “1” to “ONE” and added the corresponding description.	
	Added Section 39.8.28 “USART LIN Baud Rate Register”.	8398
	Figure 39-39 “Header Transmission” and Figure 39-42 “Slave Node Synchronization” reformatted for readability.	
	Section 39.7.1 “Baud Rate Generator”, replaced “...or 6...” with “...or 6 times lower...” in the last phrase of the introduction text.	rfo
	Section 39.6 “Product Dependencies”, added rows for USART3 in Table 39-3 “I/O Lines” and in Table 39-4 “Peripheral IDs”.	
	UART:	
	Section 40.4.3 “Interrupt Source”, replaced the term “Nested Vectored Interrupt Controller” and/or its acronym “NVIC” with “Interrupt Controller”.	8326
	Section 40.2 “Embedded Characteristics”, removed the 2nd line with redundant information.	
	Section 40.1 “Description”, updated the 2nd paragraph.	rfo

Doc. Rev. 11054C	Comments	Change Request Ref. (1)
	CAN:	
	Added information on Write Protected Registers:	8215
	- added a line in Section 41.2 “Embedded Characteristics”	
	- added rows for Write Protect Mode Register (CAN_WPMR) and Write Protect Status Register (CAN_WPSR) in Table 41-6 “Register Mapping”	
	- added Section 41.8.5 “Write Protected Registers” on page 922, Section 41.9.12 “CAN Write Protection Mode Register” on page 940 and Section 41.9.13 “CAN Write Protection Status Register” on page 941	
	- added a phrase specifying when a register can be written (restricted by CAN Write Protection Mode Register) in:	
	Section 41.9.1 “CAN Mode Register”	
	Section 41.9.6 “CAN Baudrate Register”	
	Section 41.9.14 “CAN Message Mode Register”	
	Section 41.9.15 “CAN Message Acceptance Mask Register”	
	Section 41.9.16 “CAN Message ID Register”	
	Updated offsets for reserved registers in Table 41-6 “Register Mapping”:	
	- 0x002C - 0x01FC --> 0x002C - 0x00E0	
	- added a row: - 0x00EC - 0x01FC	
	Updated the register table and the corresponding bitfield name in:	
	Section 41.9.7 “CAN Timer Register”	
	Section 41.9.8 “CAN Timestamp Register”	
	Section 41.9.14 “CAN Message Mode Register”	
	Section 41.9.18 “CAN Message Status Register”	
	Section 41.9.1 “CAN Mode Register”, fixed a typo in “LPM: Disable/Enable Low Power Mode” (‘w’ --> ‘0’).	
	Section 41.9.14 “CAN Message Mode Register”, updated the bitfield table in “MOT: Mailbox Object Type” .	
	Section 41.6.1 “I/O Lines”, added Table 41-2 “I/O Lines”.	
	ADC:	
	Section 42.7.15 “ADC Compare Window Register”, added two paragraphs about programming LOWTHRES and HIGHTHRES bitfields depending on the LOWRES bitfield settings (ADC Mode Register).	8045
	Section 42.6.4 “Conversion Results”, removed “...and EOC bit corresponding to the last converted channel” from the last phrase of the third paragraph.	8357
	Section 42.2 “Embedded Characteristics”, added the value of Conversion Rate in the 2nd line.	8385
	SSC:	
	Section 44.7.1.1 “Clock Divider”, removed Table 43-4 related to Figure 44-5 “Divided Clock Generation” (duplicated data in Section 44.7.1.4 “Serial Clock Ratio Considerations”).	7303
	Section 44.6.3 “Interrupt”, replaced AIC references with “interrupt controller”.	8466
	Section 44.9 “Synchronous Serial Controller (SSC) User Interface”:	
	- updated descriptions of CKS, CKO, and CKG bitfields in:	
	Section 44.9.3 “SSC Receive Clock Mode Register”	
	Section 44.9.5 “SSC Transmit Clock Mode Register”	
	- updated register tables and a description of FSOS bitfield in:	
	Section 44.9.4 “SSC Receive Frame Mode Register”	
	Section 44.9.6 “SSC Transmit Frame Mode Register”	
	Section 44.9.14 “SSC Interrupt Enable Register”, fixed a typo (0=0= --> 0=).	

Doc. Rev. 11054C	Comments	Change Request Ref. [1]
	Electrical Characteristics:	
	Added Section 46.12 “USB Transceiver Characteristics” (extracted from SAM9G15 - 11052C: Section 45.12 USB Transceiver Characteristics).	8016
	Section 46.5 “Main Oscillator Characteristics”, replaced minimum CCRYSTAL value of 17.5 with 15 in Table 46-7 “Main Oscillator Characteristics” and in the corresponding note. Updated the related values in the same note.	8098
	Section 46.5.1 “Crystal Oscillator Characteristics”, added maximum and minimum CCRYSTAL values for ESR in Table 46-8 “Crystal Characteristics”.	
	Section 46.2 “DC Characteristics”, updated RPULLUP parameter characteristics in Section 46-2 “DC Characteristics”.	8147
	Replaced “Input Leakage Current” with “Input Peak Current” in Table 46-26 “Analog Inputs”.	rfo
	Mechanical Overview:	
	Updated the table title in Table 47-4 “Package Information”.	8186
	Errata:	
	Section 49.1 “External Bus Interface (EBI)”, updated the problem description and fix/ workaround.	
	Removed sections concerning PIO and RTC.	8250
	Added Section 49.2 “Reset Controller (RSTC)”, Section 49.2.2 “Static Memory Controller (SMC)”, and Section 49.3 “USB High Speed Host Port (UHPHS) and Device Port (UDPHS)”.	
	Removed “Boot Sequence Controller (BSC)” section (see “Boot Strategies” and “BSC” above for the related modifications).	rfo

Doc. Rev. 11054B	Comments	Change Request Ref.
	DMAC:	
	Section 31.1 “Description”, FIFO size table removed.	8004
	Section 31-4 “Register Mapping” , added ‘USART3 TX 14’ and ‘USART3 RX 15’.	8081
	PMC:	
	Figure 22-2 “General Clock Block Diagram”:	7974
	- Prescaler /1,/2,/4,.../64 --> Prescaler /1,/2,/3,/4,.../64 (for Master Clock Controller.)	
	- SysClk DDR --> 2x MCK, and connection added above with /2 block and DDRCK.	
	Figure 22-3 “Switch Master Clock from Slow Clock to PLL Clock” , ...and the division by 6 --> ...and the division by 3.	
	Section 22.13.11 “PMC Master Clock Register”:	
	- Value 7 for PRES field no more reserved, now with CLOCK_DIV3, Selected clock divided by 3.	
	- MDIV field, references to ‘SysClk DDR’ removed (x4).	
	Section 22.2 “Embedded Characteristics”, 266 MHz DDR system clock --> 133MHz DDR system clock.	7975
	Then DDR system clock --> DDR clock.	

Doc. Rev. 11054B	Comments	Change Request Ref.
	UHPHS: Section 33.6.3 “OHCI”, added Figure 33-2 “Board Schematic to Interface UHP High-speed Host Controller” with an introducing sentence.	8016
	Errata: Section 49.1 "Boot Sequence Controller" , added as the BSC_CR register does not comply with the programmer description.	7996
	MATRIX: Figure 25.7.6.1 , description of NFD0_ON_D16 bitfield updated. Section 26.5.3.4 “Power supplies” on page 331, text added after “...same power supply range (NFD0_ON_D16 = 1).”	8008

Doc. Rev. 11054A	Comments	Change Request Ref.
	1st issue	

Note: 1. “rfo” indicates changes requested during the document review and approval loop

Table of Contents

Description 1

1. Features 2

2. Block Diagram 4

3. Signal Description 5

4. Package and Pinout 10
   - 4.1 Overview of the 217-ball BGA Package 10
   - 4.2 I/O Description 10
   - 4.3 217-ball BGA Package Pinout 12

5. Power Considerations 18
   - 5.1 Power Supplies 18

6. Memories 19
   - 6.1 Memory Mapping 20
   - 6.2 Embedded Memories 20
   - 6.3 External Memories 20

7. System Controller 22
   - 7.1 Chip Identification 24
   - 7.2 Backup Section 24

8. Peripherals 24
   - 8.1 Peripheral Mapping 24
   - 8.2 Peripheral Identifiers 24
   - 8.3 Peripheral Signal Multiplexing on I/O Lines 26

9. ARM926EJ-S™ 27
   - 9.1 Description 27
   - 9.2 Embedded Characteristics 28
   - 9.3 Block Diagram 29
   - 9.4 ARM9EJ-S Processor 30
   - 9.5 CP15 Coprocessor 37
   - 9.6 Memory Management Unit (MMU) 39
   - 9.7 Caches and Write Buffer 40
   - 9.8 Bus Interface Unit 42

10. Debug and Test 43
   - 10.1 Description 43
   - 10.2 Embedded Characteristics 43
   - 10.3 Block Diagram 44
   - 10.4 Application Examples 45
   - 10.5 Debug and Test Pin Description 47
   - 10.6 Functional Description 48

11. Boot Strategies 51
   - 11.1 ROM Code 51
   - 11.2 Flow Diagram 51

11.3 Chip Setup . . . 52
11.4 NVM Boot . . . 52
11.5 SAM-BA Monitor . . . 63

12. Boot Sequence Controller (BSC) . . . 67
12.1 Description . . . 67
12.2 Embedded Characteristics . . . 67
12.3 Product Dependencies . . . 67
12.4 Boot Sequence Controller (BSC) User Interface . . . 68

13. Advanced Interrupt Controller (AIC) . . . 70
13.1 Description . . . 70
13.2 Embedded Characteristics . . . 70
13.3 Block Diagram . . . 71
13.4 Application Block Diagram . . . 71
13.5 AIC Detailed Block Diagram . . . 71
13.6 I/O Line Description . . . 72
13.7 Product Dependencies . . . 72
13.8 Functional Description . . . 73
13.9 Write Protection Registers . . . 82
13.10 Advanced Interrupt Controller (AIC) User Interface . . . 83

14. Reset Controller (RSTC) . . . 104
14.1 Description . . . 104
14.2 Embedded Characteristics . . . 104
14.3 Block Diagram . . . 105
14.4 Functional Description . . . 106
14.5 Reset Controller (RSTC) User Interface . . . 114

15. Real-time Clock (RTC) . . . 118
15.1 Description . . . 118
15.2 Embedded Characteristics . . . 118
15.3 Block Diagram . . . 119
15.4 Product Dependencies . . . 120
15.5 Functional Description . . . 120
15.6 Real-time Clock (RTC) User Interface . . . 123

16. Periodic Interval Timer (PIT) . . . 136
16.1 Description . . . 136
16.2 Embedded Characteristics . . . 136
16.3 Block Diagram . . . 137
16.4 Functional Description . . . 138
16.5 Periodic Interval Timer (PIT) User Interface . . . 139

17. Watchdog Timer (WDT) . . . 144
17.1 Description . . . 144
17.2 Embedded Characteristics . . . 144
17.3 Block Diagram . . . 145
17.4 Functional Description . . . 146
17.5 Watchdog Timer (WDT) User Interface . . . 148

18. Shutdown Controller (SHDWC) . . . 152

18.1 Description . . . 152
18.2 Embedded Characteristics . . . 152
18.3 Block Diagram . . . 153
18.4 I/O Lines Description . . . 154
18.5 Product Dependencies . . . 154
18.6 Functional Description . . . 155
18.7 Shutdown Controller (SHDWC) User Interface . . . 156

19. General Purpose Backup Registers (GPBR) . . . 160
19.1 Description . . . 160
19.2 Embedded Characteristics . . . 160
19.3 General Purpose Backup Registers (GPBR) User Interface . . . 161

20. Slow Clock Controller (SCKC) . . . 163
20.1 Description . . . 163
20.2 Embedded Characteristics . . . 163
20.3 Block Diagram . . . 163
20.4 Slow Clock Configuration (SCKC) User Interface . . . 165

21. Clock Generator (CKGR) . . . 167
21.1 Description . . . 167
21.2 Embedded Characteristics . . . 167
21.3 CKGR Block Diagram . . . 168
21.4 Slow Clock Selection . . . 169
21.5 Main Clock . . . 172
21.6 Main Clock Selection . . . 173
21.7 Divider and PLLA Block . . . 175
21.8 UTMI Phase Lock Loop Programming . . . 176

22. Power Management Controller (PMC) . . . 177
22.1 Description . . . 177
22.2 Embedded Characteristics . . . 177
22.3 Master Clock Controller . . . 178
22.4 Block Diagram . . . 179
22.5 Processor Clock Controller . . . 179
22.6 USB Device and Host Clocks . . . 180
22.7 LP-DDR/DDR2 Clock . . . 180
22.8 Software Modem Clock . . . 180
22.9 Peripheral Clock Controller . . . 180
22.10 Programmable Clock Output Controller . . . 181
22.11 Programming Sequence . . . 181
22.12 Clock Switching Details . . . 184
22.13 Power Management Controller (PMC) User Interface . . . 187

23. Parallel Input/Output (PIO) Controller . . . 212
23.1 Description . . . 212
23.2 Embedded Characteristics . . . 212
23.3 Block Diagram . . . 213
23.4 Product Dependencies . . . 214
23.5 Functional Description . . . 215
23.6 I/O Lines Programming Example . . . 224
23.7 Parallel Input/Output Controller (PIO) User Interface . . . 225

24. Debug Unit (DBGU) 279
24.1 Description 279
24.2 Embedded Characteristics 279
24.3 Block Diagram 280
24.4 Product Dependencies 281
24.5 UART Operations 281
24.6 Debug Unit (DBGU) User Interface 288

25. Bus Matrix (MATRIX) 303
25.1 Description 303
25.2 Embedded Characteristics 303
25.3 Memory Mapping 305
25.4 Special Bus Granting Mechanism 305
25.5 Arbitration 306
25.6 Register Write Protection 309
25.7 Bus Matrix (MATRIX) User Interface 310

26. External Bus Interface (EBI) 323
26.1 Description 323
26.2 Embedded Characteristics 323
26.3 EBI Block Diagram 324
26.4 I/O Lines Description 325
26.5 Application Example 326

27. Programmable Multibit ECC Controller (PMECC) 342
27.1 Description 342
27.2 Embedded Characteristics 342
27.3 Block Diagram 343
27.4 Functional Description 344
27.5 Software Implementation 349
27.6 Programmable Multibit ECC Controller (PMECC) User Interface 354

28. Programmable Multibit ECC Error Location Controller (PMERRLOC) 370
28.1 Description 370
28.2 Embedded Characteristics 370
28.3 Block Diagram 370
28.4 Functional Description 371
28.5 Programmable Multibit ECC Error Location Controller (PMERRLOC) User Interface 372

29. Static Memory Controller (SMC) 384
29.1 Description 384
29.2 Embedded Characteristics 384
29.3 I/O Lines Description 385
29.4 Multiplexed Signals 385
29.5 Application Example 386
29.6 Product Dependencies 386
29.7 External Memory Mapping 387
29.8 Connection to External Devices 387
29.9 Standard Read and Write Protocols 391
29.10 Automatic Wait States 399

29.11 Data Float Wait States . . . 402
29.12 External Wait . . . 407
29.13 Slow Clock Mode . . . 413
29.14 Asynchronous Page Mode . . . 416
29.15 Programmable IO Delays . . . 418
29.16 Static Memory Controller (SMC) User Interface . . . 420

30. DDR SDR SDRAM Controller (DDRSDRC) . . . 429
30.1 Description . . . 429
30.2 Embedded Characteristics . . . 430
30.3 DDRSDRC Module Diagram . . . 431
30.4 Initialization Sequence . . . 432
30.5 Functional Description . . . 435
30.6 Software Interface/SDRAM Organization, Address Mapping . . . 452
30.7 DDR SDR SDRAM Controller (DDRSDRC) User Interface . . . 456

31. DMA Controller (DMAC) . . . 473
31.1 Description . . . 473
31.2 Embedded Characteristics . . . 473
31.3 Block Diagram . . . 476
31.4 Functional Description . . . 477
31.5 DMAC Software Requirements . . . 502
31.6 Write Protection Registers . . . 503
31.7 DMA Controller (DMAC) User Interface . . . 504

32. USB High Speed Device Port (UDPHS) . . . 530
32.1 Description . . . 530
32.2 Embedded Characteristics . . . 530
32.3 Block Diagram . . . 531
32.4 Typical Connection . . . 532
32.5 Product Dependencies . . . 532
32.6 Functional Description . . . 533
32.7 USB High Speed Device Port (UDPHS) User Interface . . . 555

33. USB Host High Speed Port (UHPHS) . . . 600
33.1 Description . . . 600
33.2 Embedded Characteristics . . . 600
33.3 Block Diagram . . . 601
33.4 Typical Connection . . . 602
33.5 Product Dependencies . . . 603
33.6 Functional Description . . . 605

34. High Speed MultiMedia Card Interface (HSMCI) . . . 606
34.1 Description . . . 606
34.2 Embedded Characteristics . . . 606
34.3 Block Diagram . . . 607
34.4 Application Block Diagram . . . 607
34.5 Pin Name List . . . 608
34.6 Product Dependencies . . . 608
34.7 Bus Topology . . . 609
34.8 High Speed MultiMedia Card Operations . . . 611
34.9 SD/SDIO Card Operation . . . 628

34.10 CE-ATA Operation 629
34.11 HSMCI Boot Operation Mode 630
34.12 HSMCI Transfer Done Timings 631
34.13 Write Protection Registers 632
34.14 High Speed MultiMedia Card Interface (HSMCI) User Interface 633

35. Serial Peripheral Interface (SPI) 660
35.1 Description 660
35.2 Embedded Characteristics 660
35.3 Block Diagram 661
35.4 Application Block Diagram 662
35.5 Signal Description 662
35.6 Product Dependencies 662
35.7 Functional Description 663
35.8 Serial Peripheral Interface (SPI) User Interface 674

36. Timer Counter (TC) 688
36.1 Description 688
36.2 Embedded Characteristics 688
36.3 Block Diagram 689
36.4 Pin Name List 690
36.5 Product Dependencies 690
36.6 Functional Description 691
36.7 Timer Counter (TC) User Interface 703

37. Pulse Width Modulation Controller (PWM) 721
37.1 Description 721
37.2 Embedded characteristics 721
37.3 Block Diagram 722
37.4 I/O Lines Description 722
37.5 Product Dependencies 723
37.6 Functional Description 724
37.7 Pulse Width Modulation Controller (PWM) User Interface 730

38. Two-wire Interface (TWI) 743
38.1 Description 743
38.2 Embedded Characteristics 744
38.3 List of Abbreviations 744
38.4 Block Diagram 745
38.5 Application Block Diagram 745
38.6 Product Dependencies 746
38.7 Functional Description 747
38.8 Master Mode 748
38.9 Multi-master Mode 761
38.10 Slave Mode 764
38.11 Write Protection System 771
38.12 Two-wire Interface (TWI) User Interface 772

39. Universal Synchronous Asynchronous Receiver Transmitter (USART) 789
39.1 Description 789
39.2 Embedded Characteristics 790

39.3 Block Diagram . . . 791
39.4 Application Block Diagram . . . 792
39.5 I/O Lines Description . . . 793
39.6 Product Dependencies . . . 793
39.7 Functional Description . . . 795
39.8 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface . . . 839

40. Universal Asynchronous Receiver Transmitter (UART) . . . 879
40.1 Description . . . 879
40.2 Embedded Characteristics . . . 879
40.3 Block Diagram . . . 880
40.4 Product Dependencies . . . 881
40.5 UART Operations . . . 881
40.6 Universal Asynchronous Receiver Transmitter (UART) User Interface . . . 887

41. Controller Area Network (CAN) Programmer Datasheet . . . 897
41.1 Description . . . 897
41.2 Embedded Characteristics . . . 897
41.3 Block Diagram . . . 898
41.4 Application Block Diagram . . . 898
41.5 I/O Lines Description . . . 899
41.6 Product Dependencies . . . 899
41.7 CAN Controller Features . . . 900
41.8 Functional Description . . . 911
41.9 Controller Area Network (CAN) User Interface . . . 923

42. Analog-to-Digital Converter (ADC) . . . 953
42.1 Description . . . 953
42.2 Embedded Characteristics . . . 953
42.3 Block Diagram . . . 954
42.4 Signal Description . . . 954
42.5 Product Dependencies . . . 955
42.6 Functional Description . . . 956
42.7 Analog-to-Digital Converter (ADC) User Interface . . . 962

43. Software Modem Device (SMD) . . . 983
43.1 Description . . . 983
43.2 Embedded Characteristics . . . 984
43.3 Block Diagram . . . 984

44. Synchronous Serial Controller (SSC) . . . 985
44.1 Description . . . 985
44.2 Embedded Characteristics . . . 985
44.3 Block Diagram . . . 986
44.4 Application Block Diagram . . . 986
44.5 Pin Name List . . . 987
44.6 Product Dependencies . . . 987
44.7 Functional Description . . . 988
44.8 SSC Application Examples . . . 999
44.9 Synchronous Serial Controller (SSC) User Interface . . . 1002

45. Ethernet MAC 10/100 (EMAC) 1023
45.1 Description 1023
45.2 Embedded Characteristics 1023
45.3 Block Diagram 1024
45.4 Functional Description 1025
45.5 Programming Interface 1034
45.6 Ethernet MAC 10/100 (EMAC) User Interface 1037

46. Electrical Characteristics 1091
46.1 Absolute Maximum Ratings 1091
46.2 DC Characteristics 1092
46.3 Power Consumption 1093
46.4 Clock Characteristics 1095
46.5 Main Oscillator Characteristics 1095
46.6 12 MHz RC Oscillator Characteristics 1097
46.7 32 kHz Oscillator Characteristics 1097
46.8 32 kHz RC Oscillator Characteristics 1098
46.9 PLL Characteristics 1099
46.10 I/Os 1100
46.11 USB HS Characteristics 1100
46.12 USB Transceiver Characteristics 1101
46.13 Analog-to-Digital Converter (ADC) 1102
46.14 POR Characteristics 1103
46.15 Power Sequence Requirements 1104
46.16 SMC Timings 1105
46.17 DDRSDRC Timings 1108
46.18 Peripheral Timings 1109
46.19 Two-wire Interface Characteristics 1123

47. Mechanical Overview 1125
47.1 217-ball BGA Package 1125
47.2 Marking 1126

48. SAM9X25 Ordering Information 1127

49. SAM9X25 Errata 1128
49.1 External Bus Interface (EBI) 1128
49.2 Reset Controller (RSTC) 1128
49.3 USB High Speed Host Port (UHPHS) and Device Port (UDPHS) 1128
49.4 Timer Counter (TC) 1129
49.5 Boot Strategy 1129
49.6 Real Time Clock (RTC) 1130

Revision History 1131

Table of Contents 1143

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81) (3) 6417-0300
Fax: (+81) (3) 6417-0370

© 2014 Atmel Corporation. All rights reserved. / Rev.: 11054E–ATARM–10-Mar-2014

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, SAM-BA® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM® and Thumb® are registered trademarks of ARM Ltd. Windows® and others are registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 AND 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 products 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.