ARM-based Embedded MPU

SAMA5D3 Series

DATASHEET

Description

The Atmel SAMA5D3 series is a high-performance, power-efficient embedded MPU based on the ARM® Cortex®-A5 processor, achieving 536 MHz with power consumption levels below 0.5 mW in low-power mode. The device features a floating point unit for high-precision computing and accelerated data processing, and a high data bandwidth architecture. It integrates advanced user interface and connectivity peripherals and security features.

The SAMA5D3 series features an internal multi-layer bus architecture associated with 39 DMA channels to sustain the high bandwidth required by the processor and the high-speed peripherals. The device offers support for DDR2/LPDDR/LPDDR2 and MLC NAND Flash memory with 24-bit ECC.

The comprehensive peripheral set includes an LCD controller with overlays for hardware-accelerated image composition, a touchscreen interface and a CMOS sensor interface. Connectivity peripherals include Gigabit EMAC with IEEE1588, 10/100 EMAC, multiple CAN, UART, SPI and I2C. With its secure boot mechanism, hardware accelerated engines for encryption (AES, TDES) and hash function (SHA), the SAMA5D3 ensures anti-cloning, code protection and secure external data transfers.

The SAMA5D3 series is optimized for control panel/HMI applications and applications that require high levels of connectivity in the industrial and consumer markets. Its low-power consumption levels make the SAMA5D3 particularly suited for battery-powered devices.

There are five SAMA5D3 devices in this series. Table 1-1 “SAMA5D3 Device Differences” shows the differences in the embedded features. All other features are available on all derivatives; this includes the three USB ports as well as the encryption engine and secure boot features.

1. Features

- Core
  - ARM® Cortex®-A5 Processor with ARM v7-A Thumb2® Instruction Set
    - CPU Frequency up to 536 MHz
  - 32 Kbyte Data Cache, 32 Kbyte Instruction Cache, Virtual Memory System Architecture (VMSA)
  - Fully Integrated MMU and Floating Point Unit (VFPv4)
- Memories
  - One 160 Kbyte Internal ROM Single-cycle Access at System Speed, Embedded Boot Loader: Boot on 8-bit NAND Flash, SDCard, eMMC, serial DataFlash®, selectable Order
  - One 128 Kbyte Internal SRAM, Single-cycle Access at System Speed
  - High Bandwidth 32-bit Multi-port Dynamic RAM Controller supporting 512 Mbyte 8 bank DDR2/LPDDR/LPDDR2 with datapath scrambling
  - Independent Static Memory Controller with datapath scrambling and SLC/MLC NAND Support with up to 24-bit Error Correcting Code (PMECC)
- System running up to 166 MHz
  - 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 Oscillator and Fast 12 MHz RC Oscillator
  - Selectable 32768 Hz Low-power Oscillator and 12 MHz Oscillator
  - One 400 to 1000 MHz PLL for the System and one PLL at 480 MHz optimized for USB High Speed
  - 39 DMA Channels including two 8-channel 64-bit Central DMA Controllers
  - 64-bit Advanced Interrupt Controller
  - Three Programmable External Clock Signals
  - Programmable Fuse Box with 256 fuse bits, 192 of them available for Customer
- Low Power Management
  - Shut Down Controller
  - Battery Backup Registers
  - Clock Generator and Power Management Controller
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
- Peripherals
  - LCD TFT Controller with Overlay, Alpha-blending, Rotation, Scaling and Color Space Conversion
  - ITU-R BT. 601/656 Image Sensor Interface
  - Three HS/FS/LS USB Ports with On-Chip Transceivers
    - One Device Controller
    - One Host Controller with Integrated Root Hub (3 Downstream Ports)
  - One 10/100/1000 Mbps Gigabit Ethernet MAC Controller (GMAC) with IEEE1588 support
  - One 10/100 Mbps Ethernet MAC Controller (EMAC)
  - Two CAN Controllers with 8 Mailboxes, fully Compliant with CAN 2.0 Part A and 2.0 Part B
  - Softmodem Interface
  - Three High Speed Memory Card Hosts (eMMC 4.3 and SD 2.0)
  - Two Master/Slave Serial Peripheral Interfaces
  - Two Synchronous Serial Controllers
  - Three Two-wire Interface up to 400 Kbit/s supporting I2C Protocol and SMBUS
  - Four USARTs, two UARTs, one DBGU
  - Two Three-channel 32-bit Timer/Counters
  - One 4-channel 16-bit PWM Controller

- One 12-channel 12-bit Analog-to-Digital Converter with Resistive Touch-Screen function

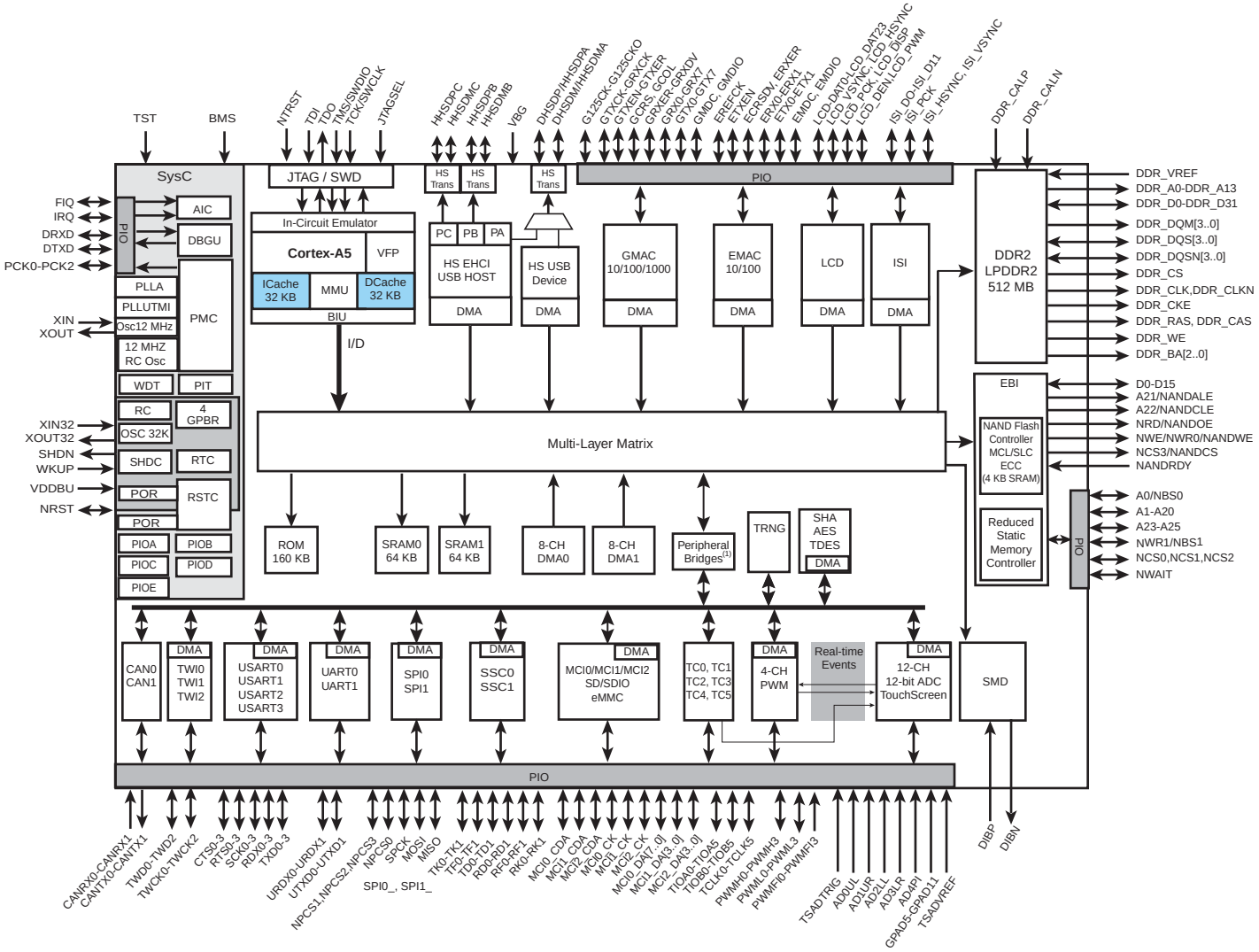
- Safety
  - Power-on Reset Cells
  - Independent Watchdog
  - Main Crystal Clock Failure Detection
  - Write Protection Registers
  - SHA: Supports Secure Hash Algorithm (SHA1, SHA224, SHA256, SHA384, SHA512)
  - Memory Management Unit
- Security
  - TRNG: True Random Number Generator
  - Encryption Engine
    - AES: 256-bit, 192-bit, 128-bit Key Algorithm, Compliant with FIPS PUB 197 Specifications
    - TDES: Two-key or Three-key Algorithms, Compliant with FIPS PUB 46-3 Specifications
  - Atmel® Secure Boot Solution
- I/O
  - Five 32-bit Parallel Input/Output Controllers
  - 160 I/Os
  - Input Change Interrupt Capability on Each I/O Line, Selectable Schmitt Trigger Input
  - Individually Programmable Open-drain, Pull-up and Pull-down Resistor, Synchronous Output, Filtering
  - Slew Rate Control on High Speed I/Os
  - Impedance Control on DDR I/Os
- Package
  - 324-ball LFBGA, 15 x 15 x 1.4 mm, pitch 0.8 mm
  - 324-ball TFBGA, 12 x 12 x 1.2 mm, pitch 0.5 mm

Table 1-1. SAMA5D3 Device Differences

Peripherals	SAMA5D31	SAMA5D33	SAMA5D34	SAMA5D35	SAMA5D36
CAN0, CAN1	—	—	X	X	X
EMAC	X	—	—	X	X
GMAC	—	X	X	X	X
HSMCI2	X	—	X	X	X
LCDC	X	X	X	—	X
TC1	—	—	—	X	X
UART0, UART1	X	—	—	X	X

2. Block Diagram

Figure 2-1. SAMA5D3 Block Diagram



Note: 1. Peripheral Bridge 0 (APB0) connects HSMCI0, SPI0, USART0, USART1, TWI0, TWI1, UART0, SSC0, SMD. Peripheral Bridge 1 (APB1) connects HSMCI1, HSMCI2, ADC, SSC1, UART1, USART2, USART3, TWI2, DBGU, SPI1, SHA, AES, TDES.

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–PCK2	Programmable Clock Output	Output	—
Shutdown, Wake-up Logic			
SHDN	Shut-Down Control	Output	—
WKUP	Wake-Up Input	Input	—
ICE and JTAG			
TCK/SWCLK	Test Clock/Serial Wire Clock	Input	—
TDI	Test Data In	Input	—
TDO	Test Data Out	Output	—
TMS/SWDIO	Test Mode Select/Serial Wire Input/Output	I/O	—
JTAGSEL	JTAG Selection	Input	—
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 - PIOE			
PA0–PAxx	Parallel IO Controller A	I/O	—
PB0–PBxx	Parallel IO Controller B	I/O	—
PC0–PCxx	Parallel IO Controller C	I/O	—
PD0–PDxx	Parallel IO Controller D	I/O	—
PE0–PExx	Parallel IO Controller E	I/O	—
External Bus Interface - EBI			

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
D0–D15	Data Bus	I/O	—
A0–A25	Address Bus	Output	—
NWAIT	External Wait Signal	Input	Low
Static Memory Controller - HSMC			
NCS0–NCS3	Chip Select Lines	Output	Low
NWR0–NWR1	Write Signal	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0–NBS1	Byte Mask Signal	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
DDR2/LPDDR Controller			
DDR_VREF	Reference Voltage	Input	—
DDR_CALP	Positive Calibration Reference	Input	—
DDR_CALN	Negative Calibration Reference	Input	—
DDR_CK, DDR_CKN	DDR2 differential clock	Output	—
DDR_CKE	DDR2 Clock Enable	Output	High
DDR_CS	DDR2 Controller Chip Select	Output	Low
DDR_BA[2..0]	Bank Select	Output	Low
DDR_WE	DDR2 Write Enable	Output	Low
DDR_RAS, DDR_CAS	Row and Column Signal	Output	Low
DDR_A[13..0]	DDR2 Address Bus	Output	—
DDR_D[31..0]	DDR2 Data Bus	I/O	—
DQS[3..0]	Differential Data Strobe	I/O	—
DQSN[3..0]	DQSN must be connected to DDR_VREF for DDR2 memories	I/O	—
DQM[3..0]	Write Data Mask	Output	—
High Speed Multimedia Card Interface - HSMCI0–2			
MCI0_CK, MCI1_CK, MCI2_CK	Multimedia Card Clock	I/O	—
MCI0_CDA, MCI1_CDA, MCI2_CDA	Multimedia Card Command	I/O	—
MCI0_DA[7..0]	Multimedia Card 0 Data	I/O	—
MCI1_DA[3..0]	Multimedia Card 1 Data	I/O	—
MCI2_DA[3..0)	Multimedia Card 2 Data	I/O	—

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
Universal Synchronous Asynchronous Receiver Transmitter - USART0–3			
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 [1..0]			
UTXDx	UARTx Transmit Data	Output	—
URXDx	UARTx Receive Data	Input	—
Synchronous Serial Controller - SSCx [1..0]			
TDx	SSC Transmit Data	Output	—
RDx	SSC Receive Data	Input	—
TKx	SSC Transmit Clock	I/O	—
RKx	SSC Receive Clock	I/O	—
TFx	SSC Transmit Frame Sync	I/O	—
RFx	SSC Receive Frame Sync	I/O	—
Timer/Counter - TCx [5..0]			
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 [1..0]			
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_NPCS[3..1]	SPI Peripheral Chip Select	Output	Low
Two-Wire Interface - TWIx [2..0]			
TWDx	Two-wire Serial Data	I/O	—
TWCKx	Two-wire Serial Clock	I/O	—
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	—
Pulse Width Modulation Controller - PWMC			
PWMH[3..0]	PWM Waveform Output High	Output	—

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
PWML[3..0]	PWM Waveform Output Low	Output	—
PWMFIx	PWM Fault Input	Input	—
USB Host High Speed Port - UHPHS			
HHSDPA	USB Host Port A High Speed Data +	Analog	—
HHSDMA	USB Host Port A High Speed Data -	Analog	—
HHSDPB	USB Host Port B High Speed Data +	Analog	—
HHSDMB	USB Host Port B High Speed Data -	Analog	—
HHSDPC	USB Host Port C High Speed Data +	Analog	—
HHSDMC	USB Host Port C High Speed Data -	Analog	—
USB Device High Speed Port - UDPHS			
DHSDP	USB Device High Speed Data +	Analog	—
DHSDM	USB Device High Speed Data -	Analog	—
GIgabit Ethernet 10/100/1000 - GMAC			
GTXCK	Transmit Clock or Reference Clock	Input	—
G125CK	125 MHz input Clock	Input	—
G125CKO	125 MHz output Clock	Output	—
GTXEN	Transmit Enable	Output	—
GTX[7..0]	Transmit Data	Output	—
GTXER	Transmit Coding Error	Output	—
GRXCK	Receive Clock	Input	—
GRXDV	Receive Data Valid	Input	—
GRX[7..0]	Receive Data	Input	—
GRXER	Receive Error	Input	—
GCRS	Carrier Sense and Data Valid	Input	—
GCOL	Collision Detect	Input	—
GMDC	Management Data Clock	Output	—
GMDIO	Management Data Input/Output	I/O	—
RMII Ethernet 10/100 - EMAC			
EREFCK	Transmit Clock or Reference Clock	Input	—
ETXEN	Transmit Enable	Output	—
ETX[1..0]	Transmit Data	Output	—
ECRSDV	Carrier Sense/Data Valid	Input	—
ERX[1..0]	Receive Data	Input	—
ERXER	Receive Error	Input	—
EMDC	Management Data Clock	Output	—
EMDIO	Management Data Input/Output	I/O	—
LCD Controller - LCDC			

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level
LCDDAT[23..0]	LCD Data Bus	Output	—
LCDVSYNC	LCD Vertical Synchronization	Output	—
LCDHSYNC	LCD Horizontal Synchronization	Output	—
LCDPCK	LCD pixel Clock	Output	—
LCDDEN	LCD Data Enable	Output	—
LCDPWM	LCDPWM for Contrast Control	Output	—
LCDDISP	LCD Display ON/OFF	Output	—
Image Sensor Interface - ISI			
ISI_D[11..0]	Image Sensor Data	Input	—
ISI_HSYNC	Image Sensor Horizontal Synchro	input	—
ISI_VSYNC	Image Sensor Vertical Synchro	input	—
ISI_PCK	Image Sensor Data clock	input	—
Touch Screen Analog-to-Digital Converter - ADC			
$AD0_{UL}$	Upper Left Touch Panel	Analog	—
$AD1_{UR}$	Upper Right Touch Panel	Analog	—
$AD2_{LL}$	Lower Left Touch Panel	Analog	—
$AD3_{LR}$	Lower Right Touch Panel	Analog	—
$AD4_{PI}$	Panel Input	Analog	—
AD5–AD11	7 Analog Inputs	Analog	—
ADTRG	ADC Trigger	Input	—
ADVREF	ADC Reference	Analog	—

4. Package and Pinout

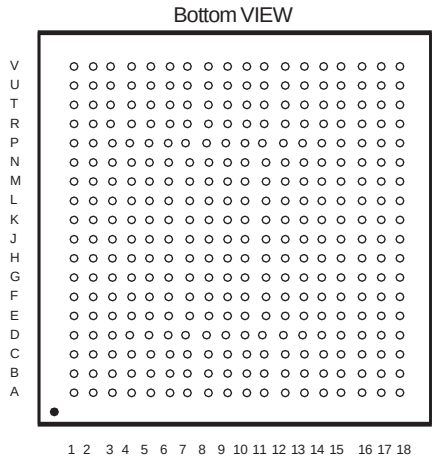
The SAMA5D3 product is available in two packages:

- 324-ball LFBGA (15 x 15 x 1.4 mm, pitch 0.8 mm)
- 324-ball TFBGA (12 x 12 x 1.2 mm, pitch 0.5 mm)

4.1 324-ball LFBGA Package (15 x 15 x 1.4 mm, pitch 0.8 mm)

Figure 4-1 shows the ball map of the 324-ball LFBGA package.

Figure 4-1. 324-ball LFBGA Ball Map



4.2 324-ball LFBGA Package Pinout

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package

			Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
Pin	Power Rail	I/O Type	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
E3	VDDIOP0	GPIO	PA0	I/O	—	—	LCDDAT0	O	—	—	—	—	PIO, I, PU, ST
F5	VDDIOP0	GPIO	PA1	I/O	—	—	LCDDAT1	O	—	—	—	—	PIO, I, PU, ST
D2	VDDIOP0	GPIO	PA2	I/O	—	—	LCDDAT2	O	—	—	—	—	PIO, I, PU, ST
F4	VDDIOP0	GPIO	PA3	I/O	—	—	LCDDAT3	O	—	—	—	—	PIO, I, PU, ST
D1	VDDIOP0	GPIO	PA4	I/O	—	—	LCDDAT4	O	—	—	—	—	PIO, I, PU, ST
J10	VDDIOP0	GPIO	PA5	I/O	—	—	LCDDAT5	O	—	—	—	—	PIO, I, PU, ST
G4	VDDIOP0	GPIO	PA6	I/O	—	—	LCDDAT6	O	—	—	—	—	PIO, I, PU, ST
J9	VDDIOP0	GPIO	PA7	I/O	—	—	LCDDAT7	O	—	—	—	—	PIO, I, PU, ST
F3	VDDIOP0	GPIO	PA8	I/O	—	—	LCDDAT8	O	—	—	—	—	PIO, I, PU, ST
J8	VDDIOP0	GPIO	PA9	I/O	—	—	LCDDAT9	O	—	—	—	—	PIO, I, PU, ST
E2	VDDIOP0	GPIO	PA10	I/O	—	—	LCDDAT10	O	—	—	—	—	PIO, I, PU, ST
K8	VDDIOP0	GPIO	PA11	I/O	—	—	LCDDAT11	O	—	—	—	—	PIO, I, PU, ST
F2	VDDIOP0	GPIO	PA12	I/O	—	—	LCDDAT12	O	—	—	—	—	PIO, I, PU, ST
G6	VDDIOP0	GPIO	PA13	I/O	—	—	LCDDAT13	O	—	—	—	—	PIO, I, PU, ST
E1	VDDIOP0	GPIO	PA14	I/O	—	—	LCDDAT14	O	—	—	—	—	PIO, I, PU, ST
H5	VDDIOP0	GPIO	PA15	I/O	—	—	LCDDAT15	O	—	—	—	—	PIO, I, PU, ST
H3	VDDIOP0	GPIO	PA16	I/O	—	—	LCDDAT16	O	—	—	ISI_D0	I	PIO, I, PU, ST
H6	VDDIOP0	GPIO	PA17	I/O	—	—	LCDDAT17	O	—	—	ISI_D1	I	PIO, I, PU, ST
H4	VDDIOP0	GPIO	PA18	I/O	—	—	LCDDAT18	O	TWD2	I/O	ISI_D2	I	PIO, I, PU, ST
H7	VDDIOP0	GPIO	PA19	I/O	—	—	LCDDAT19	O	TWCK2	O	ISI_D3	I	PIO, I, PU, ST
H2	VDDIOP0	GPIO	PA20	I/O	—	—	LCDDAT20	O	PWMH0	O	ISI_D4	I	PIO, I, PU, ST
J6	VDDIOP0	GPIO	PA21	I/O	—	—	LCDDAT21	O	PWML0	O	ISI_D5	I	PIO, I, PU, ST
G2	VDDIOP0	GPIO	PA22	I/O	—	—	LCDDAT22	O	PWMH1	O	ISI_D6	I	PIO, I, PU, ST
J5	VDDIOP0	GPIO	PA23	I/O	—	—	LCDDAT23	O	PWML1	O	ISI_D7	I	PIO, I, PU, ST
F1	VDDIOP0	GPIO	PA24	I/O	—	—	LCDPWM	O	—	—	—	—	PIO, I, PU, ST
J4	VDDIOP0	GPIO	PA25	I/O	—	—	LCDDISP	O	—	—	—	—	PIO, I, PU, ST
G3	VDDIOP0	GPIO	PA26	I/O	—	—	LCDVSYNC	O	—	—	—	—	PIO, I, PU, ST
J3	VDDIOP0	GPIO	PA27	I/O	—	—	LCDHSYNC	O	—	—	—	—	PIO, I, PU, ST
G1	VDDIOP0	GPIO_CLK2	PA28	I/O	—	—	LCDPCK	O	—	—	—	—	PIO, I, PU, ST
K4	VDDIOP0	GPIO	PA29	I/O	—	—	LCDDEN	O	—	—	—	—	PIO, I, PU, ST
H1	VDDIOP0	GPIO	PA30	I/O	—	—	TWD0	I/O	URXD1	I	ISI_VSYNC	I	PIO, I, PU, ST
K3	VDDIOP0	GPIO	PA31	I/O	—	—	TWCK0	O	UTXD1	O	ISI_HSYNC	I	PIO, I, PU, ST
T2	VDDIOP1	GMAC	PB0	I/O	—	—	GTX0	O	PWMH0	O	—	—	PIO, I, PU, ST
N7	VDDIOP1	GMAC	PB1	I/O	—	—	GTX1	O	PWML0	O	—	—	PIO, I, PU, ST
T3	VDDIOP1	GMAC	PB2	I/O	—	—	GTX2	O	TK1	I/O	—	—	PIO, I, PU, ST
N6	VDDIOP1	GMAC	PB3	I/O	—	—	GTX3	O	TF1	I/O	—	—	PIO, I, PU, ST
P5	VDDIOP1	GMAC	PB4	I/O	—	—	GRX0	I	PWMH1	O	—	—	PIO, I, PU, ST
T4	VDDIOP1	GMAC	PB5	I/O	—	—	GRX1	I	PWML1	O	—	—	PIO, I, PU, ST
R4	VDDIOP1	GMAC	PB6	I/O	—	—	GRX2	I	TD1	O	—	—	PIO, I, PU, ST
U1	VDDIOP1	GMAC	PB7	I/O	—	—	GRX3	I	RK1	I	—	—	PIO, I, PU, ST
R5	VDDIOP1	GMAC	PB8	I/O	—	—	GTXCK	I	PWMH2	O	—	—	PIO, I, PU, ST
P3	VDDIOP1	GMAC	PB9	I/O	—	—	GTXEN	O	PWML2	O	—	—	PIO, I, PU, ST

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
R6	VDDIOP1	GMAC	PB10	I/O	—	—	GTXER	O	RF1	I/O	—	—	PIO, I, PU, ST
V3	VDDIOP1	GMAC	PB11	I/O	—	—	GRXCK	I	RD1	I	—	—	PIO, I, PU, ST
P6	VDDIOP1	GMAC	PB12	I/O	—	—	GRXDV	I	PWMH3	O	—	—	PIO, I, PU, ST
V1	VDDIOP1	GMAC	PB13	I/O	—	—	GRXER	I	PWML3	O	—	—	PIO, I, PU, ST
R7	VDDIOP1	GMAC	PB14	I/O	—	—	GCRS	I	CANRX1	I	—	—	PIO, I, PU, ST
U3	VDDIOP1	GMAC	PB15	I/O	—	—	GCOL	I	CANTX1	O	—	—	PIO, I, PU, ST
P7	VDDIOP1	GMAC	PB16	I/O	—	—	GMDC	O	—	—	—	—	PIO, I, PU, ST
V2	VDDIOP1	GMAC	PB17	I/O	—	—	GMDIO	I/O	—	—	—	—	PIO, I, PU, ST
V5	VDDIOP1	GMAC	PB18	I/O	—	—	G125CK	I	—	—	—	—	PIO, I, PU, ST
T6	VDDIOP1	GMAC	PB19	I/O	—	—	MCI1_CDA	I/O	GTX4	O	—	—	PIO, I, PU, ST
N8	VDDIOP1	GMAC	PB20	I/O	—	—	MCI1_DA0	I/O	GTX5	O	—	—	PIO, I, PU, ST
U4	VDDIOP1	GMAC	PB21	I/O	—	—	MCI1_DA1	I/O	GTX6	O	—	—	PIO, I, PU, ST
M7	VDDIOP1	GMAC	PB22	I/O	—	—	MCI1_DA2	I/O	GTX7	O	—	—	PIO, I, PU, ST
U5	VDDIOP1	GMAC	PB23	I/O	—	—	MCI1_DA3	I/O	GRX4	I	—	—	PIO, I, PU, ST
M8	VDDIOP1	GMAC	PB24	I/O	—	—	MCI1_CK	I/O	GRX5	I	—	—	PIO, I, PU, ST
T5	VDDIOP1	GMAC	PB25	I/O	—	—	SCK1	I/O	GRX6	I	—	—	PIO, I, PU, ST
N9	VDDIOP1	GMAC	PB26	I/O	—	—	CTS1	I	GRX7	I	—	—	PIO, I, PU, ST
V4	VDDIOP1	GPIO	PB27	I/O	—	—	RTS1	O	G125CKO	O	—	—	PIO, I, PU, ST
M9	VDDIOP1	GPIO	PB28	I/O	—	—	RXD1	I	—	—	—	—	PIO, I, PU, ST
P8	VDDIOP1	GPIO	PB29	I/O	—	—	TXD1	O	—	—	—	—	PIO, I, PU, ST
M10	VDDIOP0	GPIO	PB30	I/O	—	—	DRXD	I	—	—	—	—	PIO, I, PU, ST
R9	VDDIOP0	GPIO	PB31	I/O	—	—	DTXD	O	—	—	—	—	PIO, I, PU, ST
D8	VDDIOP0	GPIO	PC0	I/O	—	—	ETX0	O	TIOA3	I/O	—	—	PIO, I, PU, ST
A4	VDDIOP0	GPIO	PC1	I/O	—	—	ETX1	O	TIOB3	I/O	—	—	PIO, I, PU, ST
E8	VDDIOP0	GPIO	PC2	I/O	—	—	ERX0	I	TCLK3	I	—	—	PIO, I, PU, ST
A3	VDDIOP0	GPIO	PC3	I/O	—	—	ERX1	I	TIOA4	I/O	—	—	PIO, I, PU, ST
A2	VDDIOP0	GPIO	PC4	I/O	—	—	ETXEN	O	TIOB4	I/O	—	—	PIO, I, PU, ST
F8	VDDIOP0	GPIO	PC5	I/O	—	—	ECRSDV	I	TCLK4	I	—	—	PIO, I, PU, ST
B3	VDDIOP0	GPIO	PC6	I/O	—	—	ERXER	I	TIOA5	I/O	—	—	PIO, I, PU, ST
G8	VDDIOP0	GPIO	PC7	I/O	—	—	EREFCK	I	TIOB5	I/O	—	—	PIO, I, PU, ST
B4	VDDIOP0	GPIO	PC8	I/O	—	—	EMDC	O	TCLK5	I	—	—	PIO, I, PU, ST
F7	VDDIOP0	GPIO	PC9	I/O	—	—	EMDIO	I/O	—	—	—	—	PIO, I, PU, ST
A1	VDDIOP0	GPIO	PC10	I/O	—	—	MCI2_CDA	I/O	—	—	LCDDAT20	O	PIO, I, PU, ST
D7	VDDIOP0	GPIO	PC11	I/O	—	—	MCI2_DA0	I/O	—	—	LCDDAT19	O	PIO, I, PU, ST
C6	VDDIOP0	GPIO	PC12	I/O	—	—	MCI2_DA1	I/O	TIOA1	I/O	LCDDAT18	O	PIO, I, PU, ST
E7	VDDIOP0	GPIO	PC13	I/O	—	—	MCI2_DA2	I/O	TIOB1	I/O	LCDDAT17	O	PIO, I, PU, ST
B2	VDDIOP0	GPIO	PC14	I/O	—	—	MCI2_DA3	I/O	TCLK1	I	LCDDAT16	O	PIO, I, PU, ST
F6	VDDIOP0	MCI_CLK	PC15	I/O	—	—	MCI2_CK	I/O	PCK2	O	LCDDAT21	O	PIO, I, PU, ST
B1	VDDIOP0	GPIO	PC16	I/O	—	—	TK0	I/O	—	—	—	—	PIO, I, PU, ST
E6	VDDIOP0	GPIO	PC17	I/O	—	—	TF0	I/O	—	—	—	—	PIO, I, PU, ST
C3	VDDIOP0	GPIO	PC18	I/O	—	—	TD0	O	—	—	—	—	PIO, I, PU, ST
D6	VDDIOP0	GPIO	PC19	I/O	—	—	RK0	I/O	—	—	—	—	PIO, I, PU, ST
C4	VDDIOP0	GPIO	PC20	I/O	—	—	RF0	I/O	—	—	—	—	PIO, I, PU, ST

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
D5	VDDIOP0	GPIO	PC21	I/O	—	—	RD0	I	—	—	—	—	PIO, I, PU, ST
C2	VDDIOP0	GPIO	PC22	I/O	—	—	SPI1_MISO	I/O	—	—	—	—	PIO, I, PU, ST
G9	VDDIOP0	GPIO	PC23	I/O	—	—	SPI1_MOSI	I/O	—	—	—	—	PIO, I, PU, ST
C1	VDDIOP0	GPIO_CLK	PC24	I/O	—	—	SPI1_SPCK	I/O	—	—	—	—	PIO, I, PU, ST
H10	VDDIOP0	GPIO	PC25	I/O	—	—	SPI1_NPCS0	I/O	—	—	—	—	PIO, I, PU, ST
H9	VDDIOP0	GPIO	PC26	I/O	—	—	SPI1_NPCS1	O	TWD1	I/O	ISI_D11	I	PIO, I, PU, ST
D4	VDDIOP0	GPIO	PC27	I/O	—	—	SPI1_NPCS2	O	TWCK1	O	ISI_D10	I	PIO, I, PU, ST
H8	VDDIOP0	GPIO	PC28	I/O	—	—	SPI1_NPCS3	O	PWMFI0	I	ISI_D9	I	PIO, I, PU, ST
G5	VDDIOP0	GPIO	PC29	I/O	—	—	URXD0	I	PWMFI2	I	ISI_D8	I	PIO, I, PU, ST
D3	VDDIOP0	GPIO	PC30	I/O	—	—	UTXD0	O	—	—	ISI_PCK	O	PIO, I, PU, ST
E4	VDDIOP0	GPIO	PC31	I/O	—	—	FIQ	I	PWMFI1	I	—	—	PIO, I, PU, ST
K5	VDDIOP1	GPIO	PD0	I/O	—	—	MCI0_CDA	I/O	—	—	—	—	PIO, I, PU, ST
P1	VDDIOP1	GPIO	PD1	I/O	—	—	MCI0_DA0	I/O	—	—	—	—	PIO, I, PU, ST
K6	VDDIOP1	GPIO	PD2	I/O	—	—	MCI0_DA1	I/O	—	—	—	—	PIO, I, PU, ST
R1	VDDIOP1	GPIO	PD3	I/O	—	—	MCI0_DA2	I/O	—	—	—	—	PIO, I, PU, ST
L7	VDDIOP1	GPIO	PD4	I/O	—	—	MCI0_DA3	I/O	—	—	—	—	PIO, I, PU, ST
P2	VDDIOP1	GPIO	PD5	I/O	—	—	MCI0_DA4	I/O	TIOA0	I/O	PWMH2	O	PIO, I, PU, ST
L8	VDDIOP1	GPIO	PD6	I/O	—	—	MCI0_DA5	I/O	TIOB0	I/O	PWML2	O	PIO, I, PU, ST
R2	VDDIOP1	GPIO	PD7	I/O	—	—	MCI0_DA6	I/O	TCLK0	I	PWMH3	O	PIO, I, PU, ST
K7	VDDIOP1	GPIO	PD8	I/O	—	—	MCI0_DA7	I/O	—	—	PWML3	O	PIO, I, PU, ST
U2	VDDIOP1	MCI_CLK	PD9	I/O	—	—	MCI0_CK	I/O	—	—	—	—	PIO, I, PU, ST
K9	VDDIOP1	GPIO	PD10	I/O	—	—	SPI0_MISO	I/O	—	—	—	—	PIO, I, PU, ST
M5	VDDIOP1	GPIO	PD11	I/O	—	—	SPI0_MOSI	I/O	—	—	—	—	PIO, I, PU, ST
K10	VDDIOP1	GPIO_CLK	PD12	I/O	—	—	SPI0_SPCK	I/O	—	—	—	—	PIO, I, PU, ST
N4	VDDIOP1	GPIO	PD13	I/O	—	—	SPI0_NPCS0	I/O	—	—	—	—	PIO, I, PU, ST
L9	VDDIOP1	GPIO	PD14	I/O	—	—	SCK0	I/O	SPI0_NPCS1	O	CANRX0	I	PIO, I, PU, ST
N3	VDDIOP1	GPIO	PD15	I/O	—	—	CTS0	I	SPI0_NPCS2	O	CANTX0	O	PIO, I, PU, ST
L10	VDDIOP1	GPIO	PD16	I/O	—	—	RTS0	O	SPI0_NPCS3	O	PWMFI3	I	PIO, I, PU, ST
N5	VDDIOP1	GPIO	PD17	I/O	—	—	RXD0	I	—	—	—	—	PIO, I, PU, ST
M6	VDDIOP1	GPIO	PD18	I/O	—	—	TXD0	O	—	—	—	—	PIO, I, PU, ST
T1	VDDIOP1	GPIO	PD19	I/O	—	—	ADTRG	I	—	—	—	—	PIO, I, PU, ST
N2	VDDANA	GPIO_ANA	PD20	I/O	—	—	AD0	I	—	—	—	—	PIO, I, PU, ST
M3	VDDANA	GPIO_ANA	PD21	I/O	—	—	AD1	I	—	—	—	—	PIO, I, PU, ST
M2	VDDANA	GPIO_ANA	PD22	I/O	—	—	AD2	I	—	—	—	—	PIO, I, PU, ST
L3	VDDANA	GPIO_ANA	PD23	I/O	—	—	AD3	I	—	—	—	—	PIO, I, PU, ST
M1	VDDANA	GPIO_ANA	PD24	I/O	—	—	AD4	I	—	—	—	—	PIO, I, PU, ST
N1	VDDANA	GPIO_ANA	PD25	I/O	—	—	AD5	I	—	—	—	—	PIO, I, PU, ST
L1	VDDANA	GPIO_ANA	PD26	I/O	—	—	AD6	I	—	—	—	—	PIO, I, PU, ST
L2	VDDANA	GPIO_ANA	PD27	I/O	—	—	AD7	I	—	—	—	—	PIO, I, PU, ST
K1	VDDANA	GPIO_ANA	PD28	I/O	—	—	AD8	I	—	—	—	—	PIO, I, PU, ST
K2	VDDANA	GPIO_ANA	PD29	I/O	—	—	AD9	I	—	—	—	—	PIO, I, PU, ST
J1	VDDANA	GPIO_ANA	PD30	I/O	—	—	AD10	I	PCK0	O	—	—	PIO, I, PU, ST
J2	VDDANA	GPIO_ANA	PD31	I/O	—	—	AD11	I	PCK1	O	—	—	PIO, I, PU, ST

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
P13	VDDIOM	EBI	PE0	I/O	—	—	A0/NBS0	O	—	—	—	—	A,I, PD, ST
R14	VDDIOM	EBI	PE1	I/O	—	—	A1	O	—	—	—	—	A,I, PD, ST
R13	VDDIOM	EBI	PE2	I/O	—	—	A2	O	—	—	—	—	A,I, PD, ST
V18	VDDIOM	EBI	PE3	I/O	—	—	A3	O	—	—	—	—	A,I, PD, ST
P14	VDDIOM	EBI	PE4	I/O	—	—	A4	O	—	—	—	—	A,I, PD, ST
U18	VDDIOM	EBI	PE5	I/O	—	—	A5	O	—	—	—	—	A,I, PD, ST
T18	VDDIOM	EBI	PE6	I/O	—	—	A6	O	—	—	—	—	A,I, PD, ST
R15	VDDIOM	EBI	PE7	I/O	—	—	A7	O	—	—	—	—	A,I, PD, ST
P17	VDDIOM	EBI	PE8	I/O	—	—	A8	O	—	—	—	—	A,I, PD, ST
P15	VDDIOM	EBI	PE9	I/O	—	—	A9	O	—	—	—	—	A,I, PD, ST
P18	VDDIOM	EBI	PE10	I/O	—	—	A10	O	—	—	—	—	A,I, PD, ST
R16	VDDIOM	EBI	PE11	I/O	—	—	A11	O	—	—	—	—	A,I, PD, ST
N16	VDDIOM	EBI	PE12	I/O	—	—	A12	O	—	—	—	—	A,I, PD, ST
R17	VDDIOM	EBI	PE13	I/O	—	—	A13	O	—	—	—	—	A,I, PD, ST
N17	VDDIOM	EBI	PE14	I/O	—	—	A14	O	—	—	—	—	A,I, PD, ST
R18	VDDIOM	EBI	PE15	I/O	—	—	A15	O	SCK3	I/O	—	—	A,I, PD, ST
N18	VDDIOM	EBI	PE16	I/O	—	—	A16	O	CTS3	I	—	—	A,I, PD, ST
P16	VDDIOM	EBI	PE17	I/O	—	—	A17	O	RTS3	O	—	—	A,I, PD, ST
M18	VDDIOM	EBI	PE18	I/O	—	—	A18	O	RXD3	I	—	—	A,I, PD, ST
N15	VDDIOM	EBI	PE19	I/O	—	—	A19	O	TXD3	O	—	—	A,I, PD, ST
M15	VDDIOM	EBI	PE20	I/O	—	—	A20	O	SCK2	I/O	—	—	A,I, PD, ST
N14	VDDIOM	EBI	PE21	I/O	—	—	A21/NANDALE	O	—	—	—	—	A,I, PD, ST
M17	VDDIOM	EBI	PE22	I/O	—	—	A22/NANDCLE	O	—	—	—	—	A,I, PD, ST
M13	VDDIOM	EBI	PE23	I/O	—	—	A23	O	CTS2	I	—	—	A,I, PD, ST
M16	VDDIOM	EBI	PE24	I/O	—	—	A24	O	RTS2	O	—	—	A,I, PD, ST
N12	VDDIOM	EBI	PE25	I/O	—	—	A25	O	RXD2	I	—	—	A,I, PD, ST
M14	VDDIOM	EBI	PE26	I/O	—	—	NCS0	O	TXD2	O	—	—	A,I, PD, ST
M12	VDDIOM	EBI	PE27	I/O	—	—	NCS1	O	TIOA2	I/O	LCDDAT22	O	PIO,I, PD, ST
L13	VDDIOM	EBI	PE28	I/O	—	—	NCS2	O	TIOB2	I/O	LCDDAT23	O	PIO, I, PD, ST
L15	VDDIOM	EBI	PE29	I/O	—	—	NWR1/NBS1	O	TCLK2	I	—	—	PIO, I, PD, ST
L14	VDDIOM	EBI	PE30	I/O	—	—	NWAIT	I	—	—	—	—	PIO, I, PD, ST
L16	VDDIOM	EBI	PE31	I/O	—	—	IRQ	I	PWML1	O	—	—	PIO,I, PD, ST
U15	VDDBU	SYSC	TST	I	—	—	—	—	—	—	—	—	I, PD,
U9	VDDIOP0	SYSC	BMS	I	—	—	—	—	—	—	—	—	I
U8	VDDIOP0	CLOCK	XIN	I	—	—	—	—	—	—	—	—	I
V8	VDDIOP0	CLOCK	XOUT	O	—	—	—	—	—	—	—	—	O
U16	VDDBU	CLOCK	XIN32	I	—	—	—	—	—	—	—	—	I
V16	VDDBU	CLOCK	XOUT32	O	—	—	—	—	—	—	—	—	O
T12	VDDBU	SYSC	SHDN	O	—	—	—	—	—	—	—	—	O
T10	VDDBU	SYSC	WKUP	I	—	—	—	—	—	—	—	—	I, ST
V9	VDDIOP0	RSTJTAG	NRST	I/O	—	—	—	—	—	—	—	—	I, PU, ST
P11	VDDIOP0	RSTJTAG	NTRST	I	—	—	—	—	—	—	—	—	I, PU, ST
R8	VDDIOP0	RSTJTAG	TDI	I	—	—	—	—	—	—	—	—	I, ST

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
M11	VDDIOP0	RSTJTAG	TDO	O	—	—	—	—	—	—	—	—	O
N10	VDDIOP0	RSTJTAG	TMS	I	SWDIO	I/O	—	—	—	—	—	—	I, ST
P9	VDDIOP0	RSTJTAG	TCK	I	SWCLK	I	—	—	—	—	—	—	I, ST
T9	VDDBU	SYSC	JTAGSEL	I	—	—	—	—	—	—	—	—	I, PD
V6	VDDIOP0	DIB	DIBP	O	—	—	—	—	—	—	—	—	O, PU
U6	VDDIOP0	DIB	DIBN	O	—	—	—	—	—	—	—	—	O, PU
K12	VDDIOM	EBI	D0	I/O	—	—	—	—	—	—	—	—	I, PD
K15	VDDIOM	EBI	D1	I/O	—	—	—	—	—	—	—	—	I, PD
K14	VDDIOM	EBI	D2	I/O	—	—	—	—	—	—	—	—	I, PD
K16	VDDIOM	EBI	D3	I/O	—	—	—	—	—	—	—	—	I, PD
K13	VDDIOM	EBI	D4	I/O	—	—	—	—	—	—	—	—	I, PD
K17	VDDIOM	EBI	D5	I/O	—	—	—	—	—	—	—	—	I, PD
J12	VDDIOM	EBI	D6	I/O	—	—	—	—	—	—	—	—	I, PD
K18	VDDIOM	EBI	D7	I/O	—	—	—	—	—	—	—	—	I, PD
J14	VDDIOM	EBI	D8	I/O	—	—	—	—	—	—	—	—	I, PD
J16	VDDIOM	EBI	D9	I/O	—	—	—	—	—	—	—	—	I, PD
J13	VDDIOM	EBI	D10	I/O	—	—	—	—	—	—	—	—	I, PD
J17	VDDIOM	EBI	D11	I/O	—	—	—	—	—	—	—	—	I, PD
J15	VDDIOM	EBI	D12	I/O	—	—	—	—	—	—	—	—	I, PD
J18	VDDIOM	EBI	D13	I/O	—	—	—	—	—	—	—	—	I, PD
H16	VDDIOM	EBI	D14	I/O	—	—	—	—	—	—	—	—	I, PD
H18	VDDIOM	EBI	D15	I/O	—	—	—	—	—	—	—	—	I, PD
L12	VDDIOM	EBI	NCS3/NANDCS	O	—	—	—	—	—	—	—	—	O, PU
L18	VDDIOM	EBI	NANDRDY	I	—	—	—	—	—	—	—	—	I, PU
L17	VDDIOM	EBI	NRD/NANDOE	O	—	—	—	—	—	—	—	—	O, PU
K11	VDDIOM	EBI	NWE/NANDWE	O	—	—	—	—	—	—	—	—	O, PU
C13	VDDIODDR	Reference voltage	DDR_VREF	I	—	—	—	—	—	—	—	—	I
B10	VDDIODDR	DDR_IO	DDR_A0	O	—	—	—	—	—	—	—	—	O
C11	VDDIODDR	DDR_IO	DDR_A1	O	—	—	—	—	—	—	—	—	O
A9	VDDIODDR	DDR_IO	DDR_A2	O	—	—	—	—	—	—	—	—	O
D11	VDDIODDR	DDR_IO	DDR_A3	O	—	—	—	—	—	—	—	—	O
B9	VDDIODDR	DDR_IO	DDR_A4	O	—	—	—	—	—	—	—	—	O
E10	VDDIODDR	DDR_IO	DDR_A5	O	—	—	—	—	—	—	—	—	O
D10	VDDIODDR	DDR_IO	DDR_A6	O	—	—	—	—	—	—	—	—	O
A8	VDDIODDR	DDR_IO	DDR_A7	O	—	—	—	—	—	—	—	—	O
C10	VDDIODDR	DDR_IO	DDR_A8	O	—	—	—	—	—	—	—	—	O
B8	VDDIODDR	DDR_IO	DDR_A9	O	—	—	—	—	—	—	—	—	O
F11	VDDIODDR	DDR_IO	DDR_A10	O	—	—	—	—	—	—	—	—	O
A7	VDDIODDR	DDR_IO	DDR_A11	O	—	—	—	—	—	—	—	—	O
D9	VDDIODDR	DDR_IO	DDR_A12	O	—	—	—	—	—	—	—	—	O
A6	VDDIODDR	DDR_IO	DDR_A13	O	—	—	—	—	—	—	—	—	O
H12	VDDIODDR	DDR_IO	DDR_D0	I/O	—	—	—	—	—	—	—	—	HiZ
H17	VDDIODDR	DDR_IO	DDR_D1	I/O	—	—	—	—	—	—	—	—	HiZ

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
H13	VDDIODDR	DDR_IO	DDR_D2	I/O	—	—	—	—	—	—	—	—	HiZ
G17	VDDIODDR	DDR_IO	DDR_D3	I/O	—	—	—	—	—	—	—	—	HiZ
G16	VDDIODDR	DDR_IO	DDR_D4	I/O	—	—	—	—	—	—	—	—	HiZ
H15	VDDIODDR	DDR_IO	DDR_D5	I/O	—	—	—	—	—	—	—	—	HiZ
F17	VDDIODDR	DDR_IO	DDR_D6	I/O	—	—	—	—	—	—	—	—	HiZ
G15	VDDIODDR	DDR_IO	DDR_D7	I/O	—	—	—	—	—	—	—	—	HiZ
F16	VDDIODDR	DDR_IO	DDR_D8	I/O	—	—	—	—	—	—	—	—	HiZ
E17	VDDIODDR	DDR_IO	DDR_D9	I/O	—	—	—	—	—	—	—	—	HiZ
G14	VDDIODDR	DDR_IO	DDR_D10	I/O	—	—	—	—	—	—	—	—	HiZ
E16	VDDIODDR	DDR_IO	DDR_D11	I/O	—	—	—	—	—	—	—	—	HiZ
D17	VDDIODDR	DDR_IO	DDR_D12	I/O	—	—	—	—	—	—	—	—	HiZ
C18	VDDIODDR	DDR_IO	DDR_D13	I/O	—	—	—	—	—	—	—	—	HiZ
D16	VDDIODDR	DDR_IO	DDR_D14	I/O	—	—	—	—	—	—	—	—	HiZ
C17	VDDIODDR	DDR_IO	DDR_D15	I/O	—	—	—	—	—	—	—	—	HiZ
B16	VDDIODDR	DDR_IO	DDR_D16	I/O	—	—	—	—	—	—	—	—	HiZ
B18	VDDIODDR	DDR_IO	DDR_D17	I/O	—	—	—	—	—	—	—	—	HiZ
C15	VDDIODDR	DDR_IO	DDR_D18	I/O	—	—	—	—	—	—	—	—	HiZ
A18	VDDIODDR	DDR_IO	DDR_D19	I/O	—	—	—	—	—	—	—	—	HiZ
C16	VDDIODDR	DDR_IO	DDR_D20	I/O	—	—	—	—	—	—	—	—	HiZ
C14	VDDIODDR	DDR_IO	DDR_D21	I/O	—	—	—	—	—	—	—	—	HiZ
D15	VDDIODDR	DDR_IO	DDR_D22	I/O	—	—	—	—	—	—	—	—	HiZ
B14	VDDIODDR	DDR_IO	DDR_D23	I/O	—	—	—	—	—	—	—	—	HiZ
A15	VDDIODDR	DDR_IO	DDR_D24	I/O	—	—	—	—	—	—	—	—	HiZ
A14	VDDIODDR	DDR_IO	DDR_D25	I/O	—	—	—	—	—	—	—	—	HiZ
E12	VDDIODDR	DDR_IO	DDR_D26	I/O	—	—	—	—	—	—	—	—	HiZ
A11	VDDIODDR	DDR_IO	DDR_D27	I/O	—	—	—	—	—	—	—	—	HiZ
B11	VDDIODDR	DDR_IO	DDR_D28	I/O	—	—	—	—	—	—	—	—	HiZ
F12	VDDIODDR	DDR_IO	DDR_D29	I/O	—	—	—	—	—	—	—	—	HiZ
A10	VDDIODDR	DDR_IO	DDR_D30	I/O	—	—	—	—	—	—	—	—	HiZ
E11	VDDIODDR	DDR_IO	DDR_D31	I/O	—	—	—	—	—	—	—	—	HiZ
G12	VDDIODDR	DDR_IO	DDR_DQM0	O	—	—	—	—	—	—	—	—	O
E15	VDDIODDR	DDR_IO	DDR_DQM1	O	—	—	—	—	—	—	—	—	O
B15	VDDIODDR	DDR_IO	DDR_DQM2	O	—	—	—	—	—	—	—	—	O
D12	VDDIODDR	DDR_IO	DDR_DQM3	O	—	—	—	—	—	—	—	—	O
E18	VDDIODDR	DDR_IO	DDR_DQS0	I/O	—	—	—	—	—	—	—	—	I, PD
G18	VDDIODDR	DDR_IO	DDR_DQS1	I/O	—	—	—	—	—	—	—	—	I, PD
B17	VDDIODDR	DDR_IO	DDR_DQS2	I/O	—	—	—	—	—	—	—	—	I, PD
B13	VDDIODDR	DDR_IO	DDR_DQS3	I/O	—	—	—	—	—	—	—	—	I, PD
D18	VDDIODDR	DDR_IO	DDR_DQSN0	I/O	—	—	—	—	—	—	—	—	I, PU
F18	VDDIODDR	DDR_IO	DDR_DQSN1	I/O	—	—	—	—	—	—	—	—	I, PU
A17	VDDIODDR	DDR_IO	DDR_DQSN2	I/O	—	—	—	—	—	—	—	—	I, PU
A13	VDDIODDR	DDR_IO	DDR_DQSN3	I/O	—	—	—	—	—	—	—	—	I, PU
C8	VDDIODDR	DDR_IO	DDR_CS	O	—	—	—	—	—	—	—	—	O

Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
B12	VDDIODDR	DDR_IO	DDR_CLK	O	—	—	—	—	—	—	—	—	O
A12	VDDIODDR	DDR_IO	DDR_CLKN	O	—	—	—	—	—	—	—	—	O
B7	VDDIODDR	DDR_IO	DDR_CKE	O	—	—	—	—	—	—	—	—	O
C12	VDDIODDR	DDR_IO	DDR_CALN	I	—	—	—	—	—	—	—	—	O
E13	VDDIODDR	DDR_IO	DDR_CALP	I	—	—	—	—	—	—	—	—	O
G11	VDDIODDR	DDR_IO	DDR_RAS	O	—	—	—	—	—	—	—	—	O
A5	VDDIODDR	DDR_IO	DDR_CAS	O	—	—	—	—	—	—	—	—	O
B5	VDDIODDR	DDR_IO	DDR_WE	O	—	—	—	—	—	—	—	—	O
E9	VDDIODDR	DDR_IO	DDR_BA0	O	—	—	—	—	—	—	—	—	O
B6	VDDIODDR	DDR_IO	DDR_BA1	O	—	—	—	—	—	—	—	—	O
F9	VDDIODDR	DDR_IO	DDR_BA2	O	—	—	—	—	—	—	—	—	O
R11	VBG	VBG	VBG	I	—	—	—	—	—	—	—	—	I
U14	VDDUTMII	USBHS	HHSDPC	I/O	—	—	—	—	—	—	—	—	O, PD
V14	VDDUTMII	USBHS	HHSDMC	I/O	—	—	—	—	—	—	—	—	O, PD
U12	VDDUTMII	USBHS	HHSDPB	I/O	—	—	—	—	—	—	—	—	O, PD
V12	VDDUTMII	USBHS	HHSDMB	I/O	—	—	—	—	—	—	—	—	O, PD
U10	VDDUTMII	USBHS	HHSDPA	I/O	DHSDP	—	—	—	—	—	—	—	O, PD
V10	VDDUTMII	USBHS	HHSDMA	I/O	DHSDM	—	—	—	—	—	—	—	O, PD
V15	VDDBU	power supply	VDDBU	I	—	—	—	—	—	—	—	—	I
T13	GNDBU	ground	GNDBU	I	—	—	—	—	—	—	—	—	I
C5, C7, D14, T15, T7, U17, V7	VDDCORE	power supply	VDDCORE	I	—	—	—	—	—	—	—	—	I
A16, C9, N13, T14, T8, V17	GNDCORE	ground	GNDCORE	I	—	—	—	—	—	—	—	—	I
D13, F14, G10, G13, H11	VDDIODDR	power supply	VDDIODDR	I	—	—	—	—	—	—	—	—	I
E14, F10, F13, F15, H14	GNDIODDR	ground	GNDIODDR	I	—	—	—	—	—	—	—	—	I
P12, T16	VDDIOM	power supply	VDDIOM	I	—	—	—	—	—	—	—	—	I
J11, T17	GNDIOM	ground	GNDIOM	I	—	—	—	—	—	—	—	—	I
G7, V11	VDDIOP0	power supply	VDDIOP0	I	—	—	—	—	—	—	—	—	I
L11, M4	VDDIOP1	power supply	VDDIOP1	I	—	—	—	—	—	—	—	—	I
E5, J7, N11, U7	GNDIOP	Ground	GNDIOP	I	—	—	—	—	—	—	—	—	I
V13	VDDUTMIC	Power supply	VDDUTMIC	I	—	—	—	—	—	—	—	—	I

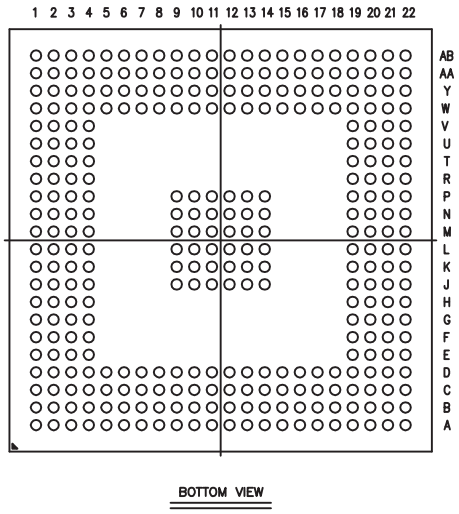
Table 4-1. SAMA5D3 Pinout for 324-ball LFBGA Package (Continued)

Pin	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, HiZ, ST
U13	VDDUTMII	Power supply	VDDUTMII	I	—	—	—	—	—	—	—	—	I
R12	GNDUTMI	Ground	GNDUTMI	I	—	—	—	—	—	—	—	—	I
R10	VDDPLLA	Power supply	VDDPLLA	I	—	—	—	—	—	—	—	—	I
P10	GNDPLL	Ground	GNDPLL	I	—	—	—	—	—	—	—	—	I
U11	VDDOSC	Power supply	VDDOSC	I	—	—	—	—	—	—	—	—	I
T11	GNDOSC	Ground	GNDOSC	I	—	—	—	—	—	—	—	—	I
L6	VDDANA	Power supply	VDDANA	I	—	—	—	—	—	—	—	—	I
L4	GNDANA	Ground	GNDANA	I	—	—	—	—	—	—	—	—	I
L5	VDDANA	Power supply	ADVREF	I	—	—	—	—	—	—	—	—	I
R3	VDDFUSE	Power supply	VDDFUSE	I	—	—	—	—	—	—	—	—	I
P4	GNDFUSE	Ground	GNDFUSE	I	—	—	—	—	—	—	—	—	I

4.3 324-ball TFBGA Package (12 x 12 x 1.2 mm, pitch 0.5 mm)

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

Figure 4-2. 324-ball TFBGA Ball Map



4.4 324-ball TFBGA Package Pinout

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package

Pin	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, HiZ, ST
D2	VDDIOP0	GPIO	PA0	I/O	—	—	LCDDAT0	O	—	—	—	—	PIO, I, PU, ST
G4	VDDIOP0	GPIO	PA1	I/O	—	—	LCDDAT1	O	—	—	—	—	PIO, I, PU, ST
C2	VDDIOP0	GPIO	PA2	I/O	—	—	LCDDAT2	O	—	—	—	—	PIO, I, PU, ST
F3	VDDIOP0	GPIO	PA3	I/O	—	—	LCDDAT3	O	—	—	—	—	PIO, I, PU, ST
F2	VDDIOP0	GPIO	PA4	I/O	—	—	LCDDAT4	O	—	—	—	—	PIO, I, PU, ST
G3	VDDIOP0	GPIO	PA5	I/O	—	—	LCDDAT5	O	—	—	—	—	PIO, I, PU, ST
B1	VDDIOP0	GPIO	PA6	I/O	—	—	LCDDAT6	O	—	—	—	—	PIO, I, PU, ST
G2	VDDIOP0	GPIO	PA7	I/O	—	—	LCDDAT7	O	—	—	—	—	PIO, I, PU, ST
C1	VDDIOP0	GPIO	PA8	I/O	—	—	LCDDAT8	O	—	—	—	—	PIO, I, PU, ST
H3	VDDIOP0	GPIO	PA9	I/O	—	—	LCDDAT9	O	—	—	—	—	PIO, I, PU, ST
D1	VDDIOP0	GPIO	PA10	I/O	—	—	LCDDAT10	O	—	—	—	—	PIO, I, PU, ST
H4	VDDIOP0	GPIO	PA11	I/O	—	—	LCDDAT11	O	—	—	—	—	PIO, I, PU, ST
E2	VDDIOP0	GPIO	PA12	I/O	—	—	LCDDAT12	O	—	—	—	—	PIO, I, PU, ST
K9	VDDIOP0	GPIO	PA13	I/O	—	—	LCDDAT13	O	—	—	—	—	PIO, I, PU, ST
H2	VDDIOP0	GPIO	PA14	I/O	—	—	LCDDAT14	O	—	—	—	—	PIO, I, PU, ST
K4	VDDIOP0	GPIO	PA15	I/O	—	—	LCDDAT15	O	—	—	—	—	PIO, I, PU, ST
G1	VDDIOP0	GPIO	PA16	I/O	—	—	LCDDAT16	O	—	—	ISI_D0	I	PIO, I, PU, ST
K10	VDDIOP0	GPIO	PA17	I/O	—	—	LCDDAT17	O	—	—	ISI_D1	I	PIO, I, PU, ST
F1	VDDIOP0	GPIO	PA18	I/O	—	—	LCDDAT18	O	TWD2	I/O	ISI_D2	I	PIO, I, PU, ST
J4	VDDIOP0	GPIO	PA19	I/O	—	—	LCDDAT19	O	TWCK2	O	ISI_D3	I	PIO, I, PU, ST
J3	VDDIOP0	GPIO	PA20	I/O	—	—	LCDDAT20	O	PWMH0	O	ISI_D4	I	PIO, I, PU, ST
K2	VDDIOP0	GPIO	PA21	I/O	—	—	LCDDAT21	O	PWML0	O	ISI_D5	I	PIO, I, PU, ST
J2	VDDIOP0	GPIO	PA22	I/O	—	—	LCDDAT22	O	PWMH1	O	ISI_D6	I	PIO, I, PU, ST
L9	VDDIOP0	GPIO	PA23	I/O	—	—	LCDDAT23	O	PWML1	O	ISI_D7	I	PIO, I, PU, ST
H1	VDDIOP0	GPIO	PA24	I/O	—	—	LCDPWM	O	—	—	—	—	PIO, I, PU, ST
K3	VDDIOP0	GPIO	PA25	I/O	—	—	LCDDISP	O	—	—	—	—	PIO, I, PU, ST
J1	VDDIOP0	GPIO	PA26	I/O	—	—	LCDVSYNC	O	—	—	—	—	PIO, I, PU, ST
L10	VDDIOP0	GPIO	PA27	I/O	—	—	LCDHSYNC	O	—	—	—	—	PIO, I, PU, ST
K1	VDDIOP0	GPIO_CLK2	PA28	I/O	—	—	LCDPCK	O	—	—	—	—	PIO, I, PU, ST
L3	VDDIOP0	GPIO	PA29	I/O	—	—	LCDDEN	O	—	—	—	—	PIO, I, PU, ST
L2	VDDIOP0	GPIO	PA30	I/O	—	—	TWD0	I/O	URXD1	I	ISI_VSYNC	I	PIO, I, PU, ST
L4	VDDIOP0	GPIO	PA31	I/O	—	—	TWCK0	O	UTXD1	O	ISI_HSYNC	I	PIO, I, PU, ST
AA1	VDDIOP1	GMAC	PB0	I/O	—	—	GTX0	O	PWMH0	O	—	—	PIO, I, PU, ST
W3	VDDIOP1	GMAC	PB1	I/O	—	—	GTX1	O	PWML0	O	—	—	PIO, I, PU, ST
Y2	VDDIOP1	GMAC	PB2	I/O	—	—	GTX2	O	TK1	I/O	—	—	PIO, I, PU, ST
Y3	VDDIOP1	GMAC	PB3	I/O	—	—	GTX3	O	TF1	I/O	—	—	PIO, I, PU, ST
AA2	VDDIOP1	GMAC	PB4	I/O	—	—	GRX0	I	PWMH1	O	—	—	PIO, I, PU, ST
W5	VDDIOP1	GMAC	PB5	I/O	—	—	GRX1	I	PWML1	O	—	—	PIO, I, PU, ST
W7	VDDIOP1	GMAC	PB6	I/O	—	—	GRX2	I	TD1	O	—	—	PIO, I, PU, ST
AB2	VDDIOP1	GMAC	PB7	I/O	—	—	GRX3	I	RK1	I	—	—	PIO, I, PU, ST
AB1	VDDIOP1	GMAC	PB8	I/O	—	—	GTXCK	I	PWMH2	O	—	—	PIO, I, PU, ST
AA3	VDDIOP1	GMAC	PB9	I/O	—	—	GTXEN	O	PWML2	O	—	—	PIO, I, PU, ST

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

			Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		Reset State
Pin	Power Rail	I/O Type	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
W6	VDDIOP1	GMAC	PB10	I/O	—	—	GTXER	O	RF1	I/O	—	—	PIO, I, PU, ST
AB3	VDDIOP1	GMAC	PB11	I/O	—	—	GRXCK	I	RD1	I	—	—	PIO, I, PU, ST
Y5	VDDIOP1	GMAC	PB12	I/O	—	—	GRXDV	I	PWMH3	O	—	—	PIO, I, PU, ST
Y4	VDDIOP1	GMAC	PB13	I/O	—	—	GRXER	I	PWML3	O	—	—	PIO, I, PU, ST
W8	VDDIOP1	GMAC	PB14	I/O	—	—	GCRS	I	CANRX1	I	—	—	PIO, I, PU, ST
AA5	VDDIOP1	GMAC	PB15	I/O	—	—	GCOL	I	CANTX1	O	—	—	PIO, I, PU, ST
AA4	VDDIOP1	GMAC	PB16	I/O	—	—	GMDC	O	—	—	—	—	PIO, I, PU, ST
Y7	VDDIOP1	GMAC	PB17	I/O	—	—	GMDIO	I/O	—	—	—	—	PIO, I, PU, ST
AB4	VDDIOP1	GMAC	PB18	I/O	—	—	G125CK	I	—	—	—	—	PIO, I, PU, ST
Y6	VDDIOP1	GMAC	PB19	I/O	—	—	MCI1_CDA	I/O	GTX4	O	—	—	PIO, I, PU, ST
Y8	VDDIOP1	GMAC	PB20	I/O	—	—	MCI1_DA0	I/O	GTX5	O	—	—	PIO, I, PU, ST
AA6	VDDIOP1	GMAC	PB21	I/O	—	—	MCI1_DA1	I/O	GTX6	O	—	—	PIO, I, PU, ST
W9	VDDIOP1	GMAC	PB22	I/O	—	—	MCI1_DA2	I/O	GTX7	O	—	—	PIO, I, PU, ST
AB6	VDDIOP1	GMAC	PB23	I/O	—	—	MCI1_DA3	I/O	GRX4	I	—	—	PIO, I, PU, ST
AB5	VDDIOP1	GMAC	PB24	I/O	—	—	MCI1_CK	I/O	GRX5	I	—	—	PIO, I, PU, ST
AB7	VDDIOP1	GMAC	PB25	I/O	—	—	SCK1	I/O	GRX6	I	—	—	PIO, I, PU, ST
AA7	VDDIOP1	GMAC	PB26	I/O	—	—	CTS1	I	GRX7	I	—	—	PIO, I, PU, ST
AB8	VDDIOP1	GPIO	PB27	I/O	—	—	RTS1	O	G125CKO	O	—	—	PIO, I, PU, ST
AA8	VDDIOP1	GPIO	PB28	I/O	—	—	RXD1	I	—	—	—	—	PIO, I, PU, ST
Y9	VDDIOP1	GPIO	PB29	I/O	—	—	TXD1	O	—	—	—	—	PIO, I, PU, ST
W10	VDDIOP0	GPIO	PB30	I/O	—	—	DRXD	I	—	—	—	—	PIO, I, PU, ST
Y12	VDDIOP0	GPIO	PB31	I/O	—	—	DTXD	O	—	—	—	—	PIO, I, PU, ST
D10	VDDIOP0	GPIO	PC0	I/O	—	—	ETX0	O	TIOA3	I/O	—	—	PIO, I, PU, ST
B8	VDDIOP0	GPIO	PC1	I/O	—	—	ETX1	O	TIOB3	I/O	—	—	PIO, I, PU, ST
D9	VDDIOP0	GPIO	PC2	I/O	—	—	ERX0	I	TCLK3	I	—	—	PIO, I, PU, ST
C8	VDDIOP0	GPIO	PC3	I/O	—	—	ERX1	I	TIOA4	I/O	—	—	PIO, I, PU, ST
B7	VDDIOP0	GPIO	PC4	I/O	—	—	ETXEN	O	TIOB4	I/O	—	—	PIO, I, PU, ST
D8	VDDIOP0	GPIO	PC5	I/O	—	—	ECRSDV	I	TCLK4	I	—	—	PIO, I, PU, ST
A6	VDDIOP0	GPIO	PC6	I/O	—	—	ERXER	I	TIOA5	I/O	—	—	PIO, I, PU, ST
A7	VDDIOP0	GPIO	PC7	I/O	—	—	EREFCK	I	TIOB5	I/O	—	—	PIO, I, PU, ST
B6	VDDIOP0	GPIO	PC8	I/O	—	—	EMDC	O	TCLK5	I	—	—	PIO, I, PU, ST
D7	VDDIOP0	GPIO	PC9	I/O	—	—	EMDIO	I/O	—	—	—	—	PIO, I, PU, ST
A5	VDDIOP0	GPIO	PC10	I/O	—	—	MCI2_CDA	I/O	—	—	LCDDAT20	O	PIO, I, PU, ST
C7	VDDIOP0	GPIO	PC11	I/O	—	—	MCI2_DA0	I/O	—	—	LCDDAT19	O	PIO, I, PU, ST
B5	VDDIOP0	GPIO	PC12	I/O	—	—	MCI2_DA1	I/O	TIOA1	I/O	LCDDAT18	O	PIO, I, PU, ST
C6	VDDIOP0	GPIO	PC13	I/O	—	—	MCI2_DA2	I/O	TIOB1	I/O	LCDDAT17	O	PIO, I, PU, ST
B4	VDDIOP0	GPIO	PC14	I/O	—	—	MCI2_DA3	I/O	TCLK1	I	LCDDAT16	O	PIO, I, PU, ST
A4	VDDIOP0	MCI_CLK	PC15	I/O	—	—	MCI2_CK	I/O	PCK2	O	LCDDAT21	O	PIO, I, PU, ST
A3	VDDIOP0	GPIO	PC16	I/O	—	—	TK0	I/O	—	—	—	—	PIO, I, PU, ST
C5	VDDIOP0	GPIO	PC17	I/O	—	—	TF0	I/O	—	—	—	—	PIO, I, PU, ST
C4	VDDIOP0	GPIO	PC18	I/O	—	—	TD0	O	—	—	—	—	PIO, I, PU, ST
D6	VDDIOP0	GPIO	PC19	I/O	—	—	RK0	I/O	—	—	—	—	PIO, I, PU, ST
B3	VDDIOP0	GPIO	PC20	I/O	—	—	RF0	I/O	—	—	—	—	PIO, I, PU, ST

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

Pin	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, HiZ, ST
D5	VDDIOP0	GPIO	PC21	I/O	—	—	RD0	I	—	—	—	—	PIO, I, PU, ST
C3	VDDIOP0	GPIO	PC22	I/O	—	—	SPI1_MISO	I/O	—	—	—	—	PIO, I, PU, ST
B2	VDDIOP0	GPIO	PC23	I/O	—	—	SPI1_MOSI	I/O	—	—	—	—	PIO, I, PU, ST
A2	VDDIOP0	GPIO_CLK	PC24	I/O	—	—	SPI1_SPCK	I/O	—	—	—	—	PIO, I, PU, ST
A1	VDDIOP0	GPIO	PC25	I/O	—	—	SPI1_NPCS0	I/O	—	—	—	—	PIO, I, PU, ST
D3	VDDIOP0	GPIO	PC26	I/O	—	—	SPI1_NPCS1	O	TWD1	I/O	ISI_D11	I	PIO, I, PU, ST
D4	VDDIOP0	GPIO	PC27	I/O	—	—	SPI1_NPCS2	O	TWCK1	O	ISI_D10	I	PIO, I, PU, ST
E4	VDDIOP0	GPIO	PC28	I/O	—	—	SPI1_NPCS3	O	PWMFI0	I	ISI_D9	I	PIO, I, PU, ST
E3	VDDIOP0	GPIO	PC29	I/O	—	—	URXD0	I	PWMFI2	I	ISI_D8	I	PIO, I, PU, ST
E1	VDDIOP0	GPIO	PC30	I/O	—	—	UTXD0	O	—	—	ISI_PCK	O	PIO, I, PU, ST
F4	VDDIOP0	GPIO	PC31	I/O	—	—	FIQ	I	PWMFI1	I	—	—	PIO, I, PU, ST
M10	VDDIOP1	GPIO	PD0	I/O	—	—	MCI0_CDA	I/O	—	—	—	—	PIO, I, PU, ST
T1	VDDIOP1	GPIO	PD1	I/O	—	—	MCI0_DA0	I/O	—	—	—	—	PIO, I, PU, ST
R4	VDDIOP1	GPIO	PD2	I/O	—	—	MCI0_DA1	I/O	—	—	—	—	PIO, I, PU, ST
U1	VDDIOP1	GPIO	PD3	I/O	—	—	MCI0_DA2	I/O	—	—	—	—	PIO, I, PU, ST
M9	VDDIOP1	GPIO	PD4	I/O	—	—	MCI0_DA3	I/O	—	—	—	—	PIO, I, PU, ST
V1	VDDIOP1	GPIO	PD5	I/O	—	—	MCI0_DA4	I/O	TIOA0	I/O	PWMH2	O	PIO, I, PU, ST
N10	VDDIOP1	GPIO	PD6	I/O	—	—	MCI0_DA5	I/O	TIOB0	I/O	PWML2	O	PIO, I, PU, ST
W1	VDDIOP1	GPIO	PD7	I/O	—	—	MCI0_DA6	I/O	TCLK0	I	PWMH3	O	PIO, I, PU, ST
R3	VDDIOP1	GPIO	PD8	I/O	—	—	MCI0_DA7	I/O	—	—	PWML3	O	PIO, I, PU, ST
Y1	VDDIOP1	MCI_CLK	PD9	I/O	—	—	MCI0_CK	I/O	—	—	—	—	PIO, I, PU, ST
T3	VDDIOP1	GPIO	PD10	I/O	—	—	SPI0_MISO	I/O	—	—	—	—	PIO, I, PU, ST
T2	VDDIOP1	GPIO	PD11	I/O	—	—	SPI0_MOSI	I/O	—	—	—	—	PIO, I, PU, ST
N9	VDDIOP1	GPIO_CLK	PD12	I/O	—	—	SPI0_SPCK	I/O	—	—	—	—	PIO, I, PU, ST
U2	VDDIOP1	GPIO	PD13	I/O	—	—	SPI0_NPCS0	I/O	—	—	—	—	PIO, I, PU, ST
T4	VDDIOP1	GPIO	PD14	I/O	—	—	SCK0	I/O	SPI0_NPCS1	O	CANRX0	I	PIO, I, PU, ST
V2	VDDIOP1	GPIO	PD15	I/O	—	—	CTS0	I	SPI0_NPCS2	O	CANTX0	O	PIO, I, PU, ST
U3	VDDIOP1	GPIO	PD16	I/O	—	—	RTS0	O	SPI0_NPCS3	O	PWMFI3	I	PIO, I, PU, ST
V3	VDDIOP1	GPIO	PD17	I/O	—	—	RXD0	I	—	—	—	—	PIO, I, PU, ST
U4	VDDIOP1	GPIO	PD18	I/O	—	—	TXD0	O	—	—	—	—	PIO, I, PU, ST
W2	VDDIOP1	GPIO	PD19	I/O	—	—	ADTRG	I	—	—	—	—	PIO, I, PU, ST
P3	VDDANA	GPIO_ANA	PD20	I/O	—	—	AD0	I	—	—	—	—	PIO, I, PU, ST
R2	VDDANA	GPIO_ANA	PD21	I/O	—	—	AD1	I	—	—	—	—	PIO, I, PU, ST
P2	VDDANA	GPIO_ANA	PD22	I/O	—	—	AD2	I	—	—	—	—	PIO, I, PU, ST
R1	VDDANA	GPIO_ANA	PD23	I/O	—	—	AD3	I	—	—	—	—	PIO, I, PU, ST
P1	VDDANA	GPIO_ANA	PD24	I/O	—	—	AD4	I	—	—	—	—	PIO, I, PU, ST
N3	VDDANA	GPIO_ANA	PD25	I/O	—	—	AD5	I	—	—	—	—	PIO, I, PU, ST
N1	VDDANA	GPIO_ANA	PD26	I/O	—	—	AD6	I	—	—	—	—	PIO, I, PU, ST
N2	VDDANA	GPIO_ANA	PD27	I/O	—	—	AD7	I	—	—	—	—	PIO, I, PU, ST
M2	VDDANA	GPIO_ANA	PD28	I/O	—	—	AD8	I	—	—	—	—	PIO, I, PU, ST
M1	VDDANA	GPIO_ANA	PD29	I/O	—	—	AD9	I	—	—	—	—	PIO, I, PU, ST
M3	VDDANA	GPIO_ANA	PD30	I/O	—	—	AD10	I	PCK0	O	—	—	PIO, I, PU, ST
L1	VDDANA	GPIO_ANA	PD31	I/O	—	—	AD11	I	PCK1	O	—	—	PIO, I, PU, ST

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

Pin	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, HiZ, ST
W17	VDDIOM	EBI	PE0	I/O	—	—	A0/NBS0	O	—	—	—	—	A,I, PD, ST
Y18	VDDIOM	EBI	PE1	I/O	—	—	A1	O	—	—	—	—	A,I, PD, ST
W18	VDDIOM	EBI	PE2	I/O	—	—	A2	O	—	—	—	—	A,I, PD, ST
AA21	VDDIOM	EBI	PE3	I/O	—	—	A3	O	—	—	—	—	A,I, PD, ST
Y16	VDDIOM	EBI	PE4	I/O	—	—	A4	O	—	—	—	—	A,I, PD, ST
Y20	VDDIOM	EBI	PE5	I/O	—	—	A5	O	—	—	—	—	A,I, PD, ST
W19	VDDIOM	EBI	PE6	I/O	—	—	A6	O	—	—	—	—	A,I, PD, ST
Y22	VDDIOM	EBI	PE7	I/O	—	—	A7	O	—	—	—	—	A,I, PD, ST
Y21	VDDIOM	EBI	PE8	I/O	—	—	A8	O	—	—	—	—	A,I, PD, ST
W22	VDDIOM	EBI	PE9	I/O	—	—	A9	O	—	—	—	—	A,I, PD, ST
V19	VDDIOM	EBI	PE10	I/O	—	—	A10	O	—	—	—	—	A,I, PD, ST
W20	VDDIOM	EBI	PE11	I/O	—	—	A11	O	—	—	—	—	A,I, PD, ST
W21	VDDIOM	EBI	PE12	I/O	—	—	A12	O	—	—	—	—	A,I, PD, ST
T19	VDDIOM	EBI	PE13	I/O	—	—	A13	O	—	—	—	—	A,I, PD, ST
V22	VDDIOM	EBI	PE14	I/O	—	—	A14	O	—	—	—	—	A,I, PD, ST
V20	VDDIOM	EBI	PE15	I/O	—	—	A15	O	SCK3	I/O	—	—	A,I, PD, ST
V21	VDDIOM	EBI	PE16	I/O	—	—	A16	O	CTS3	I	—	—	A,I, PD, ST
T20	VDDIOM	EBI	PE17	I/O	—	—	A17	O	RTS3	O	—	—	A,I, PD, ST
U20	VDDIOM	EBI	PE18	I/O	—	—	A18	O	RXD3	I	—	—	A,I, PD, ST
U21	VDDIOM	EBI	PE19	I/O	—	—	A19	O	TXD3	O	—	—	A,I, PD, ST
U22	VDDIOM	EBI	PE20	I/O	—	—	A20	O	SCK2	I/O	—	—	A,I, PD, ST
R19	VDDIOM	EBI	PE21	I/O	—	—	A21/NANDALE	O	—	—	—	—	A,I, PD, ST
R20	VDDIOM	EBI	PE22	I/O	—	—	A22/NANDCLE	O	—	—	—	—	A,I, PD, ST
T21	VDDIOM	EBI	PE23	I/O	—	—	A23	O	CTS2	I	—	—	A,I, PD, ST
T22	VDDIOM	EBI	PE24	I/O	—	—	A24	O	RTS2	O	—	—	A,I, PD, ST
P19	VDDIOM	EBI	PE25	I/O	—	—	A25	O	RXD2	I	—	—	A,I, PD, ST
R22	VDDIOM	EBI	PE26	I/O	—	—	NCS0	O	TXD2	O	—	—	A,I, PD, ST
R21	VDDIOM	EBI	PE27	I/O	—	—	NCS1	O	TIOA2	I/O	LCDDAT22	O	PIO,I, PD, ST
P20	VDDIOM	EBI	PE28	I/O	—	—	NCS2	O	TIOB2	I/O	LCDDAT23	O	PIO, I, PD, ST
P21	VDDIOM	EBI	PE29	I/O	—	—	NWR1/NBS1	O	TCLK2	I	—	—	PIO, I, PD, ST
N19	VDDIOM	EBI	PE30	I/O	—	—	NWAIT	I	—	—	—	—	PIO, I, PD, ST
N21	VDDIOM	EBI	PE31	I/O	—	—	IRQ	I	PWML1	O	—	—	PIO,I, PD, ST
Y15	VDDBU	SYSC	TST	I	—	—	—	—	—	—	—	—	I, PD,
AB14	VDDIOP0	SYSC	BMS	I	—	—			—	—	—	—	I
AB11	VDDIOP0	CLOCK	XIN	I	—	—	—	—	—	—	—	—	I
AA11	VDDIOP0	CLOCK	XOUT	O	—	—			—	—	—	—	O
AB19	VDDBU	CLOCK	XIN32	I	—	—	—	—	—	—	—	—	I
AA19	VDDBU	CLOCK	XOUT32	O	—	—			—	—	—	—	O
W16	VDDBU	SYSC	SHDN	O	—	—	—	—	—	—	—	—	O
AB16	VDDBU	SYSC	WKUP	I	—	—			—	—	—	—	I, ST
Y13	VDDIOP0	RSTJTAG	NRST	I/O	—	—	—	—	—	—	—	—	I, PU, ST
AA14	VDDIOP0	RSTJTAG	NTRST	I	—	—	—	—	—	—	—	—	I, PU, ST
W13	VDDIOP0	RSTJTAG	TDI	I	—	—	—	—	—	—	—	—	I, ST

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

Pin	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, HiZ, ST
W11	VDDIOP0	RSTJTAG	TDO	O	—	—	—	—	—	—	—	—	O
W12	VDDIOP0	RSTJTAG	TMS	I	SWDIO	I/O	—	—	—	—	—	—	I, ST
Y14	VDDIOP0	RSTJTAG	TCK	I	SWCLK	I	—	—	—	—	—	—	I, ST
AA16	VDDBU	SYSC	JTAGSEL	I	—	—	—	—	—	—	—	—	I, PD
AA9	VDDIOP0	DIB	DIBP	O	—	—	—	—	—	—	—	—	O, PU
AB9	VDDIOP0	DIB	DIBN	O	—	—	—	—	—	—	—	—	O, PU
M19	VDDIOM	EBI	D0	I/O	—	—	—	—	—	—	—	—	I, PD
M22	VDDIOM	EBI	D1	I/O	—	—	—	—	—	—	—	—	I, PD
M20	VDDIOM	EBI	D2	I/O	—	—	—	—	—	—	—	—	I, PD
L22	VDDIOM	EBI	D3	I/O	—	—	—	—	—	—	—	—	I, PD
L20	VDDIOM	EBI	D4	I/O	—	—	—	—	—	—	—	—	I, PD
L21	VDDIOM	EBI	D5	I/O	—	—	—	—	—	—	—	—	I, PD
K21	VDDIOM	EBI	D6	I/O	—	—	—	—	—	—	—	—	I, PD
H22	VDDIOM	EBI	D7	I/O	—	—	—	—	—	—	—	—	I, PD
L19	VDDIOM	EBI	D8	I/O	—	—	—	—	—	—	—	—	I, PD
J22	VDDIOM	EBI	D9	I/O	—	—	—	—	—	—	—	—	I, PD
K19	VDDIOM	EBI	D10	I/O	—	—	—	—	—	—	—	—	I, PD
J21	VDDIOM	EBI	D11	I/O	—	—	—	—	—	—	—	—	I, PD
K22	VDDIOM	EBI	D12	I/O	—	—	—	—	—	—	—	—	I, PD
H20	VDDIOM	EBI	D13	I/O	—	—	—	—	—	—	—	—	I, PD
K20	VDDIOM	EBI	D14	I/O	—	—	—	—	—	—	—	—	I, PD
J20	VDDIOM	EBI	D15	I/O	—	—	—	—	—	—	—	—	I, PD
N20	VDDIOM	EBI	NCS3/NANDCS	O	—	—	—	—	—	—	—	—	O, PU
M21	VDDIOM	EBI	NANDRDY	I	—	—	—	—	—	—	—	—	I, PU
N22	VDDIOM	EBI	NRD/NANDOE	O	—	—	—	—	—	—	—	—	O, PU
P22	VDDIOM	EBI	NWE/NANDWE	O	—	—	—	—	—	—	—	—	O, PU
J13, J14	VDDIODDR	Reference voltage	DDR_VREF	I	—	—	—	—	—	—	—	—	I
B13	VDDIODDR	DDR_IO	DDR_A0	O	—	—	—	—	—	—	—	—	O
C14	VDDIODDR	DDR_IO	DDR_A1	O	—	—	—	—	—	—	—	—	O
B16	VDDIODDR	DDR_IO	DDR_A2	O	—	—	—	—	—	—	—	—	O
C13	VDDIODDR	DDR_IO	DDR_A3	O	—	—	—	—	—	—	—	—	O
A14	VDDIODDR	DDR_IO	DDR_A4	O	—	—	—	—	—	—	—	—	O
D13	VDDIODDR	DDR_IO	DDR_A5	O	—	—	—	—	—	—	—	—	O
C12	VDDIODDR	DDR_IO	DDR_A6	O	—	—	—	—	—	—	—	—	O
B12	VDDIODDR	DDR_IO	DDR_A7	O	—	—	—	—	—	—	—	—	O
D12	VDDIODDR	DDR_IO	DDR_A8	O	—	—	—	—	—	—	—	—	O
A13	VDDIODDR	DDR_IO	DDR_A9	O	—	—	—	—	—	—	—	—	O
C11	VDDIODDR	DDR_IO	DDR_A10	O	—	—	—	—	—	—	—	—	O
B11	VDDIODDR	DDR_IO	DDR_A11	O	—	—	—	—	—	—	—	—	O
A12	VDDIODDR	DDR_IO	DDR_A12	O	—	—	—	—	—	—	—	—	O
A11	VDDIODDR	DDR_IO	DDR_A13	O	—	—	—	—	—	—	—	—	O
J19	VDDIODDR	DDR_IO	DDR_D0	I/O	—	—	—	—	—	—	—	—	HiZ
H21	VDDIODDR	DDR_IO	DDR_D1	I/O	—	—	—	—	—	—	—	—	HiZ

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

Pin	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, HiZ, ST
F20	VDDIODDR	DDR_IO	DDR_D2	I/O	—	—	—	—	—	—	—	—	HiZ
G20	VDDIODDR	DDR_IO	DDR_D3	I/O	—	—	—	—	—	—	—	—	HiZ
F21	VDDIODDR	DDR_IO	DDR_D4	I/O	—	—	—	—	—	—	—	—	HiZ
H19	VDDIODDR	DDR_IO	DDR_D5	I/O	—	—	—	—	—	—	—	—	HiZ
G21	VDDIODDR	DDR_IO	DDR_D6	I/O	—	—	—	—	—	—	—	—	HiZ
D21	VDDIODDR	DDR_IO	DDR_D7	I/O	—	—	—	—	—	—	—	—	HiZ
G19	VDDIODDR	DDR_IO	DDR_D8	I/O	—	—	—	—	—	—	—	—	HiZ
D20	VDDIODDR	DDR_IO	DDR_D9	I/O	—	—	—	—	—	—	—	—	HiZ
C22	VDDIODDR	DDR_IO	DDR_D10	I/O	—	—	—	—	—	—	—	—	HiZ
C20	VDDIODDR	DDR_IO	DDR_D11	I/O	—	—	—	—	—	—	—	—	HiZ
B21	VDDIODDR	DDR_IO	DDR_D12	I/O	—	—	—	—	—	—	—	—	HiZ
C21	VDDIODDR	DDR_IO	DDR_D13	I/O	—	—	—	—	—	—	—	—	HiZ
D19	VDDIODDR	DDR_IO	DDR_D14	I/O	—	—	—	—	—	—	—	—	HiZ
F19	VDDIODDR	DDR_IO	DDR_D15	I/O	—	—	—	—	—	—	—	—	HiZ
B20	VDDIODDR	DDR_IO	DDR_D16	I/O	—	—	—	—	—	—	—	—	HiZ
E21	VDDIODDR	DDR_IO	DDR_D17	I/O	—	—	—	—	—	—	—	—	HiZ
E19	VDDIODDR	DDR_IO	DDR_D18	I/O	—	—	—	—	—	—	—	—	HiZ
C17	VDDIODDR	DDR_IO	DDR_D19	I/O	—	—	—	—	—	—	—	—	HiZ
D18	VDDIODDR	DDR_IO	DDR_D20	I/O	—	—	—	—	—	—	—	—	HiZ
A18	VDDIODDR	DDR_IO	DDR_D21	I/O	—	—	—	—	—	—	—	—	HiZ
C19	VDDIODDR	DDR_IO	DDR_D22	I/O	—	—	—	—	—	—	—	—	HiZ
C18	VDDIODDR	DDR_IO	DDR_D23	I/O	—	—	—	—	—	—	—	—	HiZ
C16	VDDIODDR	DDR_IO	DDR_D24	I/O	—	—	—	—	—	—	—	—	HiZ
A21	VDDIODDR	DDR_IO	DDR_D25	I/O	—	—	—	—	—	—	—	—	HiZ
D15	VDDIODDR	DDR_IO	DDR_D26	I/O	—	—	—	—	—	—	—	—	HiZ
A20	VDDIODDR	DDR_IO	DDR_D27	I/O	—	—	—	—	—	—	—	—	HiZ
B14	VDDIODDR	DDR_IO	DDR_D28	I/O	—	—	—	—	—	—	—	—	HiZ
A22	VDDIODDR	DDR_IO	DDR_D29	I/O	—	—	—	—	—	—	—	—	HiZ
A16	VDDIODDR	DDR_IO	DDR_D30	I/O	—	—	—	—	—	—	—	—	HiZ
D14	VDDIODDR	DDR_IO	DDR_D31	I/O	—	—	—	—	—	—	—	—	HiZ
E20	VDDIODDR	DDR_IO	DDR_DQM0	O	—	—	—	—	—	—	—	—	O
B22	VDDIODDR	DDR_IO	DDR_DQM1	O	—	—	—	—	—	—	—	—	O
B18	VDDIODDR	DDR_IO	DDR_DQM2	O	—	—	—	—	—	—	—	—	O
C15	VDDIODDR	DDR_IO	DDR_DQM3	O	—	—	—	—	—	—	—	—	O
G22	VDDIODDR	DDR_IO	DDR_DQS0	I/O	—	—	—	—	—	—	—	—	I, PD
E22	VDDIODDR	DDR_IO	DDR_DQS1	I/O	—	—	—	—	—	—	—	—	I, PD
A19	VDDIODDR	DDR_IO	DDR_DQS2	I/O	—	—	—	—	—	—	—	—	I, PD
B17	VDDIODDR	DDR_IO	DDR_DQS3	I/O	—	—	—	—	—	—	—	—	I, PD
F22	VDDIODDR	DDR_IO	DDR_DQSN0	I/O	—	—	—	—	—	—	—	—	I, PU
D22	VDDIODDR	DDR_IO	DDR_DQSN1	I/O	—	—	—	—	—	—	—	—	I, PU
B19	VDDIODDR	DDR_IO	DDR_DQSN2	I/O	—	—	—	—	—	—	—	—	I, PU
A17	VDDIODDR	DDR_IO	DDR_DQSN3	I/O	—	—	—	—	—	—	—	—	I, PU
C9	VDDIODDR	DDR_IO	DDR_CS	O	—	—	—	—	—	—	—	—	O

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

Pin	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, HiZ, ST
D17	VDDIODDR	DDR_IO	DDR_CLK	O	—	—	—	—	—	—	—	—	O
D16	VDDIODDR	DDR_IO	DDR_CLKN	O	—	—	—	—	—	—	—	—	O
A9	VDDIODDR	DDR_IO	DDR_CKE	O	—	—	—	—	—	—	—	—	O
A15	VDDIODDR	DDR_IO	DDR_CALN	I	—	—	—	—	—	—	—	—	O
B15	VDDIODDR	DDR_IO	DDR_CALP	I	—	—	—	—	—	—	—	—	O
B10	VDDIODDR	DDR_IO	DDR_RAS	O	—	—	—	—	—	—	—	—	O
B9	VDDIODDR	DDR_IO	DDR_CAS	O	—	—	—	—	—	—	—	—	O
A8	VDDIODDR	DDR_IO	DDR_WE	O	—	—	—	—	—	—	—	—	O
D11	VDDIODDR	DDR_IO	DDR_BA0	O	—	—	—	—	—	—	—	—	O
A10	VDDIODDR	DDR_IO	DDR_BA1	O	—	—	—	—	—	—	—	—	O
C10	VDDIODDR	DDR_IO	DDR_BA2	O	—	—	—	—	—	—	—	—	O
P12	VBG	VBG	VBG	I	—	—	—	—	—	—	—	—	I
AA17	VDDUTMII	USBHS	HHSDPC	I/O	—	—	—	—	—	—	—	—	O, PD
AB17	VDDUTMII	USBHS	HHSDMC	I/O	—	—	—	—	—	—	—	—	O, PD
AA15	VDDUTMII	USBHS	HHSDPB	I/O	—	—	—	—	—	—	—	—	O, PD
AB15	VDDUTMII	USBHS	HHSDMB	I/O	—	—	—	—	—	—	—	—	O, PD
AA13	VDDUTMII	USBHS	HHSDPA	I/O	DHSDP	—	—	—	—	—	—	—	O, PD
AB13	VDDUTMII	USBHS	HHSDMA	I/O	DHSDM	—	—	—	—	—	—	—	O, PD
N13	VDDBU	Power supply	VDDBU	I	—	—	—	—	—	—	—	—	I
N12	GNDBU	Ground	GNDBU	I	—	—	—	—	—	—	—	—	I
Y17, Y19, AA20, AA22, AB20, AB22	VDDCORE	Power supply	VDDCORE	I	—	—	—	—	—	—	—	—	I
Y10, Y11, AA10, AA12, AB10, AB12	GNDCORE	Ground	GNDCORE	I	—	—	—	—	—	—	—	—	I
J12, K12, K13, K14, L12	VDDIODDR	Power supply	VDDIODDR	I	—	—	—	—	—	—	—	—	I
L13, L14, M12, M13, N11	GNDIODDR	Ground	GNDIODDR	I	—	—	—	—	—	—	—	—	I
M14, U19	VDDIOM	Power supply	VDDIOM	I	—	—	—	—	—	—	—	—	I
N14, P14	GNDIOM	Ground	GNDIOM	I	—	—	—	—	—	—	—	—	I
J9, J10	VDDIOP0	Power supply	VDDIOP0	I	—	—	—	—	—	—	—	—	I
P9, P10	VDDIOP1	Power supply	VDDIOP1	I	—	—	—	—	—	—	—	—	I
J11, K11, L11, M11	GNDIOP	Ground	GNDIOP	I	—	—	—	—	—	—	—	—	I
AB18	VDDUTMIC	Power supply	VDDUTMIC	I	—	—	—	—	—	—	—	—	I
AA18	VDDUTMII	Power supply	VDDUTMII	I	—	—	—	—	—	—	—	—	I

Table 4-2. SAMA5D3 Pinout for 324-ball TFBGA Package (Continued)

Pin	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, HiZ, ST
P13	GNDUTMI	Ground	GNDUTMI	I	—	—	—	—	—	—	—	—	I
W14	VDDPLLA	Power supply	VDDPLLA	I	—	—	—	—	—	—	—	—	I
W15	GNDPLL	Ground	GNDPLL	I	—	—	—	—	—	—	—	—	I
AB21	VDDOSC	Power supply	VDDOSC	I	—	—	—	—	—	—	—	—	I
P11	GNDOSC	Ground	GNDOSC	I	—	—	—	—	—	—	—	—	I
M4	VDDANA	Power supply	VDDANA	I	—	—	—	—	—	—	—	—	I
P4	GNDANA	Ground	GNDANA	I	—	—	—	—	—	—	—	—	I
N4	VDDANA	Power supply	ADVREF	I	—	—	—	—	—	—	—	—	I
W4	VDDFUSE	Power supply	VDDFUSE	I	—	—	—	—	—	—	—	—	I
V4	GNDFUSE	Ground	GNDFUSE	I	—	—	—	—	—	—	—	—	I

4.5 Input/Output Description

Table 4-3. SAMA5 I/O Types Description

I/O Type	Voltage Range	Analog	Pull-up	Pull-up Typ. Value (Ohm)	Pull-down	Pull-down Typ. Value (Ohm)	Schmitt Trigger
GPIO	1.65–3.6V	—	Switchable	100K	Switchable	100K	Switchable
GPIO_CLK	1.65–3.6V	—	Switchable	100K	Switchable	100K	Switchable
GPIO_CLK2	1.65–3.6V	—	Switchable	100K	Switchable	100K	Switchable
GPIO_ANA	3.0–3.6V	I	Switchable	100K	—	—	Switchable
EBI	1.65–1.95V, 3.0–3.6V	—	Switchable	100K	Switchable	100K	—
RSTJTAG	3.0–3.6V	—	Reset State	100K	Reset State	100K	Reset State
SYSC	1.65–3.6V	—	—	—	Reset State	15K	Reset State
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 indicated, the configuration is defined by the “Reset State” column of the Pin Description table (see Table 4-1 on page 11).

Table 4-4. SAMA5 I/O Type Assignation and Frequency

I/O Type	Max I/O Frequency (MHz)	Load (pF)	Fan-out	Drive Control	Signal Name
GPIO	33	40	—	High/Medium/Low	All PIO lines except the lines indicated further on in this table
MCI_CLK	52	20	—	High/Medium/Low	MCI0CK, MCI1CK, MCI2CK
GPIO_CLK	66	20	—	High/Medium/Low	SPI0CK, SPI1CK, ETXCLK, ERXCLK
GPIO_CLK2	75	20	—	High/Medium/Low	LCDDOTCK
GPIO_ANA	25	20	16 mA, 40 mA (peak)	Fixed to Medium	ADx
EBI	66	50	—	High/Medium/Low 1.8V/3.3V	All EBI signals
DDR_IO	166	20	—	High/Medium/Low	All DDR signals
RST	3	10	—	Fixed to Low	NRST, NTRST, BMS
JTAG	10	10	—	Fixed to Medium	TCK, TDI, TMS, TDO
SYSC	0.25	10	—	No	WKUP, SHDN, JTAGSEL, TST, SHDN
VBG	0.25	10	—	No	VBG
USBHS	480	20	—	No	HHSDPC, HHSDPB, HHSDPA/DHSDP, HHSDMC, HHSDMB, HHSDMA/DHSDM
CLOCK	50	50	—	No	XIN, XOUT, XIN32, XOUT32
GMAC	125	15	—	High/Medium/Low	Gigabit Ethernet I/Os

5. Power Considerations

5.1 Power Supplies

Table 5-1 defines the power supply rails and the estimated power consumption at typical voltage.

Table 5-1. SAMA5D3 Power supplies

Name	Voltage Range, Nominal	Associated Ground	Powers
VDDCORE	1.1–1.32V, 1.2V	GNDCORE	The core, including the processor, the embedded memories and the peripherals
VDDIODDR	1.7–1.9V, 1.8V 1.14–1.30, 1.2V	GNDIODDR	LPDDR/DDR2 Interface I/O lines LPDDR2 Interface I/O lines
VDDIOM	1.65–1.95V, 1.8V 3.0–3.6V, 3.3V	GNDIOM	NAND and HSMC Interface I/O lines
VDDIOP0	1.65–3.6V	GNDIOP	Peripheral I/O lines
VDDIOP1	1.65–3.6V	GNDIOP	Peripheral I/O lines
VDDBU	1.65–3.6V	GNDBU	The Slow Clock Oscillator, the internal 32 kHz RC Oscillator and a part of the System Controller
VDDUTMIC	1.1–1.32V, 1.2V	GNDUTMI	The USB device and host UTMI+ core The UTMI PLL
VDDUTMII	3.0–3.6V, 3.3V	GNDUTMI	The USB device and host UTMI+ interface
VDDPLLA	1.1–1.32V, 1.2V	GNDPLL	The PLLA cell
VDDOSC	1.65–3.6V	GNDOSC	Main Oscillator Cell and PLL UTMI. If PLL UTMI is used the range is to be 3.0V to 3.6V.
VDDANA	3.0–3.6V, 3.3V	GNDANA	The Analog-to-Digital Converter
VDDFUSE	2.25–2.75V, 2.5V	GNDFUSE	Fuse box for programming. It can be tied to ground with a 100 Ω resistor for fuse reading only.

5.2 Power-up Consideration

The user must first activate VDDIOP and VDDIOM, then VDDPLL and VDDCORE with the constraint that VDDPLL is established no later than 1 ms after VDDCORE.

The VDDCORE and VDDBU power supplies rising time must be defined according to the Core and Backup Power-On-Reset characteristics to ensure VDDCORE or VDDBU has reached V_{IH} after the POR reset time.

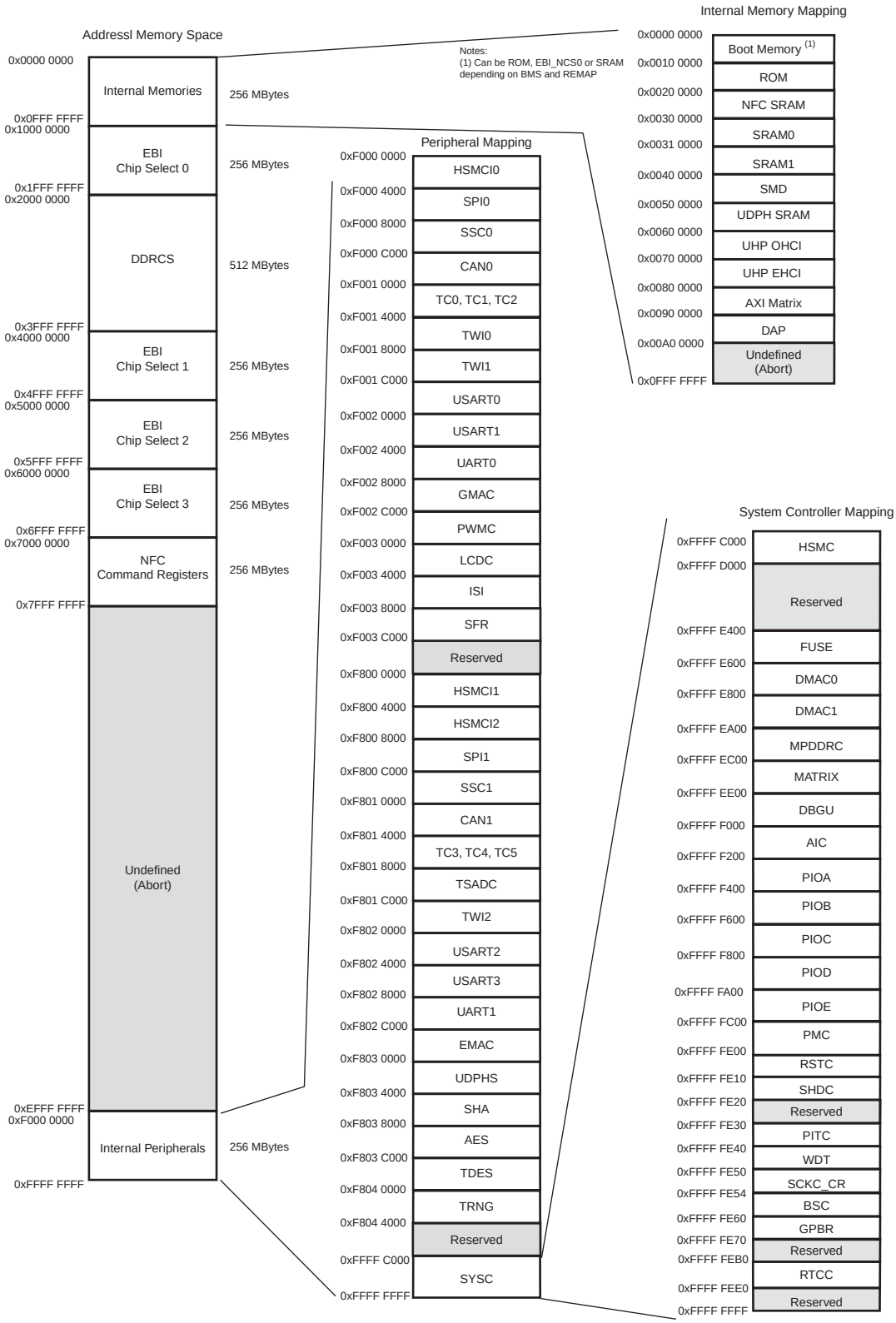
Please refer to the “Core Power Supply POR Characteristics” and “Backup Power Supply POR Characteristics” sections of the product datasheet for power-up constraints.

5.3 Power-down Consideration

The user must remove VDDPLL first, then VDDCORE, and at last VDDIOP and VDDIOM, to ensure a reliable operation of the device.

6. Memories

Figure 6-1. Memory Mapping



6.1 Embedded Memories

6.1.1 Internal SRAM

The SAMA5D3 product embeds a total of 128 Kbytes high-speed SRAM0 and SRAM1. After Remap the SRAM is accessible at address 0 but also at address 0x00300000. Only the ARM core has access to the SRAM at address 0. The others masters (DMA, peripherals, etc.) always access the SRAM at address 0x00300000.

SRAM0 and SRAM1 can be accessed in parallel to improve the overall bandwidth of the system.

6.1.2 Internal ROM

The SAMA5D3 product embeds one 160-Kbyte internal ROM containing a standard and a secure bootloader. The secure bootloader is described in a separate document, under NDA. The standard bootloader supports booting from:

- 8-bit NAND Flash with ECC management
- SPI Serial Flash
- SDCARD
- EMMC
- TWI EEPROM

The boot sequence can be selected using the boot order facility (Boot Select Control Register). The internal ROM embeds Galois field tables that are used to compute NandFlash ECC. Please refer to Figure 12-9 “Galois Field Table Mapping” in the “Boot Strategies” section of this datasheet.

6.1.3 Boot Strategies

For standard boot strategies, please refer to the “Boot Strategies” section of this datasheet.

For secure boot strategies, please refer to the Application Note “Secure Boot on SAMA5D3 Series” (NDA required).

6.2 External Memory

The SAMA5D3 features interfaces to offer connexion to a wide range of external memories or to parallel peripherals.

6.2.1 DDR2/LPDDR/LPDDR2 Interface

- 32-bit external interface
- 512 Mbytes address space on CS1
- Supports DDR2, LPDDR and LPDDR2 memories
- Drive level control
- I/O impedance control embedded
- Supports 4-banks and 8-banks and up to 512 Mbytes
- Multi-port

6.2.2 Static Memories and NAND Flash

The static memory controller is dedicated to interfacing external memory devices:

The static memory controller is able to drive up to four chip selects. NCS3 is dedicated to the NAND Flash control.

- Asynchronous SRAM-like memories and parallel peripherals
- NAND Flash (8-bit MLC and SLC)

The HSMC embeds a NAND Flash Controller (NFC). The NFC can handle automatic transfers, sending the commands and address cycles to the NAND Flash and transferring the contents of the page (for read and write) to the NFC SRAM. It minimizes the CPU overhead.

In order to improve overall system performance the DATA phase of the transfer can be DMA assisted. The static memory embeds a NAND Flash Error Correcting Code controller with the features as follows:

- Algorithm based on BCH codes
- Supports also SLC 1-bit (BCH 2-bit), SLC 4-bit (BCH 4-bit)
- Programmable Error Correcting Capability:
  - 2-bit, 4-bit, 8-bit and 16-bit errors for 512 bytes/sector (4 Kbyte page)
  - 24-bit error for 1024 bytes/sector (8 Kbyte page)
- Programmable sector size: 512 bytes or 1024 bytes
- Programmable number of sector per page: 1, 2, 4 or 8 blocks of data per page
- Programmable spare area size
- Supports spare area ECC protection
- Supports 8 Kbyte page size using 1024 bytes/sector and 4 Kbyte page size using 512 bytes/sector
- Error detection is interrupt driven
- Provides hardware acceleration for error location
- Finds roots of error-locator polynomial
- Programmable number of roots

7. Real-time Event Management

The events generated by peripherals are designed to be directly routed to peripherals managing/using these events without processor intervention. Peripherals receiving events contain logic by which to select the one required.

7.1 Embedded Characteristics

Peripherals generate event triggers which are directly routed to event managers such as ADC, for example, to start measurement/conversion without processor intervention.

7.2 Real-time Event Mapping List

Table 7-1. Real-time Event Mapping List

Event Generator	Event Manager	Function
PMC	Pulse Width Modulation (PWM)	Safety / Puts the PWM Outputs in Safe Mode (Main Crystal Clock Failure Detection)
Analog-to-Digital Converter (ADC)	PWM	Safety / Puts the PWM Outputs in Safe Mode (Overspeed, Overcurrent detection, etc.)

8. 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.

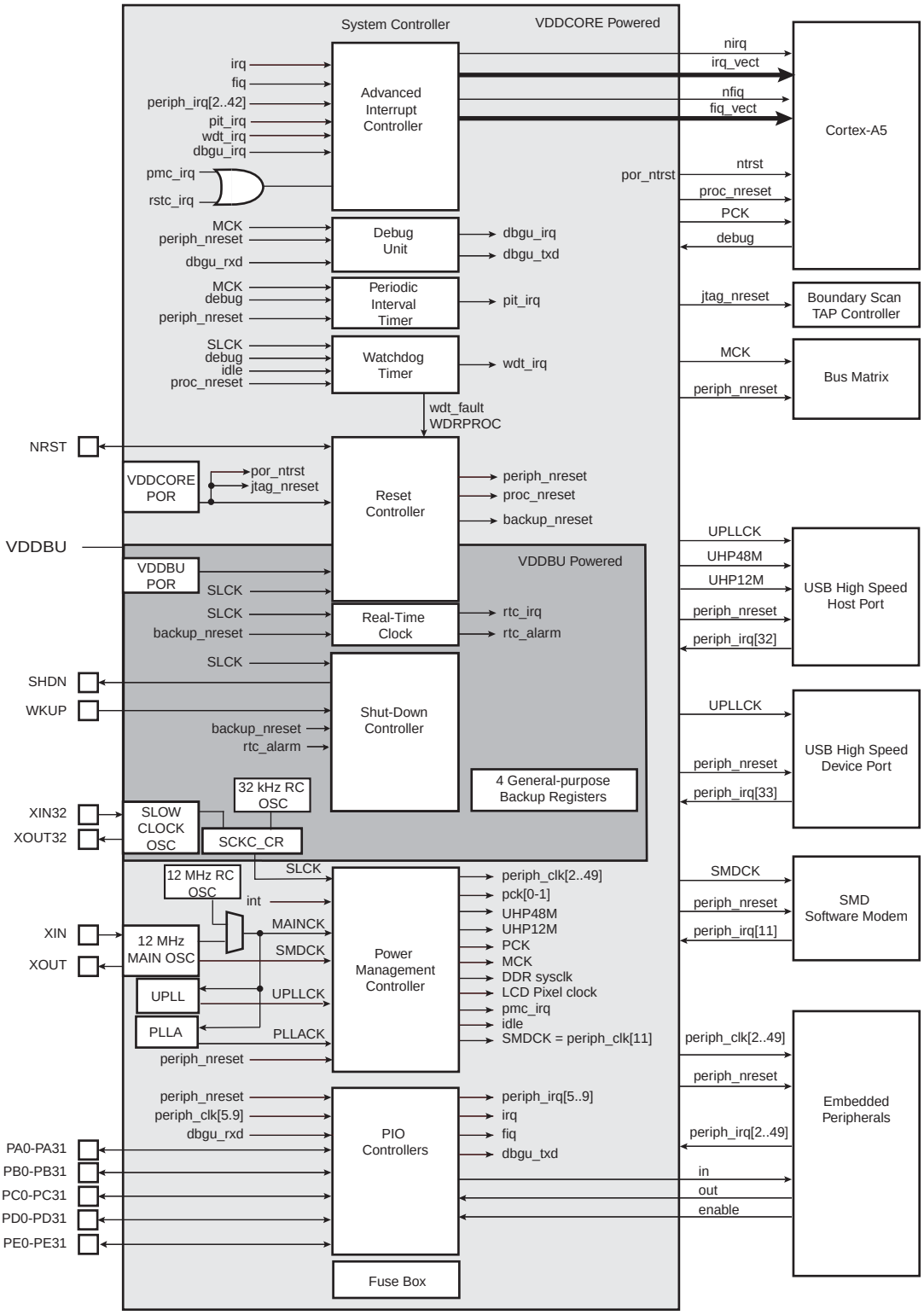
The System Controller User Interface also embeds the registers that configure the Matrix and a set of registers for the chip configuration. The chip configuration registers configure the EBI chip select assignment and voltage range for external memories.

The System Controller's peripherals are all mapped within the highest 16 KB of address space, between addresses 0xFFFF D000 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 KB.

Figure 8-1 on page 35 shows the System Controller block diagram.

Figure 8-1. SAMA5D3 System Controller Block Diagram



8.1 Chip Identification

- Chip ID: 0x8A5C07C2
- Extended ID:

Table 8-1. Chip Identification of SAMA5D3 Devices

Device	Extended ID
SAMA5D31	**0x00444300**
SAMA5D33	**0x00414300**
SAMA5D34	**0x00414301**
SAMA5D35	**0x00584300**
SAMA5D36	**0x00004301**

- Boundary JTAG ID: 0x05B3103F
- Cortex-A5 JTAG IDCODE: 0x4BA00477
- Cortex-A5 Serial Wire IDCODE: 0x2BA01477

8.2 Backup Section

The SAMA5D3 features a Backup Section that embeds:

- RC Oscillator
- Slow Clock Oscillator
- SCKR register
- Real-time Clock (RTC)
- Shutdown Controller
- 4 Backup registers
- Part of the Reset Controller (RSTC)
- Boot Select Control Register

This section is powered by the VDDBU rail.

9. Peripherals

9.1 Peripheral Mapping

As shown in Section 6. “Memories” the peripherals are mapped in the upper 256 Mbytes of the address space between the addresses 0xFFF7 8000 and 0xFFFC FFFF.

Each user peripheral is allocated 16 Kbytes of address space.

9.2 Peripheral Identifiers

Table 9-1. Peripheral Identifiers

Instance			Interrupt		
ID	Name	Description	External	Wired-OR	Clock Type
0	AIC	Advanced Interrupt Controller	FIQ	—	SYS_CLK
1	SYS	System Controller Interrupt	—	PMC, RSTC, RTC	SYS_CLK
2	DBGU	Debug Unit Interrupt	—	—	PCLOCK
3	PIT	Periodic Interval Timer Interrupt	—	—	SYS_CLK
4	WDT	Watchdog timer Interrupt	—	—	SYS_CLK
5	HSMC	Multi-bit ECC Interrupt	—	—	HCLOCK
6	PIOA	Parallel I/O Controller A	—	—	PCLOCK
7	PIOB	Parallel I/O Controller B	—	—	PCLOCK
8	PIOC	Parallel I/O Controller C	—	—	PCLOCK
9	PIOD	Parallel I/O Controller D	—	—	PCLOCK
10	PIOE	Parallel I/O Controller E	—	—	PCLOCK
11	SMD	SMD Soft Modem	—	—	HCLOCK
12	USART0	USART 0	—	—	PCLOCK
13	USART1	USART 1	—	—	PCLOCK
14	USART2	USART 2	—	—	PCLOCK
15	USART3	USART 3	—	—	PCLOCK
16	UART0	UART 0	—	—	PCLOCK
17	UART1	UART 1	—	—	PCLOCK
18	TWI0	Two-Wire Interface 0	—	—	PCLOCK
19	TWI1	Two-Wire Interface 1	—	—	PCLOCK
20	TWI2	Two-Wire Interface 2	—	—	PCLOCK
21	HSMCI0	High Speed Multimedia Card Interface 0	—	—	PCLOCK
22	HSMCI1	High Speed Multimedia Card Interface 1	—	—	PCLOCK
23	HSMCI2	High Speed Multimedia Card Interface 2	—	—	PCLOCK
24	SPI0	Serial Peripheral Interface 0	—	—	PCLOCK
25	SPI1	Serial Peripheral Interface 1	—	—	PCLOCK
26	TC0	Timer Counter 0 (ch. 0, 1, 2)	—	—	PCLOCK
27	TC1	Timer Counter 1 (ch. 3, 4, 5)	—	—	PCLOCK

Table 9-1. Peripheral Identifiers (Continued)

Instance			Interrupt		
ID	Name	Description	External	Wired-OR	Clock Type
28	PWM	Pulse Width Modulation Controller	—	—	PCLOCK
29	ADC	Touch Screen ADC Controller	—	—	PCLOCK
30	DMAC0	DMA Controller 0	—	—	HCLOCK
31	DMAC1	DMA Controller 1	—	—	HCLOCK
32	UHPHS	USB Host High Speed	—	—	HCLOCK
33	UDPHS	USB Device High Speed	—	—	HCLOCK
34	GMAC	Gigabit Ethernet MAC	—	—	HCLOCK + PCLOCK
35	EMAC	Ethernet MAC	—	—	HCLOCK + PCLOCK
36	LCDC	LCD Controller	—	—	HCLOCK
37	ISI	Image Sensor Interface	—	—	HCLOCK
38	SSC0	Synchronous Serial Controller 0	—	—	PCLOCK
39	SSC1	Synchronous Serial Controller 1	—	—	PCLOCK
40	CAN0	CAN Controller 0	—	—	PCLOCK
41	CAN1	CAN Controller 1	—	—	PCLOCK
42	SHA	Secure Hash Algorithm	—	—	PCLOCK
43	AES	Advanced Encryption Standard	—	—	PCLOCK
44	TDES	Triple Data Encryption Standard	—	—	PCLOCK
45	TRNG	True Random Number Generator	—	—	PCLOCK
46	ARM	Performance Monitor Unit	—	—	PROC_CLOCK
47	AIC	Advanced Interrupt Controller	IRQ	—	SYS_CLK
48	FUSE	Fuse Controller	—	—	PCLOCK
49	MPDDRC	MPDDR controller	—	—	HCLOCK
50-63	Reserved	—	—	—	—

9.3 Peripheral Signal Multiplexing on I/O Lines

The SAMA5D3 product features five PIO controllers (PIOA, PIOB, PIOC, PIOD and PIOE) which multiplex the I/O lines of the peripheral set.

Each PIO Controller controls 32 lines. Each line can be assigned to one of three peripheral functions: A, B or C. The multiplexing tables (Table 4-1 “SAMA5D3 Pinout for 324-ball LFBGA Package” and Table 4-2 “SAMA5D3 Pinout for 324-ball TFBGA Package”) define how the I/O lines of the peripherals A, B and C are multiplexed on the PIO controllers. Note that some output-only peripheral functions might be duplicated within the tables.

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

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

9.4 Peripheral Clock Type

The SAMA5D3 Series embeds peripherals with five different clock types:

- HCLOCK: AHB Clock, managed with the PMC_SCER, PMC_SCDR and PMC_SCSR registers of PMC System Clock
- PCLOCK: APB Clock, managed with the PMC_PCER, PMC_PCDR, PMC_PCSR and PMC_PCR registers of Peripheral Clock
- HCLOCK+PCLOCK: Both clock types coexist. The clock is managed with the PMC_PCER, PMC_PCDR, PMC_PCSR and PMC_PCR registers of Peripheral Clock
- SYS_CLOCK: This clock cannot be disabled.
- PROC_CLOCK: The clock related to Processor Clock (PCK) and managed with the PMC_SCDR and PMC_SCSR registers of PMC System Clock

Please refer to Table 9-1 “Peripheral Identifiers” for details.

10. ARM Cortex-A5 Processor

10.1 Description

The ARM Cortex-A5 processor is a high-performance, low-power, ARM macrocell with an L1 cache subsystem that provides full virtual memory capabilities. The Cortex-A5 processor implements the ARMv7 architecture and runs 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions, and 8-bit Java™ byte codes in Jazelle® state.

The *Floating-Point Unit* (FPU) supports the ARMv7 VFPv4-D16 architecture without Advanced SIMD extensions (NEON). It is tightly integrated to the Cortex-A5 processor pipeline. It provides trapless execution and is optimized for scalar operation. It can generate an Undefined instruction exception on vector instructions that enables the programmer to emulate vector capability in software.

The design can include the FPU only, in which case the *Media Processing Engine* (MPE) is not included.

See the *Cortex-A5 Floating-Point Unit Technical Reference Manual*.

10.1.1 Power Management

The Cortex-A5 design supports the following main levels of power management:

- Run Mode
- Standby Mode

10.1.1.1 Run Mode

Run mode is the normal mode of operation where all of the processor functionality is available. Everything, including core logic and embedded RAM arrays, is clocked and powered up.

10.1.1.2 Standby Mode

Standby mode disables most of the clocks of the processor, while keeping it powered up. This reduces the power drawn to the static leakage current, plus a small clock power overhead required to enable the processor to wake up from Standby mode. The transition from Standby mode to Run mode is caused by one of the following:

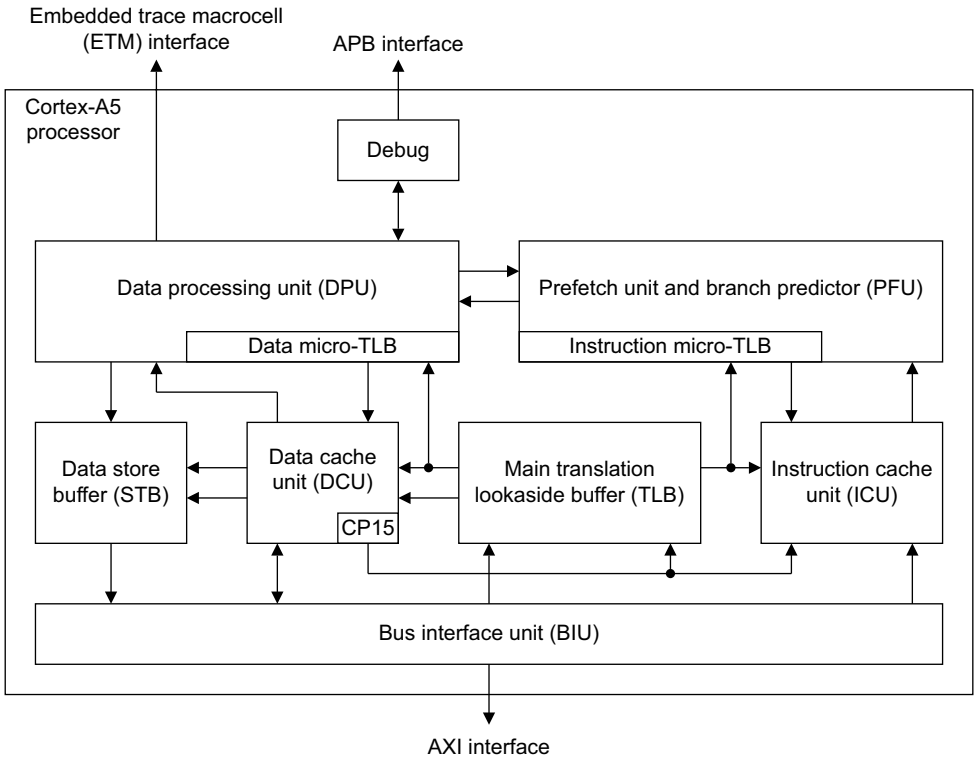
- the arrival of an interrupt, either masked or unmasked
- the arrival of an event, if standby mode was initiated by a Wait for Event (WFE) instruction
- a debug request, when either debug is enabled or disabled
- a reset.

10.2 Embedded Characteristics

- In-order pipeline with dynamic branch prediction
- ARM, Thumb, and ThumbEE instruction set support
- Harvard level 1 memory system with a Memory Management Unit (MMU)
- 32 Kbytes Data Cache
- 32 Kbytes Instruction Cache
- 64-bit AXI master interface
- ARM v7 debug architecture
- VFPv4-D16 FPU with trapless execution
- Jazelle hardware acceleration

10.3 Block Diagram

Figure 10-1. Cortex-A5 Processor Top-level Diagram



10.4 Programmer Model

10.4.1 Processor Operating Modes

The following operation modes are present in all states:

- User mode (USR) 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 (SVC) is a protected mode for the operating system.
- Abort mode (ABT) is entered after a data or instruction prefetch abort.
- System mode (SYS) is a privileged user mode for the operating system.
- Undefined mode (UND) 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.

10.4.2 Processor Operating States

The processor has the following instruction set states controlled by the T bit and J bit in the CPSR.

- ARM state:

  The processor executes 32-bit, word-aligned ARM instructions.

- Thumb state:

  The processor executes 16-bit and 32-bit, halfword-aligned Thumb instructions.
- ThumbEE state:

  The processor executes a variant of the Thumb instruction set designed as a target for dynamically generated code. This is code compiled on the device either shortly before or during execution from a portable bytecode or other intermediate or native representation.
- Jazelle state:

  The processor executes variable length, byte-aligned Java bytecodes.

The J bit and the T bit determine the instruction set used by the processor. Table 10-1 shows the encoding of these bits.

Table 10-1. CPSR J and T Bit Encoding

J	T	Instruction Set State
0	0	ARM
0	1	Thumb
1	0	Jazelle
1	1	ThumbEE

Changing between ARM and Thumb states does not affect the processor mode or the register contents. See the ARM Architecture Reference Manual for information on entering and exiting ThumbEE state.

10.4.2.1 Switching State

It is possible to change the instruction set state of the processor between:

- ARM state and Thumb state using the BX and BLX instructions.
- Thumb state and ThumbEE state using the ENTERX and LEAVEX instructions.
- ARM and Jazelle state using the BXJ instruction.
- Thumb and Jazelle state using the BXJ instruction.

See the ARM Architecture Reference Manual for more information about changing instruction set state.

10.4.3 Cortex-A5 Registers

This view provides 16 ARM core registers, R0 to R15, that include the Stack Pointer (SP), Link Register (LR), and Program Counter (PC). These registers are selected from a total set of either 31 or 33 registers, depending on whether or not the Security Extensions are implemented. The current execution mode determines the selected set of registers, as shown in Table 10-2. This shows that the arrangement of the registers provides duplicate copies of some registers, with the current register selected by the execution mode. This arrangement is described as banking of the registers, and the duplicated copies of registers are referred to as banked registers.

Table 10-2. Cortex-A5 Modes and Registers Layout

User and System	Monitor	Supervisor	Abort	Undefined	Interrupt	Fast Interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6

Table 10-2. Cortex-A5 Modes and Registers Layout (Continued)

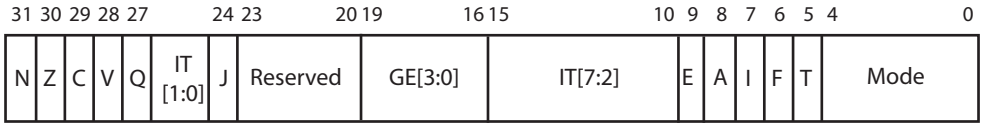
User and System	Monitor	Supervisor	Abort	Undefined	Interrupt	Fast Interrupt
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12	R12_FIQ
R13	R13_MON	R13_SVC	R13_ABT	R13_UND	R13_IRQ	R13_FIQ
R14	R14_MON	R14_SVC	R14_ABT	R14_UND	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_MON	SPSR_SVC	SPSR_ABT	SPSR_UND	SPSR_IRQ	SPSR_FIQ

	Mode-specific banked registers

The core contains one CPSR, and six 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 10-2. Status Register Format



- N: Negative, Z: Zero, C: Carry, and V: Overflow are the four ALU flags
- Q: cumulative saturation flag
- IT: If-Then execution state bits for the Thumb IT (If-Then) instruction
- J: Jazelle bit, see the description of the T bit
- GE: Greater than or Equal flags, for SIMD instructions
- E: Endianness execution state bit. Controls the load and store endianness for data accesses. This bit is ignored by instruction fetches.
  - E = 0: Little endian operation
  - E = 1: Big endian operation
- A: Asynchronous abort disable bit. Used to mask asynchronous aborts.
- I: Interrupt disable bit. Used to mask IRQ interrupts.
- F: Fast interrupt disable bit. Used to mask FIQ interrupts.
- T: Thumb execution state bit. This bit and the J execution state bit, bit [24], determine the instruction set state of the processor, ARM, Thumb, Jazelle, or ThumbEE.

- Mode: five bits to encode the current processor mode. The effect of setting M[4:0] to a reserved value is UNPREDICTABLE.

Table 10-3. Processor Mode vs. Mode Field

Mode	M[4:0]
USR	10000
FIQ	10001
IRQ	10010
SVC	10011
ABT	10111
UND	11011
SYS	11111
Reserved	Other

10.4.3.1 CP15 Coprocessor

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

- Cortex A5
- Caches (ICache, DCache and write buffer)
- MMU
- Security
- Other system options

To control these features, CP15 provides 16 additional registers. See Table 10-4.

Table 10-4. CP15 Registers

Register	Name	Read/Write
0	ID Code(1)	Read/Unpredictable
0	Cache type(1)	Read/Unpredictable
1	Control(1)	Read/Write
1	Security(1)	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	Read/Write
6	Fault Address	Read/Write
7	Cache and MMU Operations(1)	Read/Write
8	TLB operations	Unpredictable/Write
9	Cache lockdown(1)	Read/Write
10	TLB lockdown	Read/Write
11	Reserved	None
12	Interrupts management	Read/Write
13	FCSE PID(1)	Read/Write

Table 10-4. CP15 Registers (Continued)

Register	Name	Read/Write
13	Context ID[1]	Read/Write
14	Reserved	None
15	Test configuration	Read/Write

Note: 1. This register provides access to more than one register. The register accessed depends on the value of the CRm field or Opcode_2 field.

10.4.4 CP 15 Register 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 such as 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**

10.4.5 Addresses in the Cortex-A5 processor

The Cortex-A5 processor operates using *virtual addresses* (VAs). The *Memory Management Unit* (MMU) translates these VAs into the *physical addresses* (PAs) used to access the memory system. Translation tables hold the mappings between VAs and PAs.

See the ARM Architecture Reference Manual for more information.

When the Cortex-A5 processor is executing in Non-secure state, the processor performs translation table look-ups using the Non-secure versions of the Translation Table Base Registers. In this situation, any VA can only translate into a Non-secure PA. When it is in Secure state, the Cortex-A5 processor performs translation table look-ups using the Secure versions of the Translation Table Base Registers. In this situation, the security state of any VA is determined by the NS bit of the translation table descriptors for that address.

Following is an example of the address manipulation that occurs when the Cortex-A5 processor requests an instruction:

1. The Cortex-A5 processor issues the VA of the instruction as Secure or Non-secure VA accesses according to the state the processor is in.
2. The instruction cache is indexed by the bits of the VA. The MMU performs the translation table look-up in parallel with the cache access. If the processor is in the Secure state it uses the Secure translation tables, otherwise it uses the Non-secure translation tables.
3. If the protection check carried out by the MMU on the VA does not abort and the PA tag is in the instruction cache, the instruction data is returned to the processor.
4. If there is a cache miss, the MMU passes the PA to the AXI bus interface to perform an external access. The external access is always Non-secure when the core is in the Non-secure state. In the Secure state, the external access is Secure or Non-secure according to the NS attribute value in the selected translation table entry. In Secure state, both L1 and L2 translation table walk accesses are marked as Secure, even if the first level descriptor is marked as NS.

10.5 Memory Management Unit

10.5.1 About the MMU

The MMU works with the L1 and L2 memory system to translate virtual addresses to physical addresses. It also controls accesses to and from external memory.

The ARM v7 Virtual Memory System Architecture (VMSA) features include the following:

- Page table entries that support:
  - 16-Mbyte supersections. The processor supports supersections that consist of 16-Mbyte blocks of memory.
  - 1-Mbyte sections
  - 64-Kbyte large pages
  - 4-Kbyte small pages
- 16 access domains
- Global and application-specific identifiers to remove the requirement for context switch TLB flushes.
- Extended permissions checking capability.

TLB maintenance and configuration operations are controlled through a dedicated coprocessor, CP15, integrated with the core. This coprocessor provides a standard mechanism for configuring the L1 memory system.

See the ARM Architecture Reference Manual for a full architectural description of the ARMv7 VMSA.

10.5.2 Memory Management System

The Cortex-A5 processor supports the ARM v7 VMSA The translation of a Virtual Address (VA) used by the instruction set architecture to a Physical Address (PA) used in the memory system and the management of the associated attributes and permissions is carried out using a two-level MMU.

The first level MMU uses a Harvard design with separate micro TLB structures in the PFU for instruction fetches (IuTLB) and in the DPU for data read and write requests (DuTLB).

A miss in the micro TLB results in a request to the main unified TLB shared between the data and instruction sides of the memory system. The TLB consists of a 128-entry two-way set-associative RAM based structure. The TLB page-walk mechanism supports page descriptors held in the L1 data cache. The caching of page descriptors is configured globally for each translation table base register, TTBRx, in the system coprocessor, CP15.

The TLB contains a hitmap cache of the page types which have already been stored in the TLB.

10.5.2.1 Memory types

Although various different memory types can be specified in the page tables, the Cortex-A5 processor does not implement all possible combinations:

- Write-through caches are not supported. Any memory marked as write-through is treated as Non-cacheable.
- The outer shareable attribute is not supported. Anything marked as outer shareable is treated in the same way as inner shareable.
- Write-back no write-allocate is not supported. It is treated as write-back write-allocate.

Table 10-5 shows the treatment of each different memory type in the Cortex-A5 processor in addition to the architectural requirements.

Table 10-5. Treatment of Memory Attributes

Memory Type Attribute	Shareability	Other attributes	Notes
Strongly Ordered	—	—	—
Device	Non-shareable	—	—
	Shareable	—	—
Normal	Non-shareable	Non-cacheable	Does not access L1 caches
		Write-through cacheable	Treated as non-cacheable
		Write-back cacheable, write allocate	Can dynamically switch to no write allocate, if more than three full cache lines are written in succession
		Write-back cacheable, no write allocate	Treated as non-shareable write-back cacheable, write allocate
	Inner shareable	Non-cacheable	—
		Write-through cacheable	Treated as inner shareable non-cacheable
		Write-back cacheable, write allocate	Treated as inner shareable non-cacheable unless the SMP bit in the Auxiliary Control Register is set (ACTLR[6] = b1). If this bit is set the area is treated as write-back cacheable write allocate.
		Write-back cacheable, no write allocate	
	Outer shareable	Non-cacheable	Treated as inner shareable non-cacheable
		Write-through cacheable	
		Write-back cacheable, write allocate	Treated as inner shareable non-cacheable unless the SMP bit in the Auxiliary Control Register is set (ACTLR[6] = b1). If this bit is set the area is treated as write-back cacheable write allocate.
		Write-back cacheable, no write allocate	

10.5.3 TLB Organization

TLB Organization is described in the sections that follow:

- Micro TLB
- Main TLB

10.5.3.1 Micro TLB

The first level of caching for the page table information is a micro TLB of 10 entries that is implemented on each of the instruction and data sides. These blocks provide a lookup of the virtual addresses in a single cycle.

The micro TLB returns the physical address to the cache for the address comparison, and also checks the access permissions to signal either a Prefetch Abort or a Data Abort.

All main TLB related maintenance operations affect both the instruction and data micro TLBs, causing them to be flushed. In the same way, any change of the following registers causes the micro TLBs to be flushed:

- Context ID Register (CONTEXTIDR)
- Domain Access Control Register (DACR)
- Primary Region Remap Register (PRRR)
- Normal Memory Remap Register (NMRR)
- Translation Table Base Registers (TTBR0 and TTBR1)

10.5.3.2 Main TLB

Misses from the instruction and data micro TLBs are handled by a unified main TLB. Accesses to the main TLB take a variable number of cycles, according to competing requests from each of the micro TLBs and other implementation-dependent factors.

The main TLB is 128-entry two-way set-associative.

TLB match process

Each TLB entry contains a virtual address, a page size, a physical address, and a set of memory properties. Each is marked as being associated with a particular application space (ASID), or as global for all application spaces. The CONTEXTIDR determines the currently selected application space.

A TLB entry matches when these conditions are true:

- its virtual address matches that of the requested address
- its Non-secure TLB ID (NSTID) matches the Secure or Non-secure state of the MMU request
- its ASID matches the current ASID in the CONTEXTIDR or is global

The operating system must ensure that, at most, one TLB entry matches at any time. The TLB can store entries based on the following block sizes:

Supersections	Describe 16-Mbyte blocks of memory.
Sections	Describe 1-Mbyte blocks of memory.
Large pages	Describe 64-Kbyte blocks of memory.
Small pages	Describe 4-Kbyte blocks of memory

Supersections, sections and large pages are supported to permit mapping of a large region of memory while using only a single entry in the TLB. If no mapping for an address is found within the TLB, then the translation table is automatically read by hardware and a mapping is placed in the TLB.

10.5.4 Memory Access Sequence

When the processor generates a memory access, the MMU:

1. Performs a lookup for the requested virtual address and current ASID and security state in the relevant instruction or data micro TLB.
2. If there is a miss in the micro TLB, performs a lookup for the requested virtual address and current ASID and security state in the main TLB.
3. If there is a miss in main TLB, performs a hardware translation table walk.

The MMU can be configured to perform hardware translation table walks in cacheable regions by setting the IRGN bits in Translation Table Base Register 0 and Translation Table Base Register 1. If the encoding of the IRGN bits is write-back, an L1 data cache lookup is performed and data is read from the data cache. If the encoding of the IRGN bits is write-through or non-cacheable, an access to external memory is performed. For more information refer to: Cortex-A5 Technical Reference Manual.

The MMU might not find a global mapping, or a mapping for the currently selected ASID, with a matching Non-secure TLB ID (NSTID) for the virtual address in the TLB. In this case, the hardware does a translation table walk if the translation table walk is enabled by the PD0 or PD1 bit in the Translation Table Base Control Register. If translation table walks are disabled, the processor returns a Section Translation fault. For more information refer to: Cortex-A5 Technical Reference Manual.

If the TLB finds a matching entry, it uses the information in the entry as follows:

1. The access permission bits and the domain determine if the access is enabled. If the matching entry does not pass the permission checks, the MMU signals a memory abort. See the ARM Architecture Reference Manual for a

description of access permission bits, abort types and priorities, and for a description of the Instruction Fault Status Register (IFSR) and Data Fault Status Register (DFSR).

2. The memory region attributes specified in both the TLB entry and the CP15 c10 remap registers determine if the access is
   - Secure or Non-secure
   - Shared or not
   - Normal memory, Device, or Strongly-ordered

For more information refer to: Cortex-A5 Technical Reference Manual, Memory region remap.

3. The TLB translates the virtual address to a physical address for the memory access.

10.5.5 Interaction with Memory System

The MMU can be enabled or disabled as described in the ARM Architecture Reference Manual.

10.5.6 External Aborts

External memory errors are defined as those that occur in the memory system rather than those that are detected by the MMU. External memory errors are expected to be extremely rare. External aborts are caused by errors flagged by the AXI interfaces when the request goes external to the Cortex-A5 processor. External aborts can be configured to trap to Monitor mode by setting the EA bit in the Secure Configuration Register. For more information refer to: Cortex-A5 Technical Reference Manual.

10.5.6.1 External Aborts on Data Write

Externally generated errors during a data write can be asynchronous. This means that the r14_abt on entry into the abort handler on such an abort might not hold the address of the instruction that caused the exception.

The DFAR is Unpredictable when an asynchronous abort occurs.

Externally generated errors during data read are always synchronous. The address captured in the DFAR matches the address which generated the external abort.

10.5.6.2 Synchronous and Asynchronous Aborts

Chapter 4, System Control in the Cortex-A5 Technical Reference Manual describes synchronous and asynchronous aborts, their priorities, and the IFSR and DFSR. To determine a fault type, read the DFSR for a data abort or the IFSR for an instruction abort.

The processor supports an Auxiliary Fault Status Register for software compatibility reasons only. The processor does not modify this register because of any generated abort.

10.5.7 MMU Software Accessible Registers

The system control coprocessor registers, CP15, in conjunction with page table descriptors stored in memory, control the MMU.

Access all the registers with instructions of the form:

MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>

MCR p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>

CRn is the system control coprocessor register. Unless specified otherwise, CRm and Opcode_2 Should Be Zero.

11. SAMA5D3 Series Debug and Test

11.1 Description

The device 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.

A 2-pin debug port Serial Wire Debug (SWD). SWD replaces the 5-pin JTAG port and provides an easy and risk free alternative to JTAG as the two signals SWDIO and SWCLK are overlaid on the TMS and TCK pins, allowing for bi-modal devices that provide the other JTAG signals. These extra JTAG pins can be switched to other uses when in SWD mode.

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.

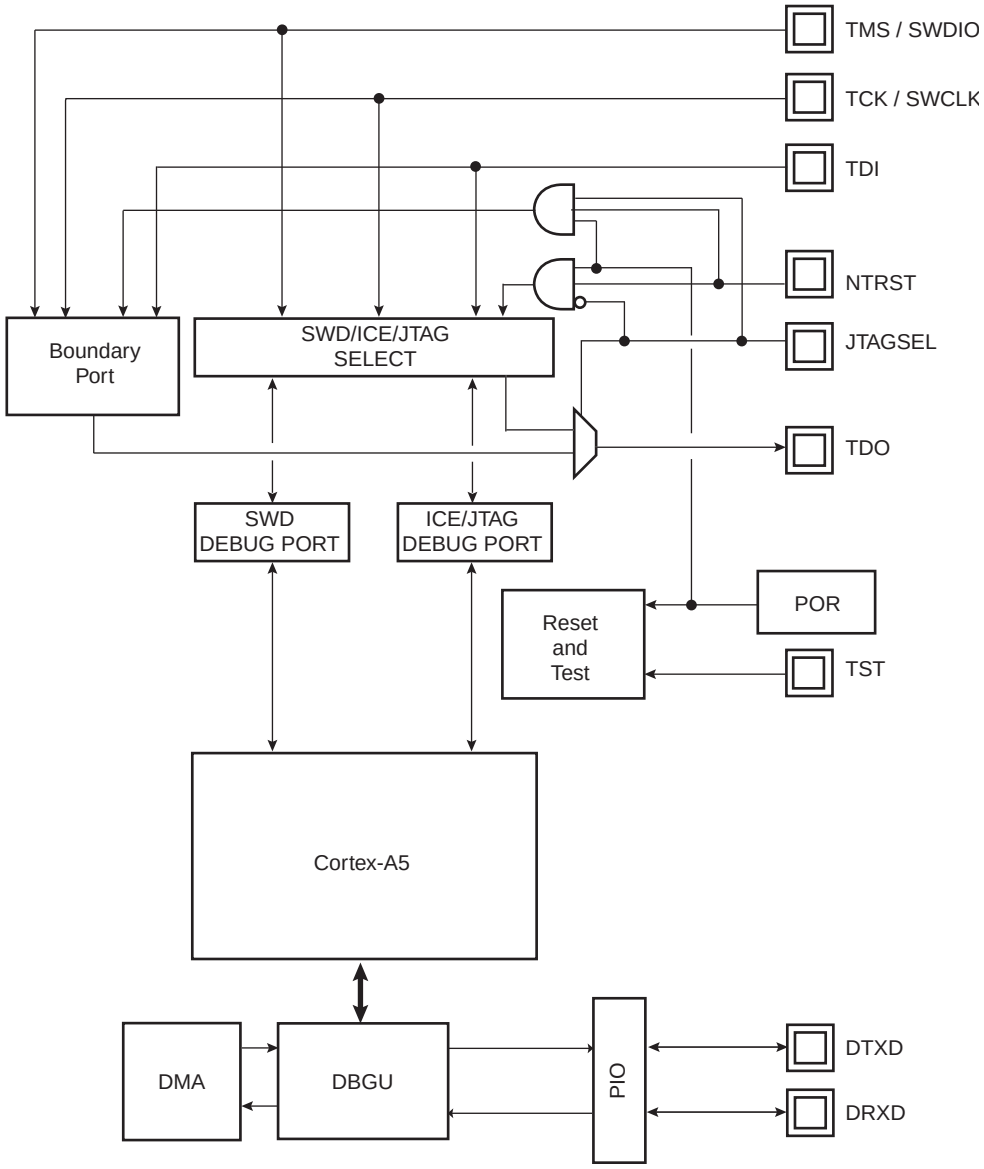
A set of dedicated debug and test input/output pins gives direct access to these capabilities from a PC-based test environment.

11.2 Embedded Characteristics

- Cortex-A5 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
  - Serial Wire Debug
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel Interrupt Handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins.

11.3 Block Diagram

Figure 11-1. Debug and Test Block Diagram

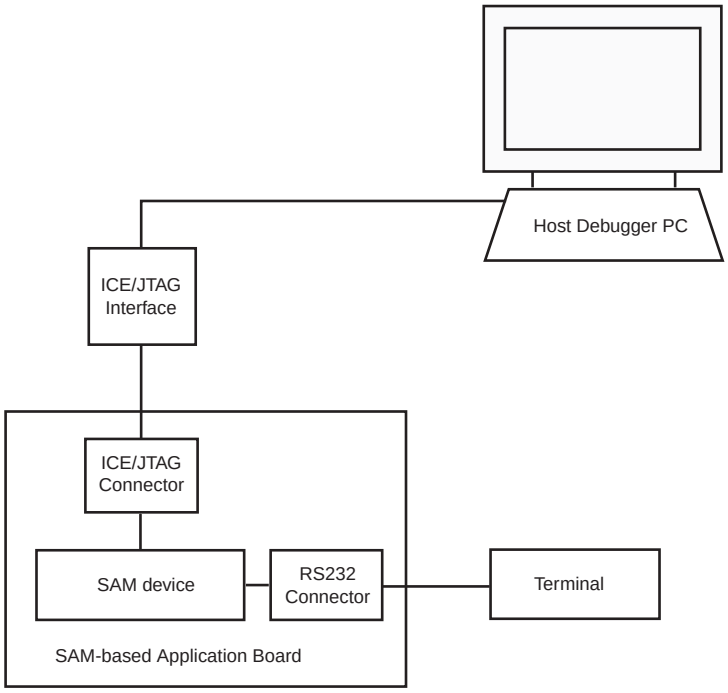


11.4 Application Examples

11.4.1 Debug Environment

Figure 11-2 on page 54 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.

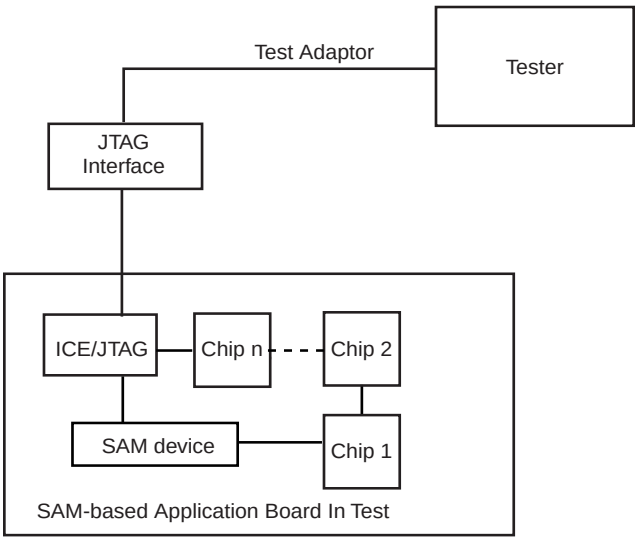
Figure 11-2. Application Debug and Trace Environment Example



11.4.2 Test Environment

Figure 11-3 on page 55 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 11-3. Application Test Environment Example



11.5 Debug and Test Pin Description

Table 11-1. 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	
JTAGSEL	JTAG Selection	Input	
SWD			
SWCLK	Serial Debug Clock	Input	
SWDIO	Serial Debug IO	Input/Output	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

11.6 Functional Description

11.6.1 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.

11.6.2 EmbeddedICE

The Cortex-A5 EmbeddedICE-RT™ is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface.The internal state of the Cortex-A5 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 Cortex-A5 registers. This data can be serially shifted out without affecting the rest of the system.

There are two scan chains inside the Cortex-A5 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:

ARM IHI 0031A_ARM_debug_interface_v5.pdf

11.6.3 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 Cortex-A5-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.

11.6.4 Chip Access Using JTAG Connection

In some cases, the JTAG connection is not allowed on this chip (JMem, SAM-BA, etc.) due to the Secure ROM Code implementation.

By default, the SAMA5D3 devices boot in Standard mode and not in Secure mode. When the Secure ROM Code starts, it disables the JTAG access for the whole boot sequence.

If the Secure ROM Code does not find any program in the external memory, it enables the USB connection and waits for a dedicated command to switch the chip into Secure mode.

If any other character is received, the Secure ROM Code starts the Standard SAM-BA Monitor, locks access to the ROM memory, and enables the JTAG.

Then you can access to the chip using the JTAG connection.

If the Secure ROM Code finds a bootable program, it disables automatically ROM access and enables JTAG just before launching the program.

The procedure to enable JTAG access is as follows:

- Connect your computer to the board with JTAG and USB (J20 USB-A)
- Power on the chip
- Open a terminal console (TeraTerm or HyperTerminal, etc.) on your computer and connect to the USB CDC Serial COM port related to the J20 connector on the board
- Send the '#' character. You will see then the prompt '>' character sent by the device (indicating that the Standard SAM-BA Monitor is running)
- Use the Standard SAM-BA Monitor to connect to the chip with JTAG

Note that you don't need to follow this sequence in order to connect the Standard SAM-BA Monitor with USB.

11.6.5 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.

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

11.6.6 IEEE 1149.1 JTAG Boundary Scan

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

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

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

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

11.7 The Boundary JTAG ID 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 0x5B31

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

Set to 0x01F.

Bit[0] required by IEEE Std. 1149.1.

Set to 0x1.

JTAG ID Code value is 0x05B3_103F.

11.8 The Cortex-A5 DP Identification Code Register IDCODE

The Identification Code Register is always present on all DP implementations. It provides identification information about the ARM Debug Interface.

11.8.1 JTAG Debug Port (JTAG-DP)

It is accessed using its own scan chain, the JTAG-DP Device 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				DESIGNER			

7	6	5	4	3	2	1	0
DESIGNER							1

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

Set to 0x0.

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

Product part Number is 0xBA00

- **DESIGNER[11:1]**

Set to 0x23B.

Bit[0] required by IEEE Std. 1149.1.

Set to 0x1.

Cortex-A5 JTAG-DP IDCODE value is 0x0BA0_0477

11.8.2 Serial Wire Debug Port (SW-DP)

It is at address 0x0 on read operations when the APnDP bit = 0. Access to the Identification Code Register is not affected by the value of the CTRLSEL bit in the Select 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				DESIGNER			

7	6	5	4	3	2	1	0
DESIGNER							1

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

Set to 0x0.

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

Product part Number is 0xBA01

- **DESIGNER[11:1]**

Set to 0x23B.

Bit[0] required by IEEE Std. 1149.1.

Set to 0x1.

Cortex-A5 SW-DP IDCODE is 0x0BA0_1477

12. Standard Boot Strategies

The system always boots from the ROM memory at address 0x0.

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

This microcontroller can be configured to run a Standard Boot Mode or a Secure Boot Mode. More information on how the Secure Boot Mode can be enabled, and how the chip operates in this mode, is provided in the application note “Secure Boot on SAMAD3 Series”, literature number 11165A. Please refer to the Atmel web site at www.atmel.com.

By default, the chip starts in Standard Boot Mode.

Note: JTAG access is disabled during the execution of ROM Code Sequence. It is re-enabled when jumping into SRAM when a valid code has been found on an external NVM, in the same time the ROM memory is hidden. If no valid boot has been found on an external NVM, the ROM Code enables the USB connection and waits for a special command to set the chip in Secure mode. If any other character is received, the ROM Code starts the Standard SAM-BA Monitor, locks access to the ROM memory and re-enables the JTAG.

The user can choose to boot from an external NOR Flash memory with the help of the BMS pin. The sampling of the BMS pin is done by hardware at reset, and the result is available in the BMS_EBI bit of the SFR_EBICFG register.

The first steps of the ROM Code program is to check the state of this pin by reading this register.

If BMS signal is tied to 0, BMS_BIT is read at 1

The ROM Code allows execution of the code contained into the memory connected to Chip Select 0 of the External Bus Interface.

To achieve that, the following sequence is preformed by the ROM Code:

- The main clock is the on-chip 12 MHz RC oscillator
- The Static Memory Controller is configured with timing allowing code execution inCS0 external memory at 12 MHz
- AXI matrix is configured to remap EBI CS0 address at 0x0
- 0x0 is loaded in the Program Counter register

The user software in the external memory must perform the next operation in order to complete the clocks and SMC timings configuration to run at a higher clock frequency:

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

If BMS signal is tied to 1, BMS_BIT is read at 0

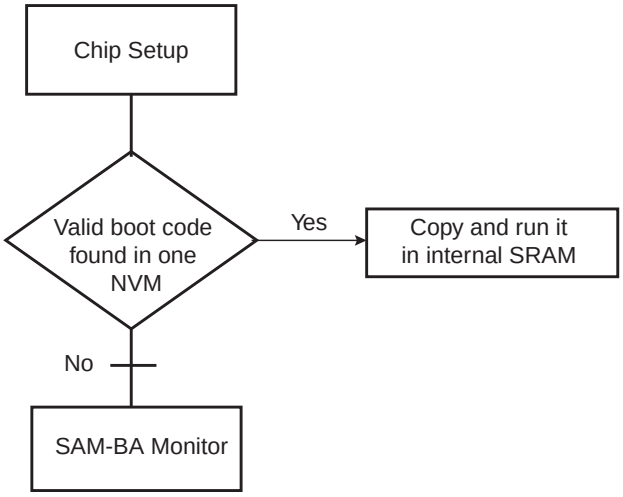
The ROM Code standard sequence is executed as follows:

- Basic chip initialization: crystal 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

12.1 Flow Diagram

The ROM Code implements the algorithm shown in Figure 12-1.

Figure 12-1. ROM Code Algorithm Flow Diagram



12.2 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 the 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 internal 12 MHz fast RC 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 oscillator, but USB will not be activated.

12.3 NVM Boot

12.3.1 NVM Boot Sequence

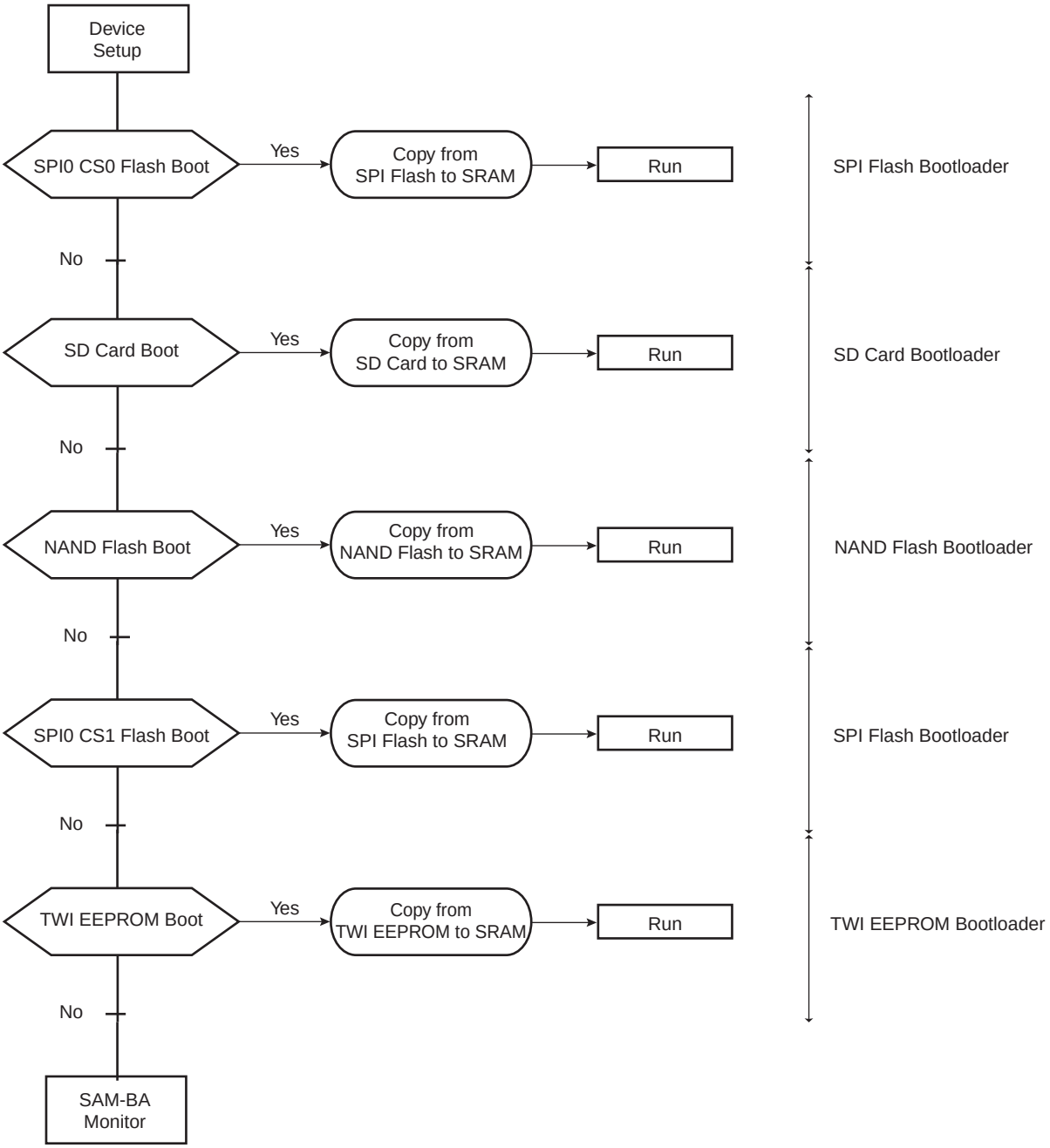
The boot sequence on external memory devices can be controlled using the Boot Sequence Configuration Register (BSC_CR).

The user can then choose to bypass some steps shown in Figure 12-2 “NVM Bootloader Sequence Diagram” according to the BOOT value in the BSC_CR.

Table 12-1. Values of the Boot Sequence Configuration Register

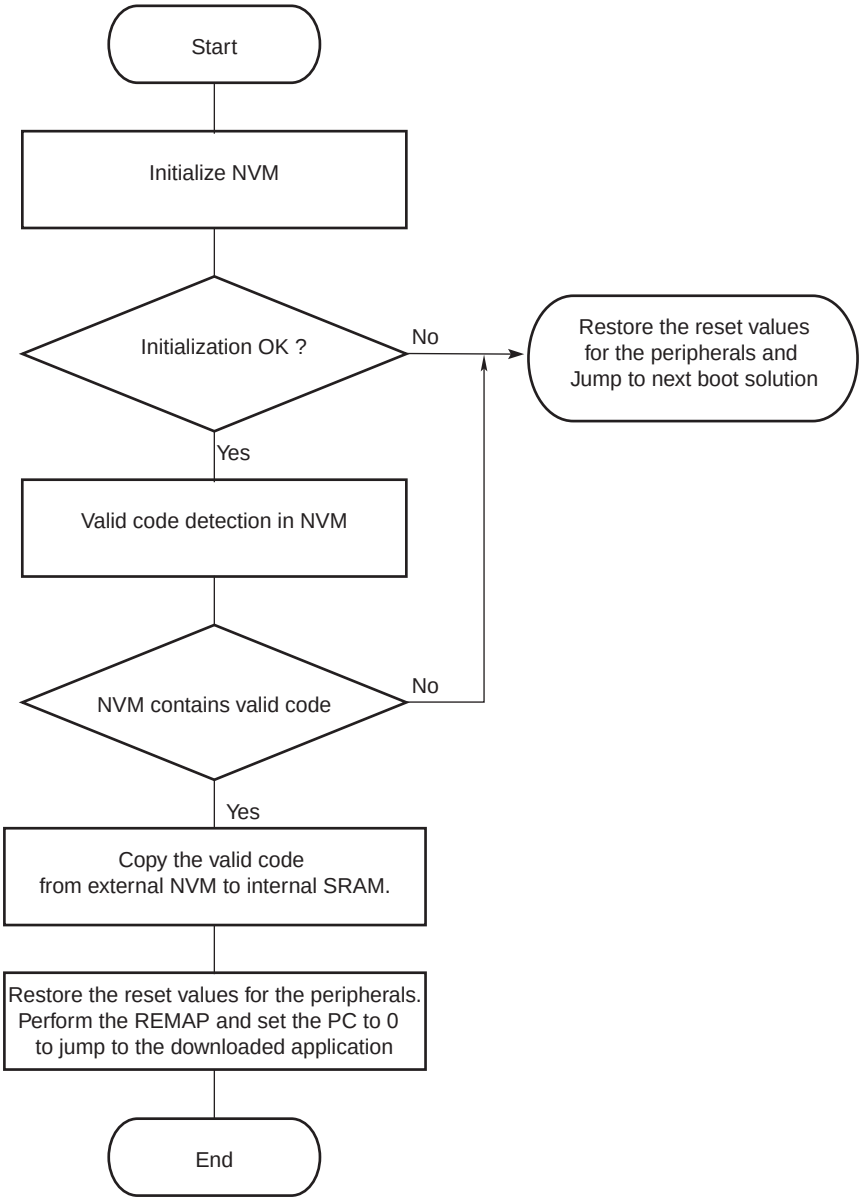
BOOT Value	SPI0 NPCS0	SD Card / eMMC (MCI0)	SD Card / eMMC (MCI1)	8-bit NAND Flash	SPI0 NPCS1	TWI EEPROM	SAM-BA Monitor
0	Y	Y	Y	Y	Y	Y	Y
1	Y	—	Y	Y	Y	Y	Y
2	Y	—	—	Y	Y	Y	Y
3	Y	—	—	—	Y	Y	Y
4	Y	—	—	—	Y	Y	Y
5	—	—	—	—	—	—	Y
6	—	—	—	—	—	—	Y
7	—	—	—	—	—	—	Y

Figure 12-2. NVM Bootloader Sequence Diagram



12.3.2 NVM Bootloader Program Description

Figure 12-3. NVM Bootloader Program Diagram



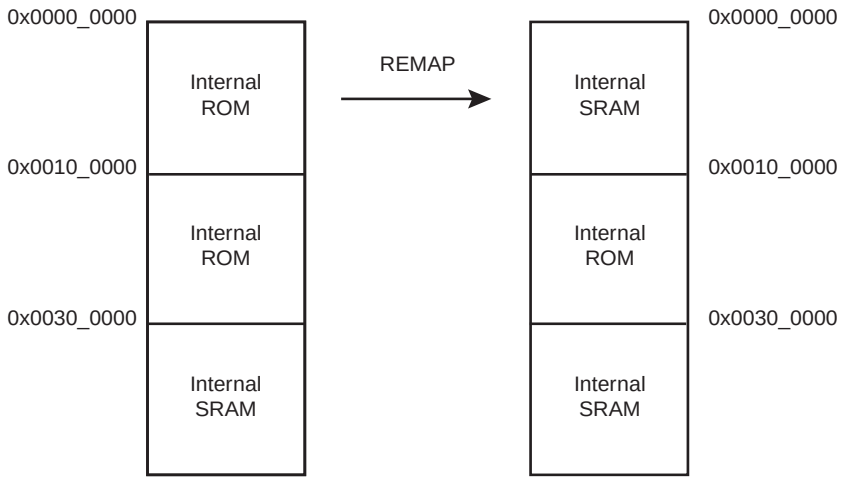
The NVM bootloader program first initializes the PIOs related to the NVM device. Then it configures the right peripheral depending on the NVM and tries to access this memory. If the initialization fails, it restores the reset values for the PIO and the peripheral, and then tries to fulfill the same operations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

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

If a valid code is found, this code is loaded from the NVM into the 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 12-4. Remap Action after Download Completion



12.3.3 Valid Code Detection

There are two kinds of valid code detection.

12.3.3.1 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 12-5. 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 12-6. B Opcode

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

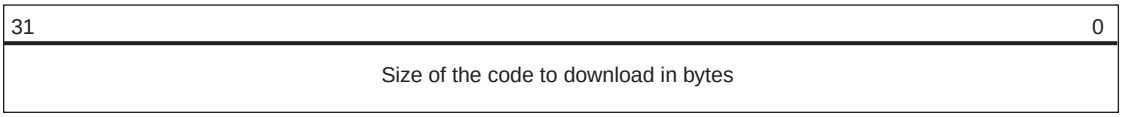
Unconditional instruction: 0xE for bits 31 to 28.

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

Figure 12-7. Structure of the ARM Vector 6



The value has to be smaller than 64 Kbytes.

Example

An example of valid vectors:

```
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
```

12.3.3.2 boot.bin File Check

This method is the one used on FAT formatted SD Card and eMMC. The boot program must be a file named "`boot.bin`" written in the root directory of the file system. Its size must not exceed the maximum size allowed: 64 Kbytes (0x10000).

12.3.4 Detailed Memory Boot Procedures

12.3.4.1 NAND Flash Boot: NAND Flash Detection

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

Hardware ECC detection and correction are provided by the PMECC peripheral. Please refer to the "PMECC Controller Functional Description" section of this datasheet for more details.

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

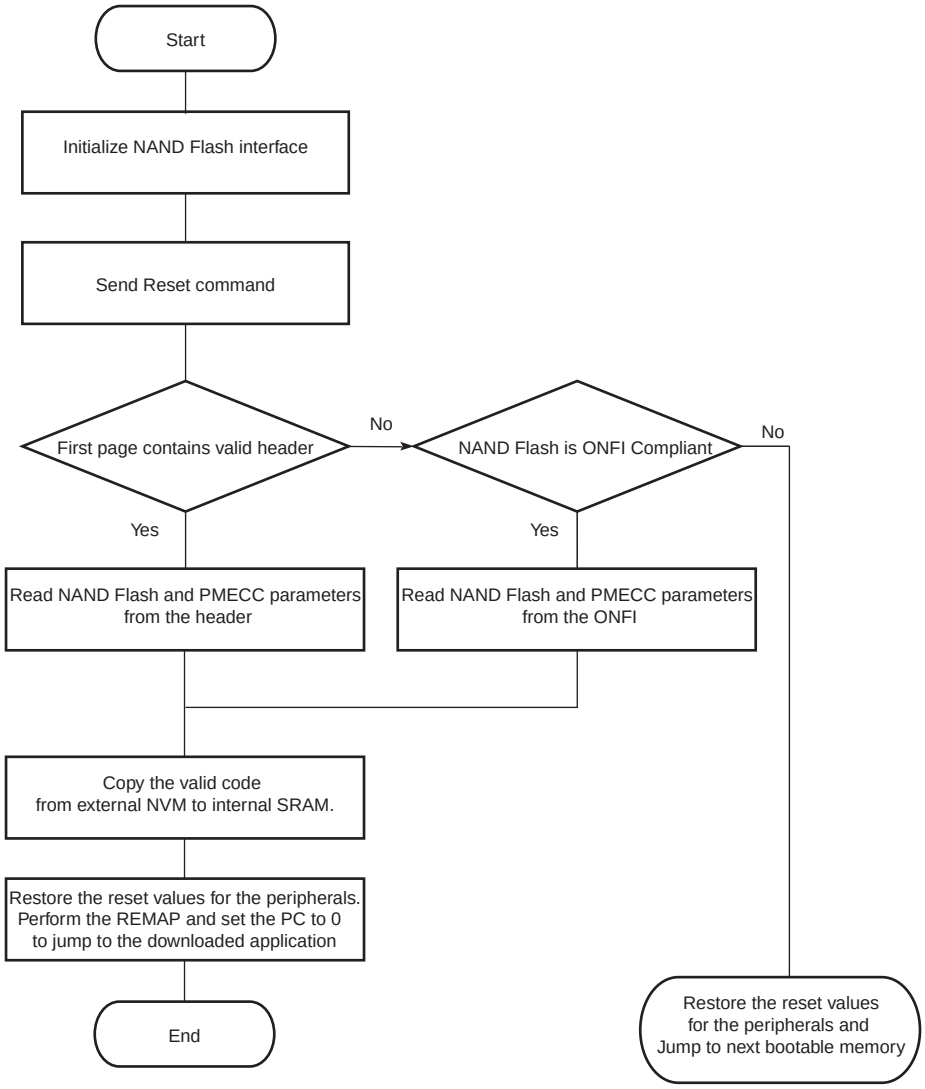
- The detection of a specific header written at the beginning of the first page of the NAND Flash,

or

- Through the ONFI parameters for the ONFI compliant memories

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

Figure 12-8. 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 an 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:

31	30	29	28	27	26	25	24
key				-	eccOffset		

23	22	21	20	19	18	17	16
eccOffset						sectorSize	

15	14	13	12	11	10	9	8
eccBitReq			spareSize				

7	6	5	4	3	2	1	0
spareSize				nbSectorPerPage			usePmecc

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

- **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**

0: 2-bit ECC

1: 4-bit ECC

2: 8-bit ECC

3: 12-bit ECC

4: 24-bit ECC

- **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 continues with the detection of a valid code.

ONFI 2.2 Parameters

In case no valid header is found, the Boot Program checks 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 to 1024 bytes if the ECC bit capability above is 0xFF

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

Once the Boot Program retrieves the parameter, using one of the two methods described above, it reads 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.

12.3.4.2 NAND Flash Boot: PMECC Error Detection and Correction

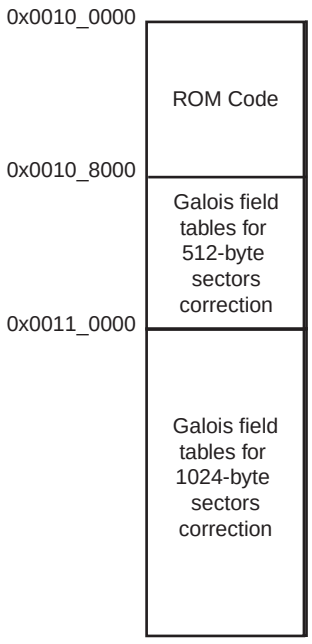
NAND Flash boot procedure uses PMECC to detect and correct errors during NAND Flash read operations in two cases:

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

The ROM memory embeds the Galois field tables. The user does not need to embed them in his own software.

The Galois field tables are mapped in the ROM just after the ROM code, as described in Figure 12-9.

Figure 12-9. 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.

12.3.4.3 SD Card / eMMC Boot

The SD Card / eMMC bootloader looks for a "`boot.bin`" file in the root directory of a FAT12/16/32 file system.

Supported SD Card Devices

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

12.3.4.4 SPI Flash Boot

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

The SPI Flash bootloader tries to boot on SPI0, 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 the 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 12-2. DataFlash Device

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

Supported Serial Flash Devices

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

12.3.4.5 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 the 7-bit device address 0x50.

12.3.5 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 connected devices could be unintentionally driven at boot time, and thus electrical conflicts between output pins used by the NVM drivers and the connected devices could occur.

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

Table 12-3 contains a list of pins that are driven during the boot program execution. These pins are driven during the boot sequence for a period of less than 1 second if no correct boot program is found.

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

Table 12-3. PIO Driven during Boot Program Execution

NVM Bootloader	Peripheral	Pin	PIO Line
NAND	EBI CS3 SMC	NANDOE	—
	EBI CS3 SMC	NANDWE	—
	EBI CS3 SMC	NANDCS	—
	EBI CS3 SMC	NAND ALE	—
	EBI CS3 SMC	NAND CLE	—
	EBI CS3 SMC	Cmd/Addr/Data	—
SD Card / eMMC	MCI0	MCI0_CK	PIOD9
	MCI0	MCI0_CDA	PIOD0
	MCI0	MCI0_D0	PIOD1
	MCI0	MCI0_D1	PIOD2
	MCI0	MCI0_D2	PIOD3
	MCI0	MCI0_D3	PIOD4
	MCI1	MCI1_CK	PIOB24
	MCI1	MCI1_CDA	PIOB19
	MCI1	MCI1_D0	PIOB20
	MCI1	MCI1_D1	PIOB21
	MCI1	MCI1_D2	PIOB22
	MCI1	MCI1_D3	PIOB23
SPI Flash	SPI0	MOSI	PIOD11
	SPI0	MISO	PIOD10
	SPI0	SPCK	PIOD12
	SPI0	NPCS0	PIOD13
	SPI0	NPCS1	PIOD14
TWI0 EEPROM	TWI0	TWD0	PIOA30
	TWI0	TWCK0	PIOA31
SAM-BA Monitor	DBGU	DRXD	PIOB30
	DBGU	DTXD	PIOB31

12.4 SAM-BA Monitor

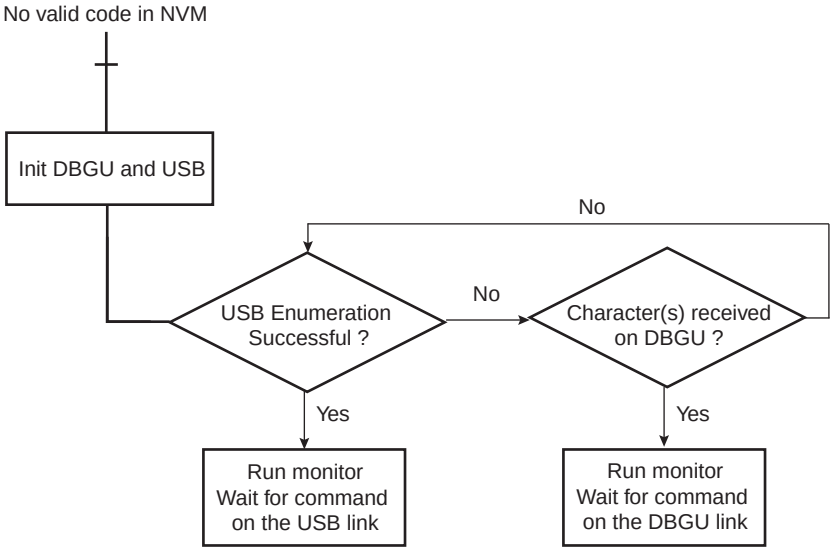
If no valid code is found in the NVM during the NVM bootloader sequence, the SAM-BA Monitor program is launched.

The SAM-BA Monitor principle is to:

- Initialize DBGU and USB
- Check if USB Device enumeration occurred
- Check if characters are 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 12-4.

Figure 12-10.SAM-BA Monitor Diagram



12.4.1 Command List

Table 12-4. 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 '>'

12.4.2 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.

12.4.2.1 Supported External Crystal/External Clocks

The SAM-BA Monitor supports a frequency of 12, 16, 24 or 48 MHz to allow DBGU communication for both external crystal and external clock.

12.4.2.2 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 maximum bit errors.

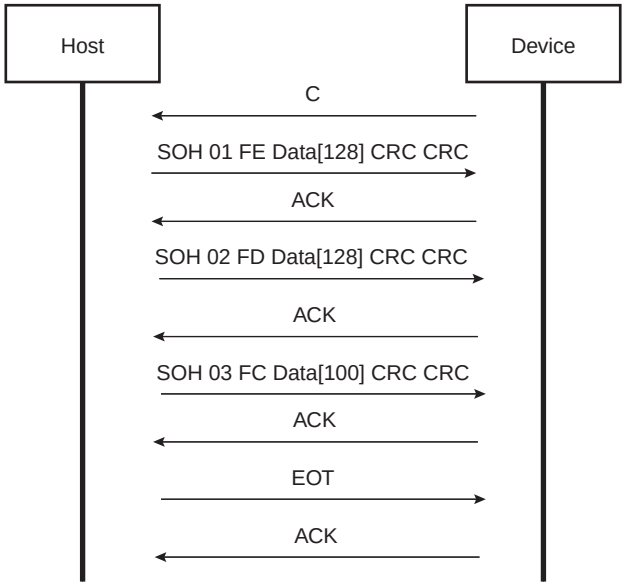
Xmodem protocol with CRC is supported by successful transmission reports provided both by a sender and by a receiver. Each transfer block is as follows:

<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 12-11 shows a transmission using this protocol.

Figure 12-11.Xmodem Transfer Example



12.4.3 USB Device Port

12.4.3.1 Supported External Crystal / External Clocks

The SAM-BA Monitor supports a frequency of 12, 16, 24 or 48 MHz to allow USB communication for both external crystal and external clock.

12.4.3.2 USB Class

The device uses the USB Communication Device Class (CDC) drivers to take advantage of the installed PC Serial Communication software to talk over the USB. The CDC class is implemented in all releases of Windows®, from Windows 98SE® to Windows 7®. 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 the 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, INF files contain the correspondence between vendor ID and product ID.

12.4.3.3 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 12-5. 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 12-6. 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 indicate to the DCE device that the DTE device is now present

Unhandled requests are STALLed.

12.4.3.4 Communication Endpoints

Endpoint 0 is used for the enumeration process.

Endpoint 1 (64-byte Bulk OUT) and endpoint 2 (64-byte Bulk IN) are used as communication endpoints.

SAM-BA Boot commands are sent by the host through Endpoint 1. If required, the message is split into several data payloads by the host driver.

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

13. Boot Sequence Controller (BSC)

13.1 Description

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

The BSC_CR is powered by VDDBU. Any modification of the register value is stored and applied after the next reset. The register defaults to the factory value in case of battery removal.

The BSC_CR is programmable with user programs or SAM-BA and is key-protected.

13.2 Embedded Characteristics

- VDDBU powered register

13.3 Product Dependencies

- Product-dependent order

13.4 Boot Sequence Controller (BSC) Registers User Interface

Table 13-1. Register Mapping

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

13.4.1 Boot Sequence Configuration Register

Name: BSC_CR

Address: 0xFFFFFE54

Access: Read-write

Factory Value: 0x0000_0000

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

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 WPKEY carries the valid value.

- **WPKEY: Write Protect Key** (Write-only)

Value	Name	Description
0x6683	PASSWD	Writing any other value in this field aborts the write operation of the BOOT field. Always reads as 0.

14. AXI Bus Matrix (AXIMX)

14.1 Description

The AXI Bus Matrix (AXIMX) comprises the embedded Advanced EXtensible Interface (AXI) bus protocol which supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

14.2 Embedded Characteristics

- High Performance AXI Network Interconnect
- 1 AXI Slave Interface
- 1 AHB-Lite Slave Interface
- 3 AXI Master Interfaces
- 1 APB3 Slave Interface
- Single-cycle Arbitration
- Full Pipelining to prevent Master Stalls
- 2 Remap States

14.3 Operation

14.3.1 Remap

There are two remap states using bits 0 and 1 in the Remap Register (AXIMX_REMAP)

- Bit 1 is used to remap EBI @ addr 0x00000000 for external boot.
- Bit 0 is used to remap RAM @ addr 0x00000000

Refer to Section 14.4 ”AXI Matrix (AXIMX) User Interface" and Table 14-1, “Register Mapping”.

The number of remap states can be defined using eight bits of the remap register, and a bit in the remap register controls each remap state.

Each remap state can be used to control the address decoding for one or more slave interfaces. If a slave interface is affected by two remap states that are both asserted, the remap state with the lowest remap bit number takes precedence.

Each slave interface can be configured independently so that a remap state can perform different functions for different masters.

A remap state can:

- Alias a memory region into two different address ranges
- Move an address region
- Remove an address region

Because of the nature of the distributed register sub-system, the masters receive the updated remap bit states in sequence, and not simultaneously.

A slave interface does not update to the latest remap bit setting until:

- The address completion handshake accepts any transaction that is pending
- Any current lock sequence completes

The BRESP from a GPV after a remap update guarantees that the next transaction issued to each slave interface, or the first one after the completion of a locked sequence, uses the updated value.

The AXI Matrix uses two remap bits. At powerup, ROM is seen at address 0 After powerup, ahbslave can be moved down to address 0 by means of the remap bits.

Figure 14-1 shows the memory map when remap is set to 000, representing no remap,

Figure 14-1. No Remap

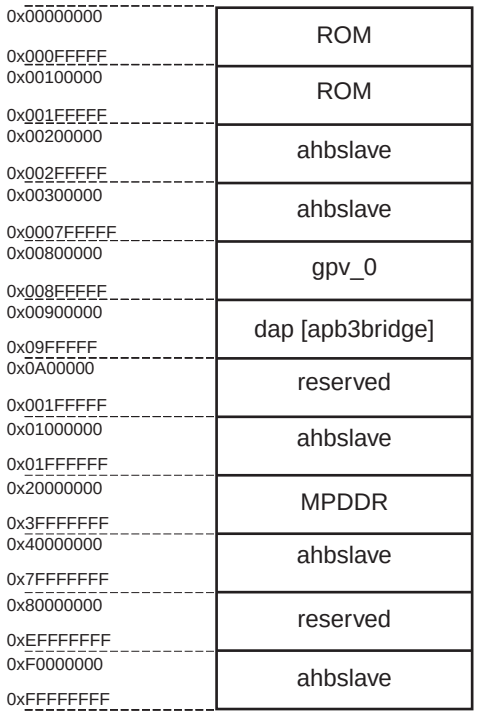
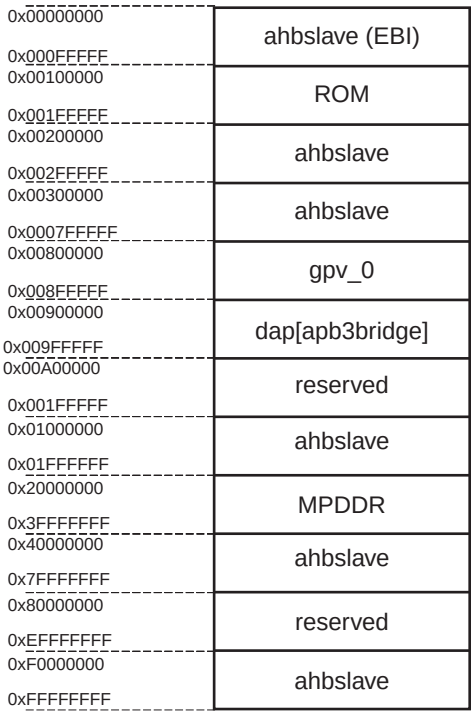


Figure 14-2 shows mapping when remap state is 01 or 11. This state is used for RAM boot. RAM is seen at address 0 through ahbslave.

Figure 14-2. Remap state is 01 or 11

Figure 14-3 shows mapping when remap state is 10. This state is used for external boot. EBI is seen at address 0 through ahbslave.

Figure 14-3. Remap State is 10



14.4 AXI Matrix (AXIMX) User Interface

Table 14-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Remap Register	AXIMX_REMAP	Write-only	0x00000000
0x04 - 0x43108	Reserved	–	–	0x00000000

14.4.1 AXI Matrix Remap Register

Name: AXIMX_REMAP

Address: 0x00800000

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
–	–	–	–	_	_	REMAP1	REMAP0

REMAP0 has higher priority than REMAP1, i.e., if both REMAP0 & REMAP1 are asserted, the matrix is in remap state 0.

- **REMAP0: Remap State 0**

SRAM is seen at address 0x00000000 (through AHB slave interface) instead of ROM.

- **REMAP1: Remap State 1**

HEBI is seen at address 0x00000000 (through AHB slave interface) instead of ROM for external boot.

15. Bus Matrix (MATRIX)

15.1 Description

The Bus Matrix (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.

15.1.1 Matrix Masters

The Bus Matrix of the SAMA5D3 product manages 15 masters, which means that each master can perform an access concurrently with others, to an available slave.

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

List of Bus Matrix Masters

Master 0	Cortex A5
Master 1, 2, 3	DMA Controller 0
Master 4, 5, 6	DMA Controller 1
Master 7	GMAC DMA
Master 8, 9	LCDC DMA
Master 10	UHP EHCI DMA
Master 11	UHP OHCI DMA
Master 12	UDPHS DMA
Master 13	EMAC DMA
Master 14	ISI DMA

15.1.2 Matrix Slaves

The Bus Matrix of the SAMA5 product manages 13 slaves. Each slave has its own arbiter, allowing a different arbitration per slave.

Table 15-1. List of Bus Matrix Slaves

Slave 0	Internal SRAM0
Slave 1	Internal SRAM1
Slave 2	NFC SRAM
Slave 3	Internal ROM
Slave 4	Soft Modem (SMD)
Slave 5	USB Device High Speed Dual Port RAM (DPR)
	USB Host OHCI registers
	USB Host EHCI registers
Slave 6	External Bus Interface/NFC
Slave 7	DDR2 Port0
Slave 8	DDR2 Port1
Slave 9	DDR2 Port2

Table 15-1. List of Bus Matrix Slaves

Slave 10	DDR2 Port3
Slave 11	Peripheral Bridge 0
Slave 12	Peripheral Bridge 1

15.1.3 Master to Slave Access

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

Table 15-2. SAMA5 Master to Slave Access

Masters		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Slaves		A5	DMAC0			DMAC1			GMAC DMA	LCDC DMA		UHPHS EHCI DMA	UHPHS OHCI DMA	UDPHS DMA	EMAC DMA	ISI DMA
0	Internal SRAM0	X	X			X			X			X	X	X	X	
1	Internal SRAM1	X	X			X			X			X	X	X	X	
2	NFC SRAM	X	X													
3	Internal ROM	X	X									X	X	X		
4	SMD	X	X													
5	UDPHS RAM	X														
	UHP OHCI Reg															
	UHP EHCI Reg															
6	EBI CS0..CS3	X	X			X	X		X	X	X	X	X	X	X	X
	NFC Command Register	X														
7	DDR2 Port 0	X														
8	DDR2 port1			X		X			X							
9	DDR2 port2		X				X			X			X			
10	DDR2 port3										X	X		X	X	X
11	APB 0	X			X											
12	APB 1	X						X								

15.2 Embedded Characteristics

- AMBA Advanced High-performance Bus (AHB Lite) Compliant Interfaces
- 32-bit or 64-bit Data Bus
- APB Compliant User Interface
- Configurable Number of Masters (Up to sixteen)
- Configurable Number of Slaves (Up to sixteen)
- One Decoder for Each Master
- Several Possible Boot Memories for Each Master before Remap
- One Remap Function for Each Master
- Support for Long Bursts of 32, 64, 128 and Up to the 256-beat Word Burst AHB Limit
- Enhanced Programmable Mixed Arbitration for Each Slave
  - Round-Robin
  - Fixed Priority
- Programmable Default Master for Each Slave
  - No Default Master
  - Last Accessed Default Master
  - Fixed Default Master
- Deterministic Maximum Access Latency for Masters
- Zero or One Cycle Arbitration Latency for the First Access of a Burst
- Bus Lock Forwarding to Slaves
- Master Number Forwarding to Slaves
- One Special Function Register for Each Slave (Not dedicated)
- Write Protection of User Interface Registers

15.3 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.

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

15.4 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 for a single transfer, as long as the slave is free from any other master access. It does not provide any benefit if 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 15.10.2 “Bus Matrix Slave Configuration Registers” on page 97.

15.5 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 whatever the number of requesting masters.

15.6 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 whatever is the number of requesting masters.

15.7 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, regardless of the number of requesting masters.

15.8 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 2 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 must be done, specific conditions apply. See Section 15.8.1 “Arbitration Scheduling” on page 91.

15.8.1 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 “Undefined Length Burst Arbitration” on page 91
4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See “Slot Cycle Limit Arbitration” on page 92

15.8.1.1 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.

The use of undefined length16-beat bursts, or less, is discouraged since this generally decreases significantly the 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).

15.8.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At 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.

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.

15.8.2 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. In the worst case, 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 the highest priority pool (round-robin), and remaining masters the lowest priority pool (round-robin), with no master for intermediate fix priority levels.

If more than one master requests the slave bus, regardless 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.

15.8.2.1 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.

15.8.2.2 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.

15.9 Write Protect Registers

To prevent any single software error that may corrupt the Bus Matrix behavior, the entire Bus Matrix address space can be write-protected by setting the WPEN bit in the Bus Matrix Write Protect Mode Register (MATRIX_WPMR).

If WPEN is at one and a write access in the Bus Matrix address space is detected, then the WPVS flag in the Bus Matrix Write Protect Status Register (MATRIX_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 Bus Matrix Write Protect Mode Register (MATRIX_WPMR) with the appropriate access key WPKEY.

The protected registers are:

“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”

15.10 Bus Matrix (MATRIX) User Interface

Table 15-3. 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	Master Configuration Register 9	MATRIX_MCFG9	Read-write	0x00000000
0x0028	Master Configuration Register 10	MATRIX_MCFG10	Read-write	0x00000000
0x002C	Master Configuration Register 11	MATRIX_MCFG11	Read-write	0x00000000
0x0030	Master Configuration Register 12	MATRIX_MCFG12	Read-write	0x00000000
0x0034	Master Configuration Register 13	MATRIX_MCFG13	Read-write	0x00000000
0x0038	Master Configuration Register 14	MATRIX_MCFG14	Read-write	0x00000000
0x003C	Master Configuration Register 15	MATRIX_MCFG15	Read-write	0x00000000
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	Slave Configuration Register 10	MATRIX_SCFG10	Read-write	0x000001FF
0x006C	Slave Configuration Register 11	MATRIX_SCFG11	Read-write	0x000001FF
0x0070	Slave Configuration Register 12	MATRIX_SCFG12	Read-write	0x000001FF
0x0074	Slave Configuration Register 13	MATRIX_SCFG13	Read-write	0x000001FF
0x0078	Slave Configuration Register 14	MATRIX_SCFG14	Read-write	0x000001FF
0x007C	Slave Configuration Register 15	MATRIX_SCFG15	Read-write	0x000001FF
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read-write	0x33333333(1)
0x0084	Priority Register B for Slave 0	MATRIX_PRBS0	Read-write	0x33333333(1)
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read-write	0x33333333(1)

Table 15-3. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x008C	Priority Register B for Slave 1	MATRIX_PRBS1	Read-write	0x33333333(1)
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read-write	0x33333333(1)
0x0094	Priority Register B for Slave 2	MATRIX_PRBS2	Read-write	0x33333333(1)
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read-write	0x33333333(1)
0x009C	Priority Register B for Slave 3	MATRIX_PRBS3	Read-write	0x33333333(1)
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read-write	0x33333333(1)
0x00A4	Priority Register B for Slave 4	MATRIX_PRBS4	Read-write	0x33333333(1)
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Read-write	0x33333333(1)
0x00AC	Priority Register B for Slave 5	MATRIX_PRBS5	Read-write	0x33333333(1)
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Read-write	0x33333333(1)
0x00B4	Priority Register B for Slave 6	MATRIX_PRBS6	Read-write	0x33333333(1)
0x00B8	Priority Register A for Slave 7	MATRIX_PRAS7	Read-write	0x33333333(1)
0x00BC	Priority Register B for Slave 7	MATRIX_PRBS7	Read-write	0x33333333(1)
0x00C0	Priority Register A for Slave 8	MATRIX_PRAS8	Read-write	0x33333333(1)
0x00C4	Priority Register B for Slave 8	MATRIX_PRBS8	Read-write	0x33333333(1)
0x00C8	Priority Register A for Slave 9	MATRIX_PRAS9	Read-write	0x33333333(1)
0x00CC	Priority Register B for Slave 9	MATRIX_PRBS9	Read-write	0x33333333(1)
0x00D0	Priority Register A for Slave 10	MATRIX_PRAS10	Read-write	0x33333333(1)
0x00D4	Priority Register B for Slave 10	MATRIX_PRBS10	Read-write	0x33333333(1)
0x00D8	Priority Register A for Slave 11	MATRIX_PRAS11	Read-write	0x33333333(1)
0x00DC	Priority Register B for Slave 11	MATRIX_PRBS11	Read-write	0x33333333(1)
0x00E0	Priority Register A for Slave 12	MATRIX_PRAS12	Read-write	0x33333333(1)
0x00E4	Priority Register B for Slave 12	MATRIX_PRBS12	Read-write	0x33333333(1)
0x00E8	Priority Register A for Slave 13	MATRIX_PRAS13	Read-write	0x33333333(1)
0x00EC	Priority Register B for Slave 13	MATRIX_PRBS13	Read-write	0x33333333(1)
0x00F0	Priority Register A for Slave 14	MATRIX_PRAS14	Read-write	0x33333333(1)
0x00F4	Priority Register B for Slave 14	MATRIX_PRBS14	Read-write	0x33333333(1)
0x00F8	Priority Register A for Slave 15	MATRIX_PRAS15	Read-write	0x33333333(1)
0x00FC	Priority Register B for Slave 15	MATRIX_PRBS15	Read-write	0x33333333(1)
0x0100	Master Remap Control Register	MATRIX_MRCR	Read-write	0x00000000
0x0104 - 0x010C	Reserved	–	–	–
0x01A0 - 0x01E0	Reserved	–	–	–
0x01E4	Write Protect Mode Register	MATRIX_WPMR	Read-write	0x00000000
0x01E8	Write Protect Status Register	MATRIX_WPSR	Read-only	0x00000000

Notes: 1. Values in the Bus Matrix Priority Registers are product dependent.

15.10.1 Bus Matrix Master Configuration Registers

Name: MATRIX_MCFG0...MATRIX_MCFG15

Address: 0xFFFFEC00

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 Protect 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.

This value should not be used in the very particular case of a master capable of performing back-to-back undefined length bursts on a single slave, since this could indefinitely freeze the slave arbitration and thus prevent another master from accessing this slave.

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 or bursts sequence.

2: 4-beat Burst

The undefined length burst or bursts sequence is split into 4-beat bursts or less, allowing re-arbitration every 4 beats.

3: 8-beat Burst

The undefined length burst or bursts sequence is split into 8-beat bursts or less, allowing re-arbitration every 8 beats.

4: 16-beat Burst

The undefined length burst or bursts sequence is split into 16-beat bursts or less, allowing re-arbitration every 16 beats.

5: 32-beat Burst

The undefined length burst or bursts sequence is split into 32-beat bursts or less, allowing re-arbitration every 32 beats.

6: 64-beat Burst

The undefined length burst or bursts sequence is split into 64-beat bursts or less, allowing re-arbitration every 64 beats.

7: 128-beat Burst

The undefined length burst or bursts sequence is split into 128-beat bursts or less, allowing re-arbitration every 128 beats.

Unless duly needed, the ULBT should be left at its default 0 value for power saving.

15.10.2 Bus Matrix Slave Configuration Registers

Name: MATRIX_SCFG0...MATRIX_SCFG15

Address: 0xFFFFEC40

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 Protect 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 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.

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 “Slot Cycle Limit Arbitration” on page 92 for details.

- **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 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 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.

15.10.3 Bus Matrix Priority Registers A For Slaves

Name: MATRIX_PRAS0...MATRIX_PRAS15

Address: 0xFFFFEC80 [0], 0xFFFFEC88 [1], 0xFFFFEC90 [2], 0xFFFFEC98 [3], 0xFFFFECA0 [4], 0xFFFFECA8 [5], 0xFFFFECB0 [6], 0xFFFFECB8 [7], 0xFFFFECC0 [8], 0xFFFFECC8 [9], 0xFFFFECD0 [10], 0xFFFFECD8 [11], 0xFFFFECE0 [12], 0xFFFFECE8 [13], 0xFFFFECF0 [14], 0xFFFFECF8 [15]

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 WPE bit is cleared in the “Write Protect 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 92 for details.

15.10.4 Bus Matrix Priority Registers B For Slaves

Name: MATRIX_PRBS0...MATRIX_PRBS15

Address: 0xFFFFEC84 [0], 0xFFFFEC8C [1], 0xFFFFEC94 [2], 0xFFFFEC9C [3], 0xFFFFECA4 [4], 0xFFFFECAC [5], 0xFFFFECB4 [6], 0xFFFFECBC [7], 0xFFFFECC4 [8], 0xFFFFECCC [9], 0xFFFFECD4 [10], 0xFFFFECDC [11], 0xFFFFECE4 [12], 0xFFFFECEC [13], 0xFFFFECF4 [14], 0xFFFFECFC [15]

Access: Read-write

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

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

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

7	6	5	4	3	2	1	0
–	–	M9PR		–	–	M8PR	

This register can only be written if the WPEN bit is cleared in the “Write Protect 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 92 for details.

15.10.5 Bus Matrix Master Remap Control Register

Name: MATRIX_MRCR

Address: 0xFFFFED00

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
RCB15	RCB14	RCB13	RCB12	RCB11	RCB10	RCB9	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 Protect Mode Register” .

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

0: Disable remapped address decoding for the selected Master

1: Enable remapped address decoding for the selected Master

15.10.6 Write Protect Mode Register

Name: MATRIX_WPMR

Address: 0xFFFFEDE4

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

For more details on MATRIX_WPMR, refer to Section 15.9 “Write Protect Registers” on page 93.

The protected registers are:

“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”

- **WPEN: Write Protect Enable**

0: Disables the Write Protect if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

1: Enables the Write Protect if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

Protects the entire Bus Matrix address space from address offset 0x000 to 0x1FC.

- **WPKEY: Write Protect KEY** (Write-only)

Should be written at value 0x4D4154 (“MAT” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

15.10.7 Write Protect Status Register

Name: MATRIX_WPSR

Address: 0xFFFFEDE8

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

For more details on MATRIX_WPSR, refer to Section 15.9 “Write Protect Registers” on page 93.

- **WPVS: Write Protect Violation Status**

0: No Write Protect Violation has occurred since the last write of the MATRIX_WPMR.

1: At least one Write Protect Violation has occurred since the last write of the MATRIX_WPMR.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the register address offset in which a write access has been attempted.

Otherwise it reads as 0.

16. Special Function Registers (SFR)

16.1 Description

Special Function Registers (SFR) manage specific aspects of the integrated memory, bridge implementations, processor and other functionality not controlled elsewhere.

16.2 Embedded Characteristics

- 32-bit Special Function Registers control specific behavior of the product

16.3 Special Function Registers (SFR) User Interface

Table 16-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00–0x04	Reserved	–	–	–
0x08–0x0C	Reserved	–	–	–
0x10	OHCI Interrupt Configuration Register	SFR_OHCIICR	Read-write	0x0
0x14	OHCI Interrupt Status Register	SFR_OHCIISR	Read-only	–
0x18	Reserved	–	–	–
0x1C	Reserved	–	–	–
0x20–0x24	Reserved	–	–	–
0x28	Security Configuration Register	SFR_SECURE	Read-write	0x0
0x2C	Reserved	–	–	–
0x30	UTMI Clock Trimming Register	SFR_UTMICKTRIM	Read-write	0x00010000
0x40	EBI Configuration Register	SFR_EBICFG	Read-write	–
0x44	Reserved	–	–	–
0x48	Reserved	–	–	–
0x4C–0x50	Reserved	–	–	–
0x54–0x3FFC	Reserved	–	–	–

16.3.1 OHCI Interrupt Configuration Register

Name: SFR_OHCIICR

Address: 0xF0038010

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
UDPPUDIS	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	APPSTART	ARIE	–	RES2	RES1	RES0

- **RESx: USB PORTx RESET**

0: Resets USB PORT.

1: Usable USB PORT.

- **ARIE: OHCI Asynchronous Resume Interrupt Enable**

0: Interrupt disabled.

1: Interrupt enabled.

- **APPSTART: Reserved**

0: Must write 0.

- **UDPPUDIS: USB DEVICE PULL-UP DISABLE**

0: USB device Pull-up connection is enabled.

1: USB device Pull-up connection is disabled.

16.3.2 OHCI Interrupt Status Register

Name: SFR_OHCIISR

Address: 0xF0038014

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
–	–	–	–	–	RIS2	RIS1	RIS0

- **RISx: OHCI Resume Interrupt Status Port x**

0: OHCI Port resume not detected.

1: OHCI Port resume detected.

16.3.3 APB Bridge Configuration Register

Name: SFR_BRIDGE

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
–	–	–	–	–	–	–	AXI2AHBSEL
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	APBTURBO

- **APBTURBO: AHB to APB Bridge Mode**

0: AHB transaction optimization disabled.

1: AHB transaction optimization enabled.

- **AXI2AHBSEL: AXI to AHB Bridge for DDR Controller Selection**

0 (SINGLE): Uses single port bridge.

1 (DUAL): Uses dual port bridge.

16.3.4 Security Configuration Register

Name: SFR_SECURE

Address: 0xF0038028

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
–	–	–	–	–	–	–	FUSE
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ROM

- **ROM: Disable Access to ROM Code**

This bit is writable once only. When the ROM is secured, only reset signal can clear this bit.

0: ROM is enabled.

1: ROM is disabled.

- **FUSE: Disable Access to Fuse Controller**

This bit is writable once only. When the Fuse Controller is secured, only reset signal can clear this bit.

0: Fuse Controller is enabled.

1: Fuse Controller is disabled.

16.3.5 UTMI Clock Trimming Register

Name: SFR_UTMICKTRIM

Address: 0xF0038030

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
–	–	–	–	–	–	FREQ	

- **FREQ: UTMI Reference Clock Frequency**

Value	Name	Description
0	12	12 MHz reference clock
1	16	16 MHz reference clock
2	24	24 MHz reference clock
3	48	48 MHz reference clock

16.3.6 EBI Configuration Register

Name: SFR_EBICFG

Address: 0xF0038040

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	BMS
15	14	13	12	11	10	9	8
–	–	–	SCH1	PULL1		DRIVE1	
7	6	5	4	3	2	1	0
–	–	–	SCH0	PULL0		DRIVE0	

This register controls EBI pins which are not multiplexed with PIO controller lines.

DRIVE0, PULL0, SCH0 control EBI Data pins when applicable.

DRIVE1, PULL1, SCH1 control other EBI pins when applicable.

- **DRIVEx: EBI Pins Drive Level**

Drive level should be programmed depending on target frequency and board characteristics. Refer to pad characteristics to set correct drive level.

Value	Name	Description
0	LOW	Low drive level
1	RESERVED	Low drive level
2	MEDIUM	Medium drive level
3	HIGH	High drive level

- **PULLx: EBI Pins Pull Value**

Value	Name	Description
0	UP	Pull-up
1	NONE	No Pull
2	Reserved	No Change (forbidden write value)
3	DOWN	Pull-down

- **SCHx: EBI Pins Schmitt Trigger**

0: Schmitt Trigger off.

1: Schmitt Trigger on.

- **BMS: BMS Sampled Value (Read Only)**

This bit examines whether boot is on EBI or ROM.

0 (ROM): Boot on ROM.

1 (EBI): Boot on EBI.

17. Advanced Interrupt Controller (AIC)

17.1 Description

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to hundred and twenty-eight 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.

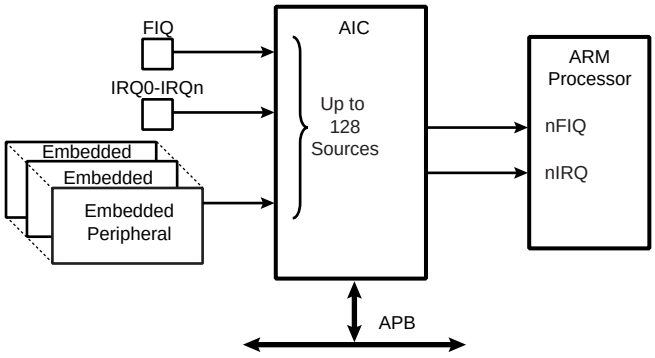
The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

17.2 Embedded Characteristics

- Controls the Interrupt Lines (nIRQ and nFIQ) of an ARM® Processor
- 128 Individually Maskable and Vectored Interrupt Sources
  - Source 0 is Reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is Reserved for System Peripheral Interrupts
  - Source 2 to Source 127, Control up to 126 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 127
  - 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 for all Interrupt Sources
  - 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

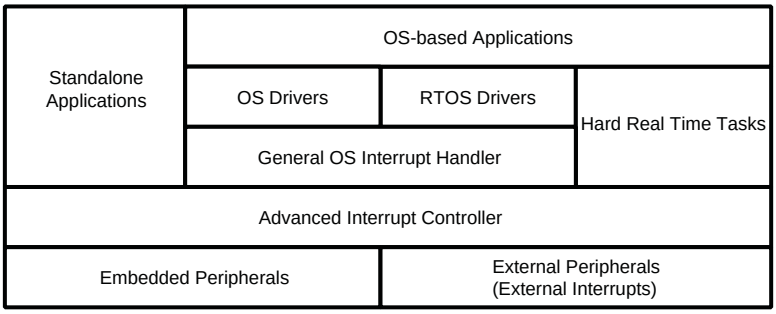
17.3 Block Diagram

Figure 17-1. Block Diagram



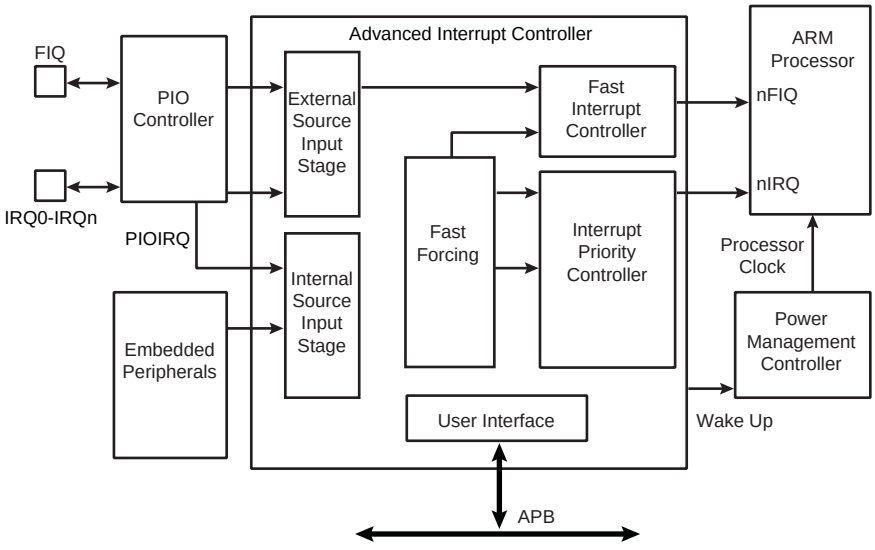
17.4 Application Block Diagram

Figure 17-2. Description of the Application Block



17.5 AIC Detailed Block Diagram

Figure 17-3. AIC Detailed Block Diagram



17.6 I/O Line Description

Table 17-1. I/O Line Description

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

17.7 Product Dependencies

17.7.1 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 17-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
AIC	FIQ	PC31	A
AIC	IRQ	PE31	A

17.7.2 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.

17.7.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature a 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 127 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 127.

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 PID127.

17.8 Functional Description

17.8.1 Interrupt Source Control

17.8.1.1 Interrupt Source Mode

The Advanced Interrupt Controller independently programs each interrupt source. The SRCTYPE field of the AIC_SMR (Source Mode Register) selects the interrupt condition of the interrupt source selected by the INTSEL field of the "AIC Source Select Register".

Note: Configuration registers such as AIC_SMR, AIC_SSR, return the values corresponding to the interrupt source selected by INTSEL.

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.

17.8.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; “AIC Interrupt Enable Command Register” on page 139 and “AIC Interrupt Disable Command Register” on page 140. The interrupt mask of the selected interrupt source can be read in the AIC_IMR register. A disabled interrupt does not affect servicing of other interrupts.

17.8.1.3 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 117.) 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 Section 17.8.4.5 ”Fast Forcing”).

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

17.8.1.4 Interrupt Status

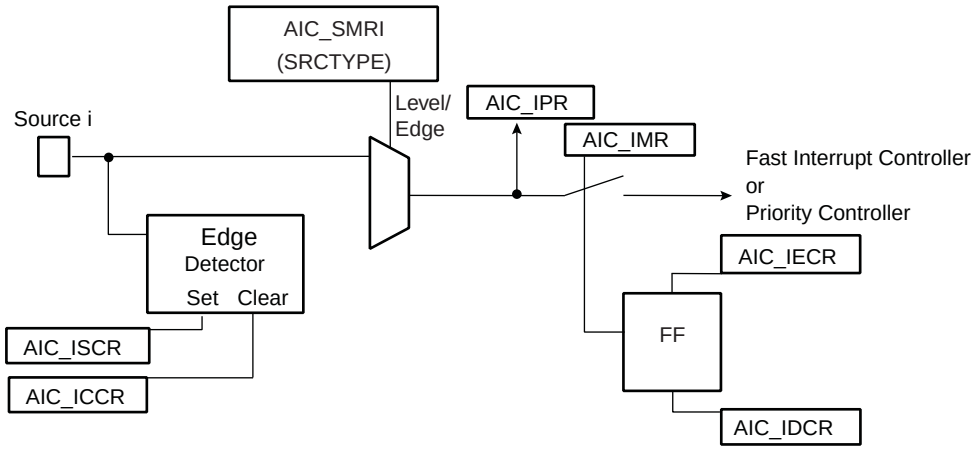
AIC_IPR registers represent the state of the interrupt lines, whether they are masked or not. The AIC_IMR register permits to define the mask of the interrupt lines.

The AIC_ISR register reads the number of the current interrupt (see “Priority Controller” on page 117) 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.

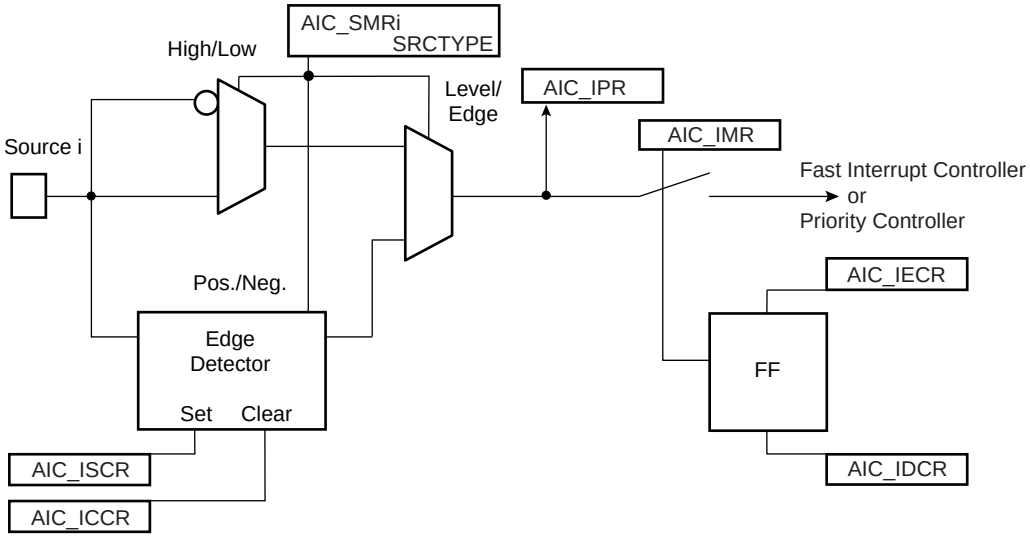
17.8.1.5 Internal Interrupt Source Input Stage

Figure 17-4. Internal Interrupt Source Input Stage



17.8.1.6 External Interrupt Source Input Stage

Figure 17-5. External Interrupt Source Input Stage



17.8.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

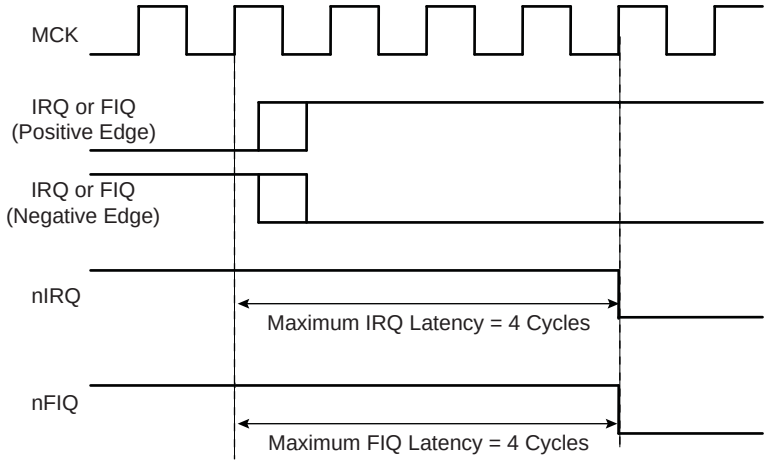
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

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.

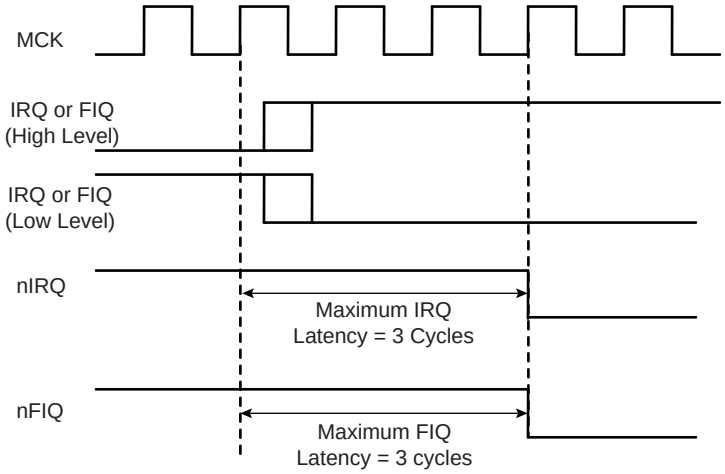
17.8.2.1 External Interrupt Edge Triggered Source

Figure 17-6. External Interrupt Edge Triggered Source



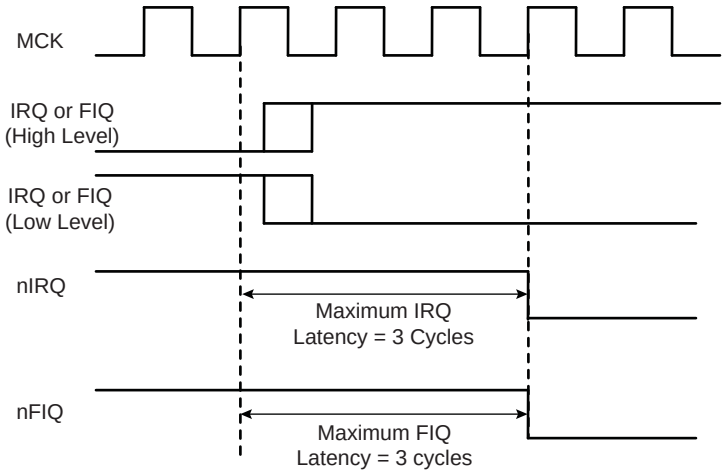
17.8.2.2 External Interrupt Level Sensitive Source

Figure 17-7. External Interrupt Level Sensitive Source



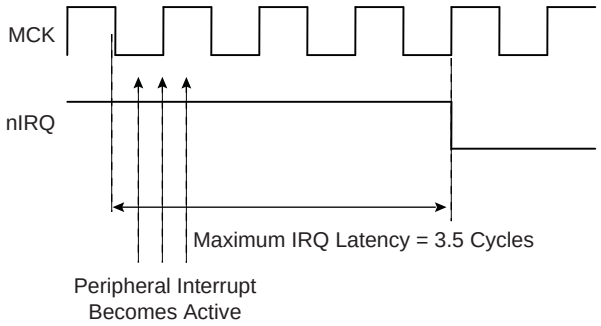
17.8.2.3 Internal Interrupt Edge Triggered Source

Figure 17-8. Internal Interrupt Edge Triggered Source



17.8.2.4 Internal Interrupt Level Sensitive Source

Figure 17-9. Internal Interrupt Level Sensitive Source



17.8.3 Normal Interrupt

17.8.3.1 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 127 (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 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.

17.8.3.2 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.

17.8.3.3 Interrupt Vectoring

The interrupt handler address corresponding to the interrupt source selected by the INTSEL field can be stored in the registers AIC_SVR (Source Vector Register). 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 the AIC_SVR of the interrupt sources 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.

17.8.3.4 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).

17.8.4 Fast Interrupt

17.8.4.1 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.

17.8.4.2 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_SMR and INTSEL = 0, the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR 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 when INTSEL = 0. The bit 0 of AIC_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

17.8.4.3 Fast Interrupt Vectoring

The fast interrupt handler address can be stored through the AIC_SVR (Source Vector Register). The value written into this register when INTSEL = 0 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.

17.8.4.4 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_SVR 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.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.
4. 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.
5. 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.

17.8.4.5 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).

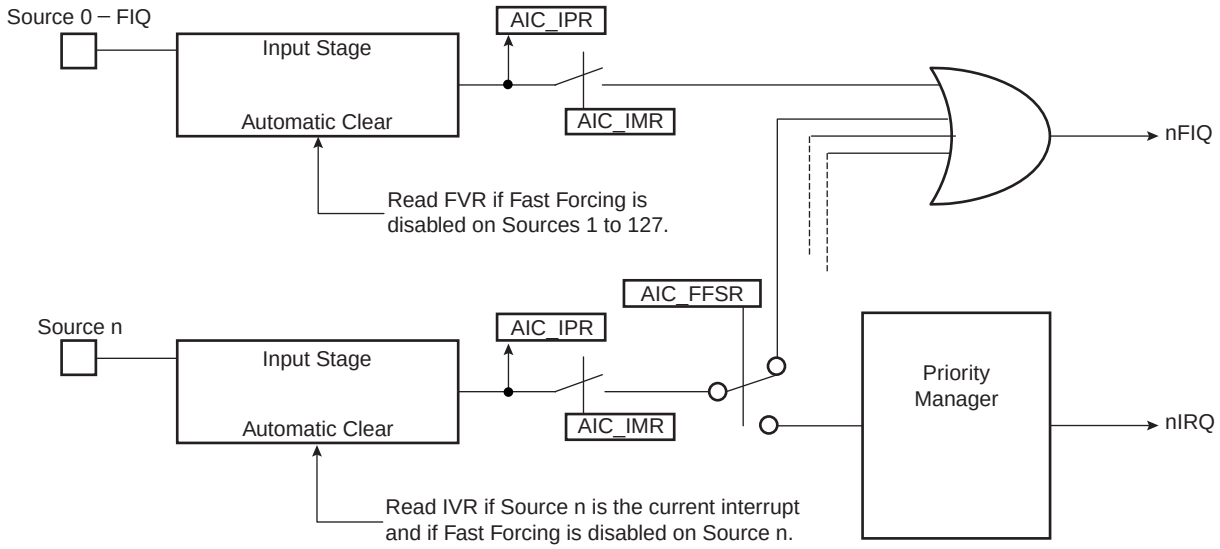
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt 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 17-10.Fast Forcing



17.8.5 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.

17.8.6 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.

17.8.7 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.

17.8.8 Write Protected 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”.

List of the write-protected registers:

- ”AIC Source Mode Register”
- ”AIC Source Vector Register”
- ”AIC Spurious Interrupt Vector Register”
- ”AIC Debug Control Register”

17.9 Advanced Interrupt Controller (AIC) User Interface

Table 17-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Source Select Register	AIC_SSR	Read-write	0x0
0x04	Source Mode Register	AIC_SMR	Read-write	0x0
0x08	Source Vector Register	AIC_SVR	Read-write	0x0
0x0C	Reserved	–	–	–
0x10	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x14	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x18	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x1C	Reserved	–	–	–
0x20	Interrupt Pending Register 0(2)	AIC_IPR0	Read-only	0x0(1)
0x24	Interrupt Pending Register 1(2)	AIC_IPR1	Read-only	0x0(1)
0x28	Interrupt Pending Register 2(2)	AIC_IPR2	Read-only	0x0(1)
0x2C	Interrupt Pending Register 3(2)	AIC_IPR3	Read-only	0x0(1)
0x30	Interrupt Mask Register	AIC_IMR	Read-only	0x0
0x34	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x38	End of Interrupt Command Register	AIC_EOICR	Write-only	–
0x3C	Spurious Interrupt Vector Register	AIC_SPU	Read-write	0x0
0x40	Interrupt Enable Command Register	AIC_IECR	Write-only	–
0x44	Interrupt Disable Command Register	AIC_IDCR	Write-only	–
0x48	Interrupt Clear Command Register	AIC_ICCR	Write-only	–
0x4C	Interrupt Set Command Register	AIC_ISCR	Write-only	–
0x50	Fast Forcing Enable Register	AIC_FFER	Write-only	–
0x54	Fast Forcing Disable Register	AIC_FFDR	Write-only	–
0x58	Fast Forcing Status Register	AIC_FFSR	Read-only	0x0
0x5C	Reserved	–	–	–
0x6C	Debug Control Register	AIC_DCR	Read-write	0x0
0xE4	Write Protect Mode Register	AIC_WPMR	Read-write	0x0
0xE8	Write Protect Status Register	AIC_WPSR	Read-only	0x0
0xEC - 0xFC	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...PID127 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.

17.9.1 AIC Source Select Register

Name: AIC_SSR

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
–	INTSEL						

- **INTSEL: Interrupt Line Selection**

0-127 = Selects the interrupt line to handle.

See Section 17.8.1.1 ”Interrupt Source Mode”.

17.9.2 AIC Source Mode Register

Name: AIC_SMR

Address: 0xFFFFF004

Access: Read-write

Reset: 0x0

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

- **PRIOR: Priority Level**

Programs the priority level of the source selected by INTSEL in except FIQ source (source 0).

The priority level can be between 0 (lowest) and 7 (highest).

The priority level is not used for the FIQ.

- **SRCTYPE: Interrupt Source Type**

The active level or edge is not programmable for the internal interrupt source selected by INTSEL.

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

Value		Internal Interrupt	External Interrupt
0	0	High level sensitive	Low level sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level sensitive	High level sensitive
1	1	Positive edge triggered	Positive edge triggered

17.9.3 AIC Source Vector Register

Name: AIC_SVR

Address: 0xFFFFF008

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
VECTOR							

23	22	21	20	19	18	17	16
VECTOR							

15	14	13	12	11	10	9	8
VECTOR							

7	6	5	4	3	2	1	0
VECTOR							

- **VECTOR: Source Vector**

The user may store in this register the address of the corresponding handler for the interrupt source selected by INTSEL.

17.9.4 AIC Interrupt Vector Register

Name: AIC_IVR

Address: 0xFFFFF010

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.

17.9.5 AIC FIQ Vector Register

Name: AIC_FVR

Address: 0xFFFFF014

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 when INTSEL = 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

17.9.6 AIC Interrupt Status Register

Name: AIC_ISR

Address: 0xFFFFF018

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.

17.9.7 AIC Interrupt Pending Register 0

Name: AIC_IPR0

Address: 0xFFFFF020

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, PIDx: Interrupt Pending**

0 = The corresponding interrupt is not pending.

1 = The corresponding interrupt is pending.

17.9.8 AIC Interrupt Pending Register 1

Name: AIC_IPR1

Address: 0xFFFFF024

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
PID63	PID62	PID61	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
PID55	PID54	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	PID38	PID37	PID36	PID35	PID34	PID33	PID32

- **PIDx: Interrupt Pending**

0 = The corresponding interrupt is not pending.

1 = The corresponding interrupt is pending.

17.9.9 AIC Interrupt Pending Register 2

Name: AIC_IPR2

Address: 0xFFFFF028

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
PID95	PID94	PID93	PID92	PID91	PID90	PID89	PID88
23	22	21	20	19	18	17	16
PID87	PID86	PID85	PID84	PID83	PID82	PID81	PID80
15	14	13	12	11	10	9	8
PID79	PID78	PID77	PID76	PID75	PID74	PID73	PID72
7	6	5	4	3	2	1	0
PID71	PID70	PID69	PID68	PID67	PID66	PID65	PID64

- **PIDx: Interrupt Pending**

0 = The corresponding interrupt is not pending.

1 = The corresponding interrupt is pending.

17.9.10 AIC Interrupt Pending Register 3

Name: AIC_IPR3

Address: 0xFFFFF02C

Access: Read-only

Reset: 0x0

31	30	29	28	27	26	25	24
PID127	PID126	PID125	PID124	PID123	PID122	PID121	PID120
23	22	21	20	19	18	17	16
PID119	PID118	PID117	PID116	PID115	PID114	PID113	PID112
15	14	13	12	11	10	9	8
PID111	PID110	PID109	PID108	PID107	PID106	PID105	PID104
7	6	5	4	3	2	1	0
PID103	PID102	PID101	PID100	PID99	PID98	PID97	PID96

- **PIDx: Interrupt Pending**

0 = The corresponding interrupt is not pending.

1 = The corresponding interrupt is pending.

17.9.11 AIC Interrupt Mask Register

Name: AIC_IMR

Address: 0xFFFFF030

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
–	–	–	–	–	–	–	INTM

- **INTM: Interrupt Mask**

0 = The interrupt source selected by INTSEL is disabled.

1 = The interrupt source selected by INTSEL is enabled.

17.9.12 AIC Core Interrupt Status Register

Name: AIC_CISR

Address: 0xFFFFF034

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.

17.9.13 AIC End of Interrupt Command Register

Name: AIC_EOICR

Address: 0xFFFFF038

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
–	–	–	–	–	–	–	ENDIT

- **ENDIT: Interrupt Processing Complete Command**

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.

17.9.14 AIC Spurious Interrupt Vector Register

Name: AIC_SPU

Address: 0xFFFFF03C

Access: Read-write

Reset: 0x0

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

- **SIVR: Spurious Interrupt Vector Register**

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC_IVR in case of a spurious interrupt and in AIC_FVR in case of a spurious fast interrupt.

17.9.15 AIC Interrupt Enable Command Register

Name: AIC_IECR

Address: 0xFFFFF040

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
–	–	–	–	–	–	–	INTEN

- **INTEN: Interrupt Enable**

0 = No effect.

1 = Enables the interrupt source selected by INTSEL.

17.9.16 AIC Interrupt Disable Command Register

Name: AIC_IDCR

Address: 0xFFFFF044

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
–	–	–	–	–	–	–	INTD

- **INTD: Interrupt Disable**

0 = No effect.

1 = Disables the interrupt source selected by INTSEL.

17.9.17 AIC Interrupt Clear Command Register

Name: AIC_ICCR

Address: 0xFFFFF048

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
–	–	–	–	–	–	–	INTCLR

- **INTCLR: Interrupt Clear**

Clears one the following depending on the setting of the INTSEL bit FIQ, SYS, PID2-PID127

0 = No effect.

1 = Clears the interrupt source selected by INTSEL.

17.9.18 AIC Interrupt Set Command Register

Name: AIC_ISCR

Address: 0xFFFFF04C

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
–	–	–	–	–	–	–	INTSET

- **INTSET: Interrupt Set**

0 = No effect.

1 = Sets the interrupt source selected by INTSEL.

17.9.19 AIC Fast Forcing Enable Register

Name: AIC_FFER

Address: 0xFFFFF050

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
–	–	–	–	–	–	–	FFEN

- **FFEN: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the interrupt source selected by INTSEL.

17.9.20 AIC Fast Forcing Disable Register

Name: AIC_FFDR

Address: 0xFFFFF054

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
–	–	–	–	–	–	–	FFDIS

- **FFDIS: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the interrupt source selected by INTSEL.

17.9.21 AIC Fast Forcing Status Register

Name: AIC_FFSR

Address: 0xFFFFF058

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
–	–	–	–	–	–	–	FFS

- **FFS: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the interrupt source selected by INTSEL.

1 = The Fast Forcing feature is enabled on the interrupt source selected by INTSEL.

17.9.22 AIC Debug Control Register

Name: AIC_DCR

Address: 0xFFFFF06C

Access: Read-write

Reset: 0x0

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

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

17.9.23 AIC Write Protect Mode Register

Name: AIC_WPMR

Address: 0xFFFFF0E4

Access: Read-write

Reset: See Table 17-3

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”
- ”AIC Source Vector Register”
- ”AIC Spurious Interrupt Vector Register”
- ”AIC Debug Control Register”

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x414943	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

17.9.24 AIC Write Protect Status Register

Name: AIC_WPSR

Address: 0xFFFFF0E8

Access: Read-only

Reset: See Table 17-3

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.

18. Watchdog Timer (WDT)

18.1 Description

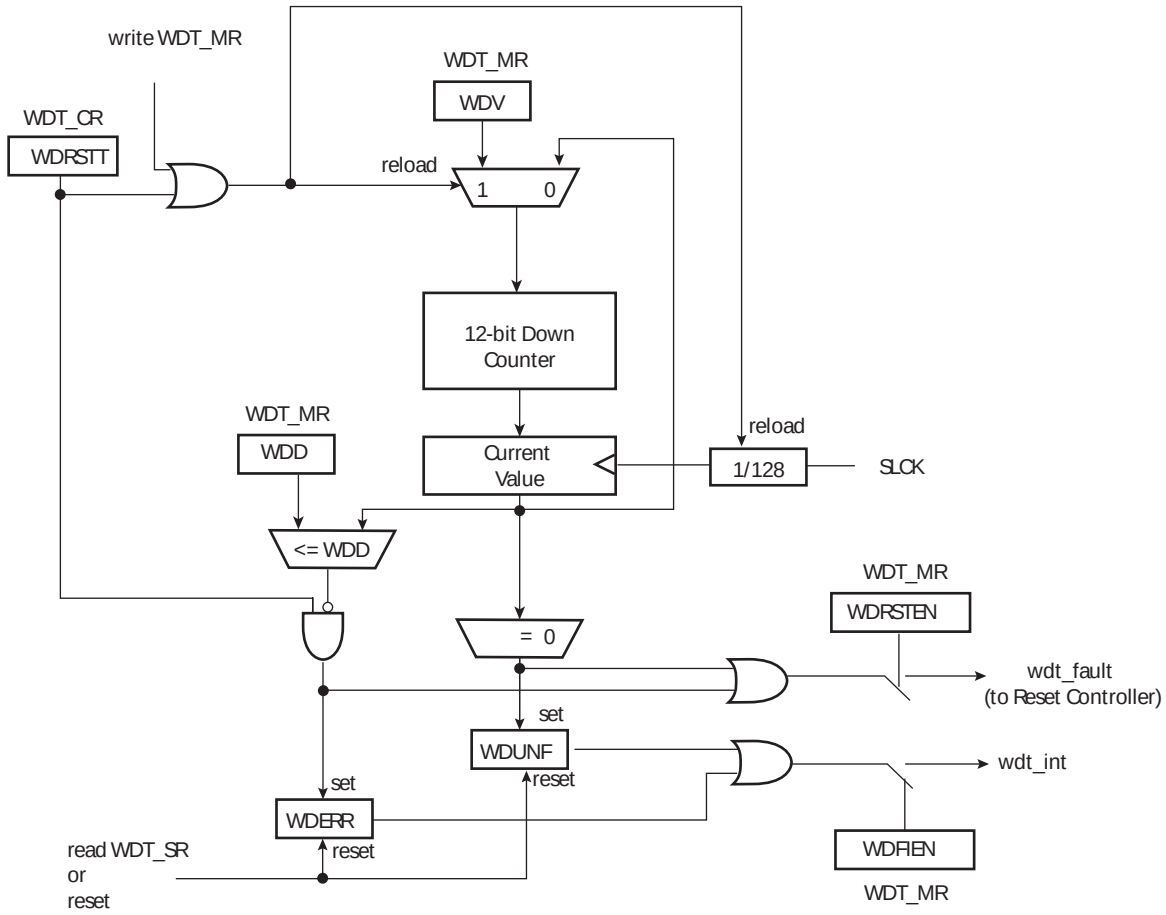
The Watchdog Timer (WDT) can be used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock around 32 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in debug mode or idle mode.

18.2 Embedded Characteristics

- 12-bit key-protected programmable counter
- Watchdog Clock is independent from Processor Clock
- Provides reset or interrupt signals to the system
- Counter may be stopped while the processor is in debug state or in idle mode

18.3 Block Diagram

Figure 18-1. Watchdog Timer Block Diagram



18.4 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).

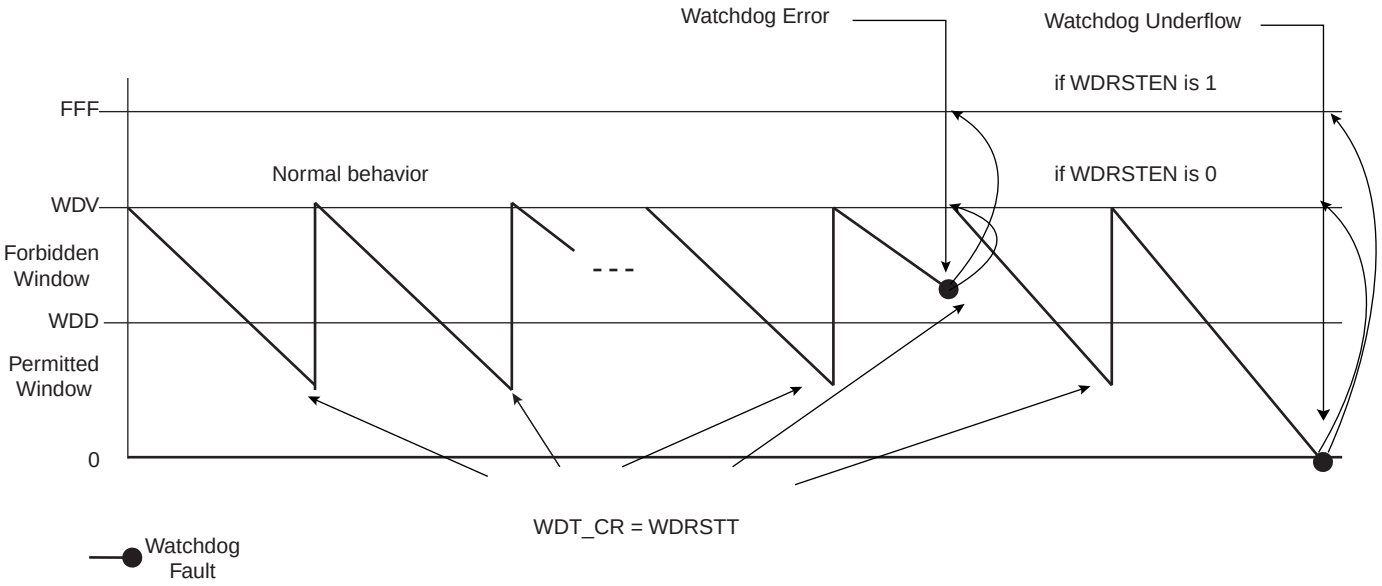
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet. 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 18-2. Watchdog Behavior



18.5 Watchdog Timer (WDT) User Interface

Table 18-1. Register Mapping

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

18.5.1 Watchdog Timer Control Register

Name: WDT_CR

Address: 0xFFFFFE40

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog if KEY is written to 0xA5.

- **KEY: Password.**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

18.5.2 Watchdog Timer Mode Register

Name: WDT_MR

Address: 0xFFFFFE44

Access: Read-write Once

31	30	29	28	27	26	25	24
		WDIDLEHLT	WDDBGHLT	WDD			

23	22	21	20	19	18	17	16
WDD							

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

7	6	5	4	3	2	1	0
WDV							

Note: The first write access prevents any further modification of the value of this register, read accesses remain possible.

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.

- **WDV: Watchdog Counter Value**

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

- **WDFIEN: Watchdog Fault Interrupt Enable**

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

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

- **WDRSTEN: Watchdog Reset Enable**

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

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

- **WDRPROC: Watchdog Reset Processor**

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

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

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer.

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

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

- **WDDBGHLT: Watchdog Debug Halt**

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

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

- **WDIDLEHLT: Watchdog Idle Halt**

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

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

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

18.5.3 Watchdog Timer Status Register

Name: WDT_SR

Address: 0xFFFFFE48

Access Read-only

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

- **WDUNF: Watchdog Underflow**

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

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

- **WDERR: Watchdog Error**

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

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

19. Reset Controller (RSTC)

19.1 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.

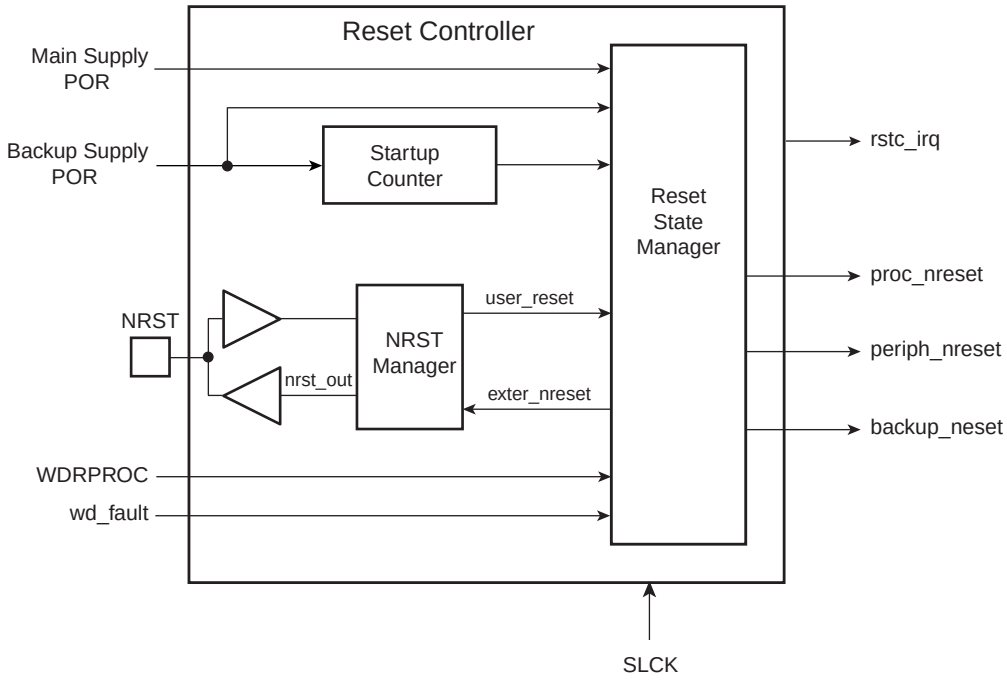
The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

19.2 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

19.3 Block Diagram

Figure 19-1. Reset Controller Block Diagram



19.4 Functional Description

19.4.1 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 datasheet.

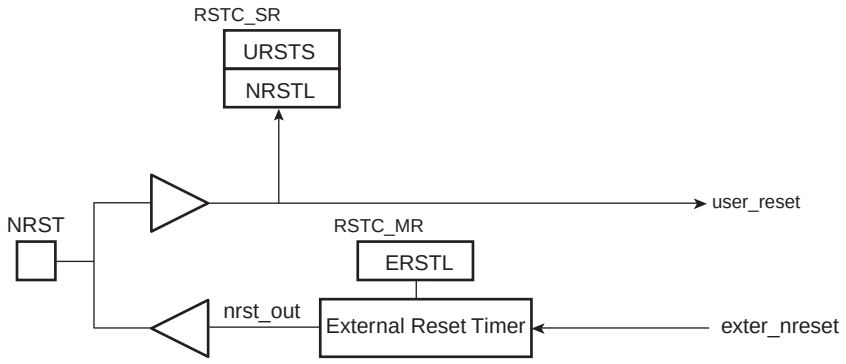
The Reset Controller Mode Register (RSTC_MR), used to configure the reset controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU is on.

19.4.2 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 19-2shows the block diagram of the NRST Manager.

Figure 19-2. NRST Manager



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 the Reset Controller Status Register (RSTC_SR). As soon as the pin NRST is asserted, the bit URSTS in the RSTC_SR is set. This bit clears only when RSTC_SR is read.

19.4.2.1 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 the 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.

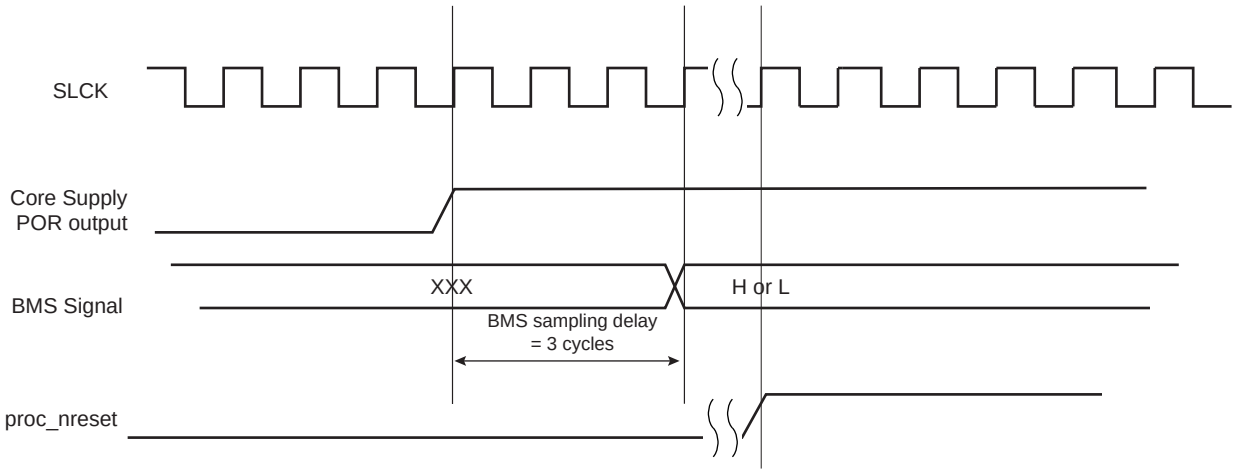
This feature allows the reset controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

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.

19.4.3 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 19-3. BMS Sampling



19.4.4 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 RSTC_SR. The update of the field RSTTYP is performed when the processor reset is released.

19.4.4.1 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 the RSTC_SR reports a General Reset. As the RSTC_MR is reset, the NRST line rises two cycles after the backup_nreset, as ERSTL defaults at value 0x0.

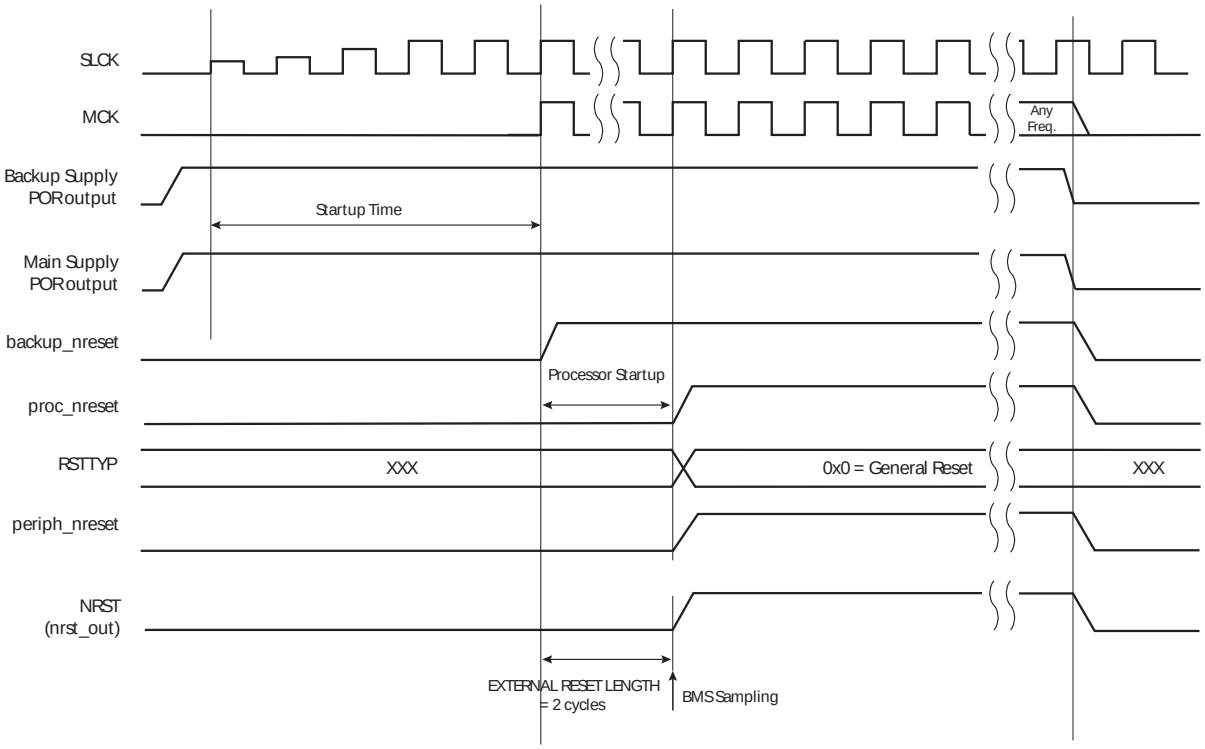
When VDDBU is detected low by the backup supply POR cell, all resets signals are immediately asserted, even if the main supply POR cell does not report a main supply shutdown.

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 19-4 shows how the General Reset affects the reset signals.

Figure 19-4. General Reset State



19.4.4.2 Wake-up Reset

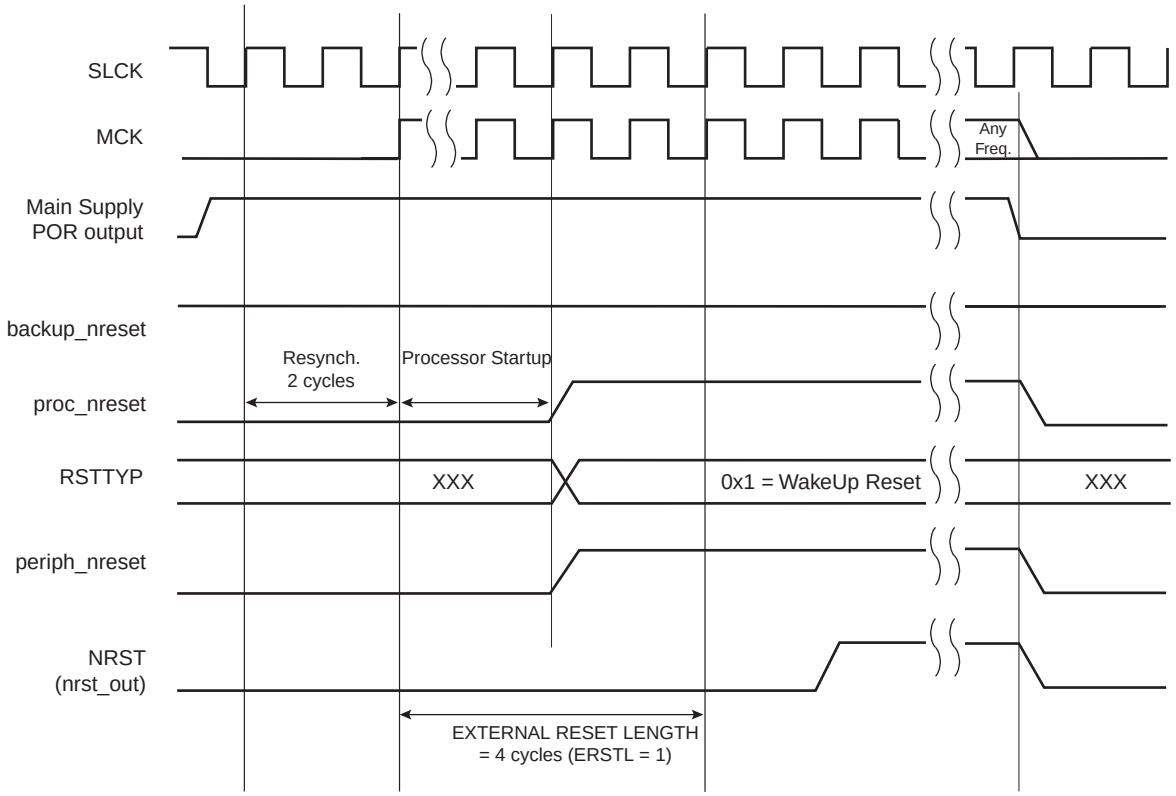
The wake-up reset occurs when the main supply is down. When the main supply POR output is active, all the reset signals are asserted except backup_nreset. When the main supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 3 Slow Clock cycles, 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 the 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 19-5. Wake-up Reset



19.4.4.3 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.

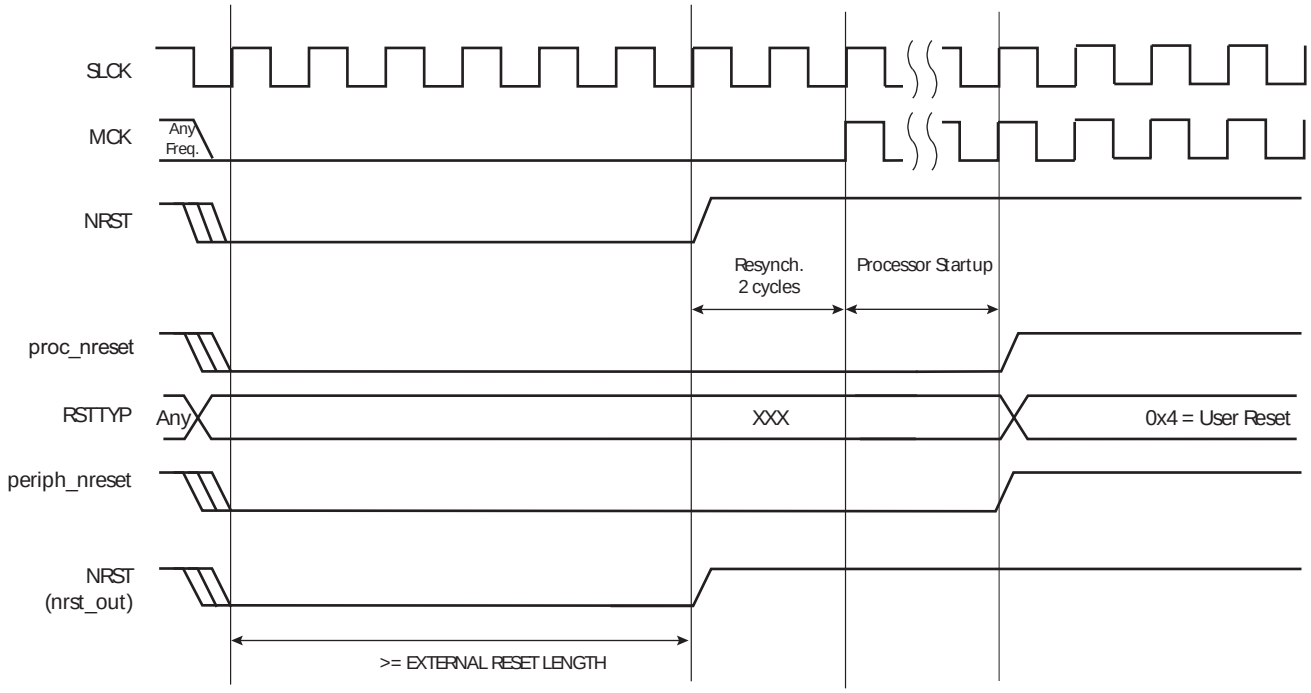
The Processor Reset and the Peripheral Reset are asserted.

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 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 19-6. User Reset State



19.4.4.4 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.
  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.

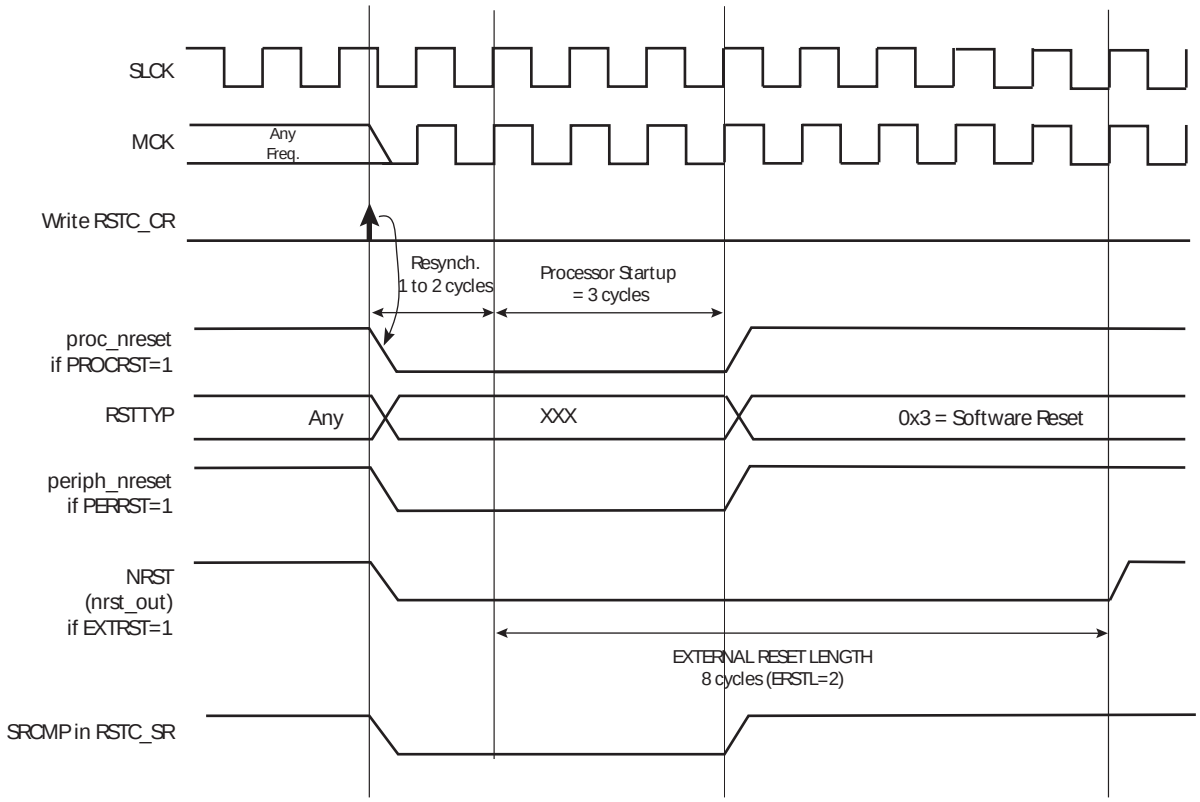
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e., synchronously to SLCK.

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 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 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 19-7. Software Reset



19.4.4.5 Watchdog Reset

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

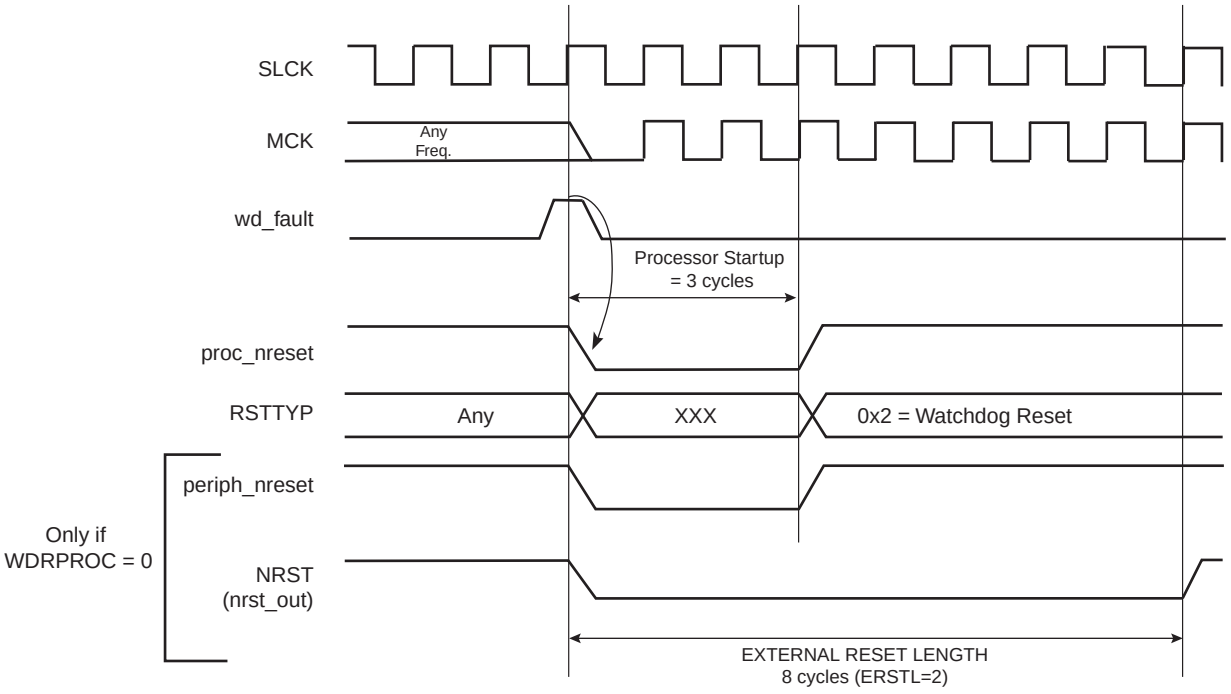
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- 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 19-8. Watchdog Reset



19.4.5 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.

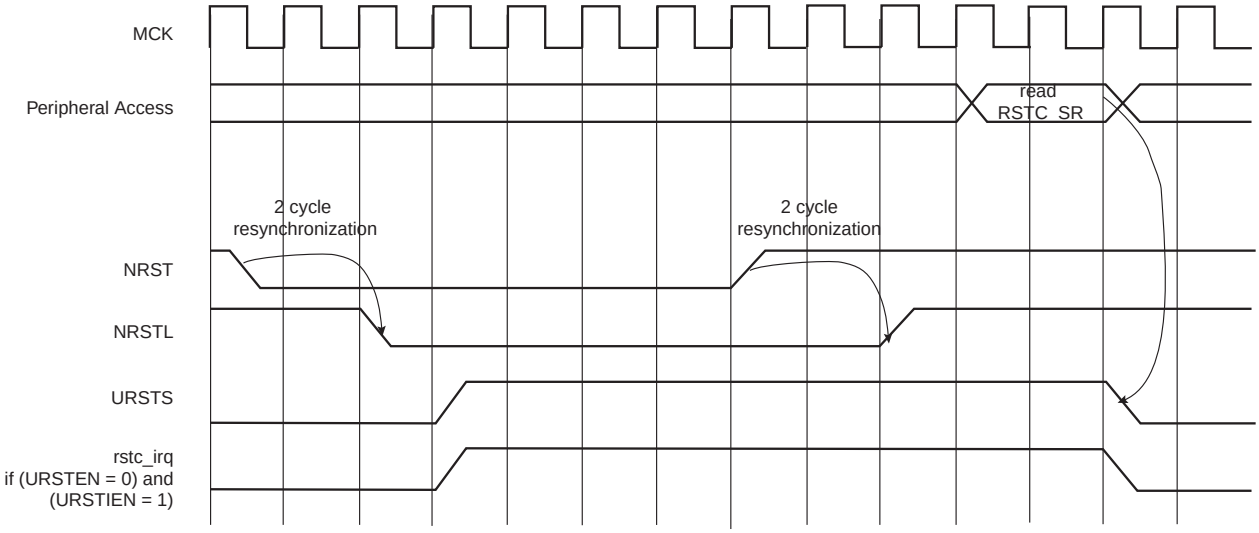
19.4.6 Reset Controller Status Register

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 bit 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: This bit 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. This transition is also detected on the Master Clock (MCK) rising edge (see Figure 19-9). Reading the RSTC_SR resets the URSTS bit.

Figure 19-9. Reset Controller Status and Interrupt



19.5 Reset Controller (RSTC) User Interface

Table 19-1. Register Mapping

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

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

19.5.1 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: Write Access Password**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

19.5.2 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.

Value	Name	Description
0	GENERAL_RST	Both VDDCORE and VDDBU rising
1	WKUP_RST	VDDCORE rising
2	WDT_RST	Watchdog fault occurred
3	SOFT_RST	Processor reset required by the software
4	USER_RST	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.

19.5.3 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 the assertion duration to be programmed between 60 µs and 2 seconds.

- **KEY: Write Access Password**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

20. Shutdown Controller (SHDWC)

20.1 Description

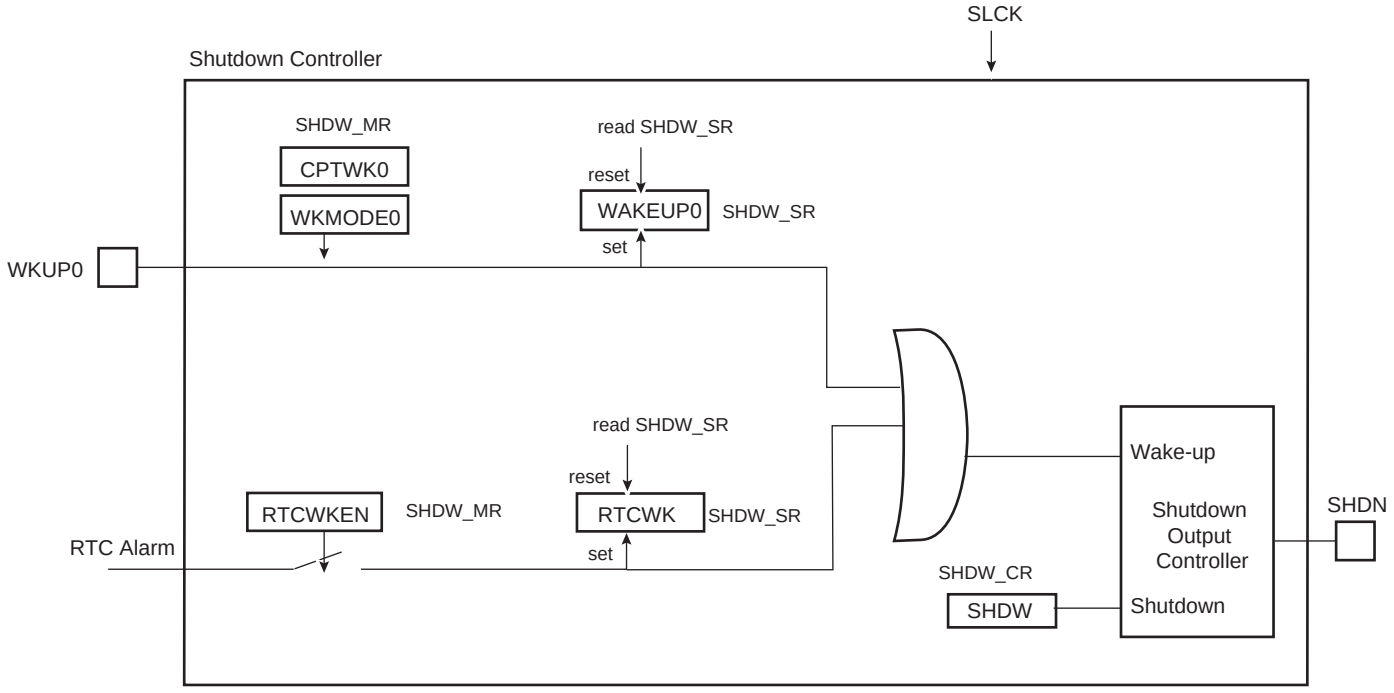
The Shutdown Controller controls the power supplies VDDIO and VDDCORE and the wake-up detection on debounced input lines.

20.2 Embedded Characteristics

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

20.3 Block Diagram

Figure 20-1. Shutdown Controller Block Diagram



20.4 I/O Lines Description

Table 20-1. I/O Lines Description

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

20.5 Product Dependencies

20.5.1 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.

20.6 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 RTC alarm is reported in the RTCWK bit of the SHDW_SR Status register. They are reset after the read of SHDW_SR. When using the RTC alarm to wake up the system, the user must ensure that RTC alarm status flag is cleared before shutting down the system. Otherwise, no rising edge of the status flags may be detected and the wake-up will fail.

20.7 Shutdown Controller (SHDWC) User Interface

Table 20-2. Register Mapping

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

20.7.1 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.**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

20.7.2 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**

Value	Name	Description
0	NO_DETECTION	No detection is performed on the wake-up input
1	RISING_EDGE	Low to high transition triggers the detection process
2	FALLING_EDGE	High to low level transition triggers the detection process
3	ANY_EDGE	Any edge on the wake-up input triggers the detection process

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

Defines the minimum duration of the WKUP1 pin after the occurrence of the selected triggering edge (WKMODE0).

The SHDN pin is released if the WKUP0 holds the selected level for (CPTWK * 16 + 1) consecutive Slow Clock cycles after the occurrence of the selected triggering edge on WKUP0.

- **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.

20.7.3 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 WKUP0 input since the last read of SHDW_SR.

1 = At least one wake-up event occurred on WKUP0 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.

21. General-Purpose Backup Registers (GPBR)

21.1 Description

The System Controller embeds 4 General-purpose Backup registers.

21.2 Embedded Characteristics

- 4 32-bit General Purpose Backup Registers

21.3 General Purpose Backup Registers (GPBR) User Interface

Table 21-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0	General Purpose Backup Register 0	SYS_GPBR0	Read-write	–
...	...	...	...	...
0x6C	General Purpose Backup Register 3	SYS_GPBR3	Read-write	–

21.3.1 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**

22. Periodic Interval Timer (PIT)

22.1 Description

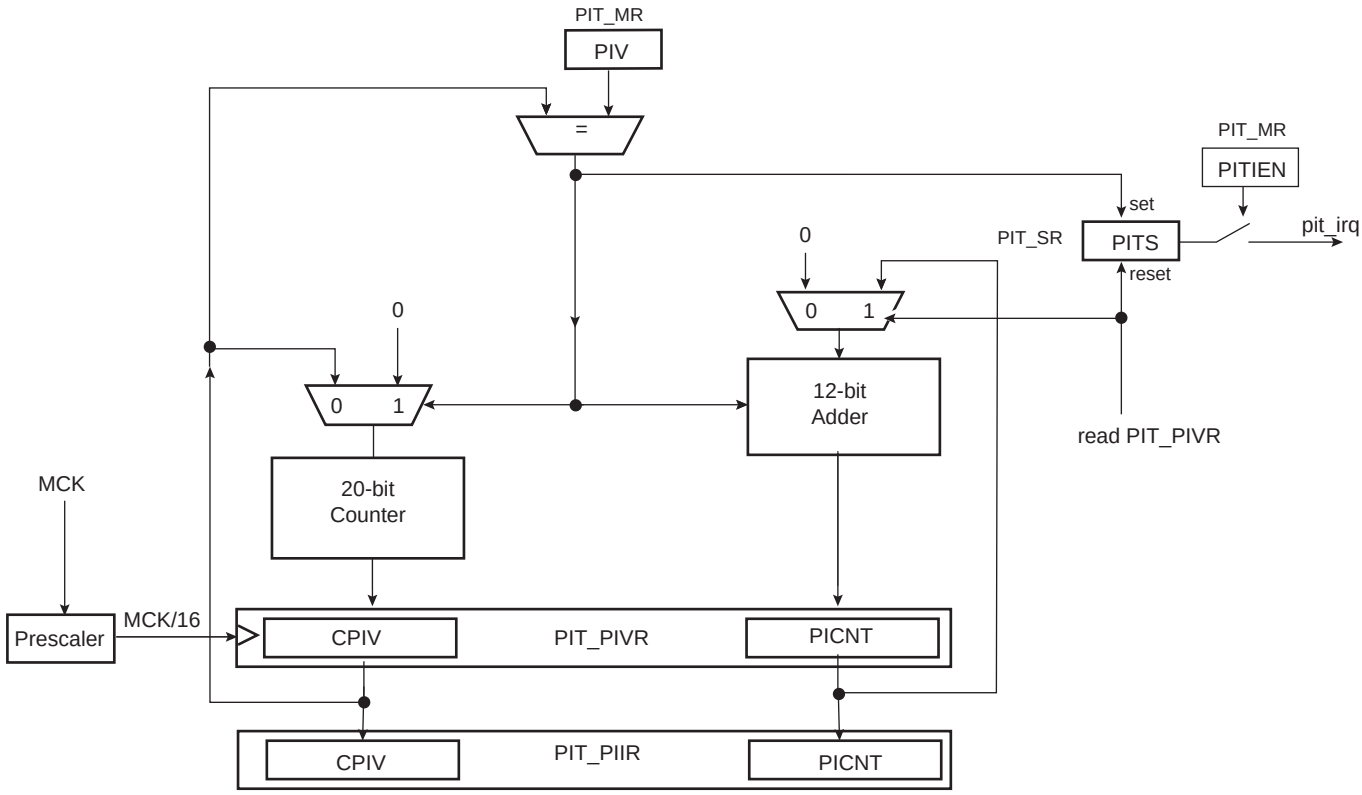
The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

22.2 Embedded Characteristics

- 20-bit Programmable Counter plus 12-bit Interval Counter
- Reset-on-read Feature
- Both Counters Work on Master Clock/16

22.3 Block Diagram

Figure 22-1. Periodic Interval Timer



22.4 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.

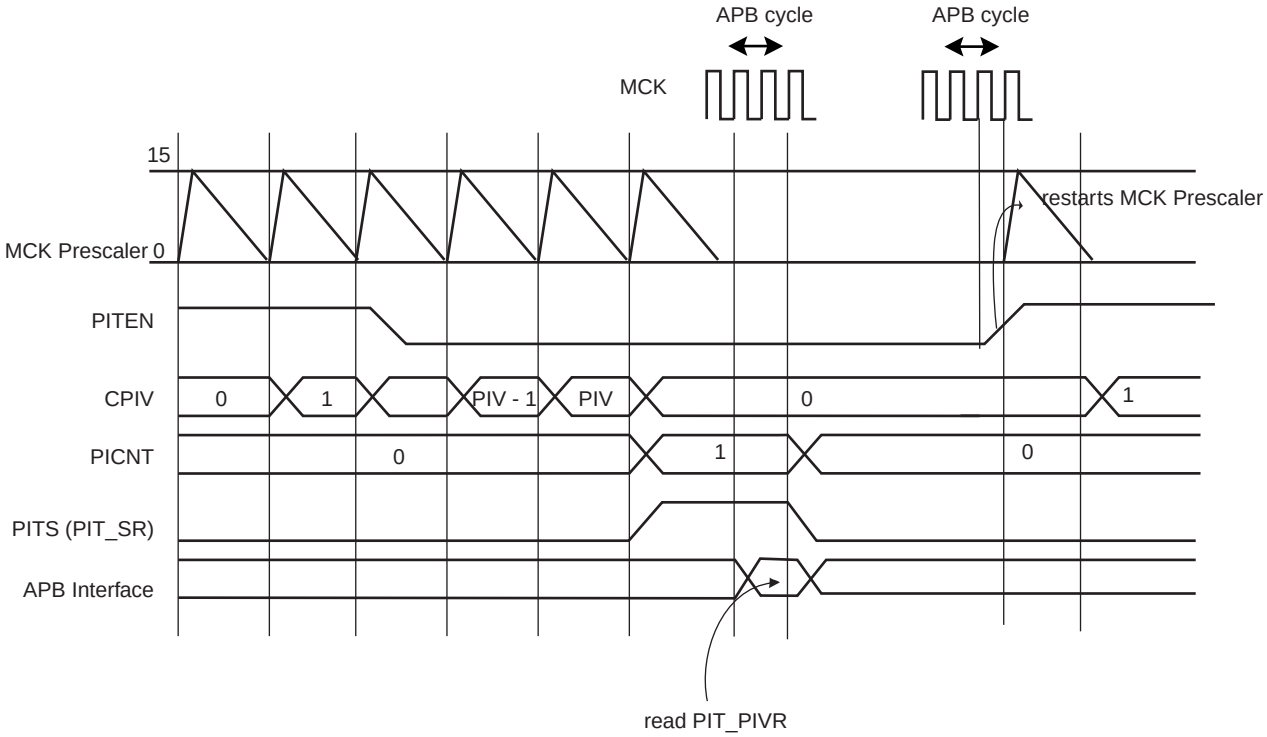
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

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 22-2 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 22-2. Enabling/Disabling PIT with PITEN



22.5 Periodic Interval Timer (PIT) User Interface

Table 22-1. 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

22.5.1 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.

22.5.2 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.

22.5.3 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.

22.5.4 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.

23. Real-time Clock (RTC)

23.1 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.

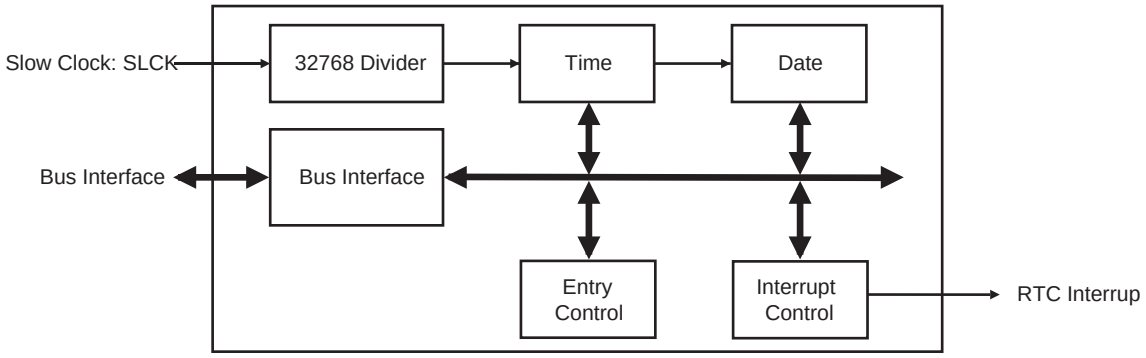
Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

23.2 Embedded Characteristics

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

23.3 Block Diagram

Figure 23-1. RTC Block Diagram



23.4 Product Dependencies

23.4.1 Power Management

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

23.4.2 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.

Table 23-1. Peripheral IDs

Instance	ID
RTC	1

23.5 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.

23.5.1 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.

23.5.2 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.

23.5.3 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.

Hour, minute and second matching alarm (SECEN, MINEN, HOUREN) can be enabled independently of SEC, MIN, HOUR fields.

Note: To change one of the SEC, MIN, HOUR, DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to 3 accesses to the RTC_TIMALR or RTC_CALALR registers. The first access clears the enable corresponding to the field to change (SECEN,MINEN,HOUREn,DATEEN,MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (SEC, MIN, HOUR, DATE, MONTH). The third access is required to re-enable the field by writing 1 in SECEN,MINEN,HOUREn,DATEEN,MTHEN fields.

23.5.4 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.

23.5.5 Updating Time/Calendar

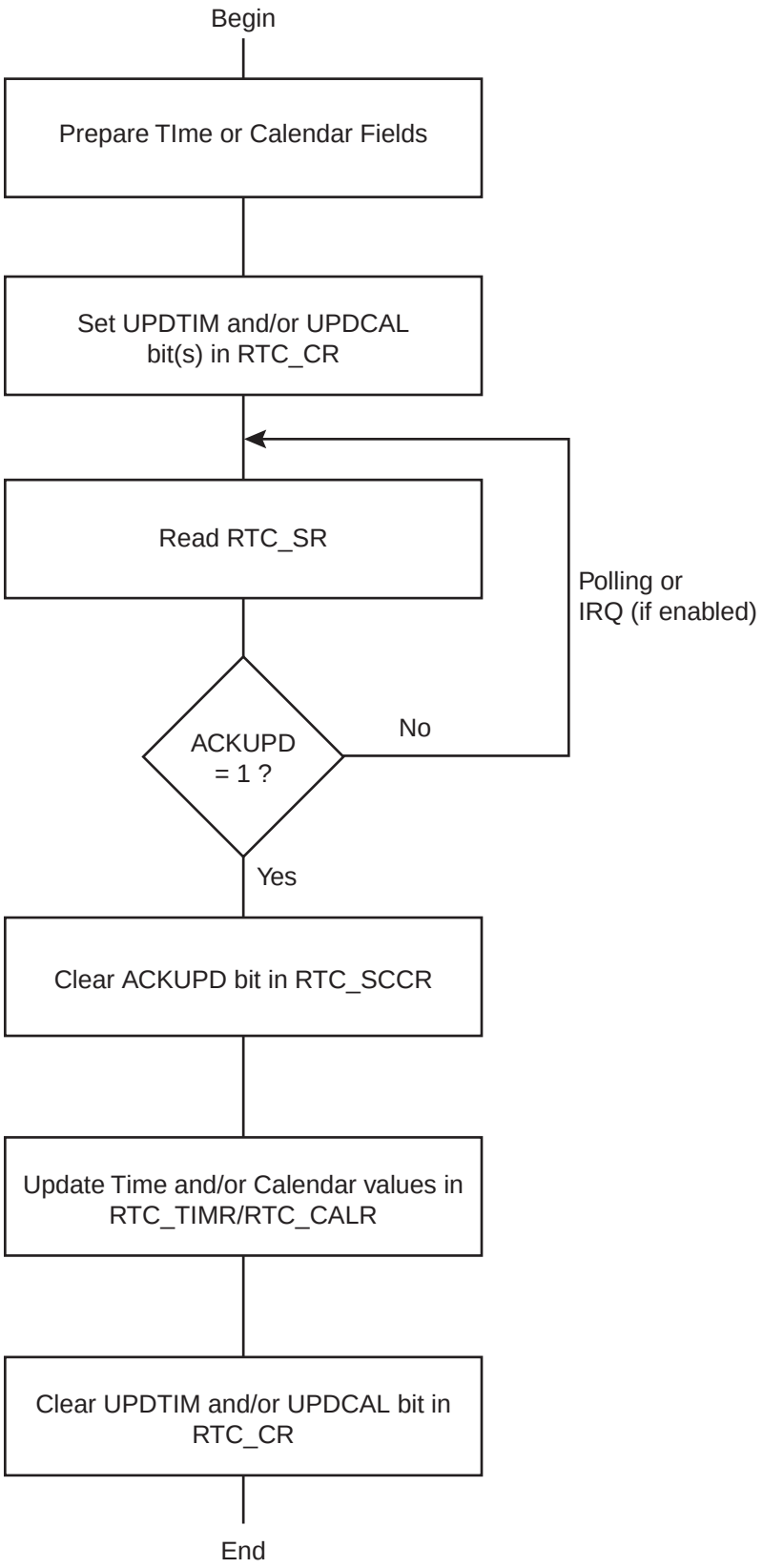
To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register. Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

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 23-2. Update Sequence



23.6 Real-time Clock (RTC) User Interface

Table 23-2. 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	0x01810720
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.

23.6.1 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)

23.6.2 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.

23.6.3 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.

23.6.4 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 (gregorian) (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.

23.6.5 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						

Note: To change one of the SEC, MIN, HOUR fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to 3 accesses to the RTC_TIMALR register. The first access clears the enable corresponding to the field to change (SECEN,MINEN,HOUREN). If the field is already cleared, this access is not required. The second access performs the change of the value (SEC, MIN, HOUR). The third access is required to re-enable the field by writing 1 in SECEN, MINEN, HOUREN fields.

- **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.

23.6.6 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
–	–	–	–	–	–	–	–

Note: To change one of the DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to 3 accesses to the RTC_CALALR register. The first access clears the enable corresponding to the field to change (DATEEN,MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (DATE,MONTH). The third access is required to re-enable the field by writing 1 in DATEEN, MTHEN fields.

- **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.

23.6.7 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.

23.6.8 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).

23.6.9 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.

23.6.10 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.

23.6.11 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.

23.6.12 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.

24. Slow Clock Controller (SCKC)

24.1 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.

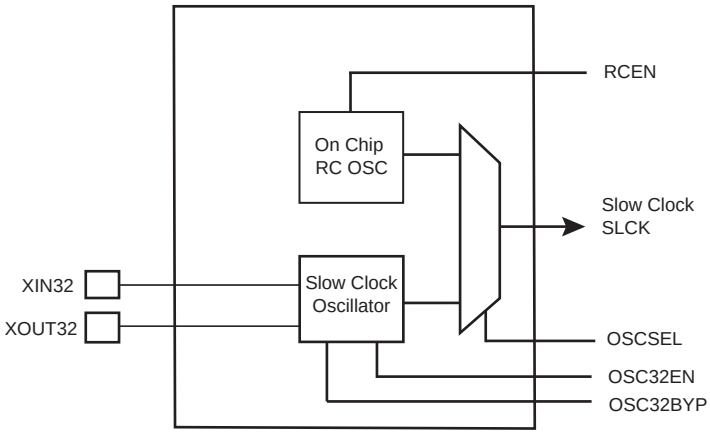
The internal 32 kHz RC oscillator and the 32768 Hz oscillator can be enabled by setting to 1, respectively, the RCEN and OSC32EN bits in the System Controller user interface. The OSCSEL command selects the slow clock source.

24.2 Embedded Characteristics

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

24.3 Block Diagram

Figure 24-1. Block Diagram



RCEN, OSC32EN, OSCSEL and OSC32BYP bits are located in the Slow Clock Controller 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 the 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.

24.3.1 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 writing a 1 to the OSC32EN bit.
- Wait 32768 Hz startup time for clock stabilization (software loop).
- Switch from internal 32 kHz RC oscillator to 32768 Hz oscillator by writing a 1 to the OSCSEL bit.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32 kHz RC oscillator by writing a 0 to the RCEN bit.

24.3.2 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 by writing a 1 to the OSC32BYP bit.
- Disable the 32768 Hz oscillator by writing a 0 to the OSC32EN bit.

24.3.3 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 writing a 1 to the RCEN bit.
- Wait internal 32 kHz RC startup time for clock stabilization (software loop).
- Switch from 32768 Hz oscillator to internal RC by writing a 0 to the OSCSEL bit.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32768 Hz oscillator by writing a 0 to the OSC32EN bit.

24.4 Slow Clock Controller (SCKC) User Interface

Table 24-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Slow Clock Controller Configuration Register	SCKC_CR	Read/Write	0x0000_0001

24.4.1 Slow Clock Controller Configuration Register

Name: SCKC_CR

Address: 0xFFFFFE50

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
–	–	–	–	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: 32768Hz 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.

25. Fuse Controller (FUSE)

25.1 Description

The Fuse Controller (FUSE) supports software fuse programming through a 32-bit register. Only fuses set to level ‘1’ are programmed.

The fast (main) RC oscillator must be enabled at startup to access the fuse matrix.

It reads the fuse states on startup and stores them into 32-bit registers. The first 8 Fuse Status registers (FUSE_SRx) can be masked and will read as a value of ‘0’ regardless of the fuse state when masked.

25.2 Embedded Characteristics

- Software Fuse Programming
- User Write Access for Fuse
- Part of Fuse can be Masked After Read
- 256 FUSE bits:
  - 192 bits are dedicated to Users
  - 3 bits are dedicated to Special Functions
  - The Fuse Controller can be hidden thanks to a SFR write-once bit.
    Please refer to the “Special Function Registers (SFR)” section of the SAMA5D3 series datasheet for details.

25.2.1 FUSE Bit Mapping

To avoid any malfunctioning, the user must not write the “DO NOT USE (DNU)” FUSE bits.

Table 25-1. FUSE Bit Mapping

255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230	229	228	227	226	225	224
DO NOT USE (DNU)																															
223	222	221	220	219	218	217	216	215	214	213	212	211	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192
USER_DATA																															
191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
DO NOT USE (DNU)																													B	J	W
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
USER_DATA																															
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
USER_DATA																															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
USER_DATA																															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
USER_DATA																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_DATA																															

25.2.2 Special Functions

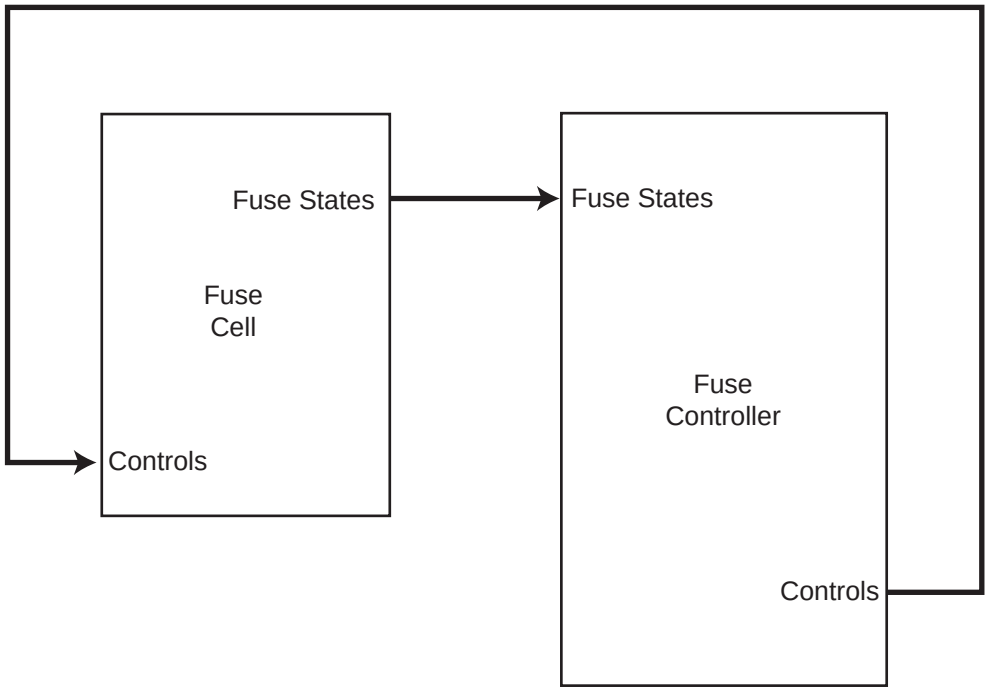
The user is allowed to set special bits as described in the following table.

Table 25-2. Special Bits

Bit Number	Bit Name	Function
162	B	BMS_SAMPLING_DISABLED - BMS sampling is disabled if set
161	J	JTAG_DISABLED - JTAG is disabled if set
160	W	FUSE_WRITE_DISABLED - FUSE bit writing is disabled if set

25.3 Block Diagram

Figure 25-1. Fuse Controller Block Diagram



25.4 Functional Description

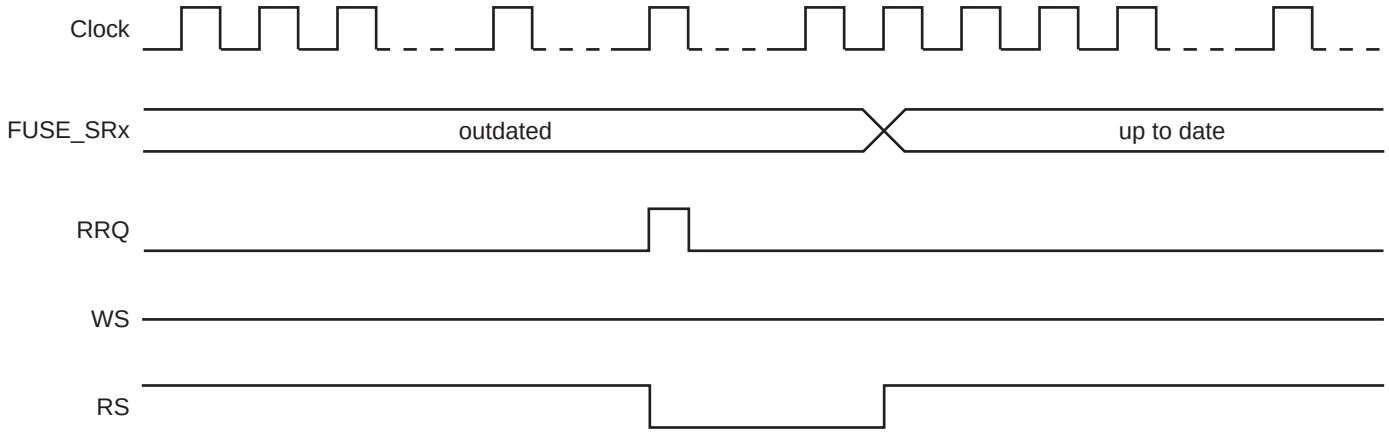
25.4.1 Fuse Reading

The fuse states are automatically read on CORE startup and are available for reading in the 8 Fuse Status (FUSE_SRx) registers.

The fuse states of bits 31 to 0 will be available at FUSE_SR0, the fuse states of bits 63 to 32 will be available at FUSE_SR1 and so on.

FUSE_SRx registers can be updated manually by using the RRQ bit of the Fuse Control register (FUSE_CR). RS and WS bits of the Fuse Index register (FUSE_IR) must be at level one before issuing the read request.

Figure 25-2. Fuse Read



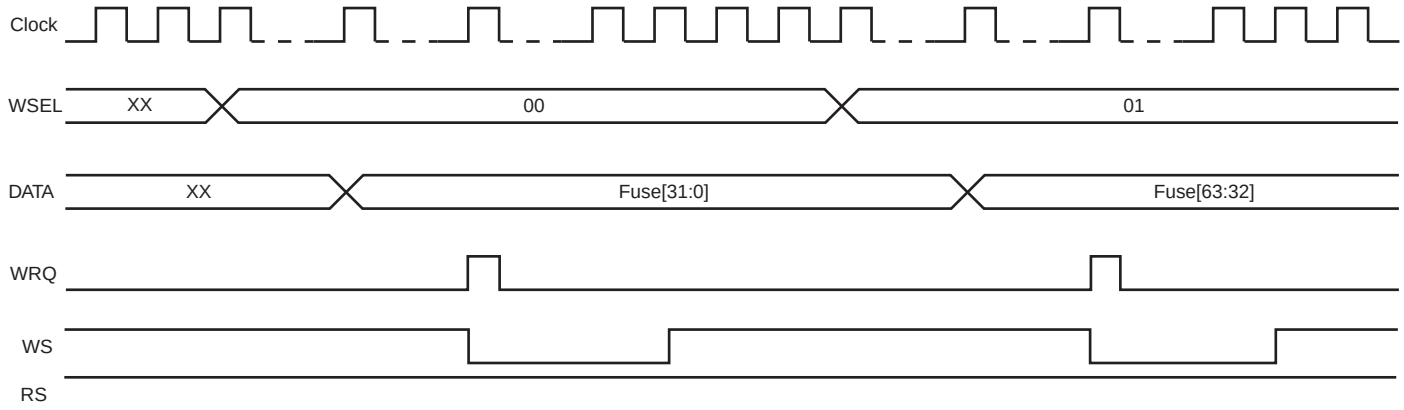
25.4.2 Fuse Programming

All the fuses, except Atmel reserved fuses, can be written by software. To program fuses, strictly follow the order of the sequence instructions as provided below:

1. Select the word to write, using the WSEL field of the Fuse Index register (FUSE_IR).
2. Write the word to program in the Fuse Data register (FUSE_DR).
3. Check that RS and WS bits of the FUSE_IR are at level one (no read and no write pending).
4. Write the WRQ bit of the Fuse Control register (FUSE_CR) to begin the fuse programming. The KEY field must be written at the same time with a value 0xFB to make the write request valid. Writing the WRQ bit will clear the WS bit.
5. Check the WS bit of FUSE_IR. When WS has a value of ‘1’ the fuse write process is over.

Only fuses to be set to level ‘1’ are written.

Figure 25-3. Fuse Write



25.4.3 Fuse Masking

It is possible to mask the first 8 FUSE_SRx registers so that they will be read at a value of ‘0’, regardless of the fuse state.

To activate fuse masking on the first 8 FUSE_SRx registers, the MSK bit of the Fuse Mode register (FUSE_MR) must be written to level ‘1’. The MSK bit is write-only. Only a general reset can disable fuse masking.

25.5 Fuse Controller (FUSE) User Interface

Table 25-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Fuse Control Register	FUSE_CR	Write-only	–
0x04	Fuse Mode Register	FUSE_MR	Write-only	–
0x08	Fuse Index Register	FUSE_IR	Read-write	0x00000000
0x0C	Fuse Data Register	FUSE_DR	Read-write	–
0x10	Fuse Status Register 0	FUSE_SR0	Read-only	0x00000000
0x14	Fuse Status Register 1	FUSE_SR1	Read-only	0x00000000
...	...	...	...	...
0x2C	Fuse Status Register 7	FUSE_SR7	Read-only	0x00000000
0x30–0xDC	Reserved	–	–	–
0xE0–0xFC	Reserved	–	–	–

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

25.5.1 Fuse Control Register

Name: FUSE_CR

Address: 0xFFFFE400

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
KEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RRQ	WRQ

- **WRQ: Write Request**

0: No effect.

1: Requests the word DATA to be programmed if KEY field value is 0xFB.

- **RRQ: Read Request**

0: No effect.

1: Requests the fuses to be read and FUSE_SRx registers to be updated if KEY field value is 0xFB.

- **KEY: Key code**

Value	Name	Description
0xFB	VALID	Writing any other value in this field aborts the write operation of the WRQ and RRQ bits. Always reads as 0.

25.5.2 Fuse Mode Register

Name: FUSE_MR

Address: 0xFFFFE404

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
–	–	–	–	–	–	–	MSK

- **MSK: Mask Fuse Status Registers**

0: No effect.

1: Masks the first 8 FUSE_SRx registers.

25.5.3 Fuse Index Register

Name: FUSE_IR

Address: 0xFFFFE408

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
–	–	–	–	WSEL			
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RS	WS

- **WS: Write Status**

0: Write is pending or no write has been requested since general reset.

1: Write of fuses is done.

- **RS: Read Status**

0: Read is pending or no read has been requested since general reset.

1: Read of fuses is done.

- **WSEL: Word Selection**

0–15: Selects the word to write.

25.5.4 Fuse Data Register

Name: FUSE_DR

Address: 0xFFFFE40C

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 Program**

Data to program. Only bits with a value of ‘1’ will be programmed.

25.5.5 Fuse Status Register

Name: FUSE_SRx [x=0..7]

Address: 0xFFFFE410

Access: Read-only

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

- **FUSE: Fuse Status**

Indicates the status of corresponding fuses:

0: Unprogrammed.

1: Programmed.

26. Power Management Controller (PMC)

26.1 Clock Generator

26.1.1 Description

The Clock Generator User Interface is embedded within the Power Management Controller and is described in Section 26.2.15 “Power Management Controller (PMC) User Interface”. However, the Clock Generator registers are named CKGR_.

26.1.2 Embedded Characteristics

The Clock Generator is made up of:

- Low-power 32768 Hz Slow Clock Oscillator with bypass mode
- Low-power 32 kHz RC Oscillator
- 8 to 48 MHz Crystal Oscillator or a 24/48 MHz XRCGB Crystal Resonator, which can be bypassed (12 MHz, 24 MHz (preferred) or 48 MHz must be used in case of USB operations)
- Fast RC Oscillator, at 12 MHz
- 480 MHz UTMI PLL providing a clock for the USB High Speed Device Controller
- 400 to 1000 MHz programmable PLL (input from 8 to 50 MHz), capable of providing the clock MCK to the processor and to the peripherals

It provides the following clocks:

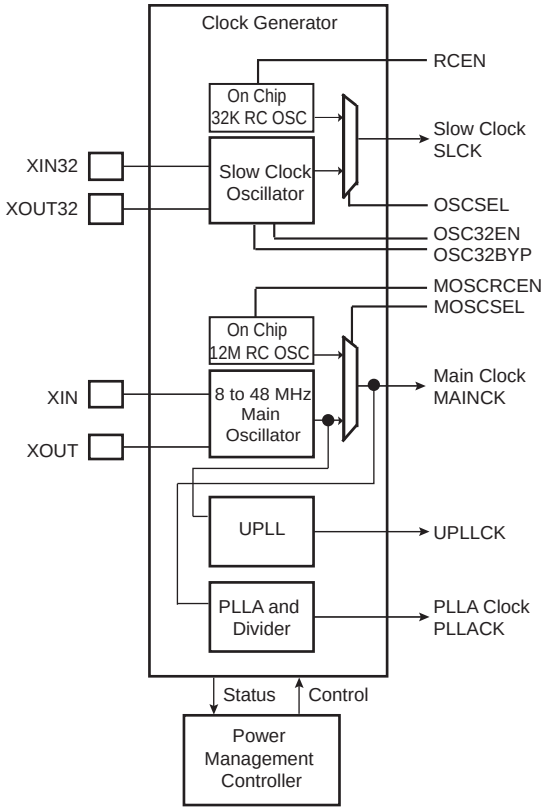
- SLCK, the Slow Clock, which is the only permanent clock within the system
- MAINCK is the output of the Main Clock Oscillator selection: either 8 to 48 MHz Crystal Oscillator or 12 MHz Fast RC Oscillator
- PLLACK is the output of the Divider and 400 to 1000 programmable PLL (PLLA)
- UPLLCK is the output of the 480 MHz UTMI PLL (UPLL)
- SMDCK is the Software Modem Clock

The Power Management Controller also provides the following operations on clocks:

- Main crystal oscillator clock failure detector
- Frequency counter on main clock and an on-the-fly adjustable main RC oscillator frequency

26.1.3 Block Diagram

Figure 26-1. Clock Generator Block Diagram

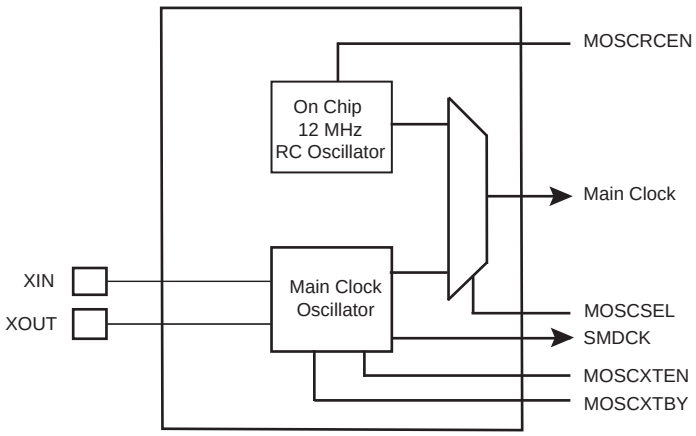


26.1.4 Main Clock Selection

The main clock can be generated either by an external 8 to 48 MHz crystal, an XRCGB crystal resonator or by the on-chip12 MHz Fast RC Oscillator. This allows the processor to start or restart in a few microseconds when 12 MHz Fast RC Oscillator is selected.

The 8 to 48 MHz crystal oscillator can be bypassed by setting the MOSCXTBY bit to accept an external main clock on XIN.

Figure 26-2. Main Clock Selection



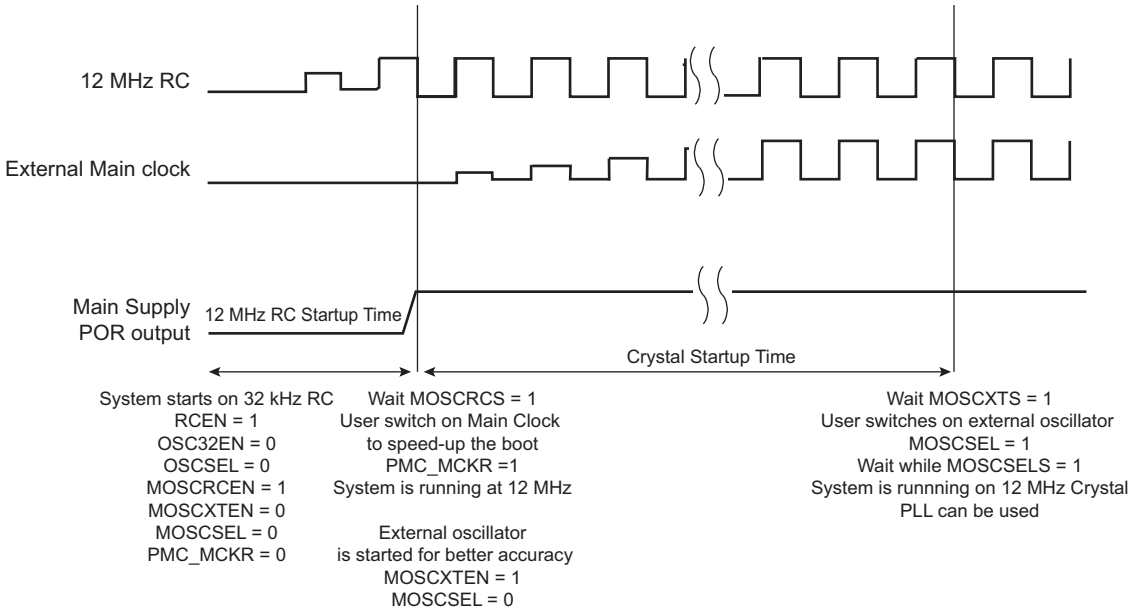
MOSCRCEN, MOSCXTEN, MOSCSEL and MOSCXTBY bits are located in the PMC Clock Generator Main Oscillator Register (CKGR_MOR).

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

26.1.4.1 Fast Wake-up

To speed up the wake-up phase, the user can switch the system clock from 32 kHz RC (SLCK) to 12 MHz RC (Main Clock). This enables the user to perform system configuration (PLL, DDR, etc.) at 12 MHz instead of 32 kHz during the12 MHz Oscillator startup.

Figure 26-3. PMC Startup



26.1.4.2 Switch from Internal 12 MHz Fast RC Oscillator to the 8 to 48 MHz Crystal

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

To switch from internal 12 MHz Fast RC Oscillator to the 8 to 48 MHz crystal, the programmer must execute the following sequence:

- Enable the 8 to 48 MHz oscillator by writing a 1 to bit MOSCXTEN.
- Wait for 8 to 48 MHz oscillator status MAINRDY is 1.
- Switch from internal 12 MHz RC to the 8 to 48 MHz oscillator by writing a 1 to bit MOSCSEL.
- If not, the PMC writes a 0 to bit MOSCSEL.
- Disable the 12 MHz RC oscillator by writing a 0 to bit MOSCRCEN.

26.1.4.3 Bypass the 8 to 48 MHz Crystal Oscillator

The following step must be added to bypass the 8 to 48 MHz crystal oscillator.

- An external clock must be connected on XIN.
- Enable the bypass path MOSCXTBY bit set to 1.
- Disable the 8 to 48 MHz oscillator by writing a 0 to bit MOSCXTEN.

26.1.4.4 Switch from the 8 to 48 MHz Crystal Oscillator to internal 12 MHz Fast RC Oscillator

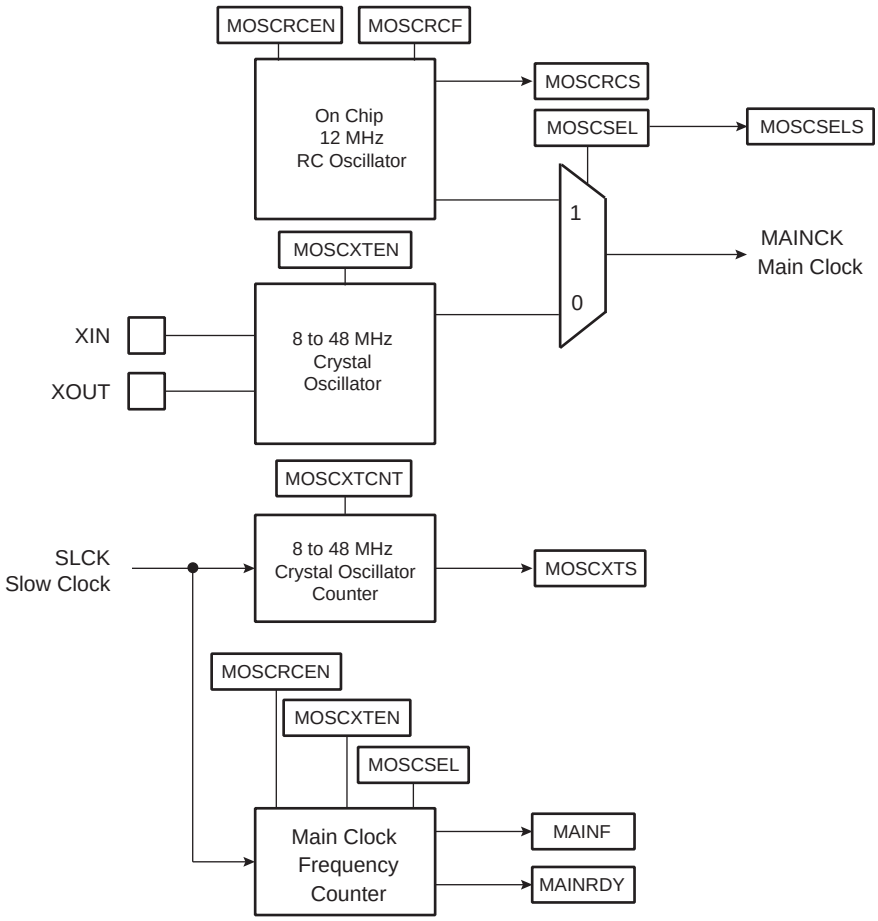
The same procedure must be followed to switch from a 8 to 48 MHz crystal oscillator to the internal 12 MHz Fast RC Oscillator.

- Enable the internal 12 MHz RC oscillator for low power by writing a 1 to bit MOSCRCEN.

- Wait internal 12 MHz RC startup time for clock stabilization (software loop).
- Switch from 8 to 48 MHz oscillator to the internal 12 MHz Fast RC Oscillator by writing a 0 to bit MOSCSEL.
- Disable the 8 to 48 MHz oscillator by writing a 0 to bit MOSCXTEN.

26.1.5 Main Clock

Figure 26-4. Main Clock Block Diagram



The Main Clock has two sources:

- 12 MHz Fast RC Oscillator which starts very quickly and is used at startup
- 8 to 48 MHz Crystal Oscillator, which can be bypassed

26.1.5.1 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 electrical characteristics.

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.

26.1.5.2 12 MHz Fast RC Oscillator Clock Frequency Adjustment

It is possible for the user to adjust the main RC oscillator frequency through PMC_OCR. By default, SEL is low, so the RC oscillator is driven with fuse calibration bits which are programmed during the chip production.

The user can adjust the trimming of the 12 MHz Fast RC oscillator through the PMC_OCR in order to obtain more accurate frequency (to compensate derating factors such as temperature and voltage).

In order to calibrate the 12 MHz oscillator frequency, SEL must be set to 1 and a correct frequency value must be configured in CAL.

It is possible to restart, at anytime, a measurement of the main frequency by means of the RCMEAS bit in Main Clock Frequency Register (CKGR_MCFR). Thus, when MAINFRDY flag reads 1, another read access on CKGR_MCFR provides an image of the frequency of the main clock on MAINF field. The software can calculate the error with an expected frequency and correct the CAL field accordingly. This may be used to compensate frequency drift due to derating factors such as temperature and/or voltage.

26.1.5.3 8 to 48 MHz Crystal Oscillator

After reset, the 8 to 48 MHz Crystal Oscillator is disabled and it is not selected as the source of MAINCK.

The user can select the 8 to 48 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 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.

26.1.5.4 Main Clock Oscillator Selection

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

The selection is made by writing the MOSCSEL bit in the CKGR_MOR. The switch of the Main Clock source is glitch free, so there is no need to run out of SLCK or PLLACK in order to change the selection. The MOSCSELS bit of the PMC_SR indicates when the switch sequence is done.

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

26.1.5.5 Switching Main Clock between the Main RC Oscillator and Fast Crystal Oscillator

Both sources must be enabled during the switch operation. Only after completion can the unused oscillator be disabled. If switching to Fast Crystal Oscillator, the clock presence must first be checked according to what is described in Section 26.1.5.6 “Software Sequence to Detect the Presence of Fast Crystal” because the source may not be reliable (crystal failure or bypass on a non-existent clock).

26.1.5.6 Software Sequence to Detect the Presence of Fast Crystal

The frequency meter carried on the CKGR_MCFR is operating on the selected main clock and not on the fast crystal clock nor on the fast RC Oscillator clock.

Therefore, to check for the presence of the fast crystal clock, it is necessary to switch the main clock on the fast crystal clock.

The following software sequence must be followed (during this sequence the Main RC oscillator must be kept enabled (MOSCRCEN = 1)):

1. MCK must select the slow clock (CSS = 0 in PMC_MCKR).
2. Wait for the MCKRDY flag in the PMC_SR to be 1.
3. The fast crystal must be enabled by programming 1 in the MOSCXTEN field in the CKGR_MOR, with the MOSCXTST field being programmed to the appropriate value (see the electrical characteristics section).
4. Wait for the MOSCXTS flag to be 1 in the PMC_SR to allow the start-up period of the fast crystal oscillator to end.
5. MOSCSEL must be programmed to 1 in the CKGR_MOR to select the fast main crystal oscillator for the main clock.
6. MOSCSEL must be read until its value equals 1.
7. The MOSCSELS status flag must be checked in the PMC_SR.

At this point, two cases may occur: either MOSCSELS = 0 or MOSCSELS = 1.

- MOSCSELS = 1: There is a valid crystal connected and its frequency can be determined by initiating a frequency measure by programming RCMEAS in the CKGR_MCFR.
- MOSCSELS = 0:
  - There is no fast crystal clock (either no crystal connected or a crystal clock out of specification). A frequency measure can reinforce this status by initiating a frequency measure by programming RCMEAS in the CKGR_MCFR.
  - The selection of the main clock must be programmed back to the main RC oscillator by writing MOSCSEL to 0 prior to disabling the fast crystal oscillator.
  - The crystal oscillator can be disabled (MOSCXTEN = 0 in CKGR_MOR).

26.1.5.7 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 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

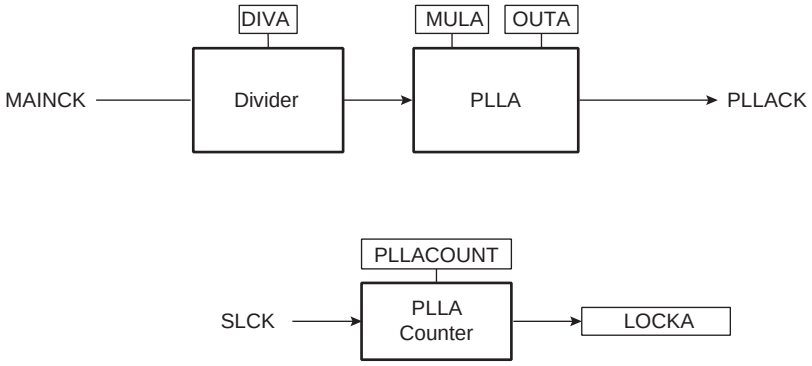
Then, at the 16th falling edge of Slow Clock, the MAINFRDY bit in the CKGR_MCFR is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the 12 MHz Fast RC Oscillator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

26.1.6 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 26-5 shows the block diagram of the divider and PLLA block.

Figure 26-5. Divider and PLLA Block Diagram



26.1.6.1 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.

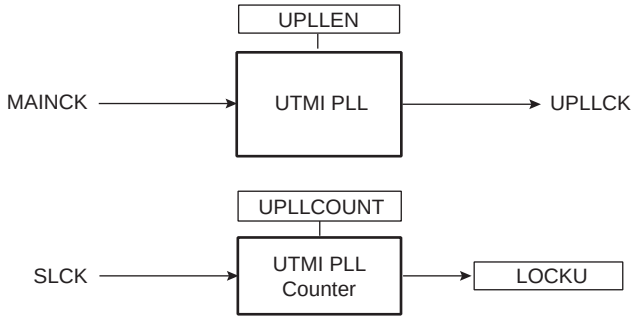
The PLLA clock must be divided by 2 by writing the PLLADIV2 bit in the PMC_MCKR.

26.1.7 UTMI Phase Lock Loop Programming

The source clock of the UTMI PLL is the Main OSC output. When the 12 MHz Fast RC Oscillator is selected as the source of the 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, 16, 24 or 48 MHz crystal is needed to use the USB. Please refer to the SFR_UTMICKTRIM register to set the value.

Figure 26-6. 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.

26.2 Power Management Controller (PMC)

26.2.1 Description

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

26.2.2 Embedded Characteristics

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently.
- Processor Clock (PCK), must be switched off when entering the processor in Sleep Mode.
- The USB Device HS Clock (UDPCK)
- The Software Modem Clock (SMDCK)
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, HSMCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

26.2.3 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.

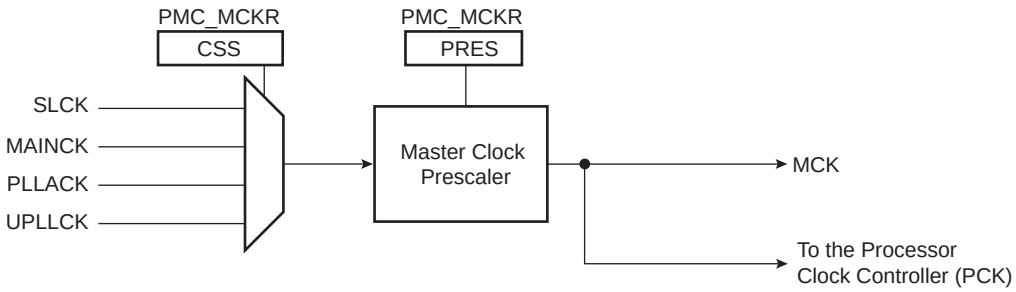
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

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 6. The PRES field in PMC_MCKR programs the prescaler.

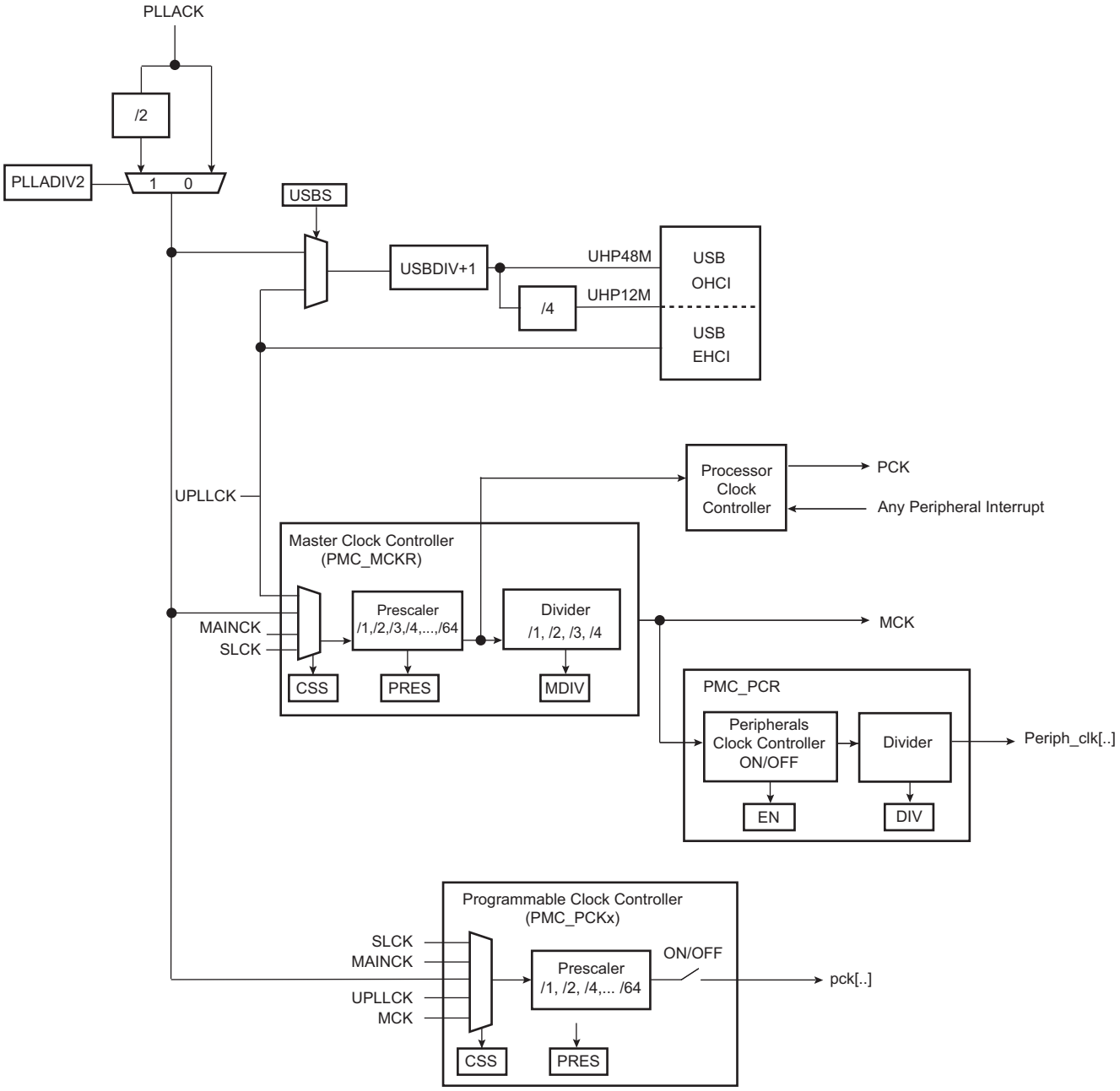
Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to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

Figure 26-7. Master Clock Controller



26.2.4 Block Diagram

Figure 26-8. General Clock Block Diagram



26.2.5 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, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

26.2.6 USB Device and Host Clocks

The USB Device and Host High Speed ports (UDPHS and UHPHS) clocks are enabled by corresponding PIDx bits in PMC_PCER. To save power on this peripheral when they are not used, the user can set these bits in the PMC_PCDR. Corresponding PIDx bits in the PMC_PCSR give the status 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 they are not used, the user can set the UHP bit in PMC_SCDR. The UHP bit in PMC_SCSR gives the status of this clock. The USB host OHCI requires both the 12/48 MHz signal and the Master Clock. The USBDIV field in PMC_USB register is to be programmed to 9 (division by 10) for normal operations.

To further reduce 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 remains possible.

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

The USB clock input is to be defined according to Main Oscillator via the FREQ field. It is defined in the UTMI Clock Trimming Register located in the SFR section (see the “Special Function Registers (SFR)” section of this datasheet for details). This input clock can be 12, 16, 24, or 48 MHz.

26.2.7 DDR2/LPDDR/LPDDR2 Clock

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

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

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

To save PLLA power consumption, the user can choose UPLLCK as an Input clock for the system. In this case the DDR Controller can drive LPDDR or LPDDR2 at up to 120 MHz.

26.2.8 Software Modem Clock

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

SMDCK is a division of UPLL or PLLA.

26.2.9 Peripheral Clock Controller

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

In order to save power consumption, the division factor can be 1, 2, 4 or 8. The PMC_PCR is a register that features a command and acts like a mailbox. To write the division factor on a particular peripheral, 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, 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,0x01030102)
```

Code Example to read the divider of peripheral 4:

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

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.

26.2.10 Programmable Clock Output Controller

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

PCKx can be independently selected between the Slow clock (SLCK), the Master Clock (MAINCK), the PLLACK, 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.

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.

26.2.11 Main Crystal Clock Failure Detector

The clock failure detector monitors the 8 to 48 MHz Crystal or Ceramic Resonator-based oscillator to identify an eventual defect of this oscillator (for example, if the crystal is unconnected).

The clock failure detector can be enabled or disabled by means of the CFDEN bit in the PMC Clock Generator Main Oscillator Register (CKGR_MOR). After reset, the detector is disabled. However, if the 8 to 48 MHz Crystal or Ceramic Resonator-based Oscillator is disabled, the clock failure detector is disabled too.

A failure is detected by means of a counter incrementing on the 8 to 48 MHz Crystal oscillator or Ceramic Resonator-based oscillator clock edge and timing logic clocked on the slow clock RC oscillator controlling the counter. The counter is cleared when the slow clock RC oscillator signal is low and enabled when the slow clock RC oscillator is high. Thus the failure detection time is one slow clock RC oscillator clock period. If, during the high level period of the slow clock RC oscillator, less than eight fast crystal oscillator clock periods have been counted, then a failure is declared.

The slow RC oscillator must be enabled.

The clock failure detection must be enabled only when system clock MCK selects the fast RC oscillator. Then the status register must be read two slow clock cycles after enabling.

The clock failure detection must be disabled when the main crystal is disabled.

If a failure of the 8 to 48 MHz Crystal or Ceramic Resonator-based oscillator clock is detected, the CFDEV flag is set in the PMC Status Register (PMC_SR), and generates an interrupt if it is not masked. The interrupt remains active until a read operation in the PMC_SR. The user can know the status of the clock failure detector at any time by reading the CFDS bit in the PMC_SR.

If the 8 to 48 MHz Crystal or Ceramic Resonator-based oscillator clock is selected as the source clock of MAINCK (MOSCSEL = 1), and if the Master Clock Source is PLLACK or UPLLCK (CSS = 2 or 3), a clock failure detection automatically forces MAINCK to be the source clock for the master clock (MCK).Then, regardless of the PMC configuration, a clock failure detection automatically forces the 4/8/12 MHz Fast RC oscillator to be the source clock for MAINCK. If the Fast RC oscillator is disabled when a clock failure detection occurs, it is automatically re-enabled by the clock failure detection mechanism.

It takes two slow clock RC oscillator cycles to detect and switch from the 8 to 48 MHz Crystal, or Ceramic Resonator-based oscillator, to the 4/8/12 MHz Fast RC Oscillator if the Master Clock source is Main Clock, or three slow clock RC oscillator cycles if the Master Clock source is PLLACK or UPLLCK.

A clock failure detection activates a fault output that is connected to the Pulse Width Modulator (PWM) Controller. With this connection, the PWM controller is able to force its outputs and to protect the driven device, if a clock failure is detected. This fault output remains active until the defect is detected and until it is cleared by the bit FOCLR in the PMC Fault Output Clear Register (PMC_FOCR).

The user can know the status of the fault output at any time by reading the FOS bit in the PMC_SR.

26.2.12 Programming Sequence

1. If the fast crystal oscillator is not required, PLL can be directly configured (begin with Step 6. or Step 7.) else the fast crystal oscillator must be started (begin with Step 2.).
2. Enabling the fast crystal oscillator:

   The fast crystal oscillator is enabled by setting the MOSCXTEN field in the Main Oscillator Register (CKGR_MOR). The user can define a start-up time. This can be achieved by writing a value in the MOSCXTST field in CKGR_MOR. Once this register has been correctly configured, the user must wait for MOSCXTS field in the PMC_SR to be set. This can be done either by polling MOSCXTS in the PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (MOSCXTS) has been enabled in the PMC_IER.
3. Switch the MAINCK to the Main Crystal Oscillator by setting MOSCSEL in CKGR_MOR.
4. Wait for the MOSCSELS to be set in PMC_SR to ensure the switchover is complete.
5. Checking the Main Clock Frequency:

   The Main Clock Frequency can be measured via the Main Clock Frequency Register (CKGR_MCFR).

   Read the CKGR_MCFR until the MAINFRDY field is set, after which the user can read the MAINF field in CKGR_MCFR by performing an additional read. This provides the number of main clock cycles that have been counted during a period of 16 slow clock cycles.

   If MAINF = 0, switch the MAINCK to the Fast RC Oscillator by clearing MOSCSEL in CKGR_MOR. If MAINF ≠ 0, proceed to Step 6.
6. Setting PLLA and divider (if not required, proceed to Step 7.):

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

   The DIVA field is used to control 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 and PLLA are turned off.

   The MULA field is the PLLA multiplier factor. This parameter can be programmed between 0 and 127. If MULA is set to 0, PLLA is 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 after the CKGR_PLLAR has been written.

   Once the CKGR_PLLAR has been written, the user must wait for the LOCKA bit to be set in the PMC_SR. This can be done either by polling LOCKA in the PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (LOCKA) has been enabled in the PMC_IER. All parameters in CKGR_PLLAR can be programmed in a single write operation. If at some stage parameter MULA or DIVA is modified, LOCKA bit goes low to indicate that PLLA is not yet ready. When PLLA is locked, LOCKA is set again.

   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. Once CKGR_PLLAR is written, the user has to write ‘3’ in the IPLL_PLLA field of the PLL Charge Pump Current Register (PMC_PLLICPR). The user must perform this step before using the PLLA output clock.

   The user must wait for the LOCKA bit to be set before using the PLLA output clock.

7. Setting Bias and High-speed PLL (UPLL) for UTMI

   The UTMI PLL is enabled by setting the UPLLEN field in the CKGR_UCKR. The UTMI Bias must is enabled by setting the BIASEN field in the CKGR_UCKR 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.

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

8. Selecting Master Clock and Processor Clock

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

   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 the main clock.

   The PRES field is used to define the Processor Clock and Master Clock prescaler. The user can choose between different values (1, 2, 3, 4, 8, 16, 32, 64). Prescaler output is the selected clock source frequency divided by the PRES value.

   The MDIV field is used to define the Master Clock divider. It is possible to choose between different values (0, 1, 2, 3). The Master Clock output is Processor Clock frequency divided by 1, 2, 3 or 4, depending on the value programmed in MDIV.

   The PMC PLLA Clock input must be divided by 2 by writing the PLLADIV2 bit.

   By default, MDIV and PLLLADIV2 are set to 0, which indicates that Processor Clock is equal to the Master Clock.

   Once the PMC_MCKR has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR. This can be done either by polling MCKRDY in the PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (MCKRDY) has been enabled in the PMC_IER.

   The PMC_MCKR must not be programmed in a single write operation. The programming sequence for PMC_MCKR is as follows:

- If a new value for CSS field corresponds to PLL Clock,
  - Program the PRES field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.
  - Program the CSS field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the PRES field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.

If at some stage parameter CSS or PRES is modified, the MCKRDY bit goes low to indicate that the Master Clock and the Processor Clock are not yet ready. The user must wait for the MCKRDY bit to be set again before using the Master and Processor Clocks.

Note: If PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR, the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCKA goes high and MCKRDY is set.
While PLL is unlocked, the Master Clock selection is automatically changed to Slow Clock. For further information, see see Section 26.2.13.2 “Clock Switching Waveforms”.

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.

9. Selecting 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. 3 programmable clocks can be used. The PMC_SCSR indicates which programmable clock is enabled. By default all programmable clocks are disabled.

   PMC_PCKx registers are used to configure programmable clocks.

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

   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 value is set to 0 which means that PCKx is equal to slow clock.

   Once the PMC_PCKx register has been configured, 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. This can be done either by polling PCKRDYx in the PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (PCKRDYx) has been enabled in the PMC_IER. 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.

10. Enabling Peripheral Clocks

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

26.2.13 Clock Switching Details

26.2.13.1 Master Clock Switching Timings

Table 26-1 and Table 26-2 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 26-1. Clock Switching Timings (Worst Case)

To	From		
	Main Clock	**SLCK**	**PLL Clock**
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Notes: 1. PLL designates either the PLLA or the UPLL Clock.

2. PLLCOUNT designates either PLLACOUNT or UPLLCOUNT.

Table 26-2. Clock Switching Timings between Two PLLs (Worst Case)

To	From	
	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

26.2.13.2 Clock Switching Waveforms

Figure 26-9. Switch Master Clock from Slow Clock to PLL Clock

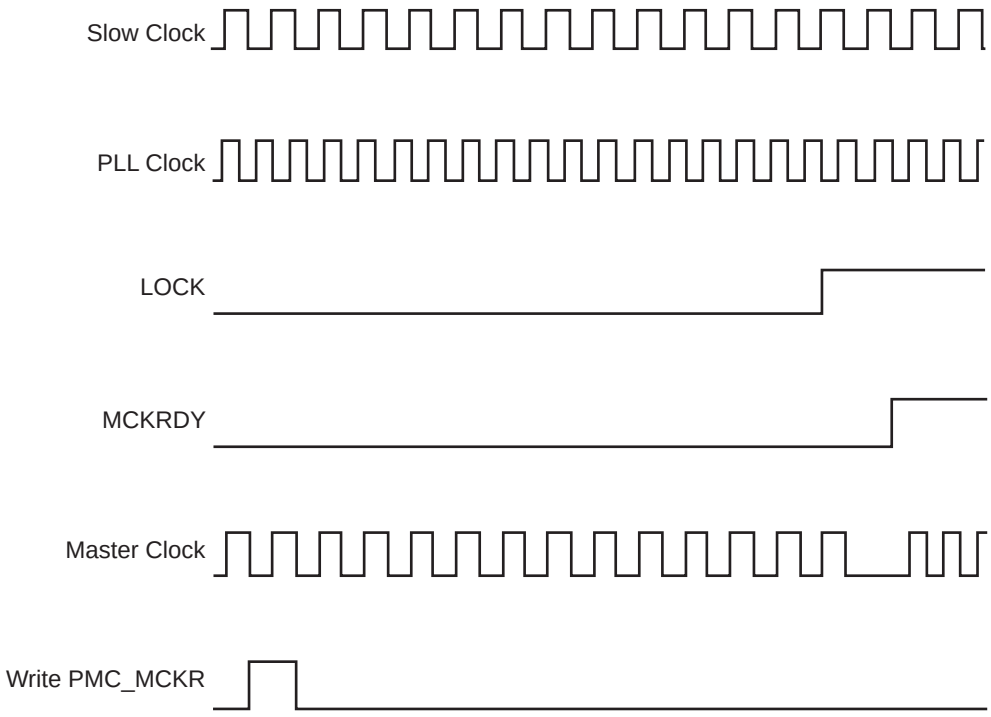


Figure 26-10.Switch Master Clock from Main Clock to Slow Clock

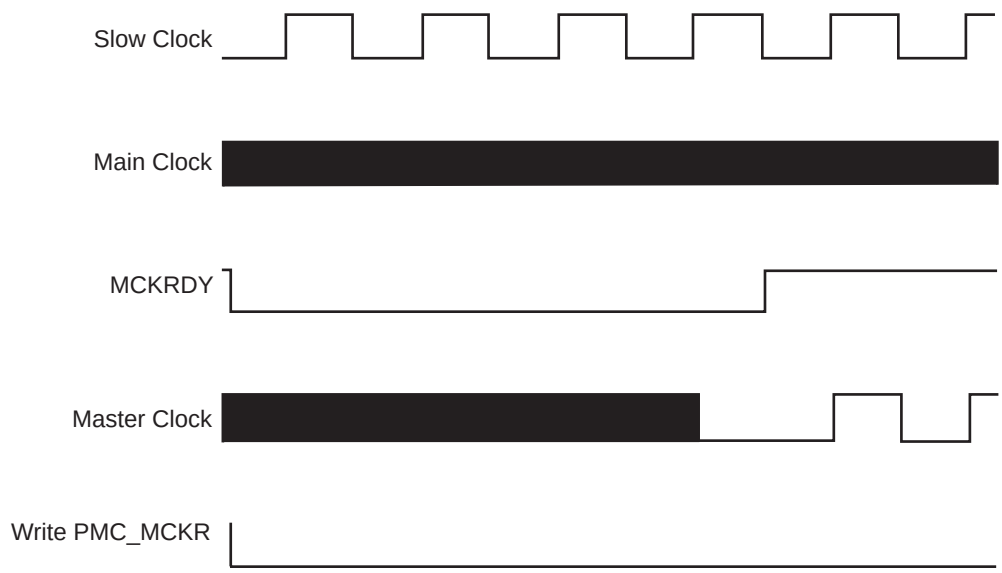


Figure 26-11.Change PLLA Programming

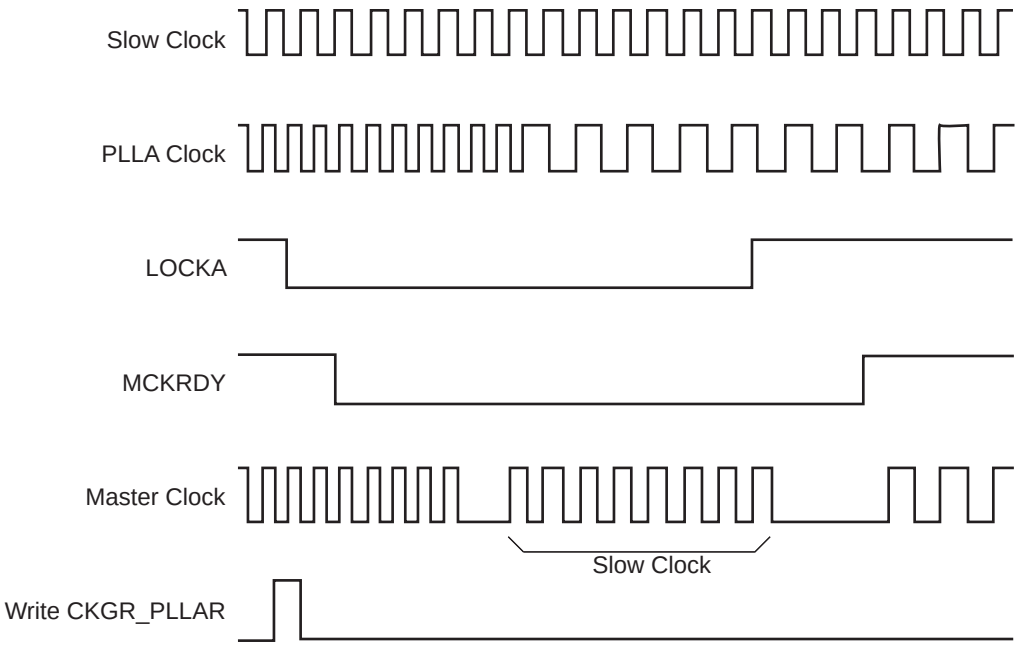
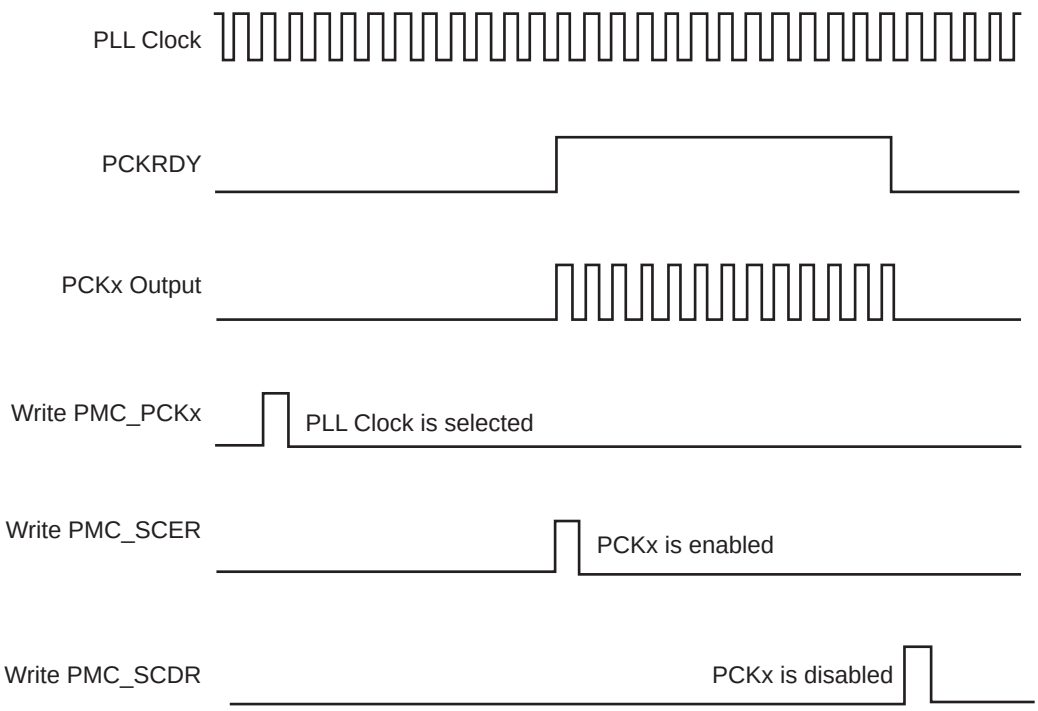


Figure 26-12.Programmable Clock Output Programming



26.2.14 Register Write Protection

To prevent any single software error from corrupting PMC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the PMC Write Protection Mode Register (PMC_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the PMC Write Protection Status Register (PMC_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 PMC_WPSR.

The following registers can be protected:

- PMC System Clock Enable Register
- PMC System Clock Disable Register
- PMC Clock Generator Main Clock Frequency Register
- PMC Clock Generator PLLA Register
- PMC Master Clock Register
- PMC USB Clock Register
- PMC Programmable Clock Register
- PLL Charge Pump Current Register
- PMC Peripheral Clock Enable Register 0
- PMC Peripheral Clock Disable Register 1
- PMC Oscillator Calibration Register

26.2.15 Power Management Controller (PMC) User Interface

Table 26-3. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC_SCSR	Read-only	0x0000_0005
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register 0	PMC_PCER0	Write-only	–
0x0014	Peripheral Clock Disable Register 0	PMC_PCDR0	Write-only	–
0x0018	Peripheral Clock Status Register 0	PMC_PCSR0	Read-only	0x0000_0000
0x001C	UTMI Clock Register	CKGR_UCKR	Read/Write	0x1020_0000
0x0020	Main Oscillator Register	CKGR_MOR	Read/Write	0x0100_0001
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read/Write	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	Programmable Clock 2 Register	PMC_PCK2	Read/Write	0x0000_0000
0x004C–0x005C	Reserved	–	–	–
0x0060	Interrupt Enable Register	PMC_IER	Write-only	–
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	–
0x0068	Status Register	PMC_SR	Read-only	0x0001_0008
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0000_0000
0x0070–0x0074	Reserved	–	–	–
0x0078	Fault Output Clear Register	PMC_FOCR	Write-only	–
0x007C	Reserved	–	–	–
0x0080	PLL Charge Pump Current Register	PMC_PLLICPR	Read/Write	0x0000_0000
0x0084–0x00E0	Reserved	–	–	–
0x00E4	Write ProtectIon Mode Register	PMC_WPMR	Read/Write	0x0000_0000
0x00E8	Write Protection Status Register	PMC_WPSR	Read-only	0x0000_0000
0x00EC–0x00FC	Reserved	–	–	–
0x0100	Peripheral Clock Enable Register 1	PMC_PCER1	Write-only	–
0x0104	Peripheral Clock Disable Register 1	PMC_PCDR1	Write-only	–

Table 26-3. Register Mapping

Offset	Register	Name	Access	Reset
0x0108	Peripheral Clock Status Register 1	PMC_PCSR1	Read-only	0x0000_0000
0x010C	Peripheral Control Register	PMC_PCR	Read/Write	0x0000_0000
0x0110	Oscillator Calibration Register	PMC_OCR	Read/Write	0x0040_4040

26.2.15.1 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
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	SMDCK	LCDCK	DDRCK	–	–

- **DDRCK: DDR Clock Enable**

0: No effect.

1: Enables the DDR clock.

- **LCDCK: LCD2x Clock Enable**

0: No effect.

1: Enables the LCD2x 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.

26.2.15.2 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
–	–	–	–	–	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
UDP	UHP	–	SMDCK	LCDCK	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.

- **LCDCK: LCD2x Clock Disable**

0: No effect.

1: Disables the LCD2x 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.

26.2.15.3 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
–	–	–	–	–	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
UDP	UHP	–	SMDCK	LCDCK	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.

- **LCDCK: LCD2x Clock Status**

0: The LCD2x clock is disabled.

1: The LCD2x 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.

26.2.15.4 PMC Peripheral Clock Enable Register 0

Name: PMC_PCER0

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	–	–

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Enable**

0: No effect.

1: Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet. Other peripherals can be enabled in PMC_PCER1.

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

26.2.15.5 PMC Peripheral Clock Disable Register 0

Name: PMC_PCDR0

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	-	-

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **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. Other peripherals can be disabled in PMC_PCDR1.

26.2.15.6 PMC Peripheral Clock Status Register 0

Name: PMC_PCSR0

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. Other peripherals status can be read in PMC_PCSR1.

26.2.15.7 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 startup time.

26.2.15.8 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
–	0			MOSCRCEN	–	MOSCXTBY	MOSCXTEN

Warning: Bit 4,5,6 must always be set to 0 when programming the CKGR_MOR.

- **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 Startup Time**

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

- **KEY: Password**

Value	Name	Description
0x37	PASSWD	Writing any other value in this field aborts the write operation.

- **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.

The clock failure detection must be disabled when the main crystal is disabled.

The slow RC oscillator must be enabled.

The clock failure detection must be enabled only when system clock MCK selects the fast RC oscillator.

26.2.15.9 PMC Clock Generator Main Clock Frequency Register

Name: CKGR_MCFR

Address: 0xFFFFFC24

Access: Read/Write

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

Note: To ensure that a correct value is read on the MAINF field, the MAINFRDY flag must be read at 1, then another read access must be performed on the register to obtain a stable value on the MAINF field.

- **RCMEAS: RC Oscillator Frequency Measure (write-only)**

0: No effect.

1: Restarts a measure of the main RC frequency, MAINF will carry the new frequency as soon as a low to high transition occurs on MAINFRDY flag.

The measure is performed on the main frequency (i.e., not limited to RC oscillator only) but if the main clock frequency source is the fast crystal oscillator, the restart of the measure is unneeded because of the well known stability of crystal oscillators.

26.2.15.10 PMC Clock Generator PLLA Register

Name: CKGR_PLLAR

Address: 0xFFFFFC28

Access: Read/Write

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

23	22	21	20	19	18	17	16
MULA						OUTA	

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.

- **DIVA: Divider A**

Value	Name	Description
0	0	Divider output is 0
1	BYPASS	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 be programmed to 0.

- **MULA: PLLA Multiplier**

0: The PLLA is deactivated.

1 up to 127: The PLLA Clock frequency is the PLLA input frequency multiplied by MULA + 1.

- **ONE: Must Be Set to 1**

Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

26.2.15.11 PMC Master Clock Register

Name: PMC_MCKR

Address: 0xFFFFFC30

Access: Read/Write

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

- **CSS: Master/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 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	Reserved	Reserved

- **MDIV: Master Clock Division**

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

- **PLLADIV2: PLLA Divisor by 2**

Bit PLLADIV2 must always be set to 1 when MDIV is set to 3.

26.2.15.12 PMC 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.

26.2.15.13 PMC 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.

26.2.15.14 PMC Programmable Clock Register

Name: PMC_PCKx[x = 0..2]

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 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

26.2.15.15 PMC Interrupt Enable Register

Name: PMC_IER

Address: 0xFFFFFC60

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	–	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	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**
- **CFDEV: Clock Failure Detector Event Interrupt Enable**

26.2.15.16 PMC Interrupt Disable Register

Name: PMC_IDR

Address: 0xFFFFFC64

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	–	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	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**
- **CFDEV: Clock Failure Detector Event Interrupt Disable**

26.2.15.17 PMC Status Register

Name: PMC_SR

Address: 0xFFFFFC68

Access: Read-only

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

23	22	21	20	19	18	17	16
–	–	–	FOS	CFDS	CFDEV	–	MOSCSELS

15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	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.

- **CFDEV: Clock Failure Detector Event**

0: No clock failure detection of the fast crystal oscillator clock has occurred since the last read of PMC_SR.

1: At least one clock failure detection of the fast crystal oscillator clock has occurred since the last read of PMC_SR.

- **CFDS: Clock Failure Detector Status**

0: A clock failure of the fast crystal oscillator clock is not detected.

1: A clock failure of the fast crystal 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.

26.2.15.18 PMC Interrupt Mask Register

Name: PMC_IMR

Address: 0xFFFFFC6C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	–	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	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**
- **CFDEV: Clock Failure Detector Event Interrupt Mask**

26.2.15.19 PMC Fault Output Clear Register

Name: PMC_FOCR

Address: 0xFFFFFC78

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
–	–	–	–	–	–	–	FOCLR

- **FOCLR: Fault Output Clear**

Clears the clock failure detector fault output.

26.2.15.20 PLL Charge Pump Current Register

Name: PMC_PLLICPR

Address: 0xFFFFFC80

Access: Read/Write

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

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

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

7	6	5	4	3	2	1	0
–	–	–	–	–	–	ICP_PLLA	

- **ICP_PLLA: Charge Pump Current PLLA**

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

- **IPLL_PLLA: Engineering Configuration PLLA**

Should be written to 3.

- **ICP_PLLU: Charge Pump Current PLL UTMI**

Should be written to 0.

- **IVCO_PLLU: Voltage Control Output Current PLL UTMI**

Should be written to 0.

26.2.15.21 PMC Write Protection Mode Register

Name: PMC_WPMR

Address: 0xFFFFFCE4

Access: Read/Write

Reset: See Table 26-3

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).

See Section 26.2.14 “Register Write Protection” for the list of registers which can be protected.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x504D43	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

26.2.15.22 PMC Write Protection Status Register

Name: PMC_WPSR

Address: 0xFFFFFCE8

Access: Read-only

Reset: See Table 26-3

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.

1: A Write Protect Violation has occurred since the last read of the PMC_WPSR. 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 = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

26.2.15.23 PMC Peripheral Clock Enable Register 1

Name: PMC_PCER1

Address: 0xFFFFFD00

Access: Write-only

31	30	29	28	27	26	25	24
PID63	PID62	PID61	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
PID55	PID54	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	PID38	PID37	PID36	PID35	PID34	PID33	PID32

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Enable**

0: No effect.

1: Enables the corresponding peripheral clock.

Notes: 1. PID32 to PID63 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.

26.2.15.24 PMC Peripheral Clock Disable Register 1

Name: PMC_PCDR1

Address: 0xFFFFFD04

Access: Write-only

31	30	29	28	27	26	25	24
PID63	PID62	PID61	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
PID55	PID54	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	PID38	PID37	PID36	PID35	PID34	PID33	PID32

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Disable**

0: No effect.

1: Disables the corresponding peripheral clock.

Note: PID32 to PID63 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

26.2.15.25 PMC Peripheral Clock Status Register 1

Name: PMC_PCSR1

Address: 0xFFFFFD08

Access: Read-only

31	30	29	28	27	26	25	24
PID63	PID62	PID61	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
PID55	PID54	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	PID38	PID37	PID36	PID35	PID34	PID33	PID32

- **PIDx: Peripheral Clock x Status**

0: The corresponding peripheral clock is disabled.

1: The corresponding peripheral clock is enabled.

Note: PID32 to PID63 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

26.2.15.26 PMC 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**

Peripheral ID selection from PID2 to the maximum PID number. This refers to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

Only the following peripherals can be configured with divided clock (DIV > 0): ADC, SSCx, CANx, USARTx, UARTx, TWIx, SPIx, and TCx.”

Among the PIDs supporting the divided clock, some require a DIV value configuration matching the maximum peripheral frequency (refer to table “Maximum Values for each Peripheral” in the “Electrical Characteristics” section of 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

26.2.15.27 PMC Oscillator Calibration Register

Name: PMC_OCR

Address: 0xFFFFFD10

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
SEL	CAL						

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **CAL: 12 MHz RC Oscillator Calibration bits**

Calibration bits applied to the RC Oscillator when SEL is set.

- **SEL: Selection of RC Oscillator Calibration bits**

0: Factory determined value.

1: Value written by user in CAL field of this register.

27. Parallel Input/Output Controller (PIO)

27.1 Description

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

27.2 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 Drive

27.3 Block Diagram

Figure 27-1. Block Diagram

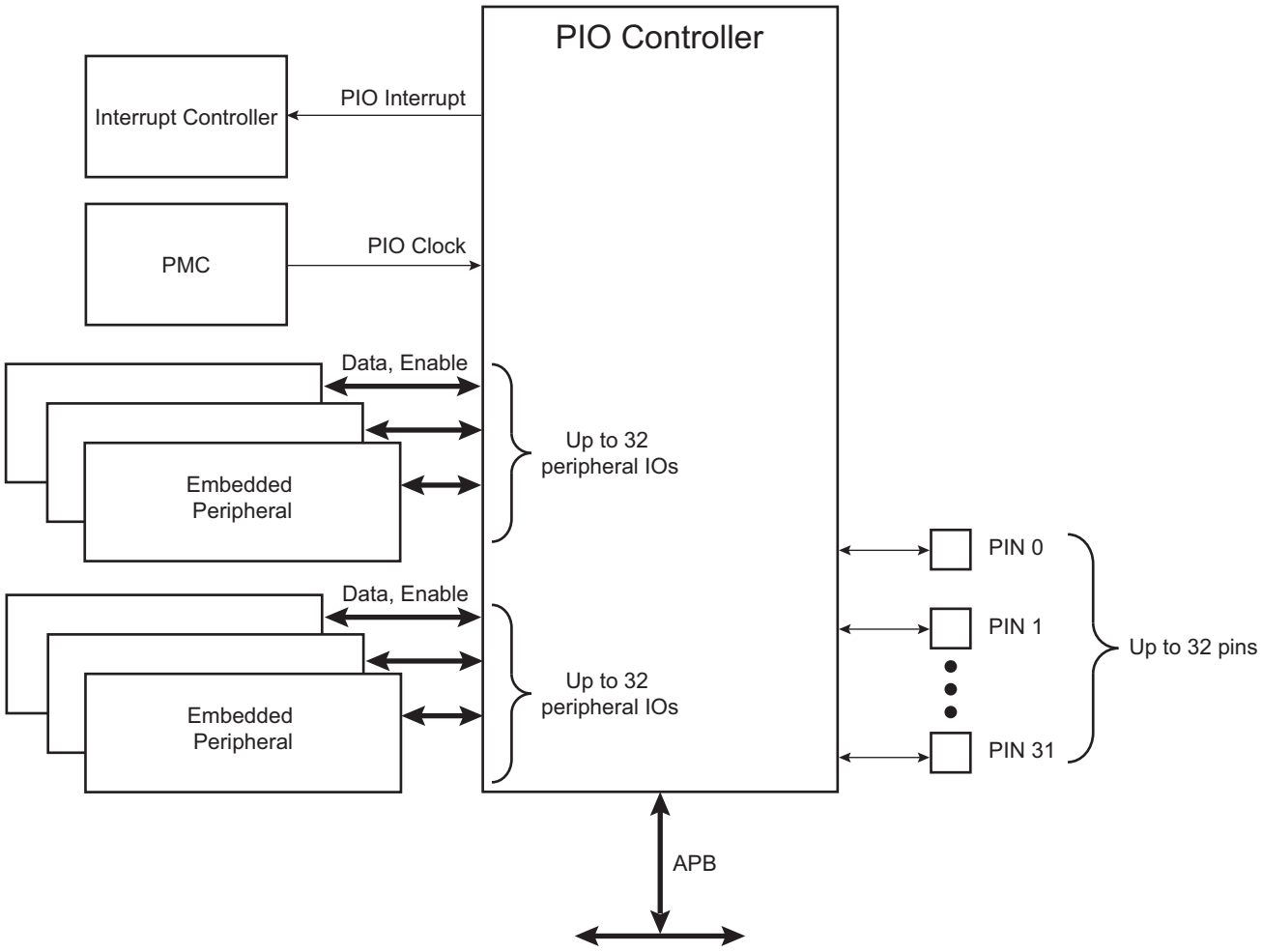
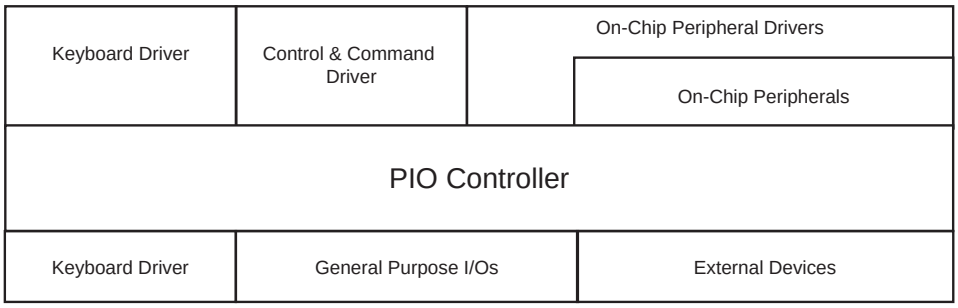


Figure 27-2. Application Block Diagram



27.4 Product Dependencies

27.4.1 Pin Multiplexing

Each pin is configurable, depending on the product, as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO Controllers required by their application. When an I/O line is general-purpose only, i.e. not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

27.4.2 External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the interrupt lines (FIQ or IRQs) are used only as inputs.

27.4.3 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, the 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.

27.4.4 Interrupt Generation

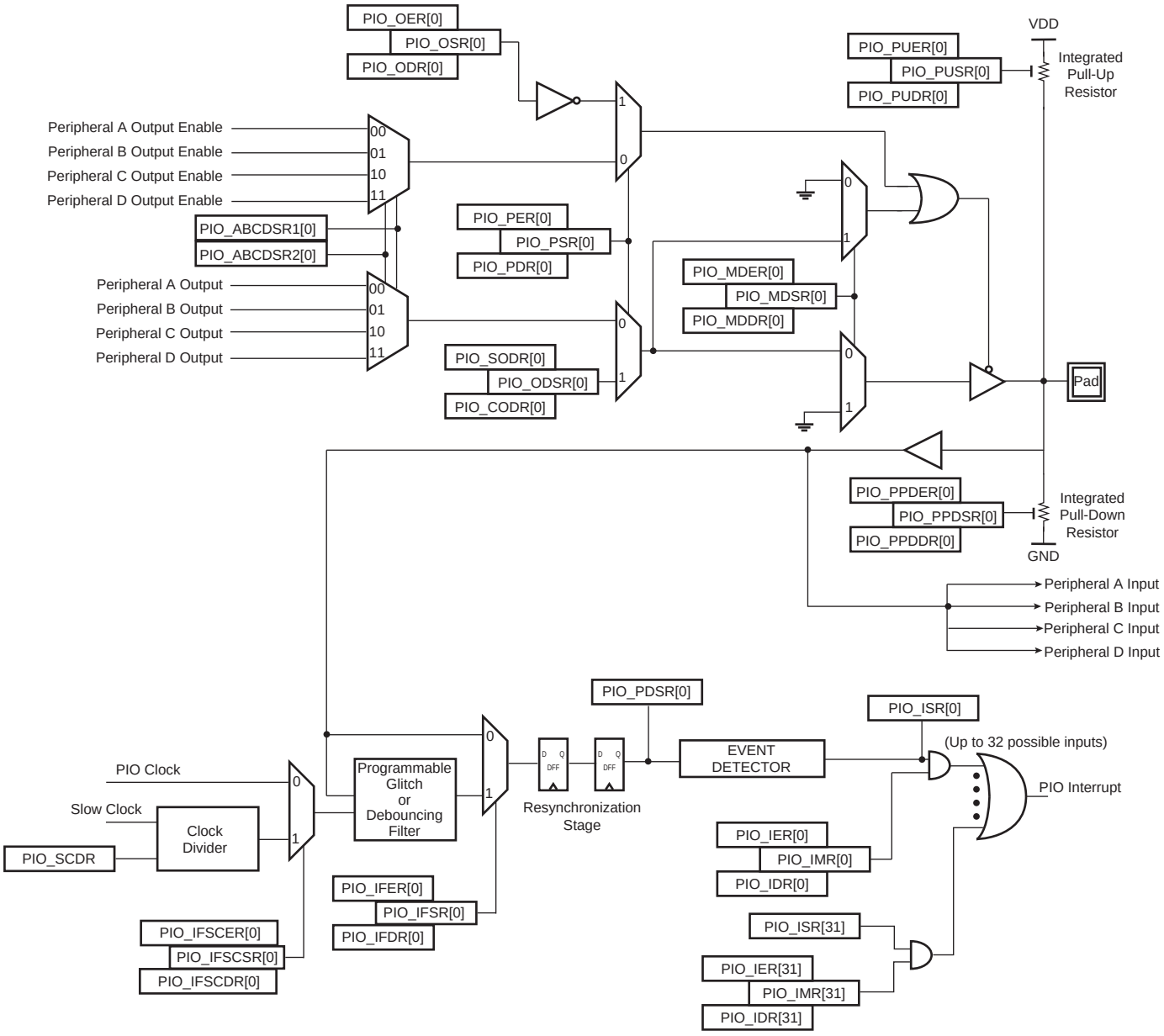
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the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

27.5 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 27-3. In this description each signal shown represents one of up to 32 possible indexes.

Figure 27-3. I/O Line Control Logic



27.5.1 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 to the Pull-up Enable register (PIO_PUER) or Pull-up Disable register (PIO_PUDR), respectively. Writing to these registers results in setting or clearing the corresponding bit in the Pull-up Status register (PIO_PUSR). Reading a one in PIO_PUSR means the pull-up is disabled and reading a zero means the pull-up is enabled. The pull-down resistor can be enabled or disabled by writing the Pull-down Enable register (PIO_PPDER) or the Pull-down Disable register (PIO_PPDDR), respectively. Writing in these registers results in setting or clearing the corresponding bit in the Pull-down Status register (PIO_PPDSR). Reading a one in PIO_PPDSR means the pull-up is disabled and reading a zero 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 relevant 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 relevant 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.

27.5.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the Enable register (PIO_PER) and the Disable register (PIO_PDR). The Status register (PIO_PSR) 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 zero indicates that the pin is controlled by the corresponding on-chip peripheral selected in the ABCD Select registers (PIO_ABCDSR1 and PIO_ABCDSR2). A value of one 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 a one for the corresponding bit.

After reset, the I/O lines are controlled by the PIO Controller, i.e. PIO_PSR resets at one. 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 and depends on the multiplexing of the device.

27.5.3 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.

For each pin:

- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral A is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral B is selected.
- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral C is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level zero 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 PIO_ABCDSR1 and PIO_ABCDSR2 in addition to a write in PIO_PDR.

After reset, PIO_ABCDSR1 and PIO_ABCDSR2 are zero, 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.

27.5.4 Output Control

When the I/O line is assigned to a peripheral function, i.e., the corresponding bit in PIO_PSR is at zero, 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 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 the Output Enable register (PIO_OER) and Output Disable register (PIO_ODR). The results of these write operations are detected in the Output Status register (PIO_OSR). When a bit in this register is at zero, the corresponding I/O line is used as an input only. When the bit is at one, 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 the Set Output Data register (PIO_SODR) and the Clear Output Data register (PIO_CODR). These write operations, respectively, set and clear the Output Data Status register (PIO_ODSR), 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 affects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

27.5.5 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. Only bits unmasked by the Output Write Status register (PIO_OWSR) are written. The mask bits in PIO_OWSR are set by writing to the Output Write Enable register (PIO_OWER) and cleared by writing to the Output Write Disable register (PIO_OWDR).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

27.5.6 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.

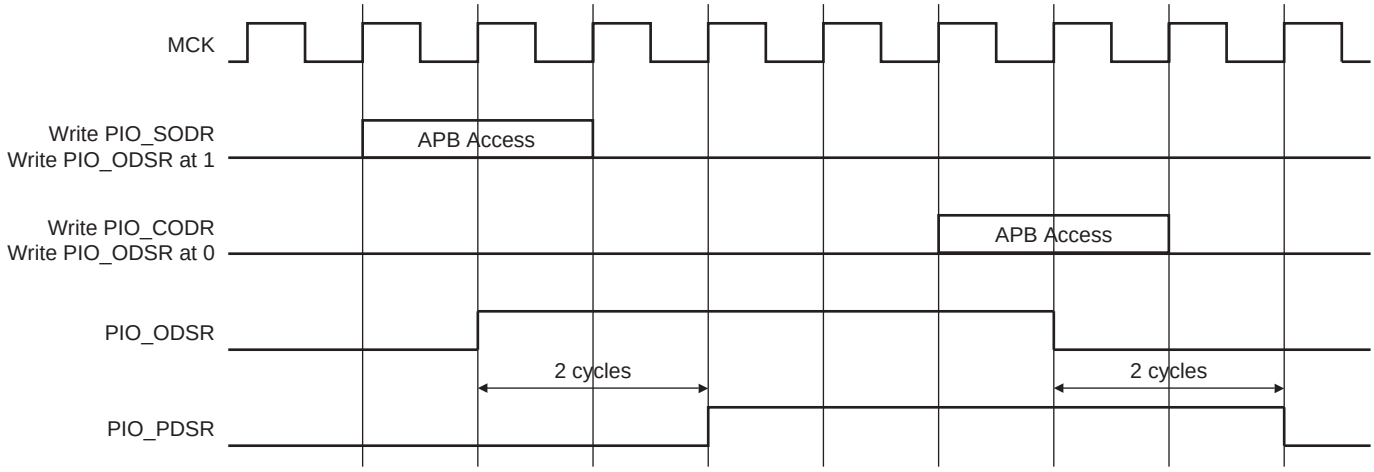
The multi-drive feature is controlled by the Multi-driver Enable register (PIO_MDER) and the Multi-driver Disable register (PIO_MDDR). The multi-drive can be selected whether the I/O line is controlled by the PIO Controller or assigned to a peripheral function. The Multi-driver Status register (PIO_MDSR) indicates the pins that are configured to support external drivers.

After reset, the multi-drive feature is disabled on all pins, i.e. PIO_MDSR resets at value 0x0.

27.5.7 Output Line Timings

Figure 27-4 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 27-4 also shows when the feedback in the Pin Data Status register (PIO_PDSR) is available.

Figure 27-4. Output Line Timings



27.5.8 Inputs

The level on each I/O line can be read through PIO_PDSR. This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input, or driven by the PIO Controller, or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

27.5.9 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 PIO Input Filter Slow Clock Disable register (PIO_IFSCDR) and the PIO Input Filter Slow Clock Enable register (PIO_IFSCER). Writing PIO_IFSCDR and PIO_IFSCER, respectively, sets and clears bits in the Input Filter Slow Clock Status register (PIO_IFSCSR).

The current selection status can be checked by reading the register PIO_IFSCSR.

- If PIO_IFSCSR[i] = 0: The glitch filter can filter a glitch with a duration of less than 1/2 master clock period.
- If PIO_IFSCSR[i] = 1: The debouncing filter can filter a pulse with a duration of less than 1/2 programmable divided slow clock period.

For the debouncing filter, the period of the divided slow clock is performed by writing in the DIV field of the Slow Clock Divider register (PIO_SCDR.

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 one selected clock (MCK or divided slow clock) cycle or more is accepted. For pulse durations between 1/2 selected clock cycle and one 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 one 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, illustrated in Figure 27-5 and Figure 27-6.

The glitch filters are controlled by the Input Filter Enable register (PIO_IFER), the Input Filter Disable register (PIO_IFDR) and the Input Filter Status register (PIO_IFSR). 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 27-5. Input Glitch Filter Timing

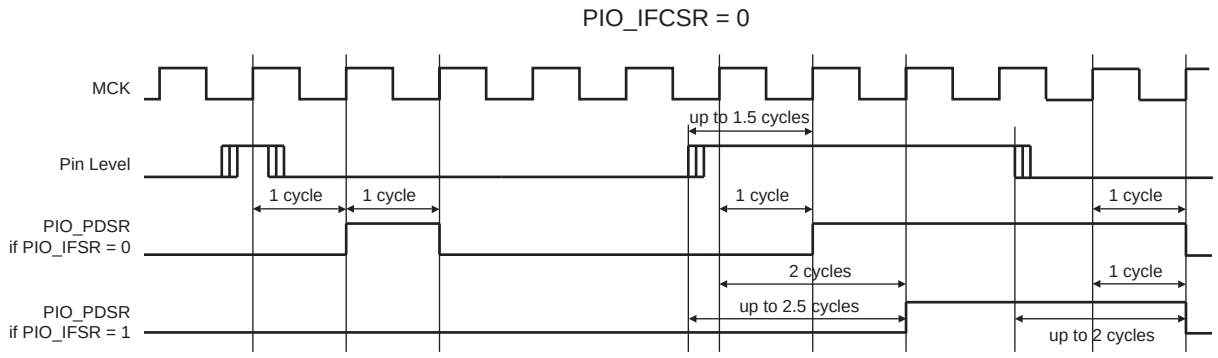
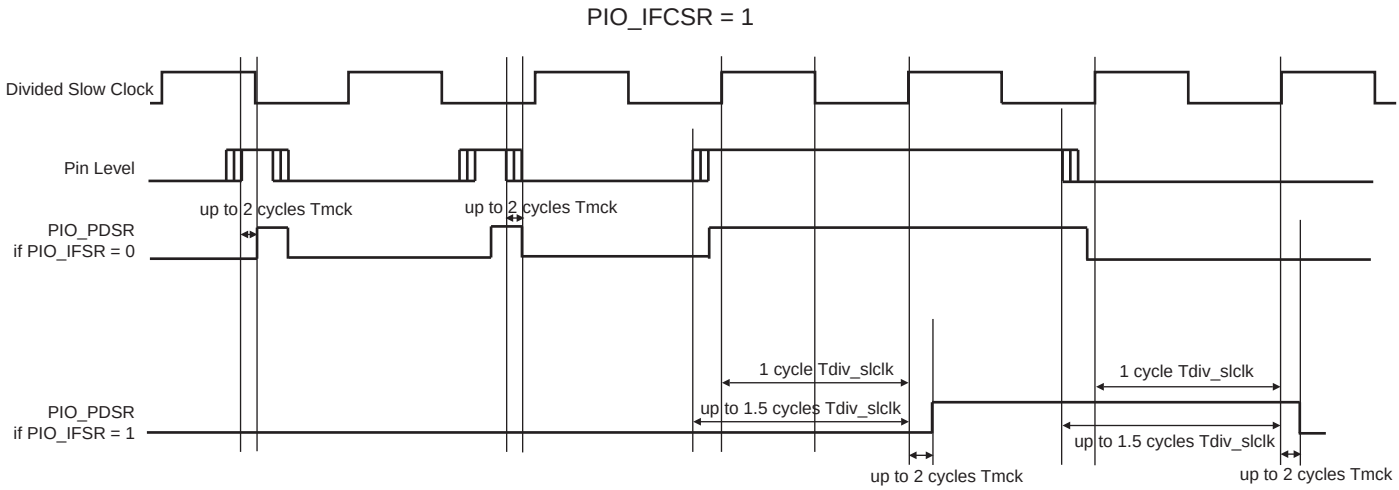


Figure 27-6. Input Debouncing Filter Timing



27.5.10 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 the Interrupt Enable register (PIO_IER) and the Interrupt Disable register (PIO_IDR), which enable and disable the input change interrupt respectively by setting and clearing the corresponding bit in the Interrupt Mask register (PIO_IMR). 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 Additional Interrupt Modes Enable register (PIO_AIMER) and Additional Interrupt Modes Disable register (PIO_AIMDR). The current state of this selection can be read through the Additional Interrupt Modes Mask register (PIO_AIMMR).

These additional modes are:

- Rising edge detection
- Falling edge detection
- Low-level detection
- High-level detection

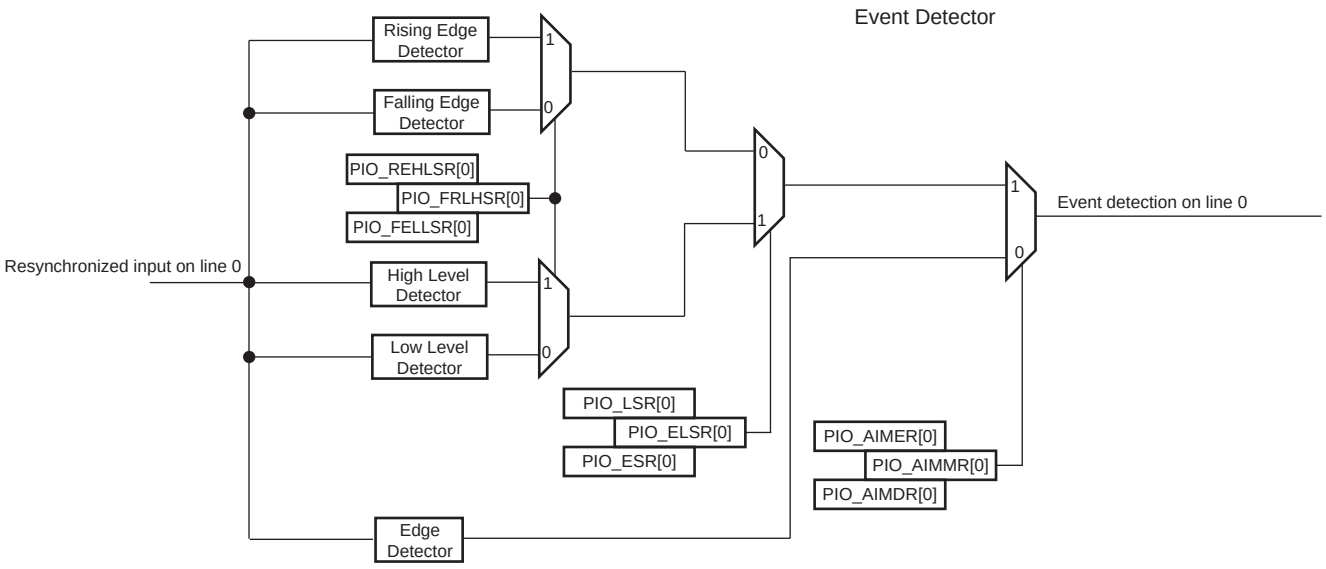
In order to select an additional interrupt mode:

- The type of event detection (edge or level) must be selected by writing in the Edge Select register (PIO_ESR) and Level Select register (PIO_LSR) which , respectively, the edge and level detection. The current status of this selection is accessible through the Edge/Level Status register (PIO_ELSR).
- The polarity of the event detection (rising/falling edge or high/low-level) must be selected by writing in the Falling Edge /Low-Level Select register (PIO_FELLSR) and Rising Edge/High-Level Select register (PIO_REHLSR) which allow to select falling or rising edge (if edge is selected in PIO_ELSR) edge or high- or low-level detection (if level is selected in PIO_ELSR). The current status of this selection is accessible through the Fall/Rise - Low/High Status register (PIO_FRLHSR).

When an input edge or level is detected on an I/O line, the corresponding bit in the Interrupt Status register (PIO_ISR) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted.The interrupt signals of the 32 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 27-7. Event Detector on Input Lines (Figure Represents Line 0)



27.5.10.1 Example

If generating an interrupt is required on the lines below, the configuration required is described in Section 27.5.10.2 "Interrupt Mode Configuration", Section 27.5.10.3 "Edge or Level Detection Configuration" and Section 27.5.10.4 "Falling/Rising Edge or Low/High-Level Detection Configuration":

- 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.

27.5.10.2 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 lines 0 to 7 by writing 32'h0000_00FF in PIO_AIMER.

27.5.10.3 Edge or Level Detection Configuration

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

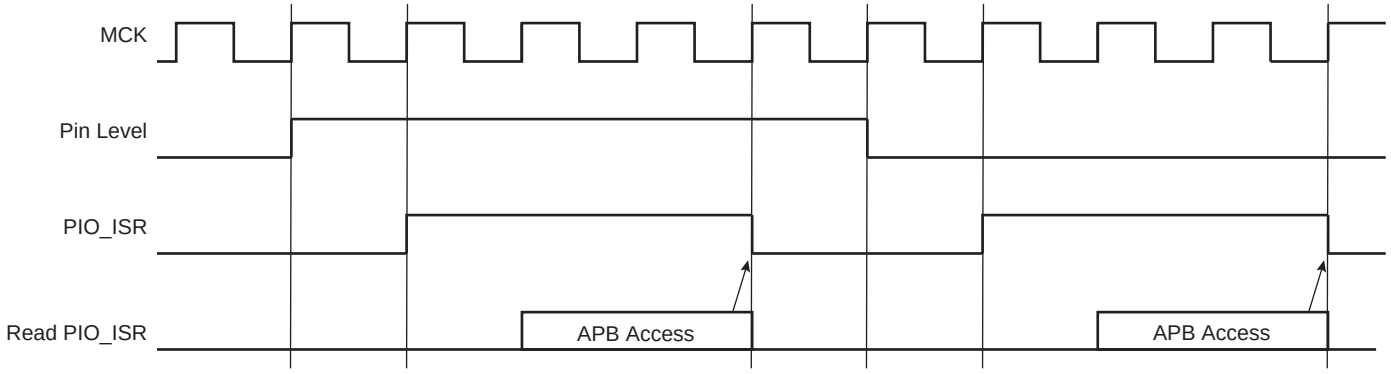
The other lines are configured in edge detection by default, if they have not been previously configured. Otherwise, lines 0, 1, 2, 6 and 7 must be configured in edge detection by writing 32'h0000_00C7 in PIO_ESR.

27.5.10.4 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 27-8. Input Change Interrupt Timings When No Additional Interrupt Modes



27.5.11 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 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.

27.5.12 Programmable I/O Drive

It is possible to configure the I/O drive for pads PA0 to PA31. For any details, refer to the product electrical characteristics.

27.5.13 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.

27.5.14 Register Write Protection

To prevent any single software error from corrupting PIO behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the "PIO Write Protection Mode Register" (PIO_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the "PIO Write Protection Status Register" (PIO_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 PIO_WPSR.

The following registers can be write-protected:

- "PIO Enable Register" on page 289
- "PIO Disable Register" on page 290
- "PIO Output Enable Register" on page 292
- "PIO Output Disable Register" on page 293
- "PIO Input Filter Enable Register" on page 295
- "PIO Input Filter Disable Register" on page 296
- "PIO Multi-driver Enable Register" on page 306
- "PIO Multi-driver Disable Register" on page 307
- "PIO Pull-Up Disable Register" on page 309
- "PIO Pull-Up Enable Register" on page 310
- "PIO Peripheral ABCD Select Register 1" on page 312
- "PIO Peripheral ABCD Select Register 2" on page 313
- "PIO Output Write Enable Register" on page 321
- "PIO Output Write Disable Register" on page 322
- "PIO Pad Pull-Down Disable Register" on page 318
- "PIO Pad Pull-Down Status Register" on page 320

27.6 I/O Lines Programming Example

The programming example shown in Table 27-1 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 27-1. 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

27.7 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 one systematically.

Table 27-2. 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 27-2. 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 Disable 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 Protection Mode Register	PIO_WPMR	Read/Write	0x0
0x00E8	Write Protection Status Register	PIO_WPSR	Read-only	0x0
0x00EC to 0x00F8	Reserved	–	–	–
0x0100	Schmitt Trigger Register	PIO_SCHMITT	Read/Write	0x00000000
0x0104-0x010C	Reserved	–	–	–
0x0110	Reserved	–	–	–
0x0114	Reserved	–	–	–

Table 27-2. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0118	I/O Drive Register 1	PIO_DRIVER1	Read/Write	0x00000000
0x011C	I/O Drive Register 2	PIO_DRIVER2	Read/Write	0x00000000
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.
5. If an offset is not listed in the table it must be considered as reserved.

27.7.1 PIO Enable Register

Name: PIO_PER

Address: 0xFFFFF200 (PIOA), 0xFFFFF400 (PIOB), 0xFFFFF600 (PIOC), 0xFFFFF800 (PIOD), 0xFFFFFA00 (PIOE)

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 Protection Mode Register” .

- **P0-P31: PIO Enable**

0: No effect.

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

27.7.2 PIO Disable Register

Name: PIO_PDR

Address: 0xFFFFF204 (PIOA), 0xFFFFF404 (PIOB), 0xFFFFF604 (PIOC), 0xFFFFF804 (PIOD), 0xFFFFFA04 (PIOE)

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 Protection Mode Register” .

- **P0-P31: PIO Disable**

0: No effect.

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

27.7.3 PIO Status Register

Name: PIO_PSR

Address: 0xFFFFF208 (PIOA), 0xFFFFF408 (PIOB), 0xFFFFF608 (PIOC), 0xFFFFF808 (PIOD), 0xFFFFFA08 (PIOE)

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).

27.7.4 PIO Output Enable Register

Name: PIO_OER

Address: 0xFFFFF210 (PIOA), 0xFFFFF410 (PIOB), 0xFFFFF610 (PIOC), 0xFFFFF810 (PIOD), 0xFFFFFA10 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Output Enable**

0: No effect.

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

27.7.5 PIO Output Disable Register

Name: PIO_ODR

Address: 0xFFFFF214 (PIOA), 0xFFFFF414 (PIOB), 0xFFFFF614 (PIOC), 0xFFFFF814 (PIOD), 0xFFFFFA14 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Output Disable**

0: No effect.

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

27.7.6 PIO Output Status Register

Name: PIO_OSR

Address: 0xFFFFF218 (PIOA), 0xFFFFF418 (PIOB), 0xFFFFF618 (PIOC), 0xFFFFF818 (PIOD), 0xFFFFFA18 (PIOE)

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.

27.7.7 PIO Input Filter Enable Register

Name: PIO_IFER

Address: 0xFFFFF220 (PIOA), 0xFFFFF420 (PIOB), 0xFFFFF620 (PIOC), 0xFFFFF820 (PIOD), 0xFFFFFA20 (PIOE)

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 Protection Mode Register” .

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

0: No effect.

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

27.7.8 PIO Input Filter Disable Register

Name: PIO_IFDR

Address: 0xFFFFF224 (PIOA), 0xFFFFF424 (PIOB), 0xFFFFF624 (PIOC), 0xFFFFF824 (PIOD), 0xFFFFFA24 (PIOE)

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 Protection Mode Register” .

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

0: No effect.

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

27.7.9 PIO Input Filter Status Register

Name: PIO_IFSR

Address: 0xFFFFF228 (PIOA), 0xFFFFF428 (PIOB), 0xFFFFF628 (PIOC), 0xFFFFF828 (PIOD), 0xFFFFFA28 (PIOE)

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.

27.7.10 PIO Set Output Data Register

Name: PIO_SODR

Address: 0xFFFFF230 (PIOA), 0xFFFFF430 (PIOB), 0xFFFFF630 (PIOC), 0xFFFFF830 (PIOD), 0xFFFFFA30 (PIOE)

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.

27.7.11 PIO Clear Output Data Register

Name: PIO_CODR

Address: 0xFFFFF234 (PIOA), 0xFFFFF434 (PIOB), 0xFFFFF634 (PIOC), 0xFFFFF834 (PIOD), 0xFFFFFA34 (PIOE)

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.

27.7.12 PIO Output Data Status Register

Name: PIO_ODSR

Address: 0xFFFFF238 (PIOA), 0xFFFFF438 (PIOB), 0xFFFFF638 (PIOC), 0xFFFFF838 (PIOD), 0xFFFFFA38 (PIOE)

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.

27.7.13 PIO Pin Data Status Register

Name: PIO_PDSR

Address: 0xFFFFF23C (PIOA), 0xFFFFF43C (PIOB), 0xFFFFF63C (PIOC), 0xFFFFF83C (PIOD), 0xFFFFFA3C (PIOE)

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.

27.7.14 PIO Interrupt Enable Register

Name: PIO_IER

Address: 0xFFFFF240 (PIOA), 0xFFFFF440 (PIOB), 0xFFFFF640 (PIOC), 0xFFFFF840 (PIOD), 0xFFFFFA40 (PIOE)

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.

27.7.15 PIO Interrupt Disable Register

Name: PIO_IDR

Address: 0xFFFFF244 (PIOA), 0xFFFFF444 (PIOB), 0xFFFFF644 (PIOC), 0xFFFFF844 (PIOD), 0xFFFFFA44 (PIOE)

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.

27.7.16 PIO Interrupt Mask Register

Name: PIO_IMR

Address: 0xFFFFF248 (PIOA), 0xFFFFF448 (PIOB), 0xFFFFF648 (PIOC), 0xFFFFF848 (PIOD), 0xFFFFFA48 (PIOE)

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.

27.7.17 PIO Interrupt Status Register

Name: PIO_ISR

Address: 0xFFFFF24C (PIOA), 0xFFFFF44C (PIOB), 0xFFFFF64C (PIOC), 0xFFFFF84C (PIOD), 0xFFFFFA4C (PIOE)

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.

27.7.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Address: 0xFFFFF250 (PIOA), 0xFFFFF450 (PIOB), 0xFFFFF650 (PIOC), 0xFFFFF850 (PIOD), 0xFFFFFA50 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Multi-Drive Enable**

0: No effect.

1: Enables multi-drive on the I/O line.

27.7.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Address: 0xFFFFF254 (PIOA), 0xFFFFF454 (PIOB), 0xFFFFF654 (PIOC), 0xFFFFF854 (PIOD), 0xFFFFFA54 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Multi-Drive Disable**

0: No effect.

1: Disables multi-drive on the I/O line.

27.7.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Address: 0xFFFFF258 (PIOA), 0xFFFFF458 (PIOB), 0xFFFFF658 (PIOC), 0xFFFFF858 (PIOD), 0xFFFFFA58 (PIOE)

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.

27.7.21 PIO Pull-Up Disable Register

Name: PIO_PUDR

Address: 0xFFFFF260 (PIOA), 0xFFFFF460 (PIOB), 0xFFFFF660 (PIOC), 0xFFFFF860 (PIOD), 0xFFFFFA60 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Pull-Up Disable**

0: No effect.

1: Disables the pull-up resistor on the I/O line.

27.7.22 PIO Pull-Up Enable Register

Name: PIO_PUER

Address: 0xFFFFF264 (PIOA), 0xFFFFF464 (PIOB), 0xFFFFF664 (PIOC), 0xFFFFF864 (PIOD), 0xFFFFFA64 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Pull-Up Enable**

0: No effect.

1: Enables the pull-up resistor on the I/O line.

27.7.23 PIO Pull-Up Status Register

Name: PIO_PUSR

Address: 0xFFFFF268 (PIOA), 0xFFFFF468 (PIOB), 0xFFFFF668 (PIOC), 0xFFFFF868 (PIOD), 0xFFFFFA68 (PIOE)

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.

27.7.24 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 Protection 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.

27.7.25 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 Protection 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.

27.7.26 PIO Input Filter Slow Clock Disable Register

Name: PIO_IFSCDR

Address: 0xFFFFF280 (PIOA), 0xFFFFF480 (PIOB), 0xFFFFF680 (PIOC), 0xFFFFF880 (PIOD), 0xFFFFFA80 (PIOE)

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.

27.7.27 PIO Input Filter Slow Clock Enable Register

Name: PIO_IFSCER

Address: 0xFFFFF284 (PIOA), 0xFFFFF484 (PIOB), 0xFFFFF684 (PIOC), 0xFFFFF884 (PIOD), 0xFFFFFA84 (PIOE)

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.

27.7.28 PIO Input Filter Slow Clock Status Register

Name: PIO_IFSCSR

Address: 0xFFFFF288 (PIOA), 0xFFFFF488 (PIOB), 0xFFFFF688 (PIOC), 0xFFFFF888 (PIOD), 0xFFFFFA88 (PIOE)

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.

27.7.29 PIO Slow Clock Divider Debouncing Register

Name: PIO_SCDR

Address: 0xFFFFF28C (PIOA), 0xFFFFF48C (PIOB), 0xFFFFF68C (PIOC), 0xFFFFF88C (PIOD), 0xFFFFFA8C (PIOE)

Access: Read/Write

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

- **DIV: Slow Clock Divider Selection for Debouncing**

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

27.7.30 PIO Pad Pull-Down Disable Register

Name: PIO_PPDDR

Address: 0xFFFFF290 (PIOA), 0xFFFFF490 (PIOB), 0xFFFFF690 (PIOC), 0xFFFFF890 (PIOD), 0xFFFFFA90 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Pull-Down Disable**

0: No effect.

1: Disables the pull-down resistor on the I/O line.

27.7.31 PIO Pad Pull-Down Enable Register

Name: PIO_PPDER

Address: 0xFFFFF294 (PIOA), 0xFFFFF494 (PIOB), 0xFFFFF694 (PIOC), 0xFFFFF894 (PIOD), 0xFFFFFA94 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Pull-Down Enable**

0: No effect.

1: Enables the pull-down resistor on the I/O line.

27.7.32 PIO Pad Pull-Down Status Register

Name: PIO_PPDSR

Address: 0xFFFFF298 (PIOA), 0xFFFFF498 (PIOB), 0xFFFFF698 (PIOC), 0xFFFFF898 (PIOD), 0xFFFFFA98 (PIOE)

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 Protection 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.

27.7.33 PIO Output Write Enable Register

Name: PIO_OWER

Address: 0xFFFFF2A0 (PIOA), 0xFFFFF4A0 (PIOB), 0xFFFFF6A0 (PIOC), 0xFFFFF8A0 (PIOD), 0xFFFFFAA0 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Output Write Enable**

0: No effect.

1: Enables writing PIO_ODSR for the I/O line.

27.7.34 PIO Output Write Disable Register

Name: PIO_OWDR

Address: 0xFFFFF2A4 (PIOA), 0xFFFFF4A4 (PIOB), 0xFFFFF6A4 (PIOC), 0xFFFFF8A4 (PIOD), 0xFFFFFAA4 (PIOE)

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 Protection Mode Register” .

- **P0-P31: Output Write Disable**

0: No effect.

1: Disables writing PIO_ODSR for the I/O line.

27.7.35 PIO Output Write Status Register

Name: PIO_OWSR

Address: 0xFFFFF2A8 (PIOA), 0xFFFFF4A8 (PIOB), 0xFFFFF6A8 (PIOC), 0xFFFFF8A8 (PIOD), 0xFFFFFAA8 (PIOE)

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.

27.7.36 PIO Additional Interrupt Modes Enable Register

Name: PIO_AIMER

Address: 0xFFFFF2B0 (PIOA), 0xFFFFF4B0 (PIOB), 0xFFFFF6B0 (PIOC), 0xFFFFF8B0 (PIOD), 0xFFFFFAB0 (PIOE)

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.

27.7.37 PIO Additional Interrupt Modes Disable Register

Name: PIO_AIMDR

Address: 0xFFFFF2B4 (PIOA), 0xFFFFF4B4 (PIOB), 0xFFFFF6B4 (PIOC), 0xFFFFF8B4 (PIOD), 0xFFFFFAB4 (PIOE)

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).

27.7.38 PIO Additional Interrupt Modes Mask Register

Name: PIO_AIMMR

Address: 0xFFFFF2B8 (PIOA), 0xFFFFF4B8 (PIOB), 0xFFFFF6B8 (PIOC), 0xFFFFF8B8 (PIOD), 0xFFFFFAB8 (PIOE)

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.

27.7.39 PIO Edge Select Register

Name: PIO_ESR

Address: 0xFFFFF2C0 (PIOA), 0xFFFFF4C0 (PIOB), 0xFFFFF6C0 (PIOC), 0xFFFFF8C0 (PIOD), 0xFFFFFAC0 (PIOE)

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.

27.7.40 PIO Level Select Register

Name: PIO_LSR

Address: 0xFFFFF2C4 (PIOA), 0xFFFFF4C4 (PIOB), 0xFFFFF6C4 (PIOC), 0xFFFFF8C4 (PIOD), 0xFFFFFAC4 (PIOE)

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.

27.7.41 PIO Edge/Level Status Register

Name: PIO_ELSR

Address: 0xFFFFF2C8 (PIOA), 0xFFFFF4C8 (PIOB), 0xFFFFF6C8 (PIOC), 0xFFFFF8C8 (PIOD), 0xFFFFFAC8 (PIOE)

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.

27.7.42 PIO Falling Edge/Low-Level Select Register

Name: PIO_FELLSR

Address: 0xFFFFF2D0 (PIOA), 0xFFFFF4D0 (PIOB), 0xFFFFF6D0 (PIOC), 0xFFFFF8D0 (PIOD), 0xFFFFFAD0 (PIOE)

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.

27.7.43 PIO Rising Edge/High-Level Select Register

Name: PIO_REHLSR

Address: 0xFFFFF2D4 (PIOA), 0xFFFFF4D4 (PIOB), 0xFFFFF6D4 (PIOC), 0xFFFFF8D4 (PIOD), 0xFFFFFAD4 (PIOE)

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.

27.7.44 PIO Fall/Rise - Low/High Status Register

Name: PIO_FRLHSR

Address: 0xFFFFF2D8 (PIOA), 0xFFFFF4D8 (PIOB), 0xFFFFF6D8 (PIOC), 0xFFFFF8D8 (PIOD), 0xFFFFFAD8 (PIOE)

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).

27.7.45 PIO Lock Status Register

Name: PIO_LOCKSR

Address: 0xFFFFF2E0 (PIOA), 0xFFFFF4E0 (PIOB), 0xFFFFF6E0 (PIOC), 0xFFFFF8E0 (PIOD), 0xFFFFFAE0 (PIOE)

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.

27.7.46 PIO Write Protection Mode Register

Name: PIO_WPMR

Address: 0xFFFFF2E4 (PIOA), 0xFFFFF4E4 (PIOB), 0xFFFFF6E4 (PIOC), 0xFFFFF8E4 (PIOD), 0xFFFFFAE4 (PIOE)

Access: Read/Write

Reset: See Table 27-2

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-protecting registers, refer to Section 27.5.14 ”Register Write Protection”.

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

See Section 27.5.14 ”Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key.**

Value	Name	Description
0x50494F	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

27.7.47 PIO Write Protection Status Register

Name: PIO_WPSR

Address: 0xFFFFF2E8 (PIOA), 0xFFFFF4E8 (PIOB), 0xFFFFF6E8 (PIOC), 0xFFFFF8E8 (PIOD), 0xFFFFFAE8 (PIOE)

Access: Read-only

Reset: See Table 27-2

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 Protection Violation Status**

0: No write protection violation has occurred since the last read of the PIO_WPSR register.

1: A write protection 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 Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

27.7.48 PIO Schmitt Trigger Register

Name: PIO_SCHMITT

Address: 0xFFFFF300 (PIOA), 0xFFFFF500 (PIOB), 0xFFFFF700 (PIOC), 0xFFFFF900 (PIOD), 0xFFFFFB00 (PIOE)

Access: Read/Write

Reset: See Table 27-2

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]: Schmitt Trigger Control**

0: Schmitt trigger is enabled.

1: Schmitt trigger is disabled.

27.7.49 PIO I/O Drive Register 1

Name: PIO_DRIVER1

Address: 0xFFFFF318 (PIOA), 0xFFFFF518 (PIOB), 0xFFFFF718 (PIOC), 0xFFFFF918 (PIOD), 0xFFFFFB18 (PIOE)

Access: Read/Write

Reset: See Table 27-2

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	LO_DRIVE	Low drive
1	LO_DRIVE	Low drive
2	ME_DRIVE	Medium drive
3	HI_DRIVE	High drive

27.7.50 PIO I/O Drive Register 2

Name: PIO_DRIVER2

Address: 0xFFFFF31C (PIOA), 0xFFFFF51C (PIOB), 0xFFFFF71C (PIOC), 0xFFFFF91C (PIOD), 0xFFFFFB1C (PIOE)

Access: Read/Write

Reset: See Table 27-2

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**

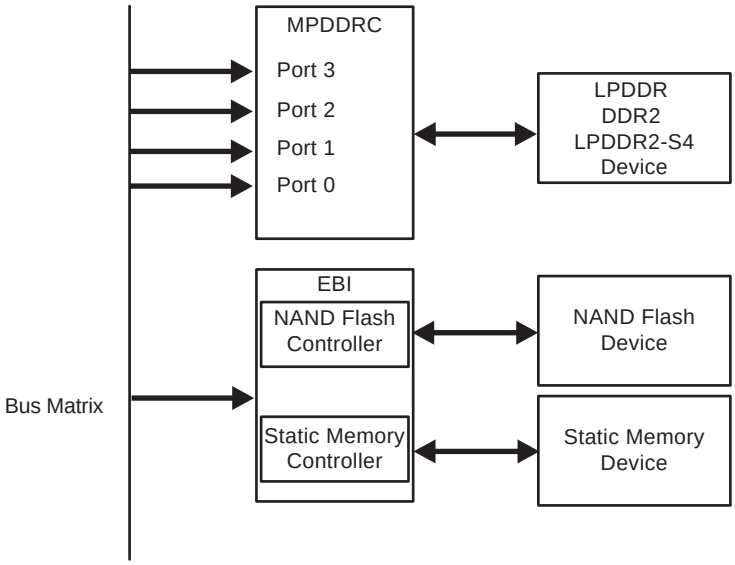
Value	Name	Description
0	LO_DRIVE	Low drive
1	LO_DRIVE	Low drive
2	ME_DRIVE	Medium drive
3	HI_DRIVE	High drive

28. External Memories

The product features:

- Multiport DDR Controller (MPDDRC)
- External Bus Interface (EBI) that embeds a NAND Flash controller and a Static Memory Controller (HSMC)

Figure 28-1. External Memory Controllers



- MPDDRC is a standalone multi-port DDRSDR controller. It supports only DDR2, LPDDR, and LPDDR2-S4 devices. Its user interface is located at 0xFFFFEA00.
- HSMC supports Static Memories and MLC/SLC NAND Flashes. It embeds Multi-Bit ECC. Its user interface is located at 0xFFFFC000.

28.1 MPDDRC Multi-port DDRSDR Controller

28.1.1 Description

The DDR2 Controller is dedicated to 8-port DDR2/LPDDR/LPDDR2 support. Data transfers are performed through a 32-bit data bus on one chip select. The Controller operates with 1.8V Power Supply for DDR2 and LP-DDR, 1.2V Power Supply for LP-DDR2.

28.1.2 Embedded Characteristics

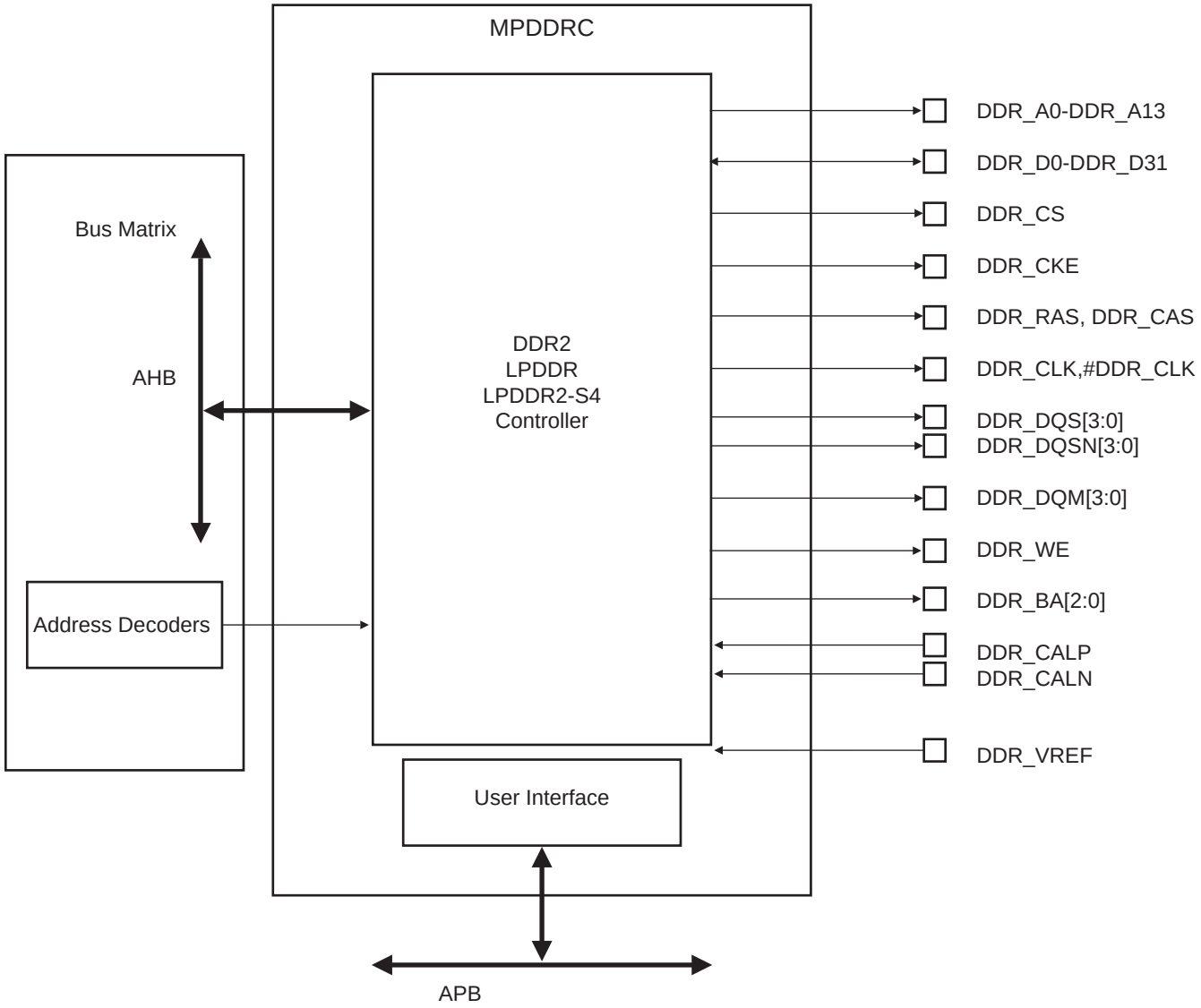
28.1.2.1 DDR2/LPDDR/LPDDR2 Controller

Four AHB Interfaces, Management of All Accesses Maximizes Memory Bandwidth and Minimizes Transaction Latency.

- Supports AHB Transfers:
  - DWord, Word, Half Word, Byte Access.
- Supports Low-Power DDR2-SDRAM-S4, DDR2-SDRAM, Low-Power DDR1-SDRAM
- Numerous Configurations Supported
  - 2K, 4K, 8K, 16K Row Address Memory Parts
  - DDR2 with Four or Eight Internal Banks (DDR2_SDRAM/Low-Power DDR2-SDRAM)
  - DDR2/LPDDR with 32-bit Data Path
  - One Chip Select for DDR2/LPDDR Device (512 Mbytes Address Space)
- Programming Facilities
  - Multibank Ping-pong Access (Up to 4 or 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 (Low-power DDR-SDRAM Devices)
- Energy-saving Capabilities
  - Self-refresh, Power-down, Active Power-down and Deep Power-down Modes Supported
- Power-up Initialization by Software
- CAS Latency of 2, 3, 4, 5, 6 supported
- Reset function supported (DDR2)
- Auto Precharge Command Not Used
- On Die Termination not supported
- OCD mode not supported

28.1.3 MPDDR Controller Block Diagram

Figure 28-2. Organization of the MPDDRC



28.1.4 I/O Lines Description

Table 28-1. DDR2 I/O Lines Description

Name	Function	Type	Active Level
DDR2/LPDDR Controller			
VDDIODDR	Power Supply of memory interface	Input	
DDR_VREF	Reference Voltage for DDR2 operations, typically 0.9V	Input	
DDR_CALP	Pad positive calibration reference for LP-DDR2	Input	
DDR_CALN	Pad negative calibration reference for LP-DDR2	Input	
DDR_D0 - DDR_D31	Data Bus	I/O	
DDR_A0 - DDR_A13	Address Bus	Output	
DDR_DQM0 - DDR_DQM3	Data Mask	Output	
DDR_DQS0 - DDR_DQS3	Data Strobe	Output	
DDR_DQSN0 - DDR_DQSN3	Negative Data Strobe	Output	
DDR_CS	Chip Select	Output	Low
DDR_CLK - DDR_CLK#	DDR2 Differential Clock	Output	
DDR_CKE	Clock enable	Output	High
DDR_RAS	Row signal	Output	Low
DDR_CAS	Column signal	Output	Low
DDR_WE	Write enable	Output	Low
DDR_BA0 - DDR_BA2	Bank Select	Output	

In LPDDR2 mode, DQS and DQSN are connected to the LPDDR2 memory.

28.1.5 Product Dependencies

The pins used for interfacing the DDR2 memory are not multiplexed with the PIO lines.

Table 28-2. DDR2 I/O Lines Usage vs Operating Modes

Signal Name	DDR2 Mode	LPDDR2 Mode	LPDDR
DDR_VREF	VDDIODDR/2	VDDIODDR/2	VDDIODDR/2
DDR_CALP	GND via 200Ω resistor	GND via 240Ω resistor	GND via 200Ω resistor
DDR_CALN	VDDIODDR via 200Ω resistor	VDDIODDR via 240Ω resistor	VDDIODDR via 200Ω resistor
DDR_CK, DDR_CKN	CLK and CLKN	CLK and CLKN	CLK and CLKN
DDR_CKE	CLKE	CLKE	CLKE
DDR_CS	CS	CS	CS
DDR_BA[2..0]	BA[2..0]	BA[2..0]	BA[2..0]
DDR_WE	WE	CA2	WE
DDR_RAS - DDR_CAS	RAS, CAS	CA0, CA1	RAS, CAS
DDR_A[13..0]	A[13:0]	CAx, with x>2	A[13:0]
DDR_D[31..0]	D[31:0]	D[31:0]	D[31:0]
DQS[3..0], DQSN[3..0]	DQS[3:0] DQSN connected to DDR_VREF	DQS[3:0] DQSN[3:0]	DQS[3:0] DQSN connected to DDR_VREF
DQM[3..0]	DQM[3..0]	DQM[3..0]	DQM[3..0]

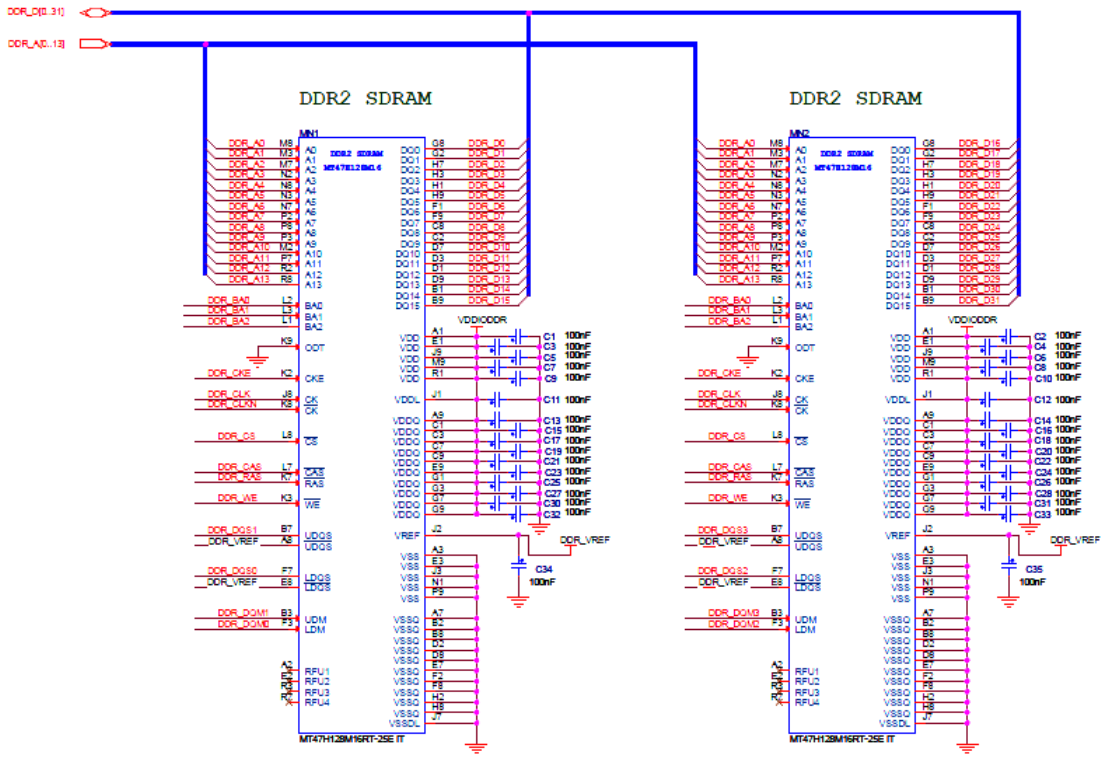
28.1.6 Implementation Example

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

28.1.6.1 2x16-bit DDR2

Hardware Configuration

Figure 28-3. 2x16-bit DDR2 Hardware Configuration



Software Configuration

The following configuration has to be performed:

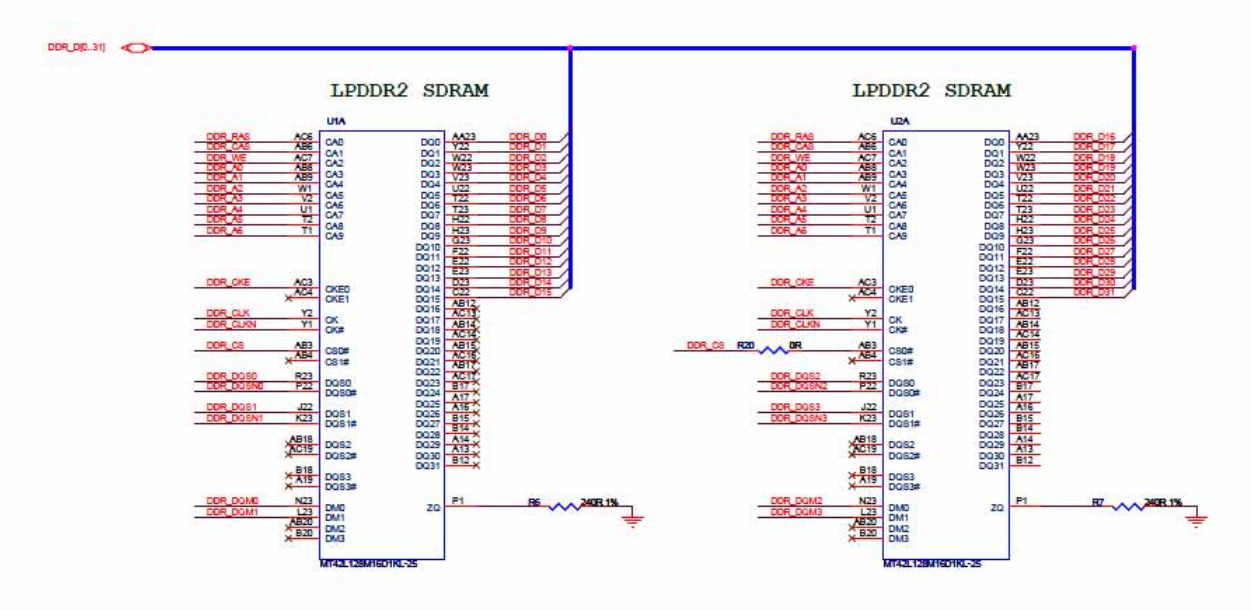
- Initialize the DDR2 Controller depending on the DDR2 device and system bus frequency.

The DDR2 initialization sequence is described in the sub-section “DDR2 Device Initialization” of the DDRSDRC section.

28.1.6.2 2x16-bit LPDDR2

Hardware Configuration

Figure 28-4. 2x16-bit LPDDR2 Hardware Configuration



CAx LPDDR2 signals are to be connected as indicated in Table 28-3:

Table 28-3. CAx LP-DDR2 Signal Connection

DDR Controller Signal	LP-DDR2 Signal
RAS	CA0
CAS	CA1
WE	CA2
DDR_A0	CA3
DDR_A1	CA4
DDR_A2	CA5
DDR_A3	CA6
DDR_A4	CA7
DDR_A5	CA8
DDR_A6	CA9
Higher addresses	Higher CAs

Software Configuration

The following configuration has to be performed:

- Initialize the DDR2 Controller depending on the LPDDR2 device and system bus frequency.

The DDR2 initialization sequence is described in the sub-section “LPDDR2 Device Initialization” of the DDRSDRC section.

28.2 External Bus Interface (EBI)

28.2.1 Description

The External Bus Interface is designed to ensure the successful data transfer between several external devices and the ARM processor-based device. The External Bus Interface of the device consists of a Static Memory Controller (HSMC).

This HSMC is capable of handling several types of external memory and peripheral devices, such as SRAM, PSRAM, PROM, EPROM, EEPROM, LCD Module, NOR Flash and NAND Flash.

The HSMC generates the signals that control the access to external memory devices or peripheral devices. It has 4 Chip Selects and a 26-bit address bus. The 16-bit data bus can be configured to interface with 8- or 16-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 HSMC can manage wait requests from external devices to extend the current access. The HSMC 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 HSMC embeds a NAND Flash Controller (NFC). The NFC can handle automatic transfers, sending the commands and address cycles to the NAND Flash and transferring the contents of the page (for read and write) to the NFC SRAM. It minimizes the CPU overhead.

The HSMC includes programmable hardware error correcting code with one-bit error correction capability and supports two-bit error detection. In order to improve the overall system performance, the DATA phase of the transfer can be DMA-assisted. The External Data Bus can be scrambled/unscrambled by means of user keys.

The full description is available in the HSMC section.

28.2.1.1 External Bus Interface (EBI)

- Integrates Two External Memory Controllers:
  - Static Memory Controller
  - SLC/MLC Nand Flash Controller
- Additional logic for NAND Flash
- Optional 16-bit External Data Bus
- Up to 26-bit Address Bus (up to 64 MBytes linear per chip select)
- Up to 4 chip selects, Configurable Assignment
- NAND Flash chip select is programmable:

28.2.1.2 Static Memory Controller (HSMC)

- 64-MByte Address Space per Chip Select
- 8- or 16-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
- External Data Bus Scrambling/Unscrambling Function
- External Wait Request
- Automatic Switch to Slow Clock Mode
- NAND Flash Controller Supporting NAND Flash with Multiplexed Data/Address Buses
- Supports SLC and MLC NAND Flash Technology
- Programmable Timing on a per Chip Select Basis

28.2.1.3 NAND Flash Controller (NFC)

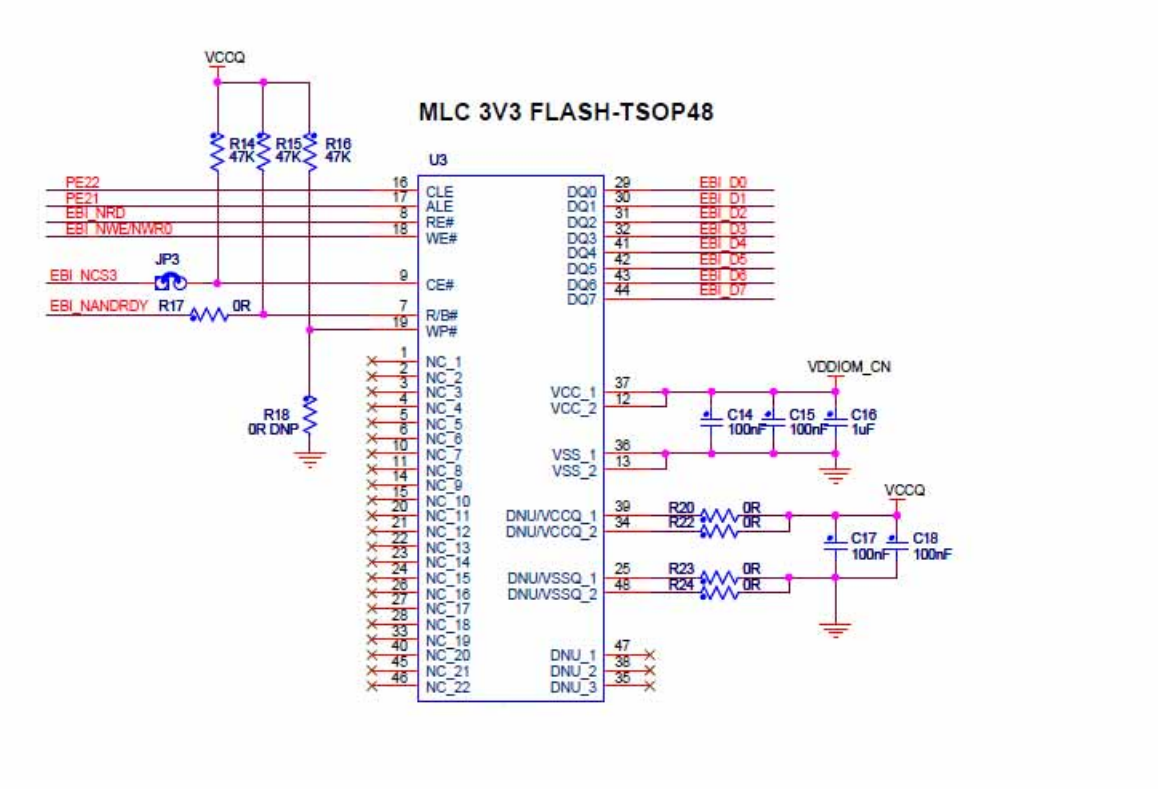
- Programmable Flash Data Width 8 bits or 16 bits.
- Supports NAND Flash and SmartMedia™ Devices with 8- or 16-bit Data Path.
- Supports 1-bit Correction for a Page of 512, 1024, 2048 and 4096 Bytes with 8- or 16-bit Data Path.
- Supports 1-bit Correction per 512 Bytes of Data for a Page Size of 512, 2048 and 4096 Bytes with 8-bit Data Path.
- Supports 1-bit Correction per 256 Bytes of Data for a Page Size of 512, 2048 and 4096 Bytes with 8-bit Data Path.
- Multibit Error Correcting Code (ECC)
  - ECC Algorithm based on binary shortened Bose, Chaudhuri and Hocquenghem (BCH) codes.
  - Programmable Error Correcting Capability: 2, 4, 8, 12 and 24 bits of errors per block.
  - Programmable block size: 512 Bytes or 1024 Bytes.
  - Programmable number of block per page: 1, 2, 4 or 8 blocks of data per page.
  - Programmable spare area size.
  - Supports spare area ECC protection.
  - Supports 8 kBytes page size using 1024 Bytes/block and 4 kBytes page size using 512 Bytes/block.
  - Multibit Error detection is interrupt driven.
  - 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 finite Field GF(2^13) or GF(2^14)
  - Finds roots of error-locator polynomial.

28.2.2 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.

28.2.2.1 8-bit NAND Flash

Hardware Configuration



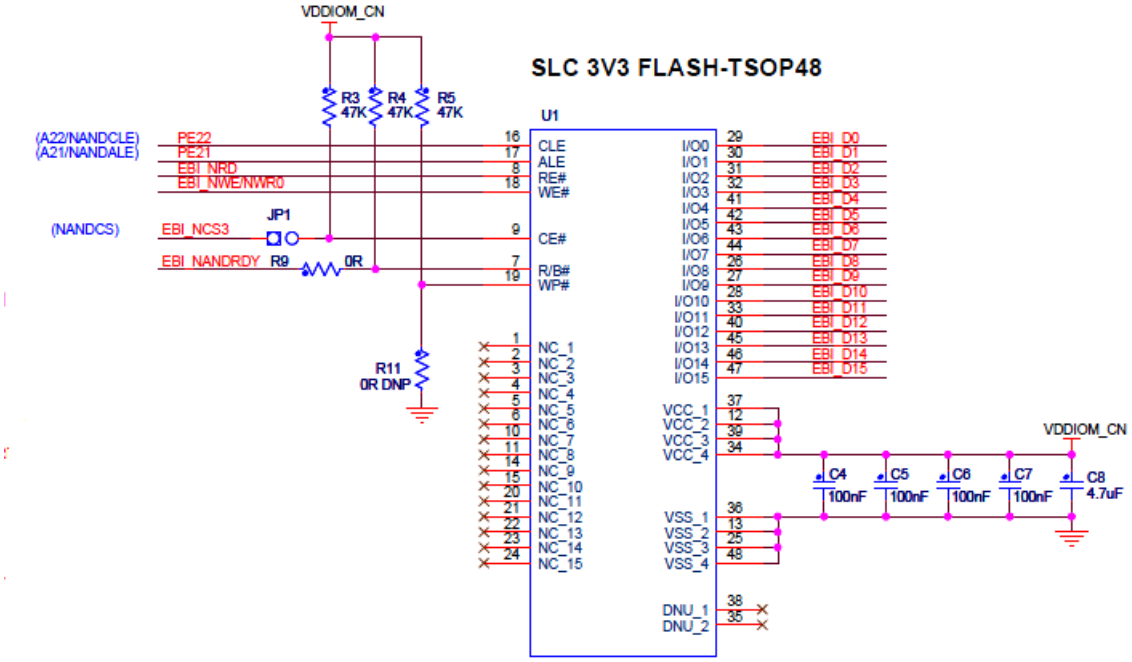
Software Configuration

The following configuration has to be performed:

- Select the NAND Flash Chip Select by setting the field CSID in NFCADDR_CMD register.
- Configure the NFC and HSMC according to the used NAND Flash.
- Enable the NFC with NFCEN bit in HSMC_CTRL register in HSMC User interface.
- 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.

28.2.2.2 16-bit NAND Flash

Hardware Configuration

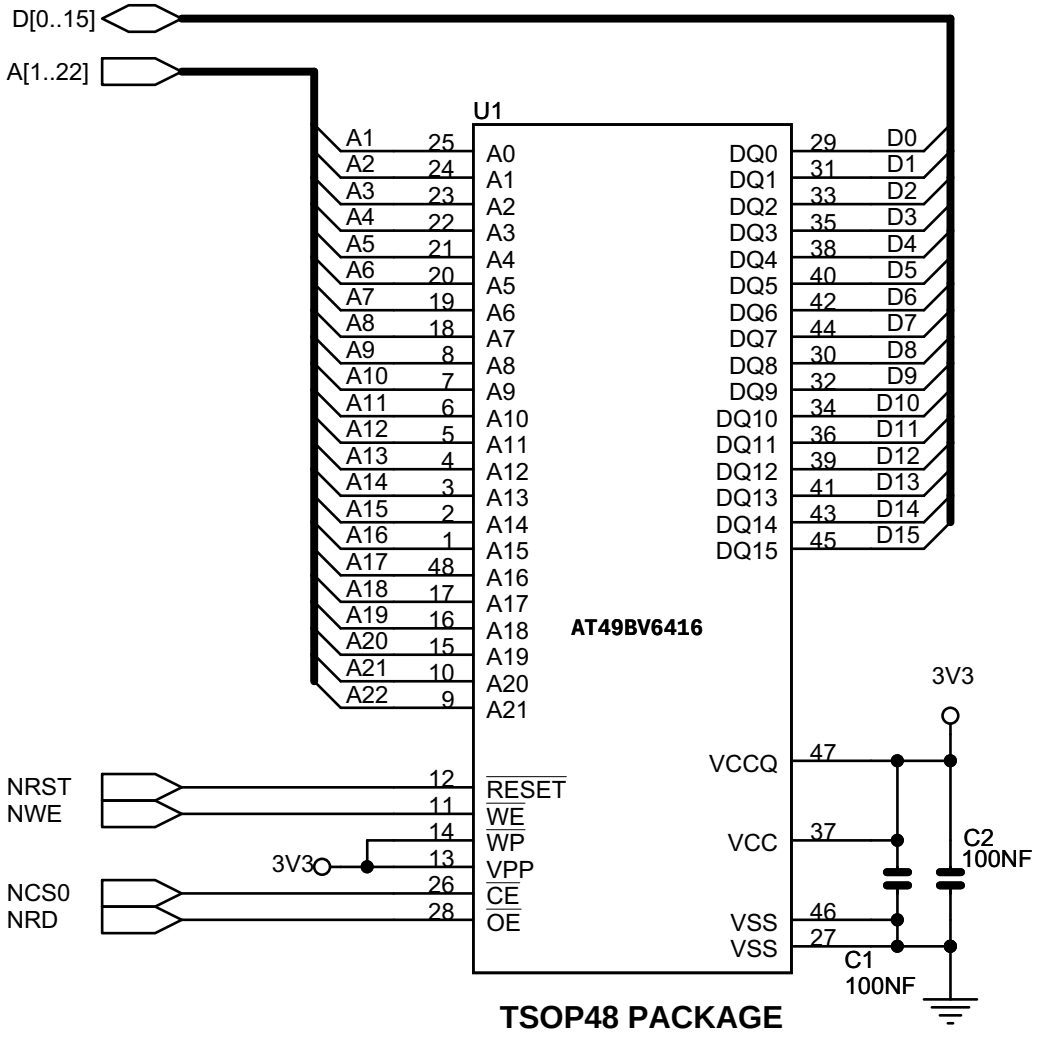


Software Configuration

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.

28.2.2.3 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.

29. Multi-port DDR-SDRAM Controller (MPDDRC)

29.1 Description

The Multi-port DDR-SDRAM Controller (MPDDRC) is a multi-port 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 DDR-SDRAM protocol.

The MPDDRC extends the memory capabilities of a chip by providing the interface to the external 16-bit or 32-bit DDR-SDRAM device. The page size support ranges from 2048 to 16384 and the number of columns from 256 to 4096. It supports dword (64-bits), word (32-bit), half-word (16-bit), and byte (8-bit) accesses.

The MPDDRC supports a read or write burst length of eight locations. This enables the command and address bus to anticipate the next command, thus reducing latency imposed by the DDR-SDRAM protocol and improving the DDR-SDRAM bandwidth. Moreover, MPDDRC keeps track of the active row in each bank, thus maximizing DDR-SDRAM performance, e.g., the application may be placed in one bank and data in other banks. To optimize performance, avoid accessing different rows in the same bank. The MPDDRC supports CAS latency of 2, 3 and optimizes the read access depending on the frequency.

Self-refresh, power-down and deep power-down modes minimize the consumption of the DDR-SDRAM device.

OCD (Off-chip Driver) and ODT (On-die Termination) modes are not supported.

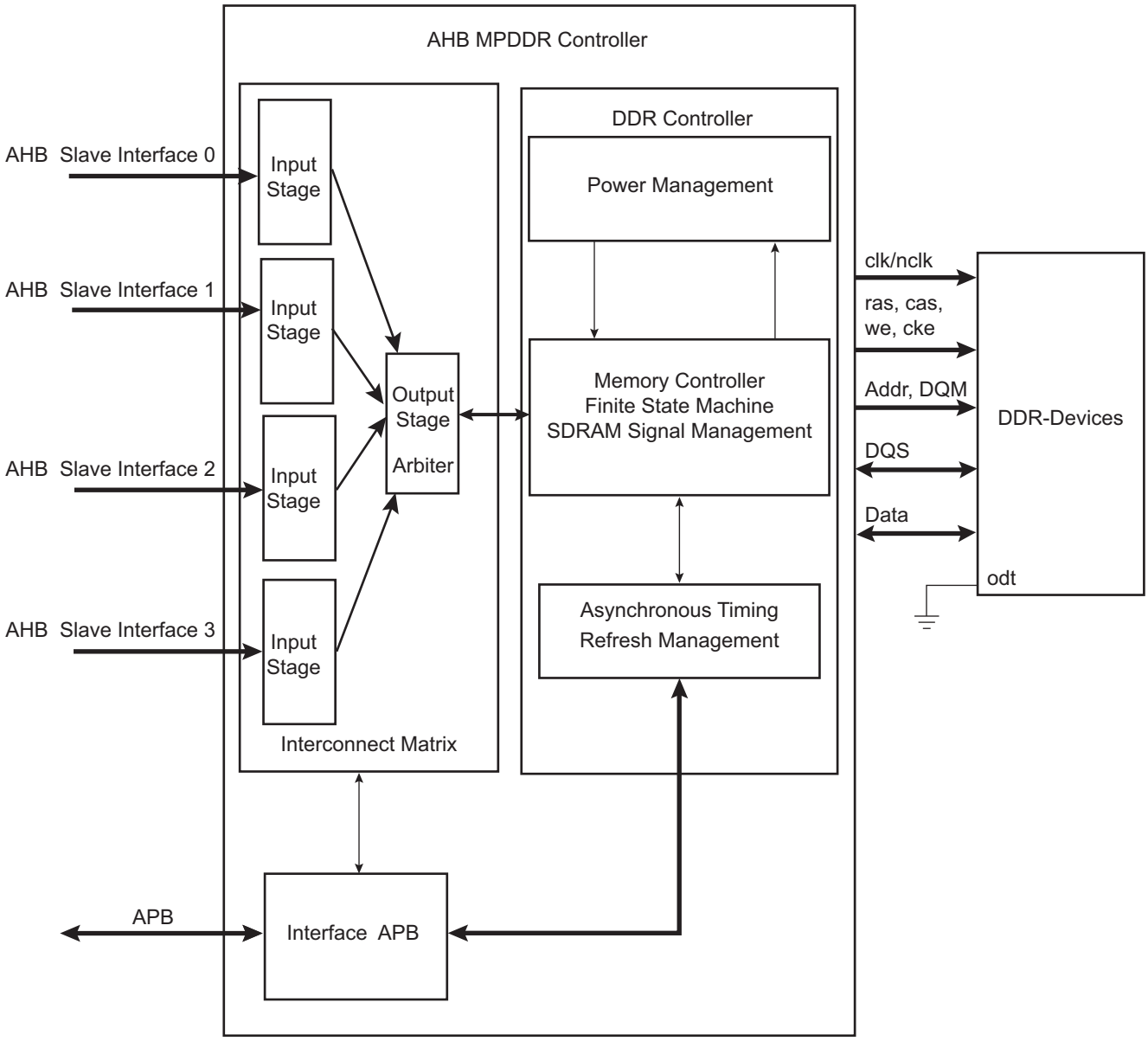
29.2 Embedded Characteristics

- Four advanced high performance bus (AHB) interfaces, management of all accesses maximizes memory bandwidth and minimizes transaction latency
- Bus transfer: word, half word, byte access
- Bus transfer: dword, word, half word, byte access
- Supports low-power DDR2-SDRAM-S4 (LPDDR2), DDR2-SDRAM, low-power DDR1-SDRAM (LPDDR1)
- Numerous configurations supported
  - 2K, 4K, 8K, 16K row address memory parts
  - DDR-SDRAM with four or eight internal banks (DDR2-SDRAM/ low-power DDR2-SDRAM-S4)
  - DDR-SDRAM with 16-bit data path for system oriented word access
  - DDR-SDRAM with 32-bit data path for system oriented dword access
  - One chip select for SDRAM device (512-Mbyte address space)
  - One chip select for SDRAM device (256-Mbyte address space)
- Programming Facilities
  - Multibank ping-pong access (up to four or eight banks opened at the same time = reduced 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 DDR-SDRAM devices)
- Energy-saving capabilities
  - Self-refresh, Power-down, active power-down and deep power-down modes supported
- DDR-SDRAM power-up initialization by software
- CAS latency of 2,3 supported
- Reset function supported (DDR2-SDRAM)
- Auto-refresh per bank supported (low-power DDR2-SDRAM-S4)
- Automatic adjust refresh rate (low-power DDR2-SDRAM-S4)
- Auto-precharge command not used
- OCD (Off-chip Driver) mode, ODT (On-die Termination) are not supported

- Dynamic Scrambling with user key (no impact on bandwidth)

29.3 MPDDRC Module Diagram

Figure 29-1. MPDDRC Module Diagram



MPDDRC is partitioned in two blocks (see Figure 29-1):

- the Interconnect Matrix block that manages concurrent accesses on the AHB bus between four AHB masters and integrates an arbiter
- the DDR controller that translates AHB requests (read/write) in the DDR-SDRAM protocol

29.4 Product Dependencies, Initialization Sequence

29.4.1 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 in the MPDDRC Memory Device Register (MPDDRC_MD).
2. Program the features of the low-power DDR1-SDRAM device in the MPDDRC Configuration Register (number of columns, rows, banks, CAS latency and output drive strength) and in the MPDDRC Timing Parameter 0 Register/MPDDRC Timing Parameter 1 Register (asynchronous timing (TRC, TRAS, etc.)).
3. Program Temperature Compensated Self-refresh (TCR), Partial Array Self-refresh (PASR) and Drive Strength (DS) parameters in the MPDDRC Low-power Register.
4. A minimum pause of 200 µs is provided to precede any signal toggle.
5. A NOP command is issued to the low-power DDR1-SDRAM. Program the NOP command in the MPDDRC Mode Register (MPDDRC_MR). The application must write a one to the MODE field in the MPDDRC_MR. Perform a write access to any low-power DDR1-SDRAM address to acknowledge this command. The clocks which drive the low-power DDR1-SDRAM device are now enabled.
6. A NOP command is issued to the low-power DDR1-SDRAM. Program the NOP command in the MPDDRC_MR. The application must write a one to the MODE field in the MPDDRC_MR. Perform a write access to any low-power DDR1-SDRAM address to acknowledge this command. A calibration request is now made to the I/O pad.
7. An All Banks Precharge command is issued to the low-power DDR1-SDRAM. Program All Banks Precharge command in the MPDDRC_MR. The application must write a two to the MODE field in the MPDDRC_MR. Perform a write access to any low-power DDR1-SDRAM address to acknowledge this command.
8. Two auto-refresh (CBR) cycles are provided. Program the Auto Refresh command (CBR) in the MPDDRC_MR. The application must write a four to the MODE field in the MPDDRC_MR. Perform a write access to any low-power DDR1-SDRAM location twice to acknowledge these commands.
9. An Extended Mode Register set (EMRS) cycle is issued to program the low-power DDR1-SDRAM parameters (TCSR, PASR, DS). The application must write a five to the MODE field in the MPDDRC_MR and perform a write access to the SDRAM to acknowledge this command. The write address must be chosen so that signal BA[1] is set to 1 and BA[0] is set to 0. For example, with a 16-bit 128 Mbits SDRAM (12 rows, 9 columns, 4 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x00800000. For example, with a 32-bit 1 Gbit SDRAM (14 rows, 10 columns, 4 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x08000000.

Note: This address is given as an example only. The real address depends on implementation in the product.

10. A Mode Register set (MRS) cycle is issued to program parameters of the DDR1-SDRAM devices, in particular CAS latency and burst length. The application must write a three to the MODE field in the MPDDRC_MR and perform a write access to the low-power DDR1-SDRAM to acknowledge this command. The write address must be chosen so that signals BA[1:0] are set to 0. For example, the SDRAM write access should be done at the address: BASE_ADDRESS_DDR.
11. The application must enter Normal mode, write a zero to the MODE field in the MPDDRC_MR and perform a write access at any location in the low-power DDR1-SDRAM to acknowledge this command.
12. Perform a write access to any low-power DDR1-SDRAM address.
13. Write the refresh rate into the COUNT field in the MPDDRC Refresh Timer Register (MPDDRC_RTR): 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, MPDDRC_RTR must be set with (15.625 * 100 MHz) = 1562 i.e., 0x061A or (7.81 * 100 MHz) = 781 i.e., 0x030D.

After initialization, the low-power DDR1-SDRAM device is fully functional.

29.4.2 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 in the MPDDRC Memory Device Register.
2. Program features of the DDR2-SDRAM device in the MPDDRC Configuration Register (number of columns, rows, banks, CAS latency and output driver impedance control) and in the MPDDRC Timing Parameter 0 Register/MPDDRC Timing Parameter 1 Register (asynchronous timing (TRC, TRAS, etc.).
3. A NOP command is issued to the DDR2-SDRAM. Program the NOP command in the MPDDRC Mode Register (MPDDRC_MR). The application must write a one to the MODE field in the MPDDRC_MR. Perform a write access to any DDR2-SDRAM address to acknowledge this command. The clocks which drive the DDR2-SDRAM device are now enabled.
4. A minimum pause of 200 µs is provided to precede any signal toggle.
5. A NOP command is issued to the DDR2-SDRAM. Program the NOP command in the MPDDRC_MR. The application must write a one to the MODE field in the MPDDRC_MR. Perform a write access to any DDR2-SDRAM address to acknowledge this command. CKE is now driven high.
6. An All Banks Precharge command is issued to the DDR2-SDRAM. Program All Banks Precharge command in the MPDDRC_MR. The application must write a two to the MODE field in the MPDDRC_MR. Perform a write access to any DDR2-SDRAM address to acknowledge this command.
7. An Extended Mode Register set (EMRS2) cycle is issued to choose between commercial or high temperature operations. The application must write a five to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signal BA[1] is set to 1 and signal BA[0] is set to 0. For example, with a 16-bit 128 Mbits DDR2-SDRAM (12 rows, 9 columns, 4 banks), the DDR2-SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x00800000. For example, with a 32-bit 1 Gbit SDRAM (14 rows, 10 columns, 8 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x08000000.

Note: This address is given as an example only. The real address depends on implementation in the product.

8. An Extended Mode Register set (EMRS3) cycle is issued to set the Extended Mode Register to 0. The application must write a five to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signal BA[1] is set to 1 and signal BA[0] is set to 1. For example, with a 16-bit 128 Mbits DDR2-SDRAM (12 rows, 9 columns, 4 banks), the DDR2-SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x00C00000. For example, with a 32-bit 1 Gbit SDRAM (14 rows, 10 columns, 8 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x0C000000.
9. An Extended Mode Register set (EMRS1) cycle is issued to enable DLL and to program D.I.C. (Output Driver Impedance Control). The application must write a five to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signal BA[1] is set to 0 and signal BA[0] is set to 1. For example, with a 16-bit 128 Mbits DDR2-SDRAM (12 rows, 9 columns, 4 banks), the DDR2-SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x00400000. For example, with a 32-bit 1 Gbit SDRAM (14 rows, 10 columns, 8 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x04000000.
10. An additional 200 cycles of clock are required for locking DLL
11. Write a one to the DLL bit (enable DLL reset) in the MPDDRC Configuration Register (MPDDRC_CR).
12. A Mode Register set (MRS) cycle is issued to reset DLL. The application must write a three to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signals BA[1:0] are set to 0. For example, the SDRAM write access should be done at the address: BASE_ADDRESS_DDR.
13. An All Banks Precharge command is issued to the DDR2-SDRAM. Program the All Banks Precharge command in the MPDDRC_MR. The application must write a two to the MODE field in the MPDDRC_MR. Perform a write access to any DDR2-SDRAM address to acknowledge this command.

14. Two auto-refresh (CBR) cycles are provided. Program the Auto Refresh command (CBR) in the MPDDRC_MR. The application must write a four to the MODE field in the MPDDRC_MR. Perform a write access to any DDR2-SDRAM location twice to acknowledge these commands.
15. Write a zero to the DLL bit (disable DLL reset) in the MPDDRC_CR.
16. A Mode Register set (MRS) cycle is issued to program parameters of the DDR2-SDRAM device, in particular CAS latency and burst length, and to disable DLL reset. The application must write a three to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signals BA[1:0] are set to 0. For example, with a 16-bit 128 Mbits SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address: BASE_ADDRESS_DDR.
17. Write a seven to the OCD field (default OCD calibration) in the MPDDRC_CR.
18. An Extended Mode Register set (EMRS1) cycle is issued to the default OCD value. The application must write a five to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signal BA[1] is set to 0 and signal BA[0] is set to 1. For example, with a 16-bit 128 Mbits DDR2-SDRAM (12 rows, 9 columns, 4 banks), the DDR2-SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x00400000. For example, with a 32-bit 1 Gbit SDRAM (14 rows, 10 columns, 8 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x04000000.
19. Write a zero to the OCD field (exit OCD calibration mode) in the MPDDRC_CR.
20. An Extended Mode Register set (EMRS1) cycle is issued to enable OCD exit. The application must write a five to the MODE field in the MPDDRC_MR and perform a write access to the DDR2-SDRAM to acknowledge this command. The write address must be chosen so that signal BA[1] is set to 0 and signal BA[0] is set to 1. For example, with a 16-bit 128 Mbits DDR2-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR2-SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x00400000. For example, with a 32-bit 1 Gbit SDRAM (14 rows, 10 columns, 8 banks), the SDRAM write access should be done at the address: BASE_ADDRESS_DDR + 0x04000000.
21. A Normal mode command is provided. Program the normal mode in the MPDDRC_MR and perform a write access to any DDR2-SDRAM address to acknowledge this command.
22. Perform a write access to any DDR2-SDRAM address.
23. Write the refresh rate into the COUNT field in the MPDDRC Refresh Timer Register (MPDDRC_RTR): 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 COUNT field in the MPDDRC_RTR must 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.

29.4.3 Low-power DDR2-SDRAM Initialization

The initialization sequence is generated by software. The low-power DDR2-SDRAM devices are initialized by the following sequence:

1. Program the memory device type in the MPDDRC Memory Device Register.
2. Program features of the low-power DDR2-SDRAM device into and in the MPDDRC Configuration Register (number of columns, rows, banks, CAS latency and output drive strength) and in the MPDDRC Timing Parameter 0 Register/MPDDRC Timing Parameter 0 Register (asynchronous timing, TRC, TRAS, etc.).
3. A NOP command is issued to the low-power DDR2-SDRAM. Program the NOP command in the MPDDRC Mode Register (MPDDRC_MR). The application must write a one to the MODE field in the MPDDRC_MR. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The clocks which drive the Low-power DDR2-SDRAM devices are now enabled.
4. A minimum pause of 100 ns must be observed to precede any signal toggle.
5. A NOP command is issued to the low-power DDR2-SDRAM. Program the NOP command in the MPDDRC_MR. The application must write a one to the MODE field in the MPDDRC_MR. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. CKE is now driven high.

6. A minimum pause of 200 µs must be satisfied before a Reset command.
7. A Reset command is issued to the low-power DDR2-SDRAM. In the MPDDRC_MR, configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and a 63 to the MRS field. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The Reset command is now issued.
8. A minimum pause of 1 µs must be satisfied before any commands.
9. A Mode Register Read command is issued to the low-power DDR2-SDRAM. In the MPDDRC_MR, configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and must write a zero to the MRS field. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The Mode Register Read command is now issued.
10. A minimum pause of 10 µs must be satisfied before any commands.
11. A Calibration command is issued to the low-power DDR2-SDRAM. Program the type of calibration in the MPDDRC Configuration Register (MPDDRC_CR): set the ZQ field to the RESET value. In the MPDDRC_MR, configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and a 10 to the MRS field. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The ZQ Calibration command is now issued. Program the type of calibration in the MPDDRC_CR: set the ZQ field to the SHORT value.
12. A Mode Register Write command is issued to the low-power DDR2-SDRAM. In the MPDDRC_MR, configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and a one to the MRS field. The Mode Register Write command cycle is issued to program parameters of the low-power DDR2-SDRAM device, in particular burst length. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The Mode Register Write command is now issued.
13. A Mode Register Write command is issued to the low-power DDR2-SDRAM. In the MPDDRC_MR, configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and a two to the MRS field. The Mode Register Write command cycle is issued to program parameters of the low-power DDR2-SDRAM device, in particular CAS latency. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The Mode Register Write command is now issued.
14. A Mode Register Write command is issued to the low-power DDR2-SDRAM. In the MPDDRC_MR, configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and a three to the MRS field. The Mode Register Write command cycle is issued to program parameters of the low-power DDR2-SDRAM device, in particular Drive Strength and Slew Rate. Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The Mode Register Write command is now issued.
15. A Mode Register Write command is issued to the low-power DDR2-SDRAM. In the MPDDRC_MR configure the MODE field to the LPDDR2_CMD value and configure the MRS field. The application must write a seven to the MODE field and a 16 to the MRS field. Mode Register Write command cycle is issued to program parameters of the low-power DDR2-SDRAM device, in particular Partial Array Self Refresh (PASR). Perform a write access to any low-power DDR2-SDRAM address to acknowledge this command. The Mode Register Write command is now issued.
16. Write the refresh rate into the COUNT field in the MPDDRC Refresh Timer Register (MPDDRC_RTR): refresh rate = delay between refresh cycles. The low-power DDR2-SDRAM device requires a refresh every 7.81 µs. With a 133 MHz frequency, the COUNT field in the MPDDRC_RTR must be set with (7.81 * 133 MHz) = 1039 i.e., 0x040F.

After initialization, the low-power DDR2-SDRAM devices are fully functional.

29.5 Functional Description

29.5.1 DDR-SDRAM Controller Write Cycle

The MPDDRC provides burst access or single access in normal mode (MODE = 0). Whatever the access type, the MPDDRC keeps track of the active row in each bank, thus maximizing performance.

The DDR-SDRAM device is programmed with a burst length (bl) 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 case of the DDR1-SDRAM devices, and it is fixed to 2 in case of DDR2-SDRAM depending on the programmed latency.

To initiate a single access, the MPDDRC 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 DDR-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 case of single access, it has to stop the burst, otherwise seven invalid values may be written. In case of the DDR-SDRAM device, the burst stop command is not supported for the burst write operation. Thus, in order to interrupt the write operation, the DM (data mask) input signal must be set to 1 to mask invalid data (see Figure 29-2 on page 357 and Figure 29-4 on page 358), and DQS must continue to toggle.

To initiate a burst access, the MPDDRC controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the DDR-SDRAM device is carried out. If the next access is a write non-sequential access, then an automatic access break is inserted, the MPDDRC generates a precharge command, activates the new row and initiates a write command. To comply with DDR-SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/write (t_{RCD}) commands.

For the definition of timing parameters, please refer to Section 29.8.4 “MPDDRC Timing Parameter 0 Register”.

Write accesses to the DDR-SDRAM device 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, eight columns are selected. All accesses for that burst take place within these eight columns, thus the burst wraps within these eight columns if a boundary is reached. These eight columns are selected by addr[13:3]. addr[2:0] is used to select the starting location within the block.

In case of incrementing burst (INCR/INCR4/INCR8/INCR16), the addresses can cross the 16-byte boundary of the DDR-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 at 0x00. Since the boundary is reached, the burst is wrapped. The MPDDRC takes this feature of the DDR-SDRAM device into account. In case of a transfer starting at address 0x04/0x08/0x0C or starting at address 0x10/0x14/0x18/0x1C, two write commands are issued to avoid wrapping 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 the write access is not done. This avoids additional writing.

Figure 29-2. Single Write Access, Row Closed, DDR-SDRAM Devices

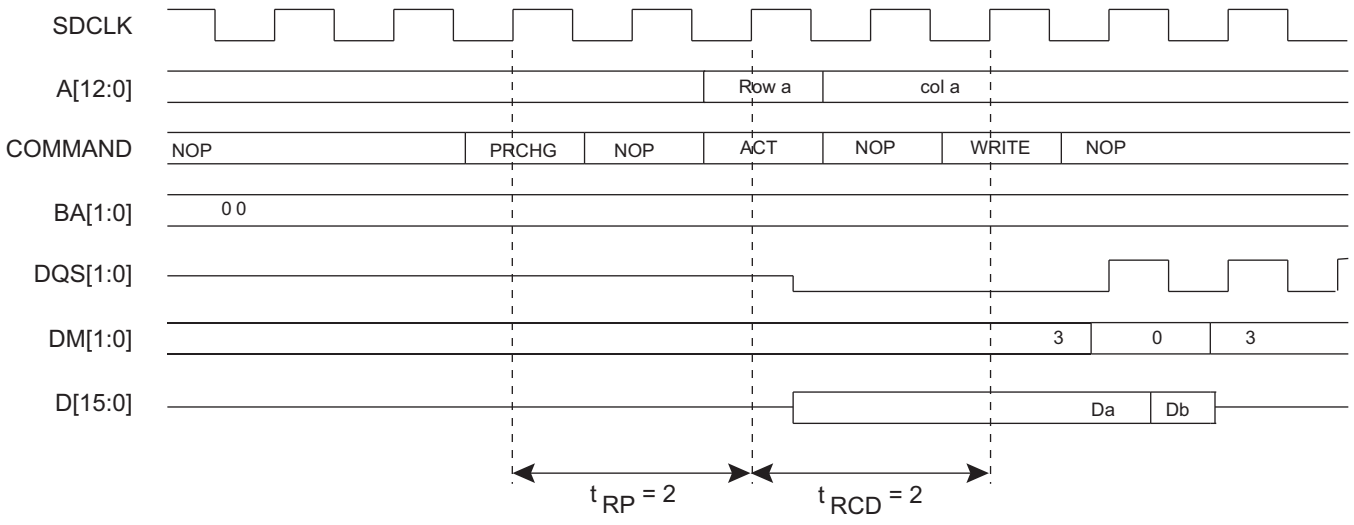


Figure 29-3. Single Write Access, Row Closed, DDR2-SDRAM Devices

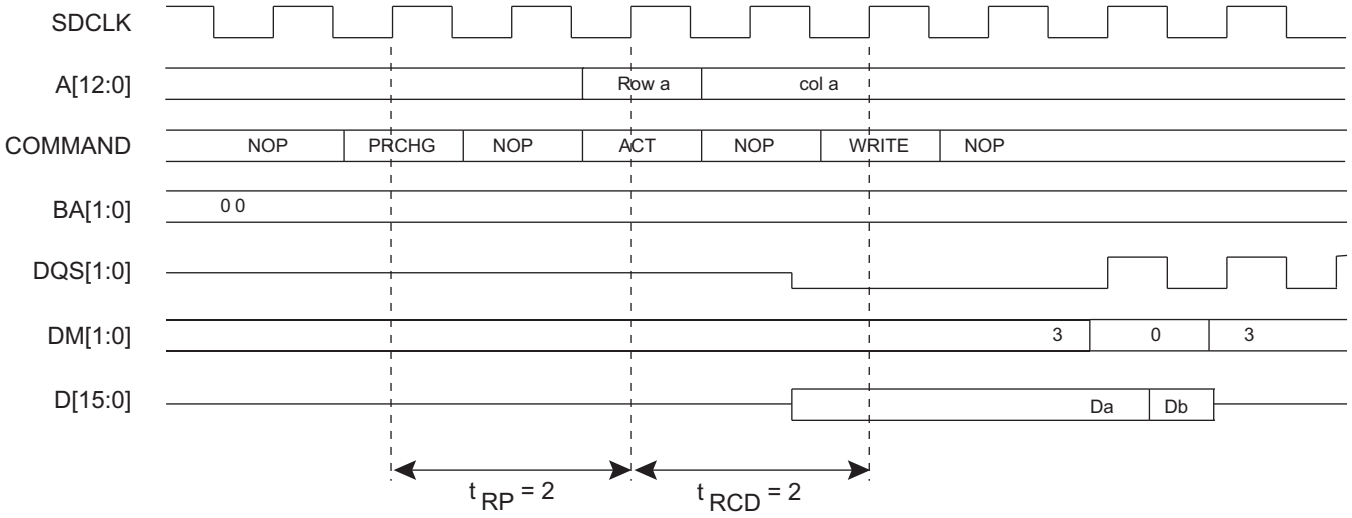


Figure 29-4. Burst Write Access, Row Closed, DDR-SDRAM Devices

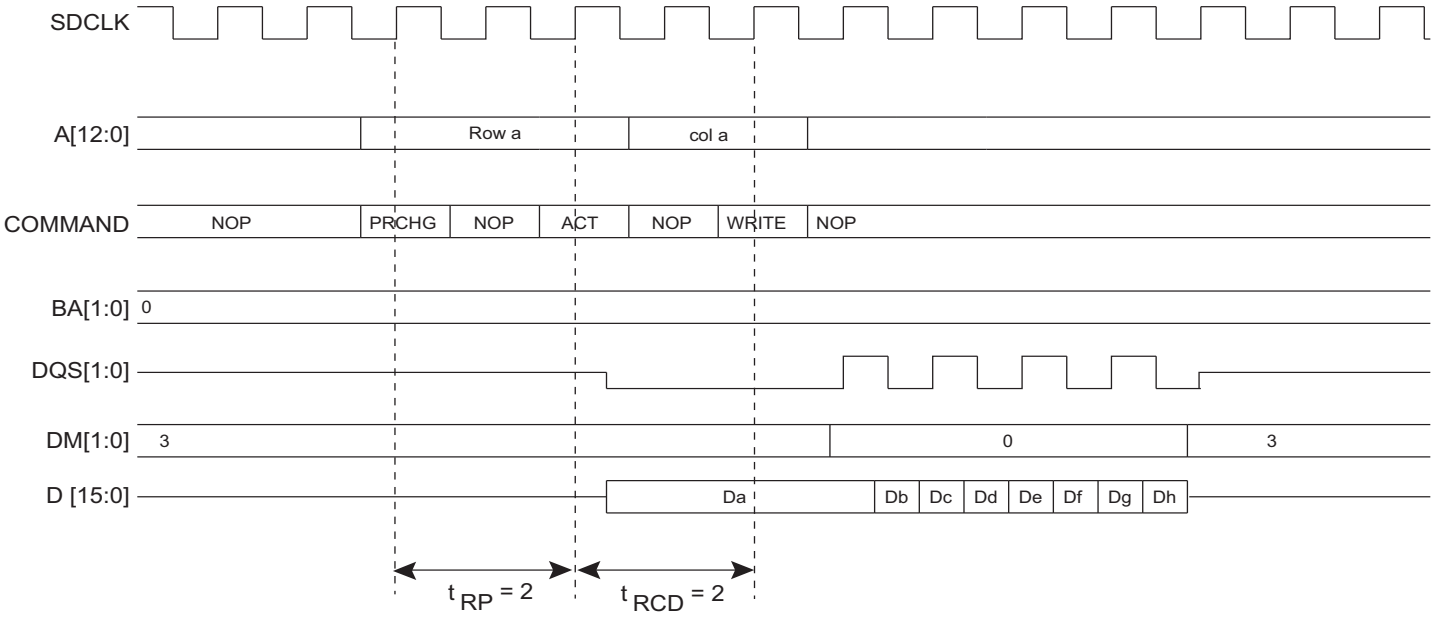
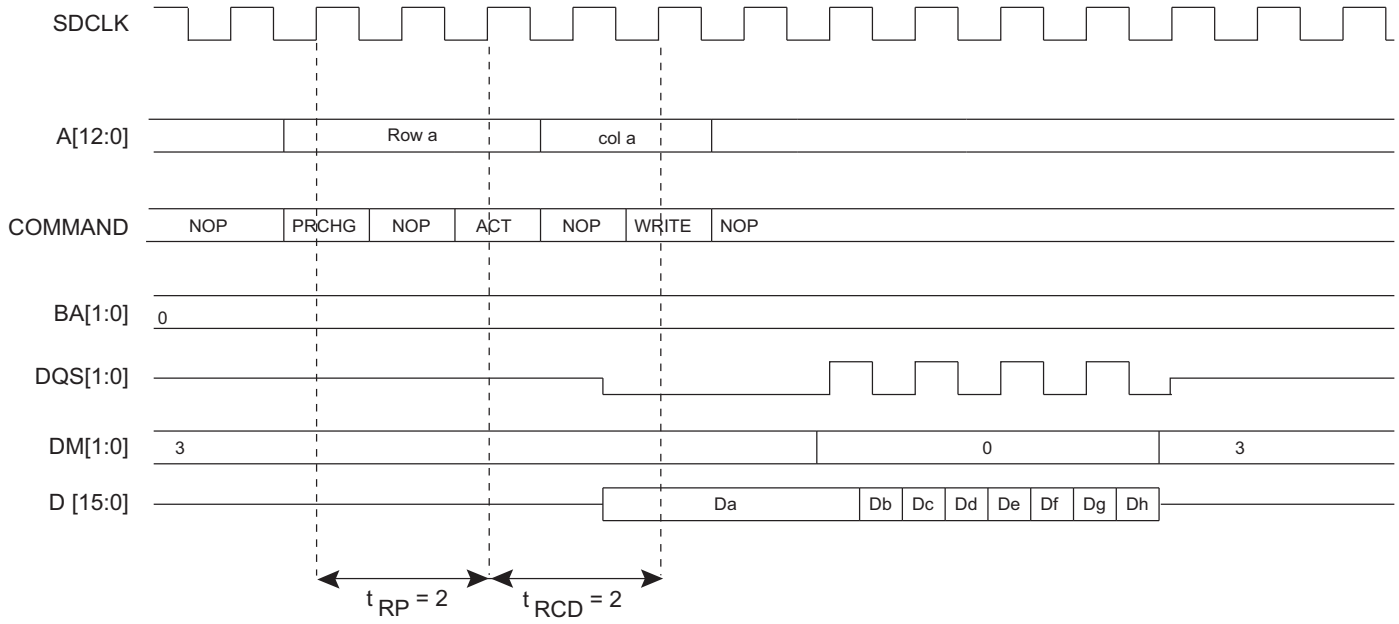
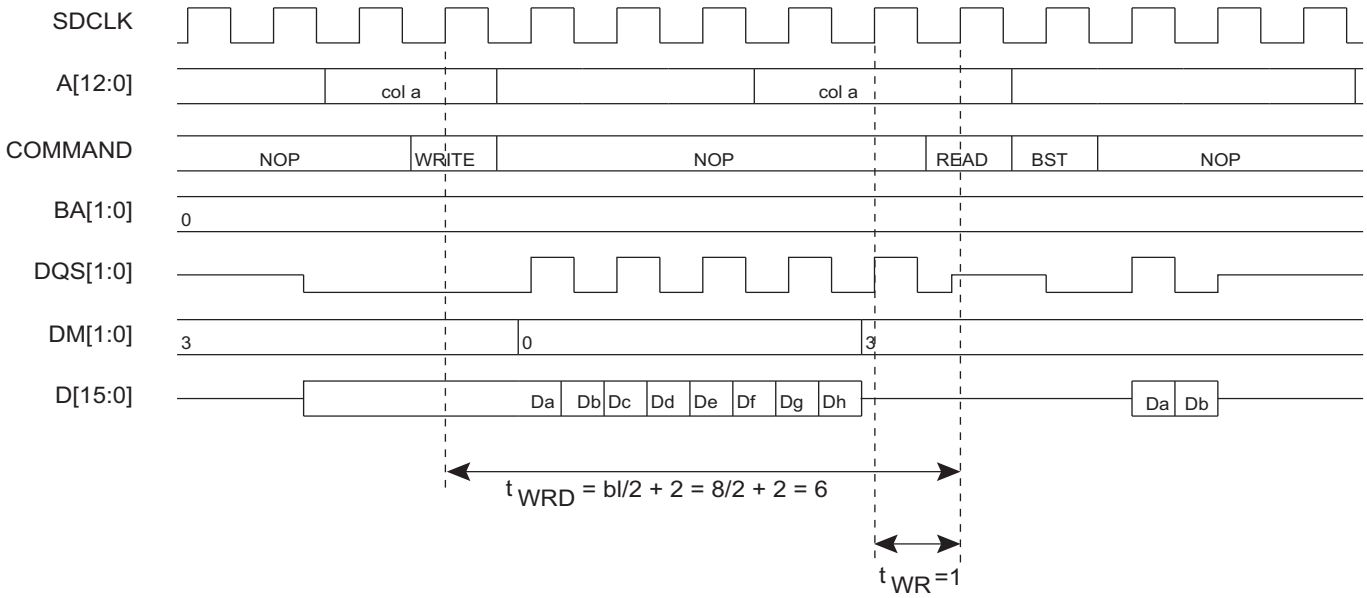


Figure 29-5. Burst Write Access, Row Closed, DDR2-SDRAM Devices



A write command can be followed by a read command. To avoid breaking the current write burst, t_{WTR}/t_{WRD} (bl/2 + 2 = 6 cycles) should be met. See Figure 29-6 on page 359.

Figure 29-6. Write Command Followed by a Read Command without Burst Write Interrupt, DDR-SDRAM Devices



In 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 29-7 on page 360.

Figure 29-7. SINGLE Write Access followed by a Read Access, DDR-SDRAM Devices

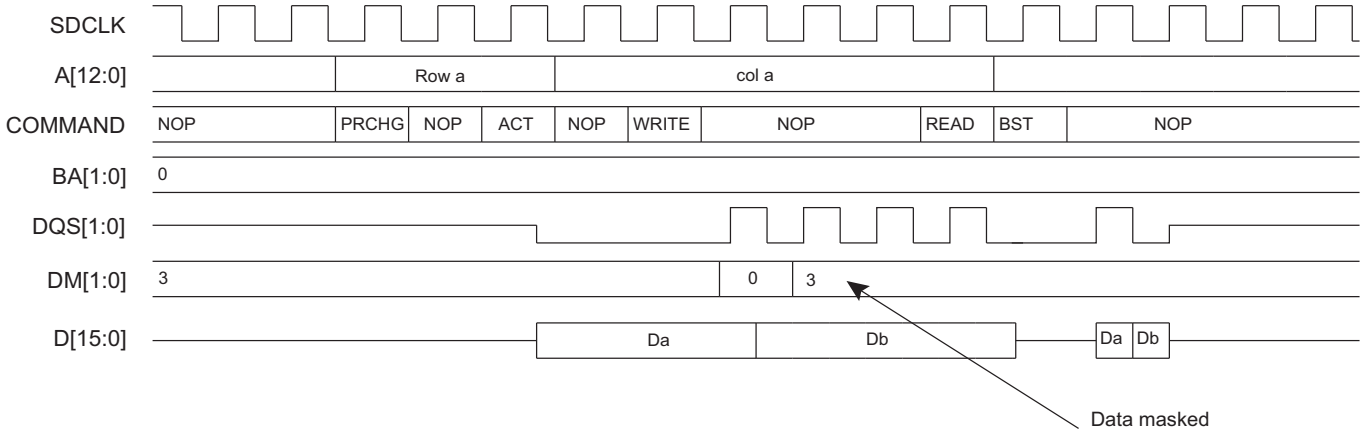
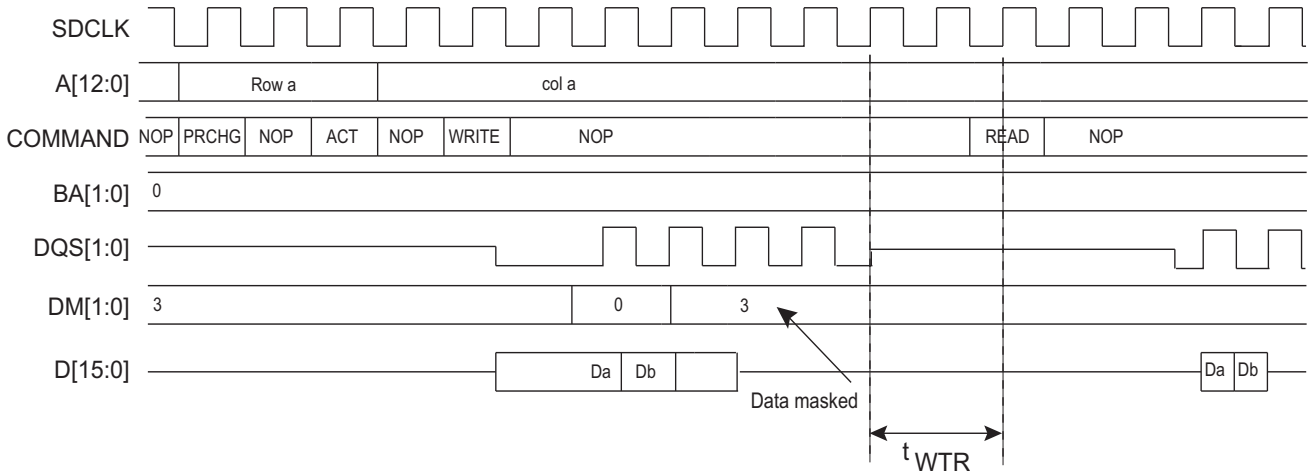


Figure 29-8. SINGLE Write Access followed by a Read Access, DDR2-SDRAM Devices



29.5.2 DDR-SDRAM Controller Read Cycle

The MPDDRC provides burst access or single access in normal mode (mode = 0). Whatever the access type, the MPDDRC keeps track of the active row in each bank, thus maximizing performance of the MPDDRC.

The DDR-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, 3. This value is programmed during the initialization phase (see Section 29.4 “Product Dependencies, Initialization Sequence”).

To initiate a single access, the MPDDRC 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 DDR-SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/read (t_{RCD}) command. After a read command, additional wait states are generated to comply with CAS latency. The MPDDRC supports a CAS latency of two to three (2 to 3 clocks delay). As the burst length is fixed to 8, in case of a single access or a burst access inferior to 8 data requests, it has to stop the burst, otherwise an additional seven or X values could be read. The Burst Stop command (BST) is used to stop output during a burst read. If the DDR2-SDRAM Burst Stop command is not supported by the JEDEC standard, in a single read access, an additional seven unwanted data will be read.

To initiate a burst access, the MPDDRC 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. If page access is already open, a read command is generated.

To comply with DDR-SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/read (t_{RCD}) commands. The MPDDRC supports a CAS latency of two to three (2 to 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, the MPDDRC anticipates two to three read accesses. In 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 depending on the latency, the MPDDRC anticipates two to three read accesses.

For the definition of timing parameters, refer to Section 29.8.3 “MPDDRC Configuration Register”.

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

In case of incrementing burst (INCR/INCR4/INCR8/INCR16), the addresses can cross the 16-byte boundary of the DDR-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 MPDDRC takes into account this feature of the SDRAM device. In case of the DDR-SDRAM device, transfers start at address 0x04/0x08/0x0C. 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).

To avoid additional reading, it is possible to use the burst stop command to truncate the read burst and to decrease power consumption. The DDR2-SDRAM devices do not support the burst stop command.

Figure 29-9. Single Read Access, Row Closed, Latency = 2, DDR-SDRAM Devices

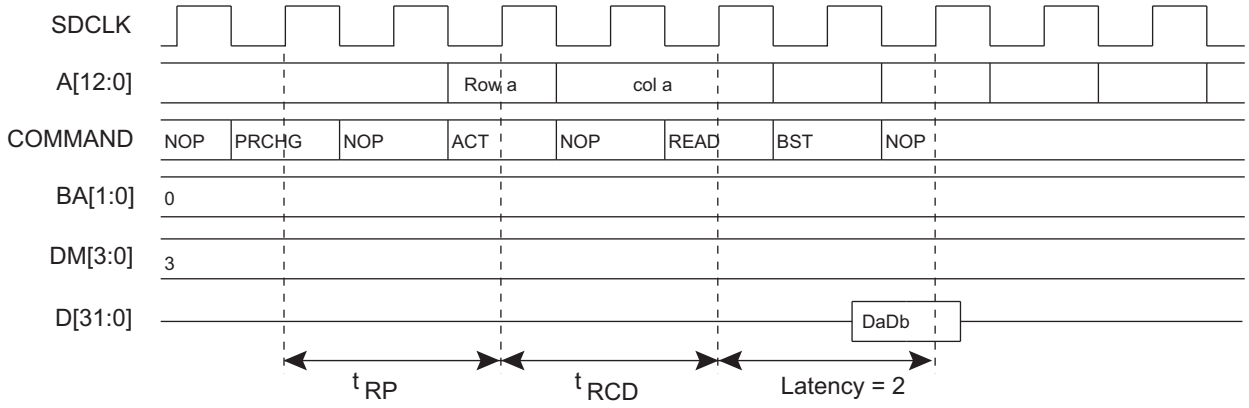


Figure 29-10.Single Read Access, Row Closed, Latency = 3, DDR2-SDRAM Devices

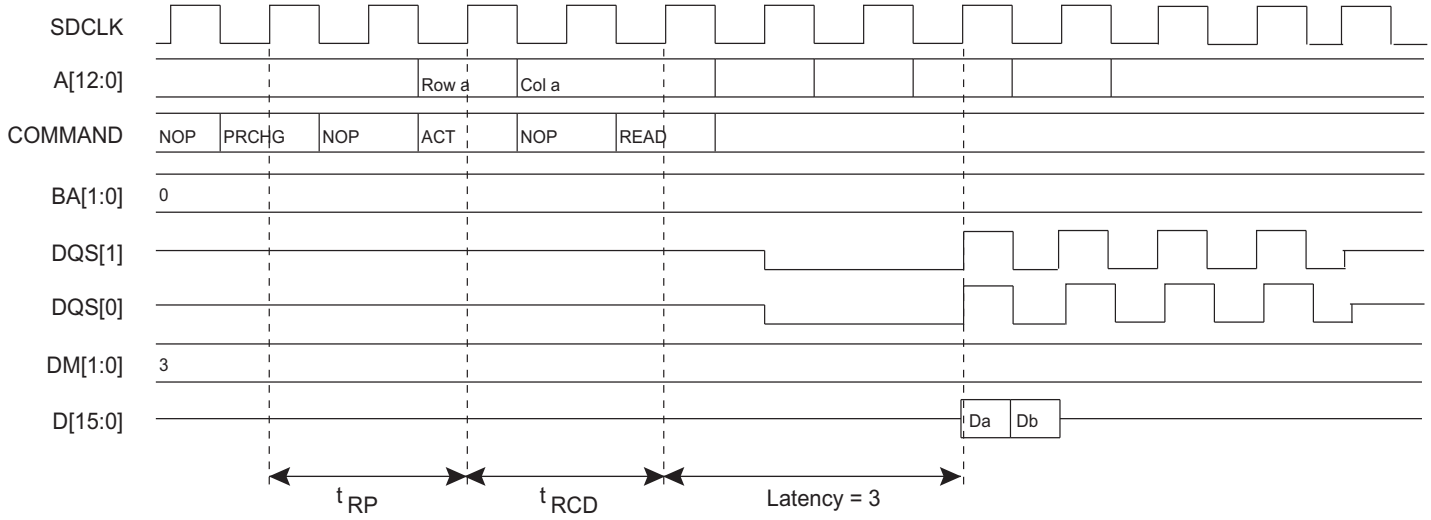


Figure 29-11.Burst Read Access, Latency = 2, DDR-SDRAM Devices

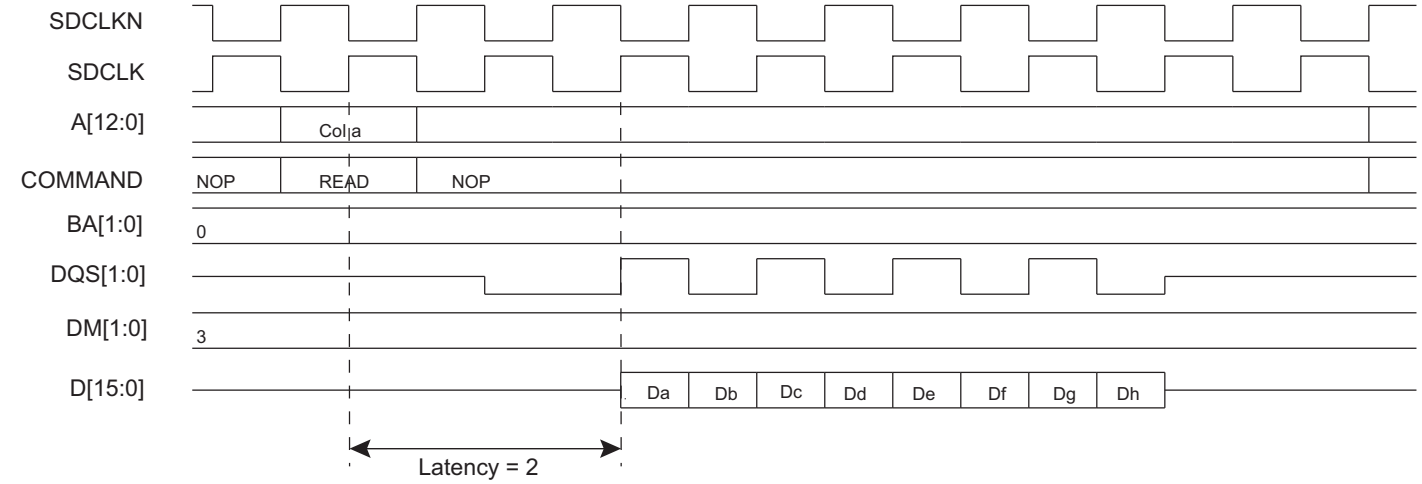
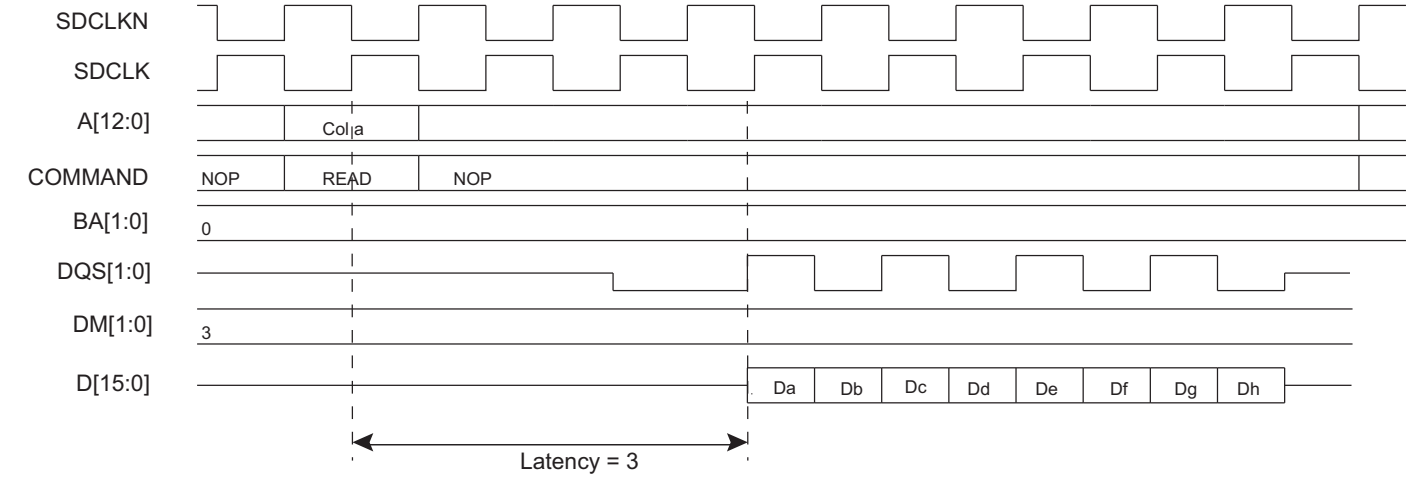


Figure 29-12.Burst Read Access, Latency = 3, DDR2-SDRAM Devices



29.5.2.1 All Banks Auto Refresh

The All Banks Auto Refresh command performs a refresh operation on all banks. An auto refresh command is used to refresh the MPDDRC. Refresh addresses are generated internally by the DDR-SDRAM device and incremented after each auto-refresh automatically. The MPDDRC generates these auto-refresh commands periodically. A timer is loaded in the MPDDRC_RTR with the value that indicates the number of clock cycles between refresh cycles (see Section 29.8.2 “MPDDRC Refresh Timer Register”). When the MPDDRC initiates a refresh of the DDR-SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the DDR-SDRAM device, the slave indicates that the device is busy. A refresh request does not interrupt a burst transfer in progress. This feature is activated by setting Per-bank Refresh bit (REF_PB) to 0 in the MPDDRC_RTR (see Section 29.8.2 “MPDDRC Refresh Timer Register”).

29.5.2.2 Per-bank Auto Refresh

The low-power DDR2-SDRAM embeds a new Per-bank Refresh command which performs a refresh operation on the bank scheduled by the bank counter in the memory device. The Per-bank Refresh command is executed in a fixed sequence order of round-robin type: “0-1-2-3-4-5-6-7-0-1-...”. The bank counter is automatically cleared upon issuing a RESET command or when exiting from self-refresh mode, in order to ensure the synchronism between SDRAM memory device and the MPDDRC controller. The bank addressing for the Per-bank Refresh count is the same as established in the Single-bank Precharge command. This feature is activated by setting the Per-bank Refresh bit (REF_PB) to 1 in the MPDDRC_RTR (see Section 29.8.2 “MPDDRC Refresh Timer Register”). This feature masks the latency due to the refresh procedure. The target bank is inaccessible during the Per-bank Refresh cycle period (t_{RFCpb}), however other banks within the device are accessible and may be addressed during the “Per-bank Refresh” cycle. During the REFpb operation, any bank other than the one being refreshed can be maintained in active state or accessed by a read or a write command. When the “Per-bank Refresh” cycle is completed, the affected bank will be in idle state.

29.5.2.3 Adjust Auto Refresh Rate

The low-power DDR2-SDRAM embeds an internal register, Mode Register 19 (Refresh Mode). The content of this register allows to adjust the interval of auto-refresh operations according to temperature variation. This feature is activated by setting the Adjust Refresh bit [ADJ_REF] to 1 in the MPDDRC_RTR (see Section 29.8.2 “MPDDRC Refresh Timer Register”). When this feature is enabled, a mode register read command (MRR) is performed every 16 * t_{REFI} (average time between REFRESH commands). Depending on the read value, the auto refresh interval will be modified. In case of high temperature, the interval is reduced and in case of low temperature, the interval is increased.

29.5.3 Power Management

29.5.3.1 Self-refresh Mode

This mode is activated by writing a one to the Low-power Command bit (LPCB) in the MPDDRC_LPR register.

Self-refresh mode is used in power-down mode, i.e., when no access to the DDR-SDRAM device is possible. In this case, power consumption is very low. In self-refresh mode, the DDR-SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto refresh cycles. During self-refresh period CKE is driven low. As soon as the DDR-SDRAM device is selected, the MPDDRC provides a sequence of commands and exits self-refresh mode.

The MPDDRC re-enables self-refresh mode as soon as the DDR-SDRAM device is not selected. It is possible to define when self-refresh mode is to be enabled by configuring the TIMEOUT field in the MPDDRC_LPR register:

0: Self-refresh mode is enabled as soon as the DDR-SDRAM device is not selected.

1: Self-refresh mode is enabled 64 clock cycles after completion of the last access.

2: Self-refresh mode is enabled 128 clock cycles after completion of the last access.

This controller also interfaces the low-power DDR-SDRAM. To optimize power consumption, the Low Power DDR SDRAM provides programmable self-refresh options comprised of Partial Array Self Refresh (full, half, quarter and 1/8 and 1/16 array).

Disabled banks are not refreshed in self-refresh mode. This feature permits to reduce the self-refresh current. In case of low-power DDR1-SDRAM, the Extended Mode register controls this feature. It includes Temperature Compensated Self-refresh (TSCR) and Partial Array Self-refresh (PASR) parameters and the drive strength (DS) (see Section 29.8.7

"MPDDRC Low-power Register"). In case of low-power DDR2-SDRAM, the mode registers 16 and 17 control this feature, including PASR Bank Mask (BK_MASK) and PASR Segment Mask (SEG_MASK) parameters and drives strength (DS) (see Section 29.8.8 "MPDDRC Low-power DDR2 Low-power Register" on page 388). These parameters are set during the initialization phase. After initialization, as soon as the PASR/DS/TCSR fields or BK_MASK/SEG_MASK/DS are modified, the memory device Extended Mode Register or Mode Registers 3/16/17 are automatically accessed. Thus if MPDDRC does not share an external bus with another controller, PASR/DS/TCSR and BK_MASK/SEG_MASK/DS bits are updated before entering self-refresh mode or during a refresh command. If MPDDRC does share an external bus with another controller, PASR/DS/TCSR and BK_MASK/SEG_MASK/DS bits are also updated during a pending read or write access. This type of update is a function of the UPD_MR bit (see Section 29.8.7 "MPDDRC Low-power Register").

The low-power DDR1-SDRAM must remain in self-refresh mode during the minimum of TRFC periods (see Section 29.8.5 "MPDDRC Timing Parameter 1 Register"), and may remain in self-refresh mode for an indefinite period.

The DDR2-SDRAM must remain in self-refresh mode during the minimum of t_{CKE} periods (see the memory device datasheet), and may remain in self-refresh mode for an indefinite period.

The low-power DDR2-SDRAM must remain in self-refresh mode for the minimum of t_{CKESR} periods (see the memory device datasheet) and may remain in self-refresh mode for an indefinite period.

Figure 29-13.Self-refresh Mode Entry, Time-out = 0

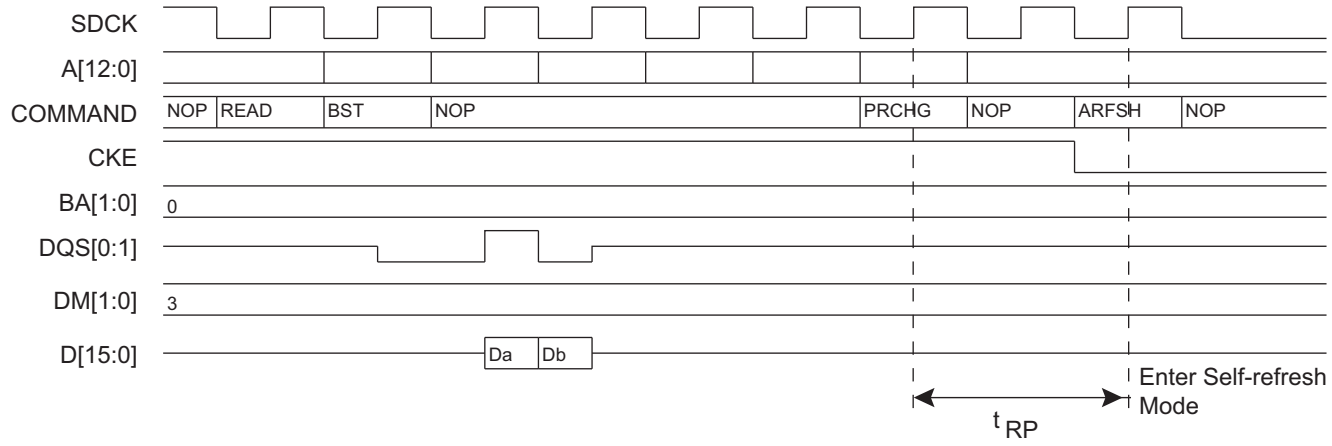


Figure 29-14.Self-refresh Mode Entry, Time-out = 1 or 2

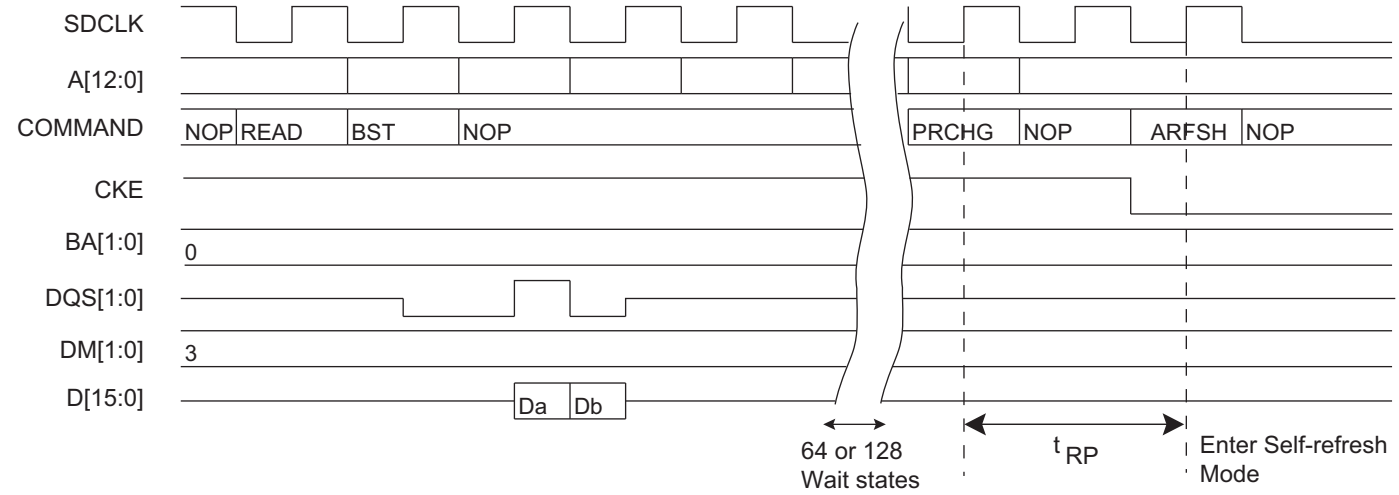
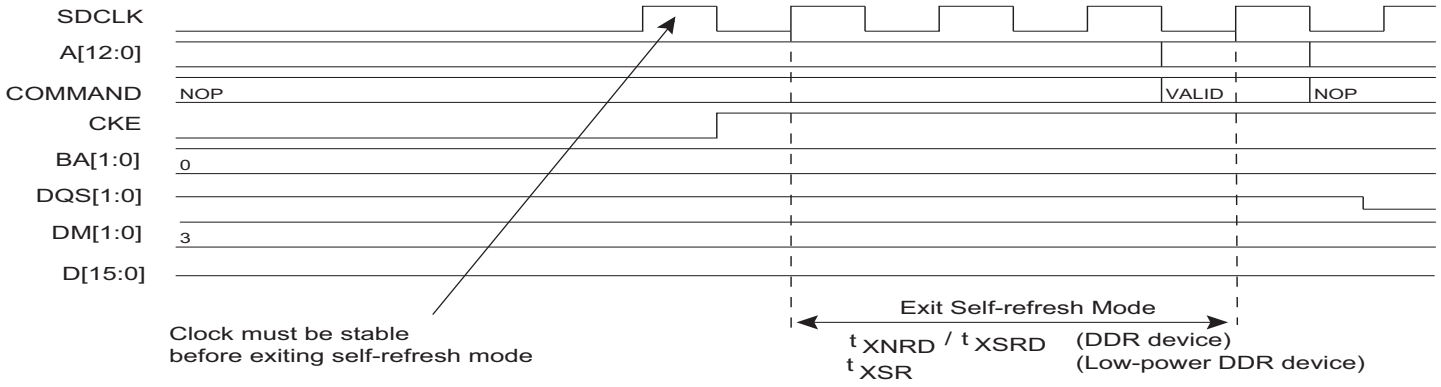


Figure 29-15.Self-refresh Mode Exit



29.5.3.2 Power-down Mode

This mode is activated by writing a 10 to the Low-power Command bit (LPCB).

Power-down mode is used when no access to the DDR-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 DDR-SDRAM device is no longer accessible. In contrast to self-refresh mode, the DDR-SDRAM device cannot remain in low-power mode longer than one refresh period (64 ms). As no auto-refresh operations are performed in this mode, the MPDDRC carries out the refresh operation. For DDR1-SDRAM, in order to exit low-power mode, a NOP command is required. For the low-power DDR1-SDRAM devices, a NOP command must be generated for a minimum period defined in the TXP field of the MPDDRC Timing Parameter 1 Register. In addition, low-power DDR-SDRAM and DDR2-SDRAM must remain in power-down mode for a minimum period corresponding to tCKE (see the memory device datasheet).

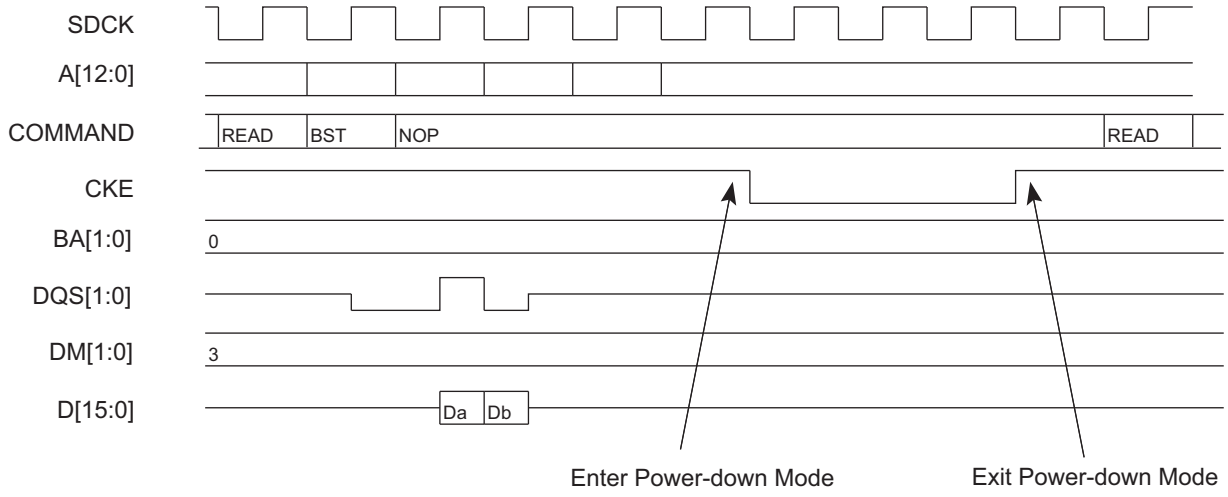
The exit procedure is faster than in self-refresh mode. See Figure 29-16 on page 366. The MPDDRC returns to power-down mode as soon as the DDR-SDRAM device is not selected. It is possible to define when power-down mode is enabled by configuring the TIMEOUT field in the MPDDRC_LPR register:

0: Power-down mode is enabled as soon as the DDR-SDRAM device is not selected.

1: Power-down mode is enabled 64 clock cycles after completion of the last access.

2: Power-down mode is enabled 128 clock cycles after completion of the last access.

Figure 29-16.Power-down Entry/Exit, Time-out = 0



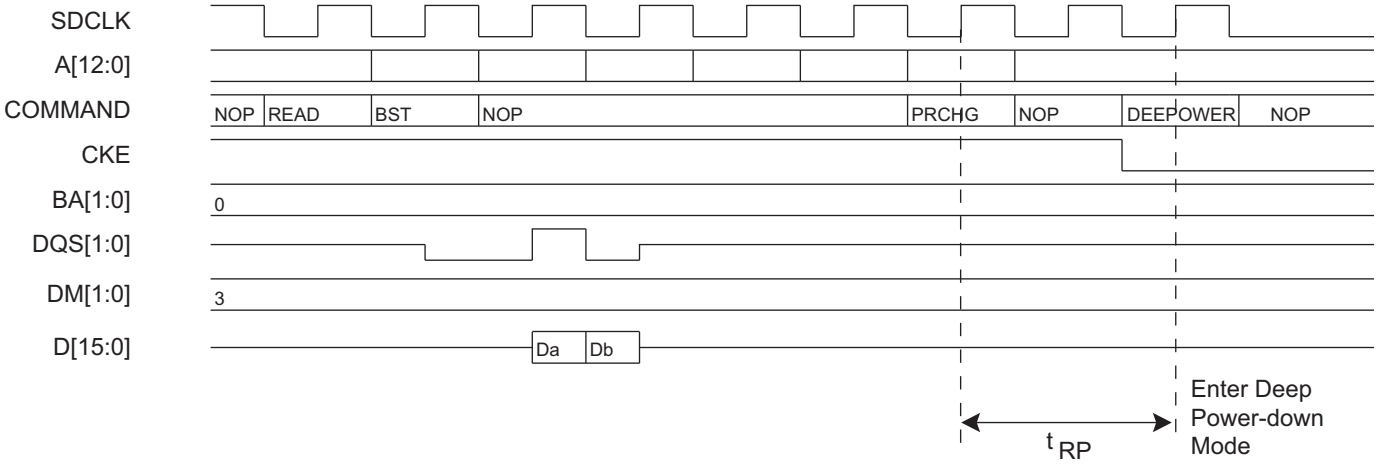
29.5.3.3 Deep Power-down Mode

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

Deep power-down mode is activated by writing a three to the Low-power Command bit (LPCB). When this mode is enabled, the MPDDRC leaves normal mode (mode = 0) and the controller is frozen.

Before enabling this mode, the user must assume there is no access in progress. To exit deep power-down mode, the Low-power Command bit (LPCB) must be written to zero and the initialization sequence must be generated by software. See Section 29.4.1 “Low-power DDR1-SDRAM Initialization” or Section 29.4.3 “Low-power DDR2-SDRAM Initialization”.

Figure 29-17.Deep Power-down Mode Entry



29.5.3.4 Change Frequency During Power-down Mode with Low-power DDR2-SDRAM Devices

To change frequency, power-down mode must be activated by writing a two to the Low-power Command bit (LPCB) and a one to the Change Frequency Command bit (CHG_FR) in the MPDDRC Low-power Register.

Once the low-power DDR2-SDRAM is in precharge power-down mode, the clock frequency may change. The device input clock frequency changes only within minimum and maximum operating frequencies as specified by low-power DDR2-SDRAM providers. Once the input clock frequency is changed, new stable clocks must be provided to the device before exiting from the precharge power-down mode.

Depending on the new clock frequency, the user can change the CAS latency in the user interface. (See “CAS: CAS Latency” on page 378.) It is recommended to check that no access is in progress. Once the controller detects a change of latency during the change frequency procedure, a Load Mode Register command is performed.

During a change frequency procedure, the Change Frequency Command bit (CHG_FR) is set to 0 automatically.

29.5.3.5 Reset Mode

The reset mode is a feature of DDR2-SDRAM. This mode is activated by writing a three to the Low-power Command bit (LPCB) and a one to the Clock Frozen Command bit (CLK_FR) in the MPDDRC Low-power Register.

When this mode is enabled, the MPDDRC leaves normal mode (mode = 0) and the controller is frozen. Before enabling this mode, the user must assume there is no access in progress.

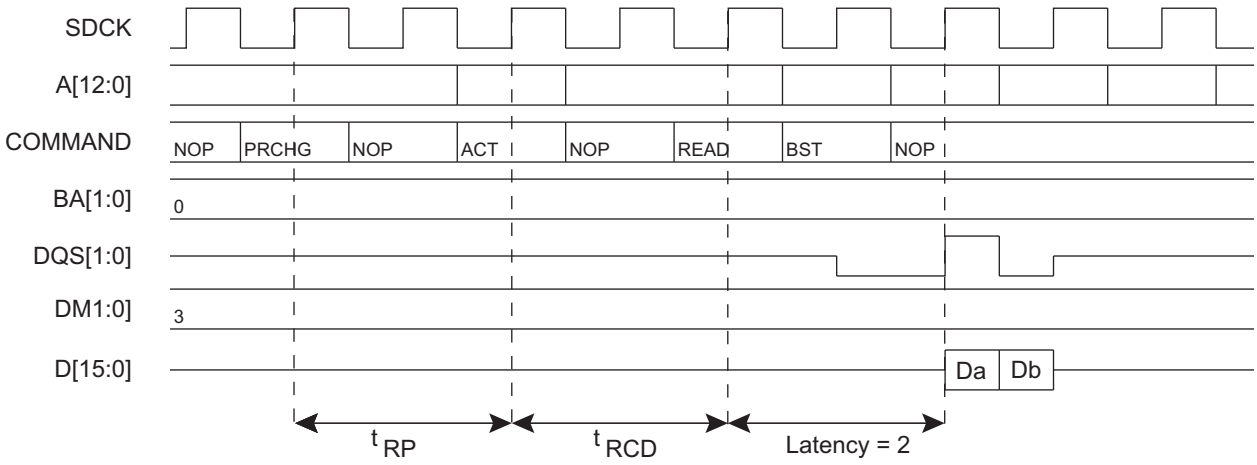
To exit reset mode, the Low-power Command bit (LPCB) must be written to zero, the Clock Frozen Command bit (CLK_FR) must be written to zero and the initialization sequence must be generated by software. (See Section 29.4.2 “DDR2-SDRAM Initialization”).

29.5.4 Multi-port Functionality

The DDR-SDRAM protocol imposes a check of timings prior to performing a read or a write access, thus decreasing system performance. An access to DDR-SDRAM is performed if banks and rows are open (or active). To activate a row in a particular bank, the last open row must be deactivated and a new row must be open. Two DDR-SDRAM commands must be performed to open a bank: Precharge command and Activate command with respect to t_{RP} timing. Before performing a read or write command, t_{RCD} timing must be checked.

This operation generates a significant bandwidth loss (see Figure 29-18.).

Figure 29-18.t_{RP} and t_{RCD} Timings

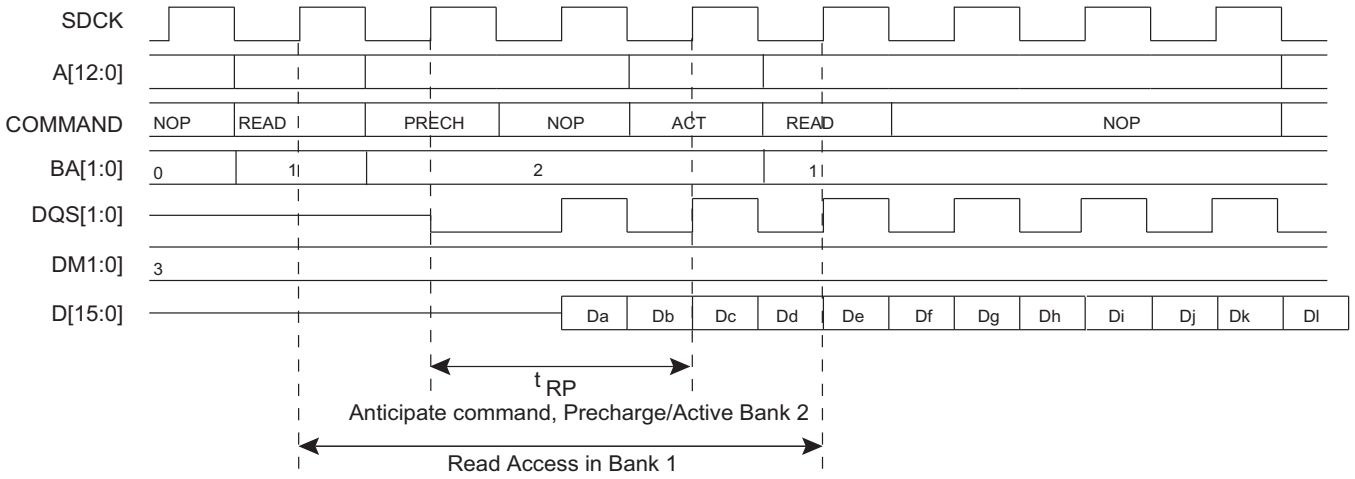


4 cycles before performing a read command

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

The MPDDRC is a multi-port controller whereby 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 command and Activate command in bank X during the current access in bank Y. This masks t_{RP} and t_{RCD} timings (see Figure 29-19). 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 or eight banks and associated rows are open, the controller reads with a continuous flow and masks the CAS latency for each access. To allow a continuous flow, the read command must be set at 2 or 3 cycles (CAS latency) before the end of the current access. The arbitration scheme must be changed since the round-robin arbitration cannot be respected. If the controller anticipates a read access, and thus a master with a high priority arises before the end of the current access, then this master will not be serviced.

Figure 29-19.Anticipate Precharge/Activate Command in Bank 2 during Read Access in Bank 1



The arbitration mechanism reduces latency when a conflict occurs, that is when two or more masters try to access the DDR-SDRAM device at the same time.

The arbitration type is round-robin arbitration. This algorithm dispatches requests from different masters to the DDR-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 DDR-SDRAM device, arbitration must only take place during the following cycles:

1. Idle cycles: when no master is connected to the DDR-SDRAM device.
2. Single cycles: when a slave is currently performing 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 anticipated read access is done while the 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.

29.6 Scrambling/Unscrambling Function

The external data bus can be scrambled in order to prevent intellectual property data located in off-chip memories from being easily recovered by analyzing data at the package pin level of either the microcontroller or the memory device.

The scrambling and unscrambling are performed on-the-fly without additional wait states.

The scrambling method depends on two user-configurable key registers, MPDDRC_KEY1 in the "MPDDRC OCMS KEY1 Register" and MPDDRC_KEY2 in the "MPDDRC OCMS KEY2 Register" . These key registers are only accessible in write mode.

The key must be securely stored in a reliable non-volatile memory in order to recover data from the off-chip memory. Any data scrambled with a given key cannot be recovered if the key is lost.

The scrambling/unscrambling function can be enabled or disabled by programming the "MPDDRC OCMS Register" .

29.7 Software Interface/SDRAM Organization, Address Mapping

The DDR-SDRAM address space is organized into banks, rows and columns. The MPDDRC maps different memory types depending on values set in the MPDDRC Configuration Register (see Section 29.8.3 "MPDDRC Configuration Register"). The tables that follow 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 MPDDRC supports address mapping in linear mode.

Sequential mode is a method for address mapping where banks alternate at each last DDR-SDRAM page of the current bank.

Interleaved mode is a method for address mapping where banks alternate at each SDRAM end page of the current bank.

The MPDDRC makes the DDR-SDRAM device access protocol transparent to the user. The tables that follow illustrate the DDR-SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

29.7.1 DDR-SDRAM Address Mapping for 16-bit Memory Data Bus Width

Table 29-1. Sequential Mapping for DDR-SDRAM Configuration, 2K Rows, 512/1024/2048/4096 Columns, 4 banks

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 29-2. Interleaved Mapping for DDR-SDRAM Configuration, 2K Rows, 512/1024/2048/4096 Columns, 4 banks

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 29-3. Sequential Mapping for DDR-SDRAM Configuration: 4K Rows, 512/1024/2048/4096 Columns, 4 banks

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 29-4. Interleaved Mapping for DDR-SDRAM Configuration: 4K Rows, 512/1024/2048/4096 Columns, 4 banks

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 29-5. Sequential Mapping for DDR-SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns, 4 banks

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 29-6. Interleaved Mapping for DDR-SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns, 4 banks

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 29-7. Sequential Mapping for DDR-SDRAM Configuration: 16K Rows, 512/1024/2048 Columns, 4 banks

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 29-8. Interleaved Mapping for DDR-SDRAM Configuration: 16K Rows, 512/1024/2048 Columns, 4 banks

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

Table 29-9. Sequential Mapping for DDR-SDRAM Configuration: 8K Rows,1024/ Columns, 8 banks

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 29-10. Interleaved Mapping for DDR-SDRAM Configuration: 8K Rows,1024/ Columns, 8 banks

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 29-11. Sequential Mapping for DDR-SDRAM Configuration: 16K Rows,1024/ Columns, 8 banks

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 29-12. Interleaved Mapping for DDR-SDRAM Configuration: 16K Rows,1024/ Columns, 8 banks

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[2:0]			Column[9:0]										M0

29.7.2 DDR-SDRAM Address Mapping for 32-bit Memory Data Bus Width

Table 29-13. Sequential Mapping DDR-SDRAM Configuration Mapping: 2K Rows /512/1024/2048 Columns, 4 banks

CPU Address Line																												
28	27	26	25	24	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 29-14. Interleaved Mapping DDR-SDRAM Configuration Mapping: 2K Rows /512/1024/2048 Columns, 4 banks

CPU Address Line																												
28	27	26	25	24	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]									M[1:0]	
				Row[10:0]											Bk[1:0]			Column[9:0]									M[1:0]	
			Row[10:0]											Bk[1:0]				Column[10:0]									M[1:0]	

Table 29-15. Sequential Mapping DDR-SDRAM Configuration Mapping: 4K Rows /512/1024/2048 Columns, 4 banks

CPU Address Line																												
28	27	26	25	24	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]									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 29-16. Interleaved Mapping DDR-SDRAM Configuration Mapping: 4K Rows /512/1024/2048 Columns, 4 banks

CPU Address Line																												
28	27	26	25	24	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]									M[1:0]	
			Row[11:0]												Bk[1:0]			Column[9:0]									M[1:0]	
		Row[11:0]												Bk[1:0]				Column[10:0]									M[1:0]	

Table 29-17. Sequential Mapping DDR-SDRAM Configuration Mapping: 8K Rows /512/1024/2048 Columns, 4 banks

CPU Address Line																												
28	27	26	25	24	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]									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]	

Table 29-18. Interleaved Mapping DDR-SDRAM Configuration Mapping: 8K Rows /512/1024/2048 Columns, 4 banks

CPU Address Line																												
28	27	26	25	24	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]									M[1:0]	
		Row[13:0]													Bk[1:0]			Column[9:0]									M[1:0]	
	Row[13:0]													Bk[1:0]				Column[10:0]									M[1:0]	

Table 29-19. Sequential Mapping DDR-SDRAM Configuration Mapping: 8K Rows /1024 Columns, 8 banks

CPU Address Line																											
28	27	26	25	24	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]										M[1:0]	

Table 29-20. Interleaved Mapping DDR-SDRAM Configuration Mapping: 8K Rows /1024 Columns, 8 banks

CPU Address Line																											
28	27	26	25	24	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]										M[1:0]	

Table 29-21. Sequential Mapping DDR-SDRAM Configuration Mapping: 16K Rows /1024 Columns, 4 banks

CPU Address Line																											
28	27	26	25	24	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[9:0]										M[1:0]	

Table 29-22. Interleaved Mapping DDR-SDRAM Configuration Mapping: 16K Rows /1024 Columns, 4 banks

CPU Address Line																											
28	27	26	25	24	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[9:0]										M[1:0]	

Table 29-23. Sequential Mapping DDR-SDRAM Configuration Mapping: 16K Rows /1024 Columns, 8 banks

CPU Address Line																											
28	27	26	25	24	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]										M[1:0]	

Table 29-24. Interleaved Mapping DDR-SDRAM Configuration Mapping: 16K Rows /1024 Columns, 8 banks

CPU Address Line																											
28	27	26	25	24	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[2:0]			Column[9:0]										M[1:0]	

Notes: 1. M[1:0] is the byte address inside a 32-bit word.
2. Bk[2] = BA2, Bk[1] = BA1, Bk[0] = BA0

29.8 AHB Multi-port DDR-SDRAM Controller (MPDDRC) User Interface

The User Interface is connected to the APB bus.

The MPDDRC is programmed using the registers listed in Table 29-25.

Table 29-25. Register Mapping

Offset	Register	Name	Access	Reset
0x00	MPDDRC Mode Register	MPDDRC_MR	Read/Write	0x00000000
0x04	MPDDRC Refresh Timer Register	MPDDRC_RTR	Read/Write	0x00000000
0x08	MPDDRC Configuration Register	MPDDRC_CR	Read/Write	0x024
0x0C	MPDDRC Timing Parameter 0 Register	MPDDRC_TPR0	Read/Write	0x20227225
0x10	MPDDRC Timing Parameter 1 Register	MPDDRC_TPR1	Read/Write	0x3C80808
0x14	MPDDRC Timing Parameter 2 Register	MPDDRC_TPR2	Read/Write	0x00042062
0x18	Reserved	–	–	–
0x1C	MPDDRC Low-power Register	MPDDRC_LPR	Read/Write	0x0
0x20	MPDDRC Memory Device Register	MPDDRC_MD	Read/Write	0x10
0x24	MPDDRC High Speed Register	MPDDRC_HS	Read/Write	0x00000000
0x28	MPDDRC LPDDR2 Low-power Register	MPDDRC_LPDDR2_LPR	Read/Write	0x00000000
0x2C	MPDDRC LPDDR2 Calibration and MR4 Register	MPDDRC_LPDDR2_CAL_MR4	Read/Write	0x00000000
0x30	MPDDRC LPDDR2 Timing Calibration Register	MPDDRC_LPDDR2_TIM_CAL	Read/Write	0x040
0x34	MPDDRC IO Calibration	MPDDRC_IO_CALIBR	Read/Write	0x00870002
0x38	MPDDRC OCMS Register	MPDDRC_OCMS	Read/Write	0x00000000
0x3C	MPDDRC OCMS KEY1 Register	MPDDRC_OCMS_KEY1	Write-only	0x00000000
0x40	MPDDRC OCMS KEY2 Register	MPDDRC_OCMS_KEY2	Write-only	0x00000000
0x44–0x70	Reserved	–	–	–
0x74	MPDDRC DLL Master Offset Register	MPDDRC_DLL_MO	Read/Write	0x-(1)
0x78	MPDDRC DLL Slave Offset Register	MPDDRC_DLL_SOF	Read/Write	0x-(1)
0x7C	MPDDRC DLL Status Master Register	MPDDRC_DLL_MS	Read-only	0x00000000
0x80	MPDDRC DLL Status Slave 0 Register	MPDDRC_DLL_SS0	Read-only	0x00000000
0x84	MPDDRC DLL Status Slave 1 Register	MPDDRC_DLL_SS1	Read-only	0x00000000
0x88	MPDDRC DLL Status Slave 2 Register	MPDDRC_DLL_SS2	Read-only	0x00000000
0x8C	MPDDRC DLL Status Slave 3 Register	MPDDRC_DLL_SS3	Read-only	0x00000000
0x94xE0	Reserved	–	–	–
0xE4	MPDDRC Write Protect Control Register	MPDDRC_WPMR	Read/Write	0x00000000
0xE8	MPDDRC Write Protect Status Register	MPDDRC_WPSR	Read-only	0x00000000
0x158–0x1CC	Reserved	–	–	–
0x1DC–0x1F8	Reserved	–	–	–

Note: 1. Values in the DLL Master Offset Register and in the DLL Slave Offset Register vary with the product implementation.

29.8.1 MPDDRC Mode Register

Name: MPDDRC_MR

Address: 0xFFFFEA00

Access: Read/Write

Reset: See Table 29-25

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

- **MODE: MPDDRC Command Mode**

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

Value	Name	Description
0	NORMAL_CMD	Normal Mode. Any access to the MPDDRC is decoded normally. To activate this mode, the command must be followed by a write to the DDR-SDRAM.
1	NOP_CMD	The MPDDRC issues a NOP command when the DDR-SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the DDR-SDRAM.
2	PRCGALL_CMD	The MPDDRC issues the All Banks Precharge command when the DDR-SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the SDRAM.
3	LMR_CMD	The MPDDRC issues a Load Mode Register command when the DDR-SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the DDR-SDRAM.
4	RFSH_CMD	The MPDDRC issues an Auto-Refresh command when the DDR-SDRAM device is accessed regardless of the cycle. Previously, an All Banks Precharge command must be issued. To activate this mode, the command must be followed by a write to the DDR-SDRAM.
5	EXT_LMR_CMD	The MPDDRC issues an Extended Load Mode Register command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the DDR-SDRAM. The write in the DDR-SDRAM must be done in the appropriate bank.
6	DEEP_CMD	Deep power mode: Access to deep power-down mode
7	LPDDR2_CMD	The MPDDRC issues an LPDDR2 Mode Register command when the low-power DDR2-SDRAM device is accessed regardless of the cycle. To activate this mode, the Mode Register command must be followed by a write to the low-power DDR2-SDRAM.

- **MRS: Mode Register Select LPDDR2**

Configure this 8-bit field to program all mode registers included in the low-power DDR2-SDRAM device. This field is unique to the low-power DDR2-SDRAM devices.

29.8.2 MPDDRC Refresh Timer Register

Name: MPDDRC_RTR

Address: 0xFFFFEA04

Access: Read/Write

Reset: See Table 29-25

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

23	22	21	20	19	18	17	16
	MR4_VALUE			–	–	REF_PB	ADJ_REF

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

7	6	5	4	3	2	1	0
COUNT							

- **COUNT: MPDDRC 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.

The SDRAM devices require a refresh of all rows every 64 ms. The value to be loaded depends on the MPDDRC 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 the COUNT field is configured: $((64 \times 10^{-3})/8192) \times 100 \times 10^{6} = 781$ or 0x030D.

The low-power DDR2-SDRAM devices support Per Bank Refresh operation. In this configuration, average time between refresh command is 0.975 µs. The value of the COUNT field is configured depending on this value. For example, the value of a 100 MHz Master clock refresh timer is 98 or 0x0062.

- **ADJ_REF: Adjust Refresh Rate**

Reset value is 0.

0: Adjust refresh rate is not enabled.

1: Adjust refresh rate is enabled.

This mode is unique to the low-power DDR2-SDRAM devices.

- **REF_PB: Refresh Per Bank**

Reset value is 0.

0: Refresh all banks during auto-refresh operation.

1: Refresh the scheduled bank by the bank counter in the memory interface.

This mode is unique to the low-power DDR2-SDRAM devices.

- **MR4_VALUE: Content of MR4 Register**

Reset value is 3.

This field (read-only) gives the content of MR4 register. This field is updated when MRR command is generated and Adjust Refresh Rate bit is enabled. Update is done when read value is different from MR4_VALUE.

LP-DDR2 JEDEC memory standards impose derating LP-DDR2 AC timings (t_{RCD}, t_{RC}, t_{RAS}, t_{RP} and t_{RRD}) when the value of MR4 is equal to 6. If the application needs to work in extreme conditions, the derating value must be added to AC timings before the power up sequence.

This mode is unique to the low-power DDR2-SDRAM devices.

29.8.3 MPDDRC Configuration Register

Name: MPDDRC_CR

Address: 0xFFFFEA08

Access: Read/Write

Reset: See Table 29-25

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

23	22	21	20	19	18	17	16
UNAL	DECOD	NDQS	NB		–	ENRDM	DQMS

15	14	13	12	11	10	9	8
–	OCD			ZQ		DIS_DLL	DIC_DS

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

- **NC: Number of Column Bits**

Reset value is 9 column bits.

Value	Name	Description
0	9_COL_BITS	9 bit to define the column number, up to 512 column
1	10_COL_BITS	10 bit to define the column number, up to 1024 columns
2	11_COL_BITS	11 bit to define the column number, up to 2048 columns
3	12_COL_BITS	12 bit to define the column number, up to 4096 columns

- **NR: Number of Row Bits**

Reset value is 12 row bits.

Value	Name	Description
0	11_ROW_BITS	11 bit to define the row number, up to 2048 rows
1	12_ROW_BITS	12 bit to define the row number, up to 4096 rows
2	13_ROW_BITS	13 bit to define the row number, up to 8192 rows
3	14_ROW_BITS	14 bit to define the row number, up to 16384 rows

- **CAS: CAS Latency**

Reset value is 2 cycles.

Value	Name	Description
2	DDR_CAS2	LPDDR1 CAS Latency 2
3	DDR_CAS3	DDR2/LPDDR2/LPDDR1 CAS Latency 3

- **DLL: Reset DLL**

Reset value is 0.

This bit defines the value of Reset DLL.

0 (RESET_DISABLED): Disable DLL reset.

1 (RESET_ENABLED): Enable DLL reset.

This value is used during the power-up sequence.

This bit is found only in the DDR2-SDRAM devices.

- **DIC_DS: Output Driver Impedance Control (Drive Strength)**

Reset value is 0.

This bit name is described as “DS” in some memory datasheets. It defines the output drive strength. This value is used during the power-up sequence.

Value	Name	Description
0	DDR2_NORMALSTRENGTH	Normal driver strength
1	DDR2_WEAKSTRENGTH	Weak driver strength

This bit is found only in the DDR2-SDRAM devices.

- **DIS_DLL: DISABLE DLL**

Reset value is 0.

0: Enable DLL.

1: Disable DLL.

This value is used during the power-up sequence. It is only found in the DDR2-SDRAM devices.

- **ZQ: ZQ Calibration**

Reset value is 0.

Value	Name	Description
0	INIT	Calibration command after initialization
1	LONG	Long calibration
2	SHORT	Short calibration
3	RESET	ZQ Reset

This parameter is used to calibrate DRAM On resistance (Ron) values over PVT.

This field is found only in the low-power DDR2-SDRAM devices.

- **OCD: Off-chip Driver**

Reset value is 7.

Note: SDRAM Controller supports only two values for OCD (default calibration and exit from calibration). These values MUST always be programmed during the initialization sequence. The default calibration must be programmed first, after which the exit calibration and maintain settings must be programmed.

This field is found only in the DDR2-SDRAM devices.

Value	Name	Description
0	DDR2_EXITCALIB	Exit from OCD calibration mode and maintain settings
7	DDR2_DEFAULT_CALIB	OCD calibration default

- **DQMS: Mask Data is Shared**

Reset value is 0.

0 (NOT_SHARED): DQM is not shared with another controller.

1 (SHARED): DQM is shared with another controller.

- **ENRDM: Enable Read Measure**

Reset value is 0.

0 (OFF): DQS/DDR_DATA phase error correction is disabled.

1 (ON): DQS/DDR_DATA phase error correction is enabled.

- **NB: Number of Banks**

Reset value is 4 banks. If LC_LPDDR1 is set to 1, NB is not relevant.

Value	Name	Description
0	4_BANKS	4 banks memory devices
1	8_BANKS	8 banks. Only possible when using the DDR2-SDRAM and low-power DDR2-SDRAM devices.

- **NDQS: Not DQS**

Reset value is 1; not DQS is disabled.

0: (ENABLED) Not DQS is enabled.

1: (DISABLED) Not DQS is disabled.

This field is found only in the DDR2-SDRAM devices.

- **DECOD: Type of Decoding**

Reset value is 0.

Value	Name	Description
0	SEQUENTIAL	Method for address mapping where banks alternate at each last DDR-SDRAM page of the current bank.
1	INTERLEAVED	Method for address mapping where banks alternate at each SDRAM end page of the current bank.

- **UNAL: Support Unaligned Access**

Reset value is 0; unaligned access is not supported.

0 (UNSUPPORTED): Unaligned access is not supported.

1 (SUPPORTED): Unaligned access is supported.

This mode is enabled with masters which have an AXI interface.

29.8.4 MPDDRC Timing Parameter 0 Register

Name: MPDDRC_TPR0

Address: 0xFFFFEA0C

Access: Read/Write

Reset: See Table 29-25

31	30	29	28	27	26	25	24
TMRD				RDC_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			

- **TRAS: Active to Precharge Delay**

Reset value is 5 SDCK[(1)] clock cycles.

This field defines the delay between an Activate command and a Precharge command in number of SDCK[(1)] clock cycles. The number of cycles is between 0 and 15.

- **TRCD: Row to Column Delay**

Reset value is 2 SDCK[(1)] clock cycles.

This field defines the delay between an Activate command and a Read/Write command in number of SDCK[(1)] clock cycles. The number of cycles is between 0 and 15.

- **TWR: Write Recovery Delay**

Reset value is 2 SDCK[(1)] clock cycles.

This field defines the Write Recovery Time in number of SDCK[(1)] clock cycles. The number of cycles is between 1 and 15.

- **TRC: Row Cycle Delay**

Reset value is 7 SDCK[(1)] clock cycles.

This field defines the delay between an Activate command and Refresh command in number of SDCK[(1)] clock cycles. The number of cycles is between 0 and 15

- **TRP: Row Precharge Delay**

Reset value is 2 SDCK[(1)] clock cycles.

This field defines the delay between a Precharge command and another command in number of SDCK[(1)] clock cycles. The number of cycles is between 0 and 15.

- **TRRD: Active BankA to Active BankB**

Reset value is 2 SDCK[(1)] clock cycles.

This field defines the delay between an Activate command in BankA and an Activate command in BankB in number of SDCK[(1)] clock cycles. The number of cycles is between 1 and 15.

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

Reset value is 0.

This field defines the internal Write to Read command time in number of SDCK[1] clock cycles. The number of cycles is between 1 and 7.

- **RDC_WRRD: Reduce Write to Read Delay**

Reset value is 0.

This field reduces the delay between write to read access for the low-power DDR-SDRAM devices with a latency equal to 2. To use this feature, the TWTR field must be equal to 0. Note that some devices do not support this feature.

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

Reset value is 2 SDCK[1] clock cycles.

This field defines the delay between a Load mode register command and an Activate or Refresh command in number of SDCK[1] clock cycles. The number of cycles is between 0 and 15. For low-power DDR2-SDRAM, this field is equivalent to TMRW timing.

Note: 1. SDCK is the clock that drives the SDRAM device.

29.8.5 MPDDRC Timing Parameter 1 Register

Name: MPDDRC_TPR1

Address: 0xFFFFEA10

Access: Read/Write

Reset: See Table 29-25

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

23	22	21	20	19	18	17	16
TXSRD							

15	14	13	12	11	10	9	8
TXSNR							

7	6	5	4	3	2	1	0
–	–	–	TRFC				

- **TRFC: Row Cycle Delay**

Reset value is 8 SDCK(1) clock cycles.

This field defines the delay between a Refresh command or a Refresh and Activate command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 31.

In case of low-power DDR2-SDRAM, this field is equivalent to tRFCab timing. If the user enables the function “Refresh Per Bank” (see “REF_PB: Refresh Per Bank” on page 376), this field is equivalent to tRFCpb.

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

Reset value is 8 SDCK(1) clock cycles.

This field defines the delay between CKE set high and a Non Read command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 255. This field is used by the DDR-SDRAM devices. In case of low-power DDR-SDRAM, this field is equivalent to t_{XSR} timing.

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

Reset value is 200 SDCK(1) clock cycles.

This field defines the delay between CKE set high and a Read command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 255.

This field is found only in the DDR2-SDRAM devices .

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

Reset value is 3 SDCK(1) clock cycles.

This field defines the delay between CKE set high and a Valid command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 15.

Note: 1. SDCK is the clock that drives the SDRAM device.

29.8.6 MPDDRC Timing Parameter 2 Register

Name: MPDDRC_TPR2

Address: 0xFFFFEA14

Access: Read/Write

Reset: See Table 29-25

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			

- **TXARD: Exit Active Power Down Delay to Read Command in Mode “Fast Exit”**

Reset value is 2 SDCK(1) clock cycles.

This field defines the delay between CKE set high and a Read command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 15.

This field is found only in the DDR2-SDRAM devices.

- **TXARDS: Exit Active Power Down Delay to Read Command in Mode “Slow Exit”**

Reset value is 6 SDCK(1) clock cycles.

This field defines the delay between CKE set high and a Read command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 15.

This field is found only in the DDR2-SDRAM devices.

- **TRPA: Row Precharge All Delay**

Reset value is 0 SDCK(1) clock cycles.

This field defines the delay between a Precharge All Banks command and another command in number of SDCK(1) clock cycles. The number of cycles is between 0 and 15.

This field is found only in the DDR2-SDRAM devices.

- **TRTP: Read to Precharge**

Reset value is 2 SDCK(1) clock cycles.

This field defines the delay between Read command and a Precharge command in number of SDCK(1) clock cycles.

The number of cycles is between 0 and 7.

- **TFAW: Four Active Windows**

Reset value is 4 SDCK(1) clock cycles.

DDR2 devices with eight banks (1 Gbit or larger) have an additional requirement concerning t_{FAW} timing. This requires that no more than four Activate commands may be issued in any given t_{FAW} (MIN) period.

The number of cycles is between 0 and 15.

This field is found only in the DDR2-SDRAM and LPDDR2-SDRAM devices.

Note: 1. SDCK is the clock that drives the SDRAM device.

29.8.7 MPDDRC Low-power Register

Name: MPDDRC_LPR

Address: 0xFFFFEA1C

Access: Read/Write

Reset: See Table 29-25

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			LPDDR2_PWOFF	CLK_FR	LPCB	

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

Reset value is 0.

Value	Name	Description
0	NOLOWPOWER	Low-power feature is inhibited. No power-down, self-refresh and deep-power modes are issued to the DDR-SDRAM device.
1	SELFREFRESH	The MPDDRC issues a self-refresh command to the DDR-SDRAM device, the clock(s) is/are deactivated and the CKE signal is set low. The DDR-SDRAM device leaves the self-refresh mode when accessed and reenters it after the access.
2	POWERDOWN	The MPDDRC issues a Power-down command to the DDR-SDRAM device after each access, the CKE signal is set low. The DDR-SDRAM device leaves the power-down mode when accessed and reenters it after the access.
3	DEEPPOWERDOWN	The MPDDRC issues a Deep Power-down command to the low-power DDR-SDRAM device.

- **CLK_FR: Clock Frozen Command Bit**

Reset value is 0.

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

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

1 (ENABLED): Clock(s) is/are frozen.

- **LPDDR2_PWOFF: LPDDR2 Power Off Bit**

Reset value is 0.

LPDDR2 power off sequence must be controlled to preserve the LPDDR2 device. The power failure is handled at system level (IRQ or FIQ) and the LPDDR2 power off sequence is applied using the LPDDR2_PWOFF bit.

LPDDR2_PWOFF bit is used to impose CKE low before a power off sequence. Uncontrolled power off sequence can be applied only up to 400 times in the life of a LPDDR2 device.

1 (ENABLED): A power off sequence is applied to the LPDDR2 device. CKE is forced low.

0 (DISABLED): No power off sequence applied to LPDDR2.

- **PASR: Partial Array Self-refresh**

Reset value is 0.

This field is unique to low-power DDR1-SDRAM. It is used to specify whether only one-quarter, one-half or all banks of the DDR-SDRAM array are enabled. Disabled banks are not refreshed in self-refresh mode.

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

After the initialization sequence, as soon as the PASR field is modified, the Extended Mode Register in the external device memory is accessed automatically and PASR bits are updated. Depending on the UPD_MR bit, update is done before entering 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 DDR-SDRAM. It selects the driver strength of the DDR- SDRAM output.

After the initialization sequence, as soon as the DS field is modified, the Extended Mode Register is accessed automatically and DS bits are updated. Depending on the UPD_MR bit, update is done before entering self-refresh mode or during a refresh command and a pending read or write access.

- **TIMEOUT: Time Between Last Transfer and Low Power Mode**

Reset value is 0. This field defines when low-power mode is activated.

Value	Name	Description
0	NONE	SDRAM low-power mode is activated immediately after the end of the last transfer.
1	DELAY_64_CLK	SDRAM low-power mode is activated 64 clock cycles after the end of the last transfer.
2	DELAY_128_CLK	SDRAM low-power mode is activated 128 clock cycles after the end of the last transfer.

- **APDE: Active Power Down Exit Time**

Reset value is 1.

This mode is unique to the DDR2-SDRAM devices.

This mode manages the active power-down mode which determines performance versus power saving**.**

Value	Name	Description
0	DDR2_FAST_EXIT	Fast Exit from Power Down. The DDR2-SDRAM devices only.
1	DDR2_SLOW_EXIT	Slow Exit from Power Down. The DDR2-SDRAM devices only.

After the initialization sequence, as soon as the APDE field is modified, the Extended Mode Register (located in the memory of the external device) is accessed automatically and APDE bits are updated. Depending on the UPD_MR bit, update is done before entering 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, an external bus with another controller.

Value	Name	Description
0	NO_UPDATE	Update of Load Mode and Extended Mode registers is disabled.
1	UPDATE_SHAREDBUS	DDRSDRC shares an external bus. Automatic update is done during a refresh command and a pending read or write access in the SDRAM device.
2	UPDATE_NOSHAREDBUS	DDRSDRC does not share an external bus. Automatic update is done before entering in self-refresh mode.

29.8.8 MPDDRC Low-power DDR2 Low-power Register

Name: MPDDRC_LPDDR2_LPR

Address: 0xFFFFEA28

Access: Read/Write

Reset: See Table 29-25

31	30	29	28	27	26	25	24
				DS			

23	22	21	20	19	18	17	16
SEG_MASK							

15	14	13	12	11	10	9	8
SEG_MASK							

7	6	5	4	3	2	1	0
BK_MASK_PASR							

- **BK_MASK_PASR: Bank Mask Bit/PASR**

Partial Array Self-Refresh (the low-power DDR2-SDRAM-S4 devices only)

Reset value is 0.

After the initialization sequence, as soon as the BK_MASK_PASR field is modified, Mode Register 16 is accessed automatically and BK_MASK_PASR bits are updated. Depending on the UPD_MR bit, update is done before entering self-refresh mode or during a refresh command and a pending read or write access.

0: Refresh is enabled (= unmasked).

1: Refresh is disabled (= masked).

This mode is unique to the low-power DDR2-SDRAM-S4 devices. In self-refresh mode, each bank of LPDDR2 can be independently configured whether a self-refresh operation is taking place or not.

After the initialization sequence, as soon as the BK_MASK_PASR field is modified, the Extended Mode Register is accessed automatically and BK_MASK_PASR bits are updated. Depending on the UPD_MR bit, update is done before entering self-refresh mode or during a refresh command and a pending read or write access.

- **SEG_MASK: Segment Mask Bit**

Reset value is 0.

After the initialization sequence, as soon as the SEG_MASK field is modified, Mode Register 17 is accessed automatically and SEG_MASK bits are updated. Depending on the UPD_MR bit, update is done before entering self-refresh mode or during a refresh command and a pending read or write access.

0: Segment is refreshed (= unmasked).

1: Segment is not refreshed (= masked).

This mode is unique to the low-power DDR2-SDRAM-S4 devices. The number of Segment Mask bits differs with the density. For 1 Gbit density, 8 segments are used. In self-refresh mode, when the Segment Mask bit is configured, the refresh operation is masked in the segment.

- **DS: Drive strength**

Reset value is 0.

After the initialization sequence, as soon as the DS field is modified, Mode Register 3 is accessed automatically and DS bits are updated. Depending on the UPD_MR bit, update is done before entering self-refresh mode or during a refresh command and a pending read or write access.

This field is unique to low-power DDR2-SDRAM. It selects the driver strength of DDR2-SDRAM I/O.

- **SR: Slew Rate**

Reset value is 0.

After the initialization sequence, as soon as the SR field is modified, Mode Register 3 is accessed automatically and SR bits are updated. Depending on the UPD_MR bit, update is done before entering self-refresh mode or during a refresh command and a pending read or write access

This field is unique to low-power DDR2-SDRAM. It selects the slew rate of low-power DDR2-SDRAM I/O.

29.8.9 MPDDRC Low-power DDR2 Calibration and MR4 Register

Name: MPDDRC_LPDDR2_CAL_MR4

Address: 0xFFFFEA2C

Access: Read/Write

Reset: See Table 29-25

31	30	29	28	27	26	25	24
MR4_READ							

23	22	21	20	19	18	17	16
MR4_READ							

15	14	13	12	11	10	9	8
COUNT_CAL							

7	6	5	4	3	2	1	0
COUNT_CAL							

- **COUNT_CAL: LPDDR2 Calibration Timer Count**

This 16-bit field is loaded into a timer which generates the calibration pulse. Each time the calibration pulse is generated, a ZQCS calibration sequence is initiated.

The ZQCS Calibration command is used to calibrate DRAM Ron values over PVT.

One method for calculating the interval between ZQCS commands gives the temperature ($T_{driftrate}$) and voltage ($V_{driftrate}$) drift rates that the SDRAM is subject to in the application. The interval could be defined by the following formula: ZQCorrection/((T_{Sens} x $T_{driftrate}$) + (V_{Sens} x $V_{driftrate}$))

Where T_{Sens} = max(dRONdTM) and V_{Sens} = max(dRONdVM) define the SDRAM temperature and voltage sensitivities.

For example, if T_{Sens} = 0.75%/C, V_{Sens} = 0.2%/mV, $T_{driftrate}$ = 1C/sec and $V_{driftrate}$ = 15 mV/s, then the interval between ZQCS commands is calculated as:

1.5/((0.75 x 1) + (0.2 x 15)) = 0.4s

In this example, the LPDDR2-SDRAM devices require a calibration every 0.4s. The value to be loaded depends on average time between REFRESH commands, t_{REF}.

For example, for an LPDDR2-SDRAM with the time between refresh of 7.8 µs, the value of the Calibration Timer Count field is programmed: ($0.4/7.8 \times 10^{-6}$) = 0xC852.

- **MR4_READ: Mode Register 4 Read Interval**

MR4_READ defines the time period between MR4 reads (for LPDDR2-SDRAM). The formula is (MR4_READ+1) * t_{REF}. The value to be loaded depends on the average time between REFRESH commands, t_{REF}. For example, for an LPDDR2-SDRAM with the time between refresh of 7.8 µs, if the MR4_READ value is 2, the time period between MR4 reads is 23.4 µs.

The LPDDR2-SDRAM devices feature a temperature sensor whose status can be read from MR4 register. This sensor can be used to determine an appropriate refresh rate. Temperature sensor data may be read from MR4 register using the Mode Register Read protocol. The Adjust Refresh Rate bit (ADJ_REF) in the Refresh Timer Register (MPDDRC_RTR) must be written to a one to activate these reads.

29.8.10 MPDDRC Low-power DDR2 Timing Calibration Register

Name: MPDDRC_LPDDR2_TIM_CAL

Address: 0xFFFFEA30

Access: Read/Write

Reset: See Table 29-25

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

- **ZQCS: ZQ Calibration Short**

Reset value is 6 SDCK[1] clock cycles

This field defines the delay between ZQ Calibration command and any Valid commands in number of SDCK[1] clock cycles.

The number of cycles is between 0 and 255.

Note: 1. SDCK is the clock that drives the SDRAM device.

29.8.11 MPDDRC I/O Calibration Register

Name: MPDDRC_IO_CALIBR

Address: 0xFFFFEA34

Access: Read/Write

Reset: See Table 29-25

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

- **RDIV: Resistor Divider, Output Driver Impedance**

Reset value is 2. This corresponds to 40 ohms.

With the LPDDR2-SDRAM device, the RDIV field must be equal to the DS (Drive Strength) field (see “DS: Drive Strength” on page 386).

RDIV is used with the external precision resistor RZQ to define the output driver impedance. The value of RZQ is either 240 ohms (LPDDR2 mode) or 200 ohms (the DDR2/LPDDR1 device).

Value	Name	Description
1	RZQ_34	LPDDR2 RZQ = 34.3 ohms, DDR2/LPDDR1: Not applicable
2	RZQ_40_RZQ_33_3	LPDDR2:RZQ = 40 ohms, DDR2/LPDDR1: RZQ = 33.3 ohms
3	RZQ_48_RZQ_40	LPDDR2:RZQ = 48 ohms, DDR2/LPDDR1: RZQ = 40 ohms
4	RZQ_60_RZQ_50	LPDDR2:RZQ = 60 ohms, DDR2/LPDDR1: RZQ = 50 ohms
6	RZQ_80_RZQ_66_7	LPDDR2: RZQ = 80 ohms, DDR2/LPDDR1: RZQ = 66.7 ohms
7	RZQ_120_RZQ_100	LPDDR2:RZQ = 120 ohms, DDR2/LPDDR1: RZQ = 100 ohms

- **TZQIO: IO Calibration**

This field defines the delay between an IO Calibration command and any valid commands in number of SDCK[1] clock cycles.

The number of cycles is between 0 and 7.

The TZQIO configuration code must be correctly set depending on the clock frequency using the following formula:

TZQIO = (DDRCLK * 20 ns) + 1.

- **CALCODEP: Number of Transistor P**

This register is read-only. Reset value is 7.

This value gives the number of transistor P to perform the calibration.

- **CALCODEN: Number of Transistor N**

This register is read-only. Reset value is 8.

This value gives the number of transistor N to perform the calibration.

Note: 1. SDCK is the clock that drives the SDRAM device.

29.8.12 MPDDRC OCMS Register

Name: MPDDRC_OCMS

Address: 0xFFFFEA38

Access: Read/Write

Reset: See Table 29-25

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

- **SCR_EN: Scrambling Enable**

0: Disables “Off-chip” scrambling for SDRAM access.

1: Enables “Off-chip” scrambling for SDRAM access.

29.8.13 MPDDRC OCMS KEY1 Register

Name: MPDDRC_OCMS_KEY1

Address: 0xFFFFEA3C

Access: Write once

Reset: See Table 29-25

31	30	29	28	27	26	25	24
KEY1							

23	22	21	20	19	18	17	16
KEY1							

15	14	13	12	11	10	9	8
KEY1							

7	6	5	4	3	2	1	0
KEY1							

- **KEY1: Off-chip Memory Scrambling (OCMS) Key Part 1**

When Off-chip Memory Scrambling is enabled, the data scrambling depends on KEY1 and KEY2 values.

29.8.14 MPDDRC OCMS KEY2 Register

Name: MPDDRC_OCMS_KEY2

Address: 0xFFFFEA40

Access: Write once

Reset: See Table 29-25

31	30	29	28	27	26	25	24
KEY2							
23	22	21	20	19	18	17	16
KEY2							
15	14	13	12	11	10	9	8
KEY2							
7	6	5	4	3	2	1	0
KEY2							

- **KEY2: Off-chip Memory Scrambling (OCMS) Key Part 2**

When Off-chip Memory Scrambling is enabled, the data scrambling depends on KEY1 and KEY2 values.

29.8.15 MPDDRC Memory Device Register

Name: MPDDRC_MD

Address: 0xFFFFEA20

Access: Read/Write

Reset: See Table 29-25

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

- **MD: Memory Device**

Indicates the type of memory used.

Reset value is that for the DDR-SDRAM device.

Value	Name	Description
3	LPDDR_SDRAM	Low-power DDR1-SDRAM
6	DDR2_SDRAM	DDR2-SDRAM
7	LPDDR2_SDRAM	Low-power DDR2-SDRAM

- **DBW: Data Bus Width**

Reset value is 16 bits.

0 (DBW_32_BITS): Data bus width is 32 bits.

1 (DBW_16_BITS): Data bus width is 16 bits.(1)

Note: 1. Only 32-bit value is used.

29.8.16 DDRSDRC High Speed Register

Name: MPDDRC_HS

Address: 0xFFFFEA24

Access: Read/Write

Reset: See Table 29-25

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

- **DIS_ANTICIP_READ: Disable Anticip Read Access**

This field allows DDR read access optimization with the multi-port.

As this feature is based on the “bank open policy”, the software must map different buffers in different DDR banks to take advantage of that feature.

0: Anticip_read access is enabled (default).

1: Anticip_read access is disabled.

29.8.17 MPDDRC Write Protect Mode Register

Name: MPDDRC_WPMR

Address: 0xFFFFEAE4

Access: Read/Write

Reset: See Table 29-25

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 0x444452 (“DDR” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x444452 (“DDR” in ASCII).

- **WPKEY: Write Protection KEY**

Value	Name	Description
0x444452	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

29.8.18 MPDDRC Write Protect Status Register

Name: MPDDRC_WPSR

Address: 0xFFFFEAE8

Access: Read-only

Reset: See Table 29-25

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 Enable**

0: No Write Protection Violation occurred since the last read of this register (MPDDRC_WPSR).

1: A Write Protection Violation occurred since the last read of this register (MPDDRC_WPSR). If this violation is an unauthorized attempt to write a control register, the associated violation is reported into the WPVSRC field.

- **WPVSRC: Write Protection Violation Source**

When WPVS is active, it indicates the register (through address or code) that should have been written if Write Protection had not been previously enabled.

29.8.19 MPDDRC DLL Master Offset Register

Name: MPDDRC_DLL_MO

Address: 0xFFFFEA74

Access: Read/Write

Reset: See Table 29-25

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SELOFF
15	14	13	12	11	10	9	8
–	–	–	CLK90OFF				
7	6	5	4	3	2	1	0
–	–	–	–	MOFF			

- **MOFF: DLL Master Delay Line Offset**

The value stored by this field is unsigned.

When this field is written, the programmable Master delay line offset is written.

When this field is read:

– If SELOFF = 0: the hard-coded Master delay line offset is read.

– If SELOFF = 1: the programmable Master delay line offset is read.

- **CLK90OFF: DLL CLK90 Delay Line Offset**

The value stored by this field is signed.

When this field is written, the programmable CLK90 delay line offset is written.

When this field is read:

– If SELOFF = 0: the hard-coded CLK90 delay line offset is read.

– If SELOFF = 1: the programmable CLK90 delay line offset is read.

- **SELOFF: DLL Offset Selection**

0: The hard-coded Master/Slave x/CLK90 delay line offsets are selected.

1: The programmable Master/Slave x/CLK90 delay line offsets are selected.

29.8.20 MPDDRC DLL Slave Offset Register

Name: MPDDRC_DLL_SOF

Address: 0xFFFFEA78

Access: Read/Write

Reset: See Table 29-25

31	30	29	28	27	26	25	24
–	–	–	S3OFF				
23	22	21	20	19	18	17	16
–	–	–	S2OFF				
15	14	13	12	11	10	9	8
–	–	–	S1OFF				
7	6	5	4	3	2	1	0
–	–	–	S0OFF				

- **SxOFF: DLL Slave x Delay Line Offset ([x = 0..3])**

The value stored by this field is signed.

When this field is written, the programmable Slave x delay line offset is written.

When this field is read:

– If MPDDRC_DLL MOR.SELOFF = 0: the hard-coded Slave x delay line offset is read.

– If MPDDRC_DLL MOR.SELOFF = 1: the programmable Slave x delay line offset is read.

29.8.21 MPDDRC DLL Master Status Register

Name: MPDDRC_DLL_MS

Address: 0xFFFFEA7C

Access: Read-only

Reset: See Table 29-25

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

- **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 MPDDRC clock frequency is too low compared to Master delay line number of elements.

- **MDVAL: DLL Master Delay Value**

Value of the Master delay counter.

29.8.22 MPDDRC DLL Status Slave x Register

Name: MPDDRC_DLL_SSx

Address: 0xFFFFEA80

Access: Read-only

Reset: See Table 29-25

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

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

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

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

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

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

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

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

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

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

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

Value of the Slave x delay counter.

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

Value of the correction applied to the Slave x delay.

30. Static Memory Controller (SMC)

30.1 Description

This SMC is capable of handling several types of external memory and peripheral devices, such as SRAM, PSRAM, PROM, EPROM, EEPROM, LCD Module, NOR Flash and NAND Flash.

The SMC generates the signals that control the access to external memory devices or peripheral devices. It has 4 Chip Selects and a 26-bit address bus. The 16-bit data bus can be configured to interface with 8- or 16-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 embeds a NAND Flash Controller (NFC). The NFC can handle automatic transfers, sending the commands and address cycles to the NAND Flash and transferring the contents of the page (for read and write) to the NFC SRAM. It minimizes the CPU overhead.

The SMC includes programmable hardware error correcting code with one-bit error correction capability and supports two-bit error detection. In order to improve the overall system performance, the DATA phase of the transfer can be DMA-assisted.

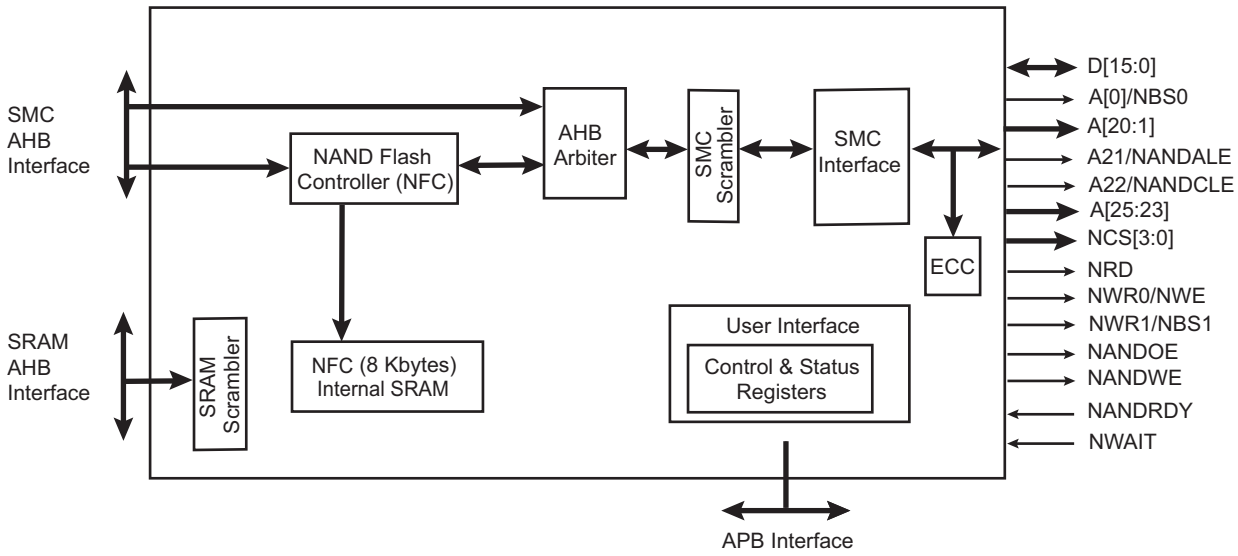
The External Data Bus can be scrambled/unscrambled by means of user keys.

30.2 Embedded Characteristics

- 64-Mbyte Address Space per Chip Select
- 8- or 16-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
- External Data Bus Scrambling/Unscrambling Function
- External Wait Request
- Automatic Switch to Slow Clock Mode
- Hardware Configurable Number of Chip Selects from 1 to 4
- Programmable Timing on a per Chip Select Basis
- NAND Flash Controller Supporting NAND Flash with Multiplexed Data/Address Buses
- Supports SLC and MLC NAND Flash Technology
- Supports NAND Flash Devices with 8-bit Data Path
- Multibit Error Correcting Code (ECC)
- ECC Algorithm Based on Binary Shortened Bose, Chaudhuri and Hocquenghem (BCH) Codes
- Programmable Error Correcting Capability: 2, 4, 8, 12 and 24 Bits of Errors per Block
- Programmable Block Size: 512 Bytes or 1024 Bytes
- Programmable Number of Block per Page: 1, 2, 4 or 8 Blocks of Data per Page
- Programmable Spare Area Size up to 512 bytes
- Supports Spare Area ECC Protection
- Supports 8 Kbytes Page Size Using 1024 Bytes/block and 4 Kbytes Page Size Using 512 Bytes/block
- MultibIt Error Detection Is Interrupt Driven
- 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

30.3 Block Diagram

Figure 30-1. Block Diagram



30.4 I/O Lines Description

Table 30-1. I/O Line Description

Name	Description	Type	Active Level
NCS[3:0]	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
A[25:1]	Address Bus	Output	–
D[15:0]	Data Bus	I/O	–
NWAIT	External Wait Signal	Input	Low
NANDRDY	NAND Flash Ready/Busy	Input	–
NANDWE	NAND Flash Write Enable	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDALE	NAND Flash Address Latch Enable	Output	–
NANDCLE	NAND Flash Command Latch Enable	Output	–

30.5 Multiplexed Signals

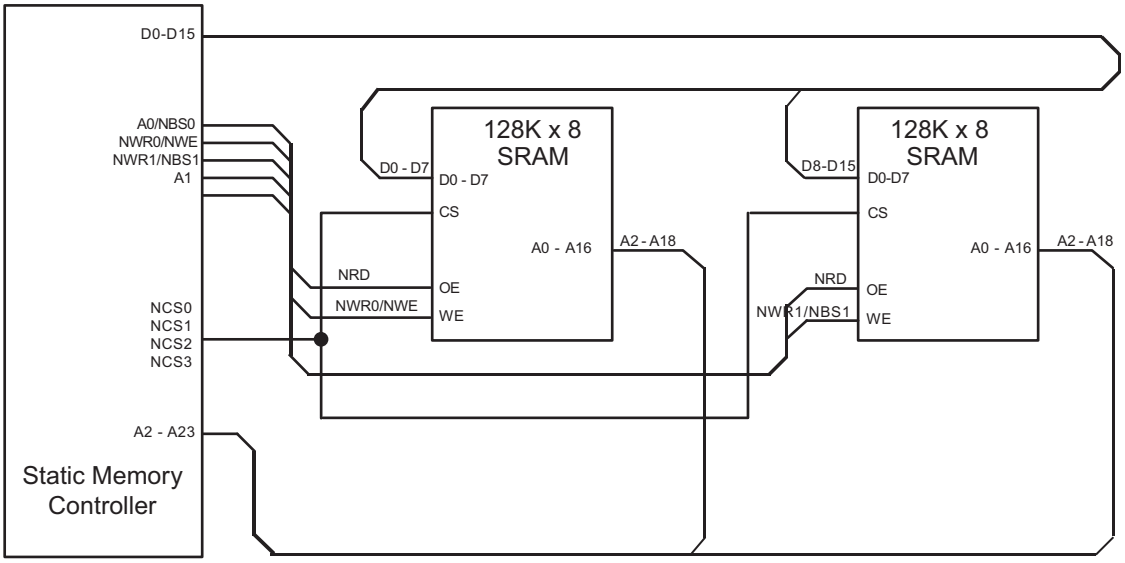
Table 30-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals		Related Function
NWR0	NWE	Byte-write or Byte-select access, see Figure 30-4 "Memory Connection for an 8-bit Data Bus" and Figure 30-5 "Memory Connection for a 16-bit Data Bus"
A0	NBS0	8-bit or 16-bit data bus, see Section 30.9.1 “Data Bus Width”
A22	NANDCLE	NAND Flash Command Latch Enable
A21	NANDALE	NAND Flash Address Latch Enable
NWR1	NBS1	Byte-write or Byte-select access, see Figure 30-4 and Figure 30-5
A1	–	8-/16-bit data bus, see Section 30.9.1 “Data Bus Width” Byte-write or Byte-select access, see Figure 30-4 and Figure 30-5

30.6 Application Example

30.6.1 Hardware Interface

Figure 30-2. SMC Connections to Static Memory Devices



30.7 Product Dependencies

30.7.1 I/O Lines

The pins used for interfacing the Static Memory Controller are 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.

30.7.2 Power Management

The SMC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SMC clock.

30.7.3 Interrupt

The SMC has an interrupt line connected to the Nested Vector Interrupt Controller (NVIC). Handling the SMC interrupt requires programming the NVIC before configuring the SMC.

Table 30-3. Peripheral IDs

Instance	ID
SMC	5

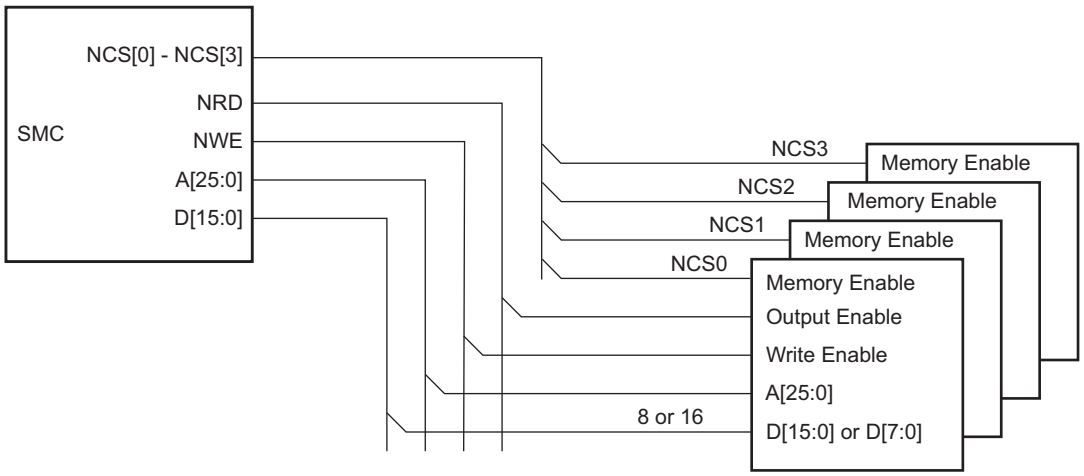
30.8 External Memory Mapping

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 30-3).

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

Figure 30-3. Memory Connections for External Devices



30.9 Connection to External Devices

30.9.1 Data Bus Width

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

Figure 30-4 shows how to connect a 512K x 8-bit memory on NCS2. Figure 30-5 shows how to connect a 512K x 16-bit memory on NCS2.

30.9.2 Byte Write or Byte Select Access

Each chip select with a 16-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 HSMC_MODE register for the corresponding chip select.

Figure 30-4. Memory Connection for an 8-bit Data Bus

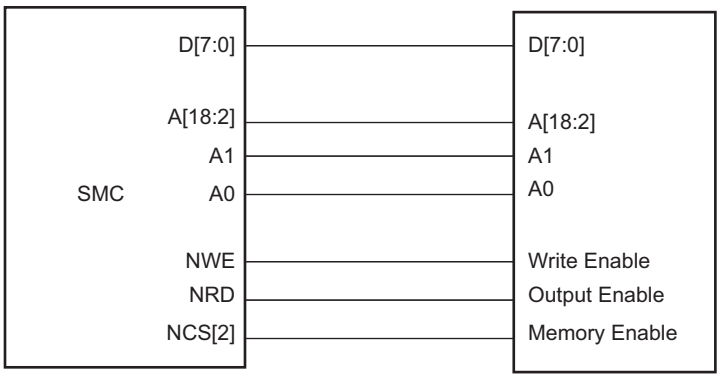
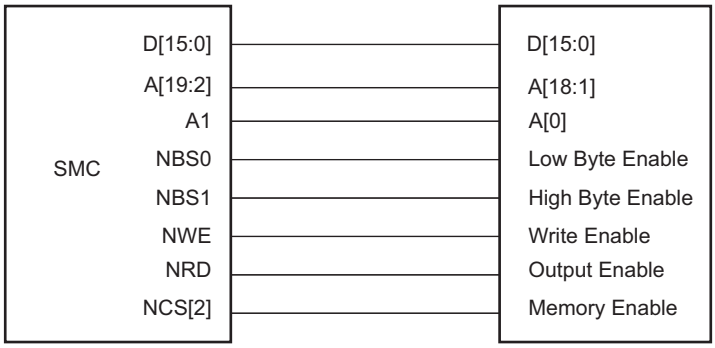


Figure 30-5. Memory Connection for a 16-bit Data Bus



30.9.2.1 Byte Write Access

Byte write access supports one 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.

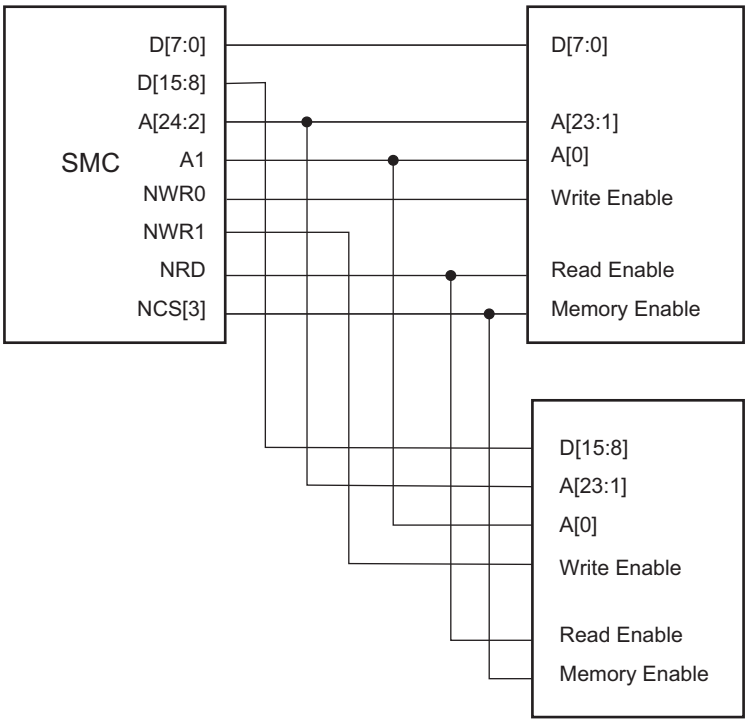
30.9.2.2 Byte Select Access

In this mode, read/write operations can be enabled/disabled at 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.

Figure 30-6. Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option



30.9.2.3 Signal Multiplexing

Depending on the Byte Access Type (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 30-4 shows signal multiplexing depending on the data bus width and the Byte Access Type.

For 16-bit devices, bit A0 of address is unused. When Byte Select Option is selected, NWR1 is unused. When Byte Write option is selected, NBS0 is unused.

Table 30-4. SMC Multiplexed Signal Translation

Device Type	Signal Name		
	16-bit Bus		8-bit Bus
	1 x 16-bit	2 x 8-bit	1 x 8-bit
Byte Access Type (BAT)	Byte Select	Byte Write	–
NBS0_A0	NBS0	–	A0
NWE_NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NWR1	–
A1	A1	A1	A1

30.10 Standard Read and Write Protocols

In the following sections, the Byte Access Type is not considered. Byte select lines (NBS0 to NBS1) 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 NWR1) in byte write access type. NWR0 to NWR1 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..3] chip select lines.

30.10.1 Read Waveforms

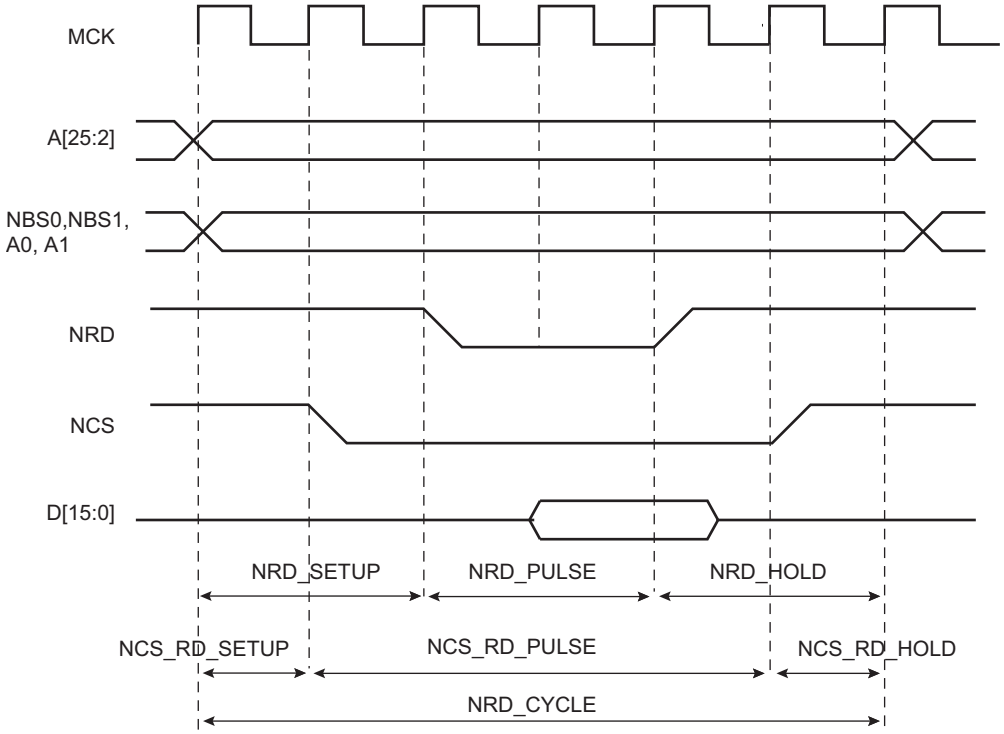
The read cycle is shown on Figure 30-7.

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

Figure 30-7. Standard Read Cycle



30.10.1.1 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.

30.10.1.2 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.

30.10.1.3 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, the 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

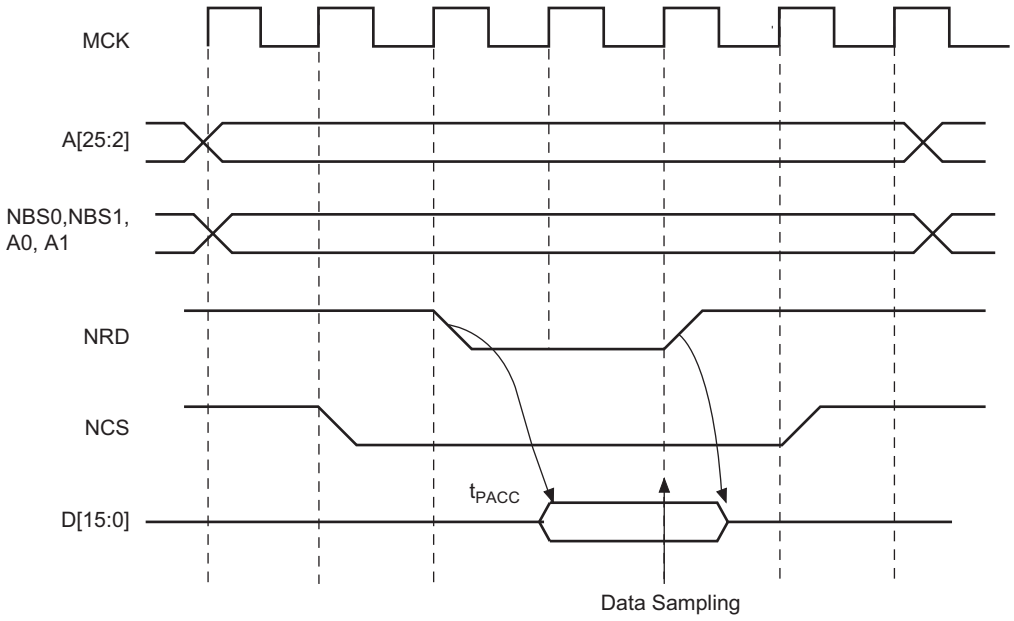
30.10.2 Read Mode

As NCS and NRD waveforms are defined independently of one another, 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 HSMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

30.10.2.1 Read is Controlled by NRD (READ_MODE = 1)

Figure 30-8 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 the Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS.

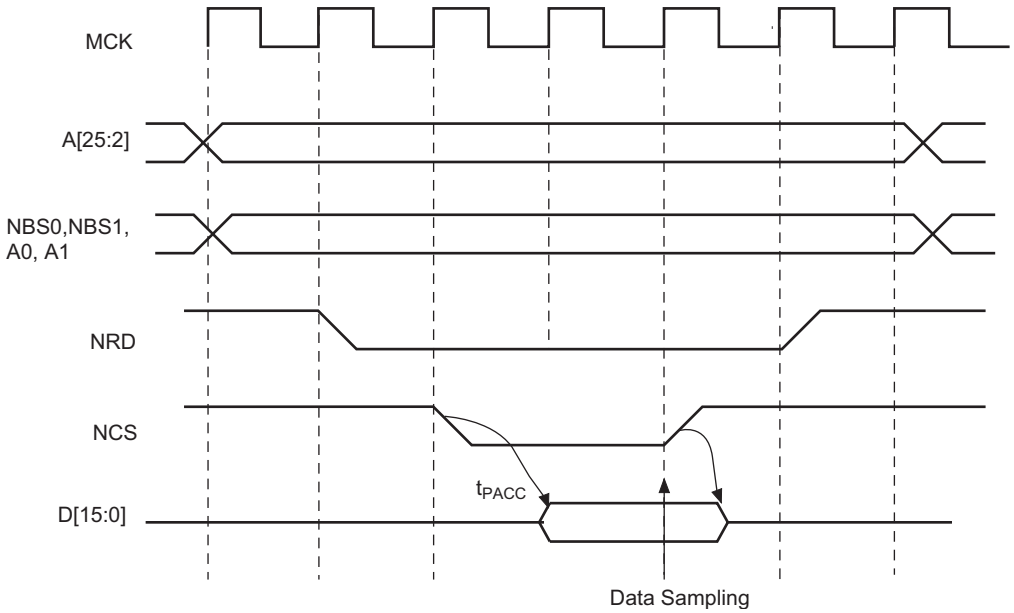
Figure 30-8. READ_MODE = 1: Data is Sampled by SMC before the Rising Edge of NRD



30.10.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 30-9 shows the typical read cycle. 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 the Master Clock that generates the rising edge of NCS, whatever the programmed waveform of NRD.

Figure 30-9. READ_MODE = 0: Data is Sampled by SMC before the Rising Edge of NCS



30.10.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 30-10. The write cycle starts with the address setting on the memory address bus.

30.10.3.1 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.

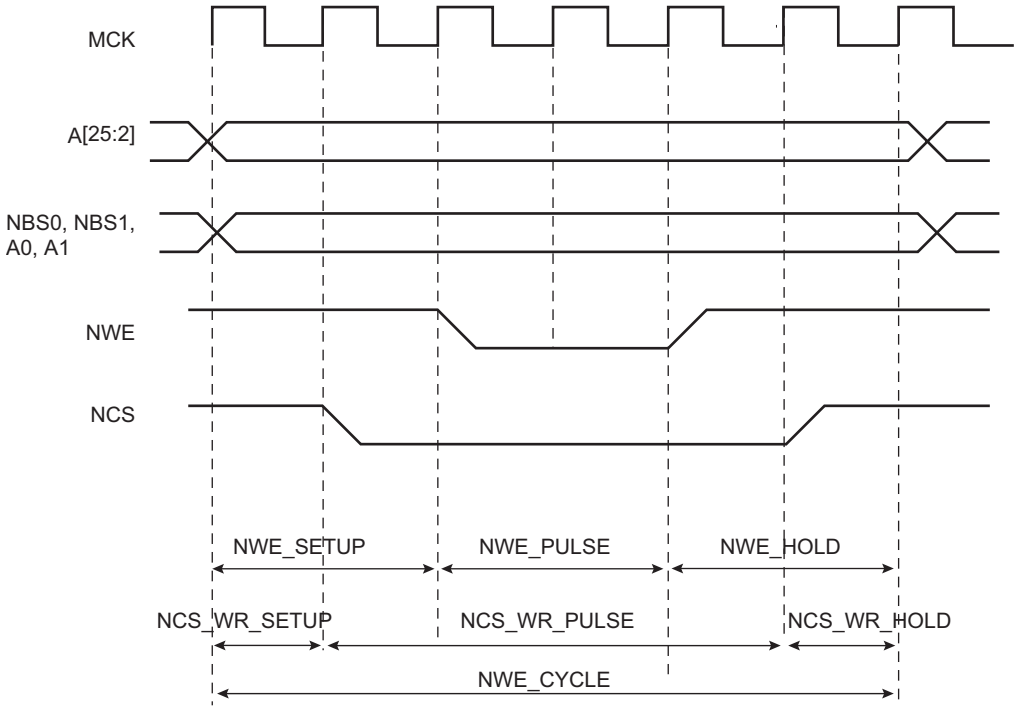
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

30.10.3.2 NCS Waveforms

The NCS signal waveforms in write operations are not the same as 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 30-10.Write Cycle



30.10.3.3 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

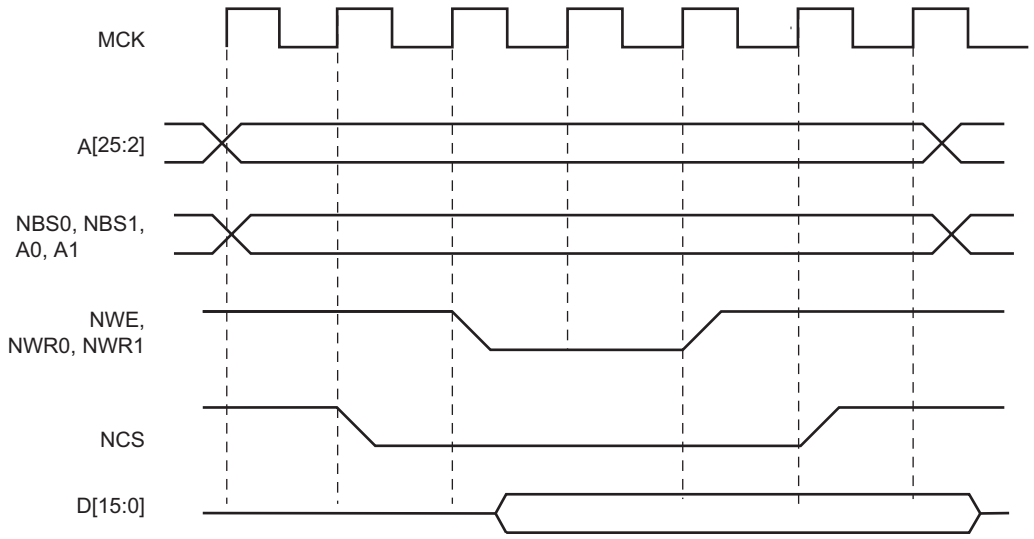
30.10.4 Write Mode

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

30.10.4.1 Write is Controlled by NWE (WRITE_MODE = 1)

Figure 30-11 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.

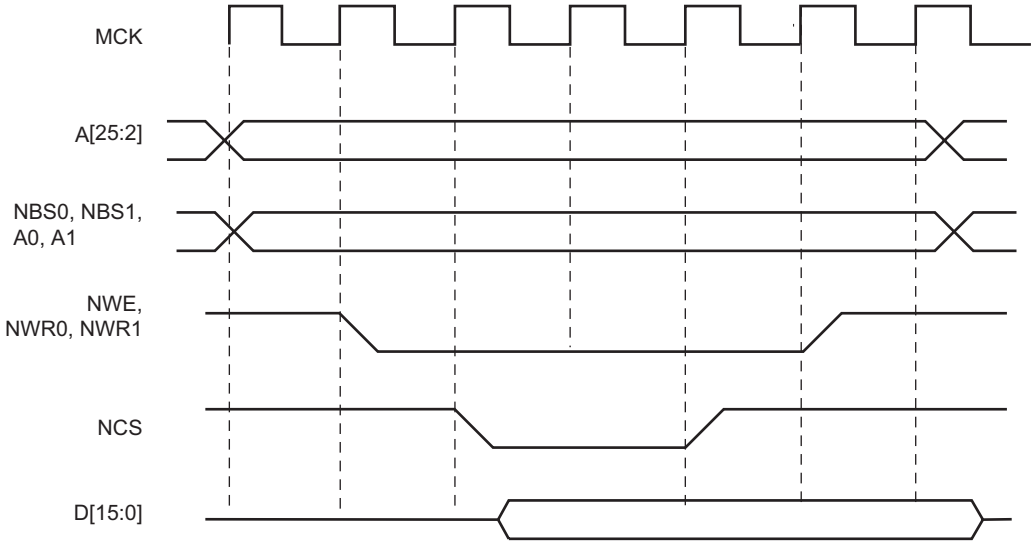
Figure 30-11.WRITE_MODE = 1. The write operation is controlled by NWE



30.10.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 30-12 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 30-12.WRITE_MODE = 0. The write operation is controlled by NCS



30.10.5 Coding Timing Parameters

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

- The HSMC_SETUP register groups the definition of all setup parameters:
  NRD_SETUP, NCS_RD_SETUP, NWE_SETUP, NCS_WR_SETUP
- The HSMC_PULSE register groups the definition of all pulse parameters:
  NRD_PULSE, NCS_RD_PULSE, NWE_PULSE, NCS_WR_PULSE
- The HSMC_CYCLE register groups the definition of all cycle parameters:
  NRD_CYCLE, NWE_CYCLE

Table 30-5 shows how the timing parameters are coded and their permitted range.

Table 30-5. 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 ≤ setup ≤ 31	0..31
			32 ≤ setup ≤ 63	128..(128 + 31)
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤ pulse ≤ 63	0..63
			64 ≤ pulse ≤ 127	256..(256 + 63)
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤ cycle ≤ 127	0..127
			128 ≤ cycle ≤ 255	256..(256 + 127)
			256 ≤ cycle ≤ 383	512..(512 + 127)
			384 ≤ cycle ≤ 511	768..(768 + 127)

30.10.6 Reset Values of Timing Parameters

Table 30-6 gives the default value of timing parameters at reset.

Table 30-6. Reset Values of Timing Parameters

Register	Reset Value	Description
HSMC_SETUP	–	All setup timings are set to 1
HSMC_PULSE	–	All pulse timings are set to 1
HSMC_CYCLE	–	The read and write operations last 3 Master Clock cycles and provide one hold cycle
WRITE_MODE	1	Write is controlled with NWE
READ_MODE	1	Read is controlled with NRD

30.10.7 Usage Restriction

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

30.10.7.1 For Read Operations

Null but positive setup and hold of address and NRD and/or NCS cannot be guaranteed at the memory interface because of the propagation delay of these signals through external logic and pads. When 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.

30.10.7.2 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 Section 30.12.2 “Early Read Wait State” on page 419.

30.10.7.3 For Read and Write Operations

A null value for pulse parameters is forbidden and may lead to an 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.

30.11 Scrambling/Unscrambling Function

The external data bus D[15:0] can be scrambled in order to prevent intellectual property data located in off-chip memories from being easily recovered by analyzing data at the package pin level of either the microcontroller or the memory device.

The scrambling and unscrambling are performed on-the-fly without additional wait states.

The scrambling method depends on two user-configurable key registers, HSMC_KEY1 and HSMC_KEY2. These key registers are only accessible in write mode.

The key must be securely stored in a reliable non-volatile memory in order to recover data from the off-chip memory. Any data scrambled with a given key cannot be recovered if the key is lost.

The scrambling/unscrambling function can be enabled or disabled by programming the HSMC_OCMS register.

One bit is dedicated to enabling/disabling the NAND Flash scrambling, and one bit is dedicated to enabling/disabling the off chip SRAM scrambling. When at least one external SRAM is scrambled, the SMSC field must be set in the HSMC_OCMS register.

When multiple chip selects (external SRAM) are handled, it is possible to configure the scrambling function per chip select using the OCMS field in the HSMC_TIMINGS registers.

To scramble the NAND Flash contents, the SRSE field must be set in the HSMC_OCMS register.

When the NAND Flash memory content is scrambled, the on-chip SRAM page buffer associated for the transfer is also scrambled.

30.12 Automatic Wait States

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

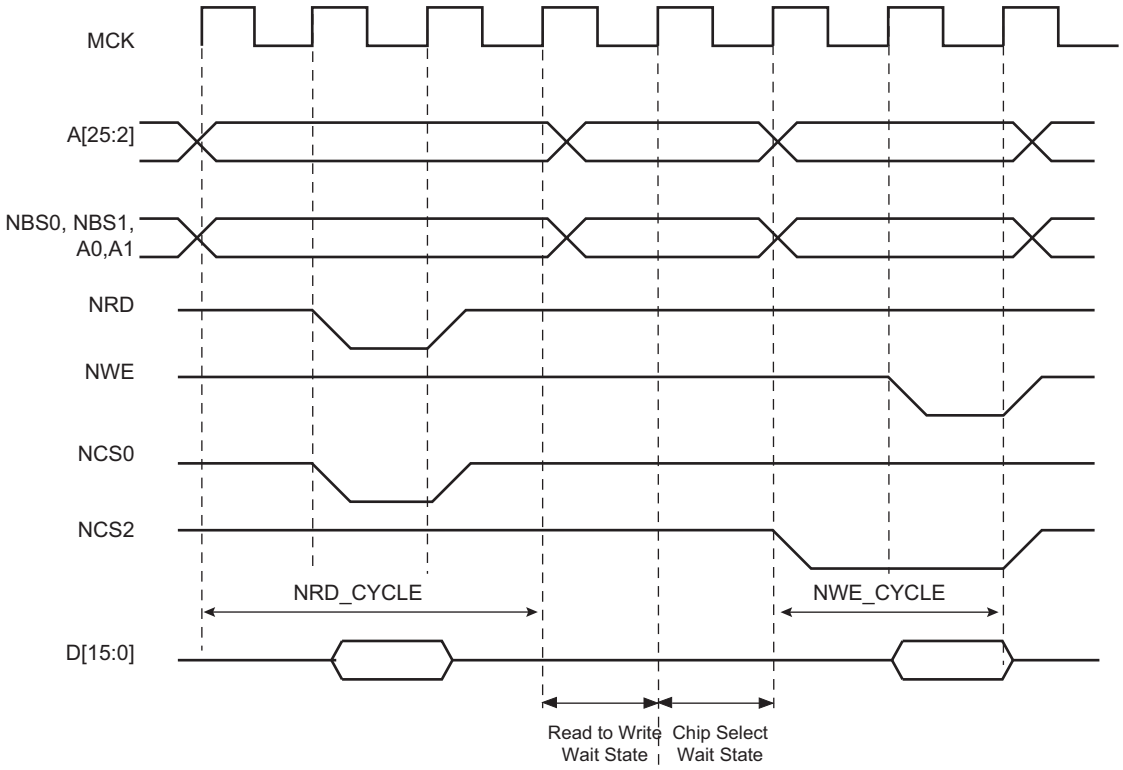
30.12.1 Chip Select Wait States

The SMC always inserts an idle cycle between two 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 NBS1, NWR0 to NWR1, NCS[0..3], and NRD lines. They are all set to 1.

Figure 30-13 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

Figure 30-13.Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



30.12.2 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 30-14).

- 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 30-15). The write operation must end with an 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 30-16.

Figure 30-14.Early Read Wait State: Write with No Hold Followed by Read with No Setup

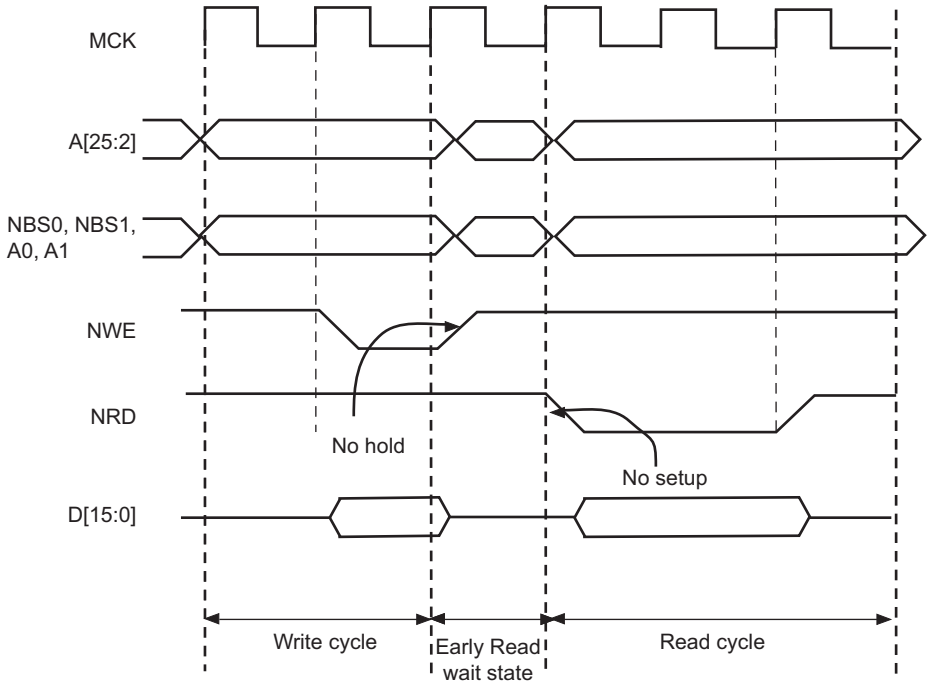


Figure 30-15.Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup

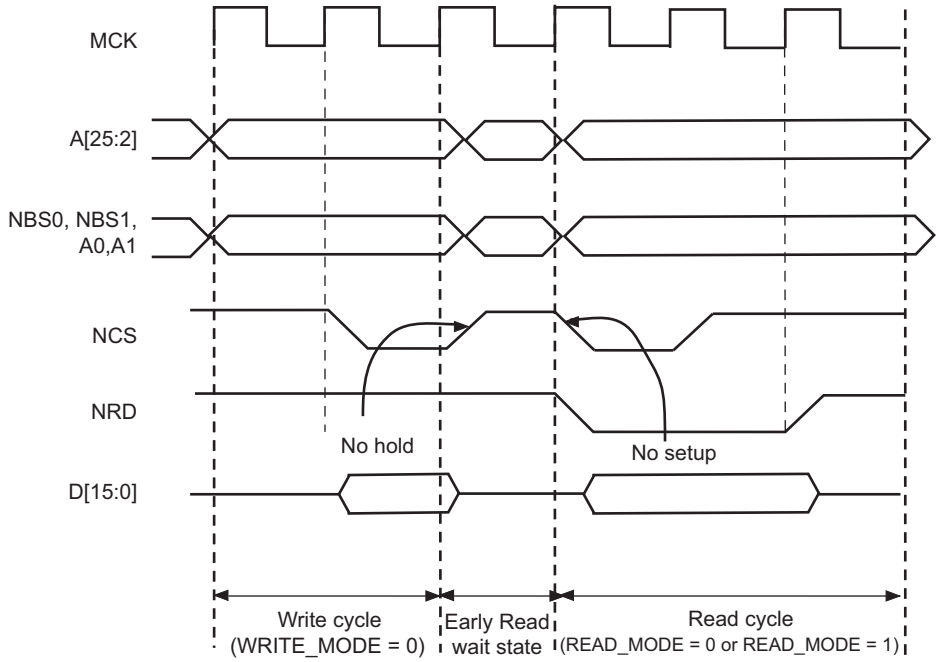
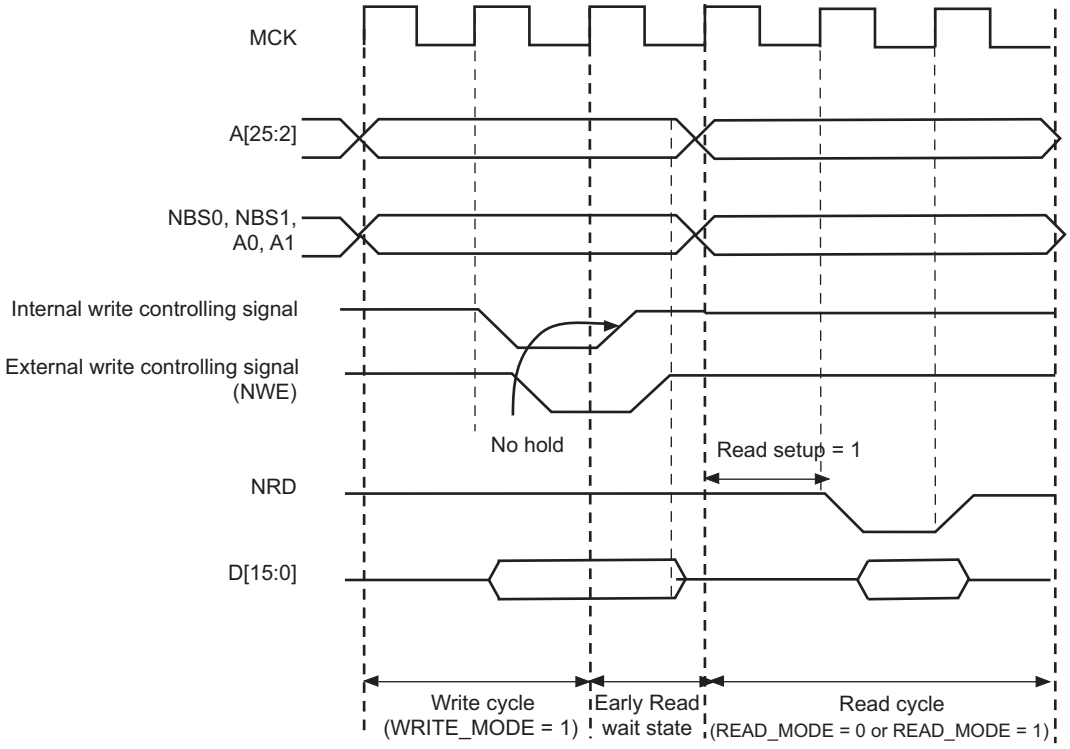


Figure 30-16.Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



30.12.3 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.

30.12.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any HSMC_MODE register of the user interface. If only the timing registers are modified (HSMC_SETUP, HSMC_PULSE, HSMC_CYCLE registers) in the user interface, the user must validate the modification by writing the HSMC_MODE register, even if no change was made on the mode parameters.

30.12.3.2 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 Section 30.15 “Slow Clock Mode” on page 432).

30.12.4 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 30-13 on page 419.

30.13 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 HSMC_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 HSMC_MODE register for the corresponding chip select.

30.13.1 READ_MODE

Setting 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 30-17 illustrates the Data Float Period in NRD-controlled mode (READ_MODE = 1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 30-18 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

Figure 30-17.TDF Period in NRD Controlled Read Access (TDF = 2)

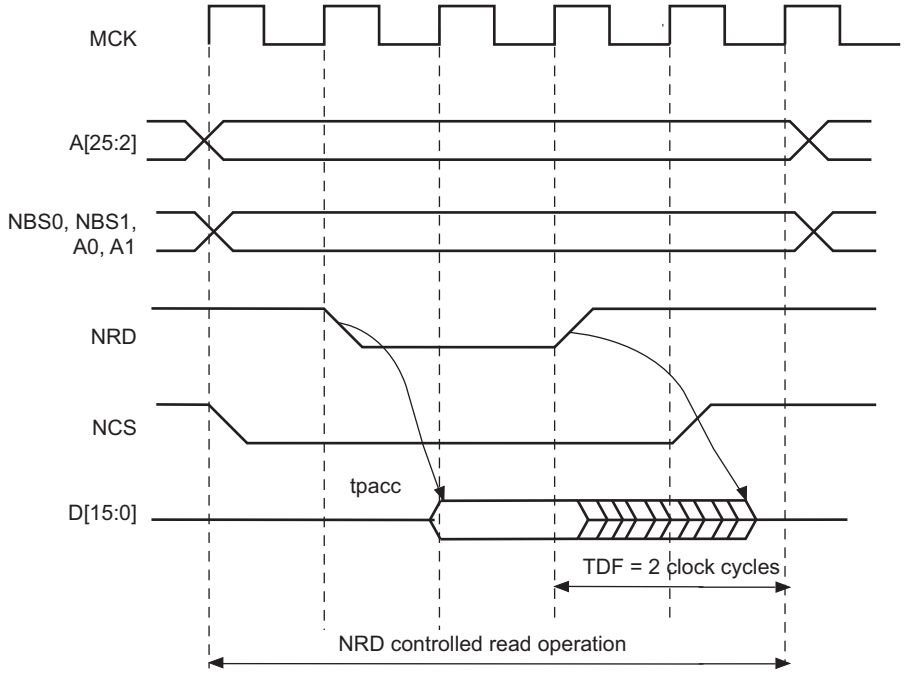
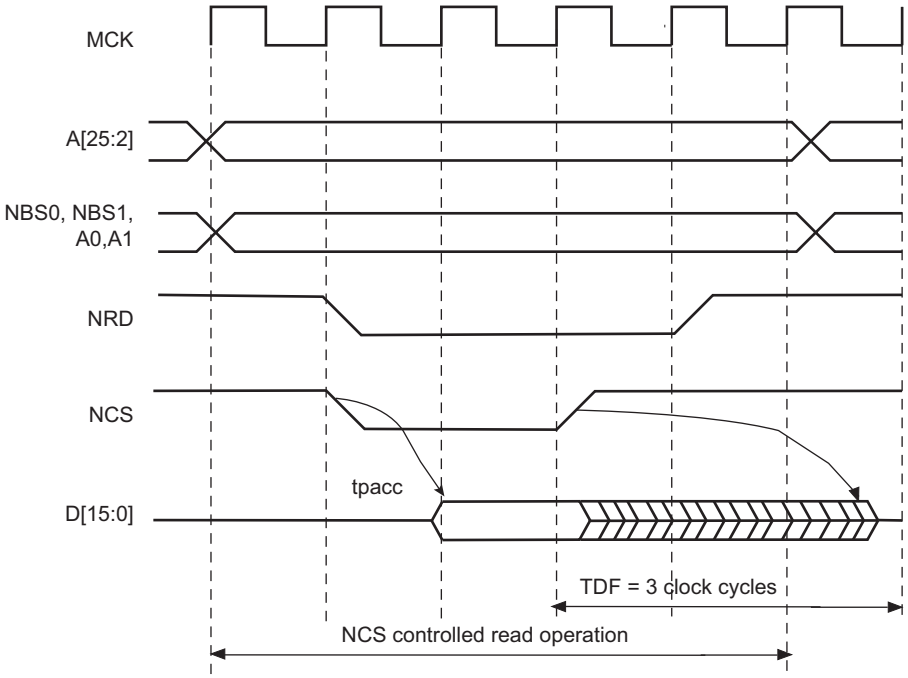


Figure 30-18.TDF Period in NCS Controlled Read Operation (TDF = 3)



30.13.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the HSMC_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.

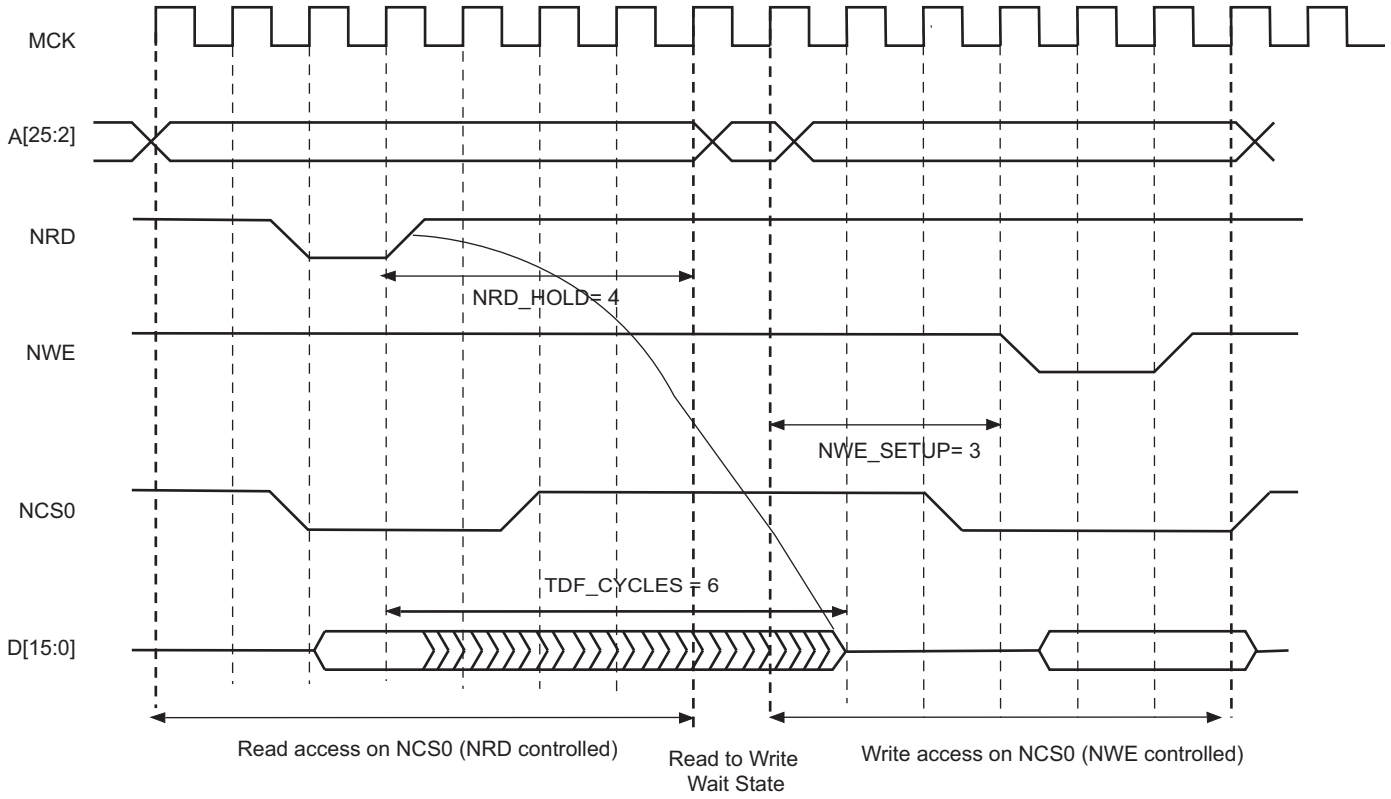
Figure 30-19 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

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 30-19.TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



30.13.3 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 ends 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 30-20, Figure 30-21 and Figure 30-22 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 30-20.TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

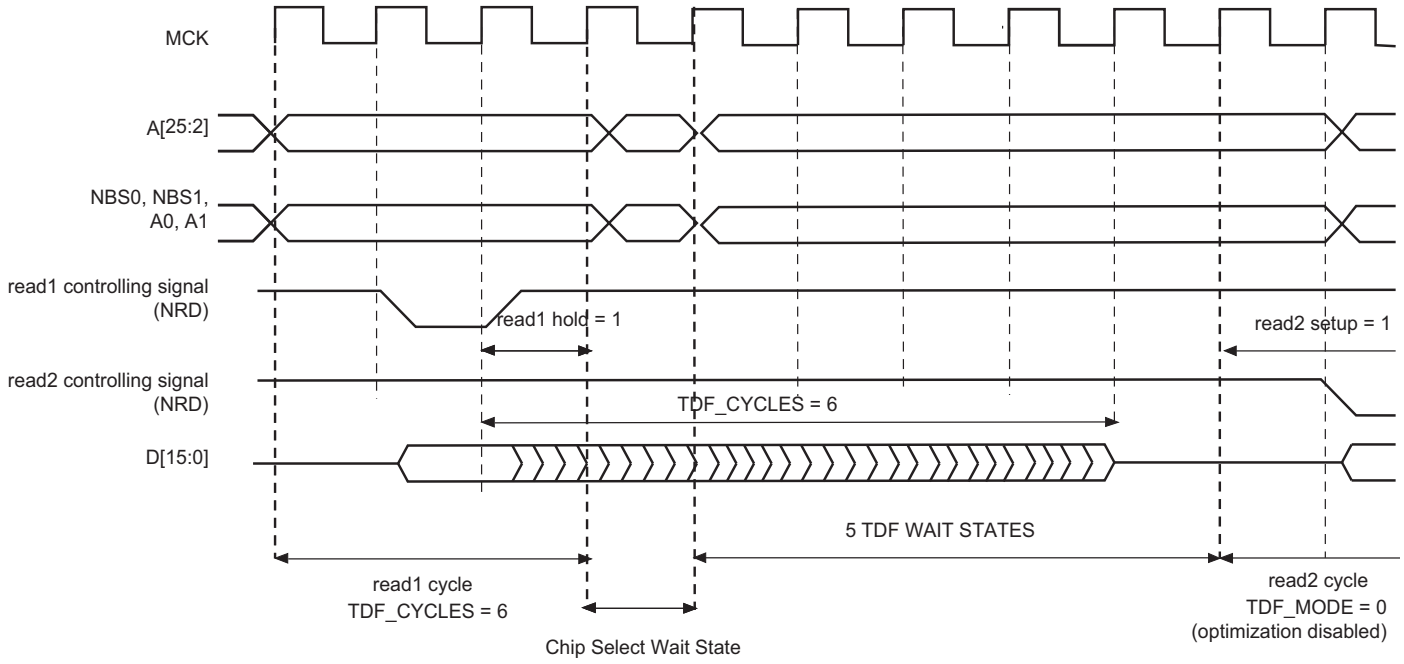


Figure 30-21. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

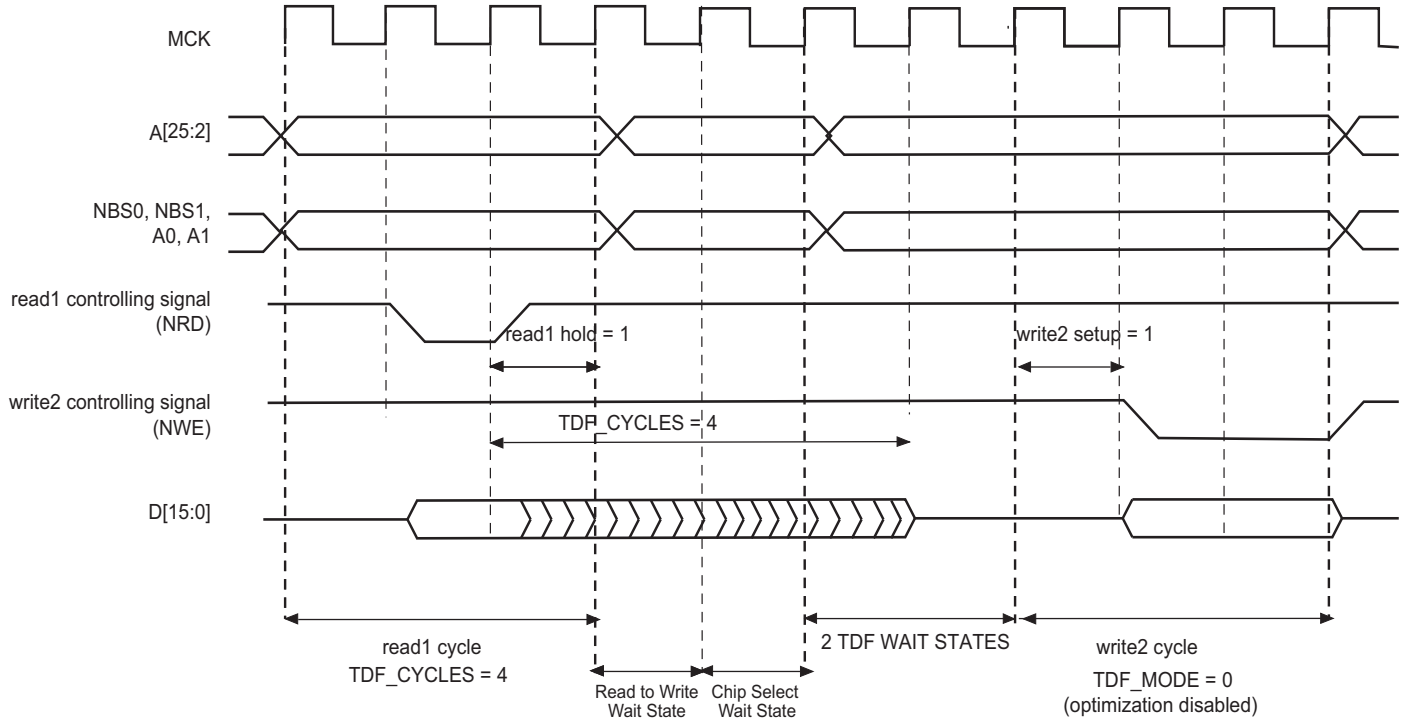
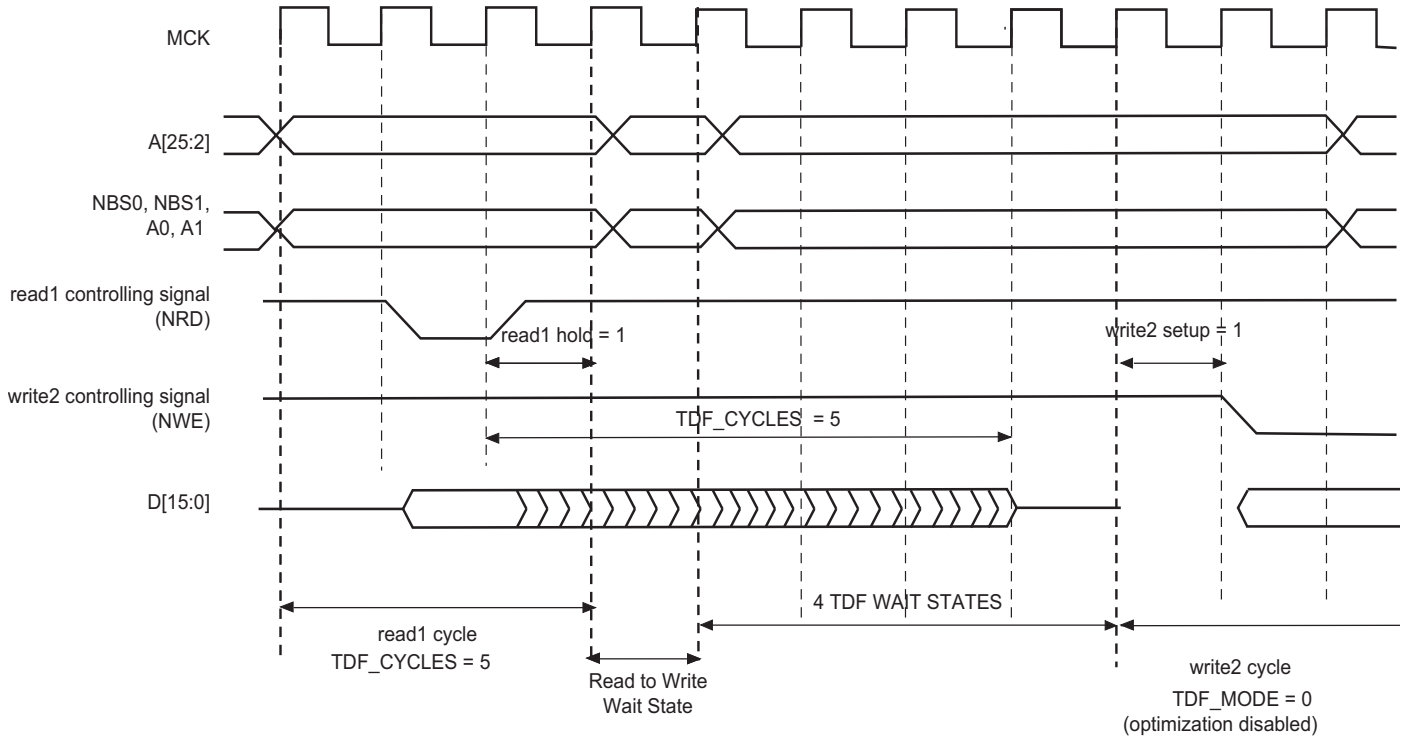


Figure 30-22.TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



30.14 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 HSMC_MODE register on the corresponding chip select must be set to either ‘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.

30.14.1 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 Slow Clock Mode (Section 30.15 “Slow Clock Mode” on page 432).

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. NWAIT is then examined by the SMC 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 the SMC behavior.

30.14.2 Frozen Mode

When the external device asserts the NWAIT signal (active low), and after an 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 30-23. 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 30-24.

Figure 30-23.Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

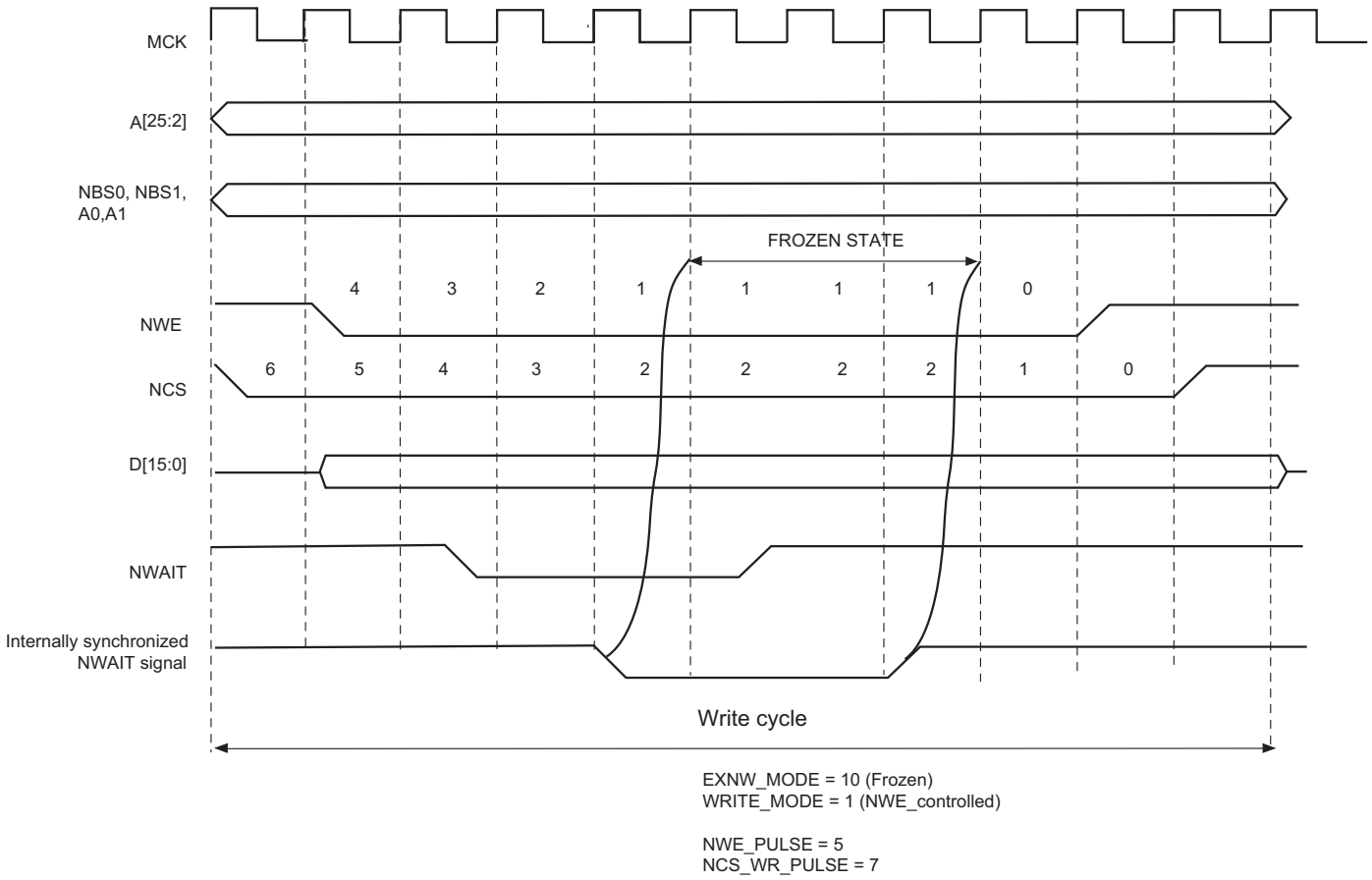
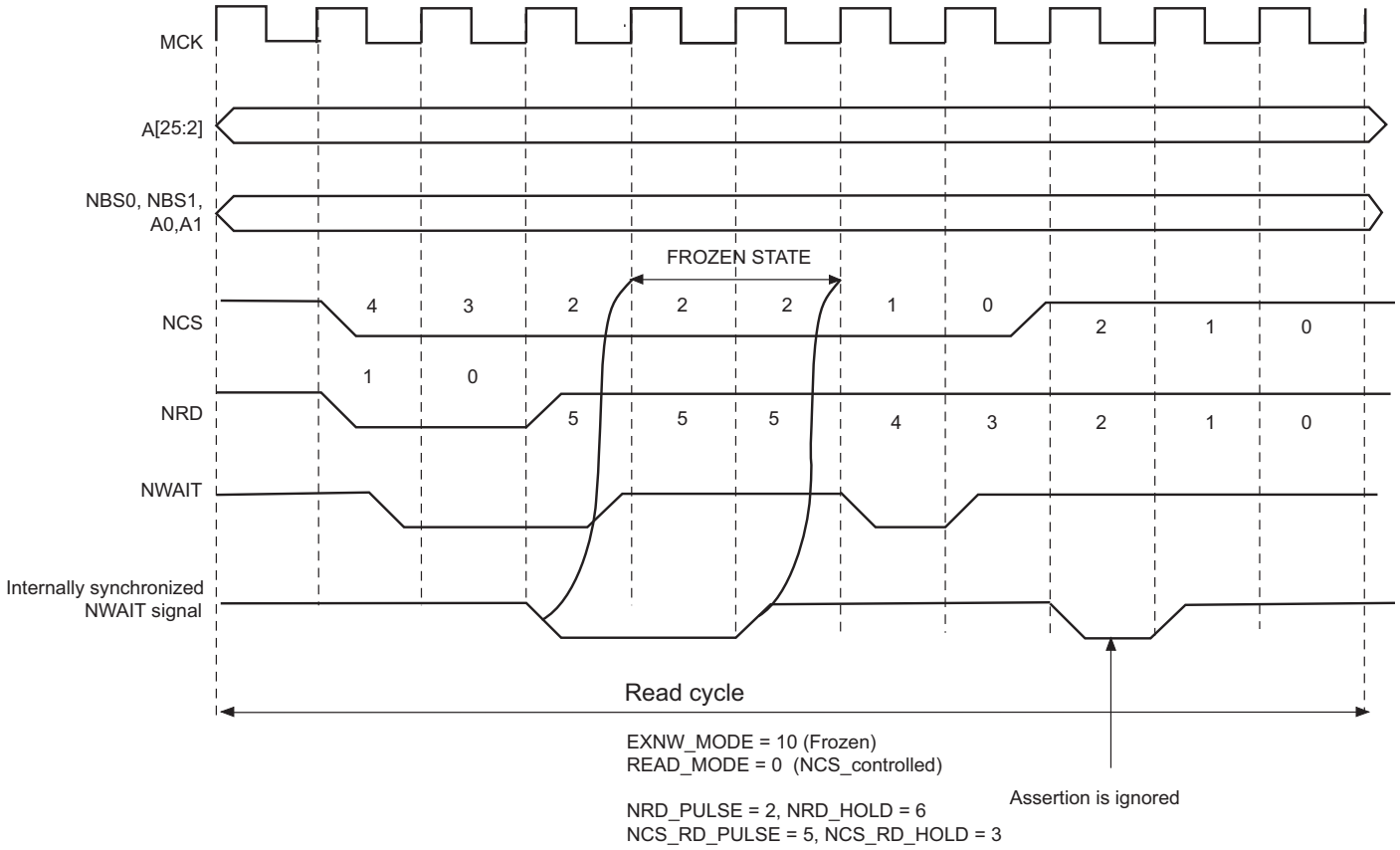


Figure 30-24.Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



30.14.3 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 30-25 and Figure 30-26. 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 30-26.

Figure 30-25.NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)

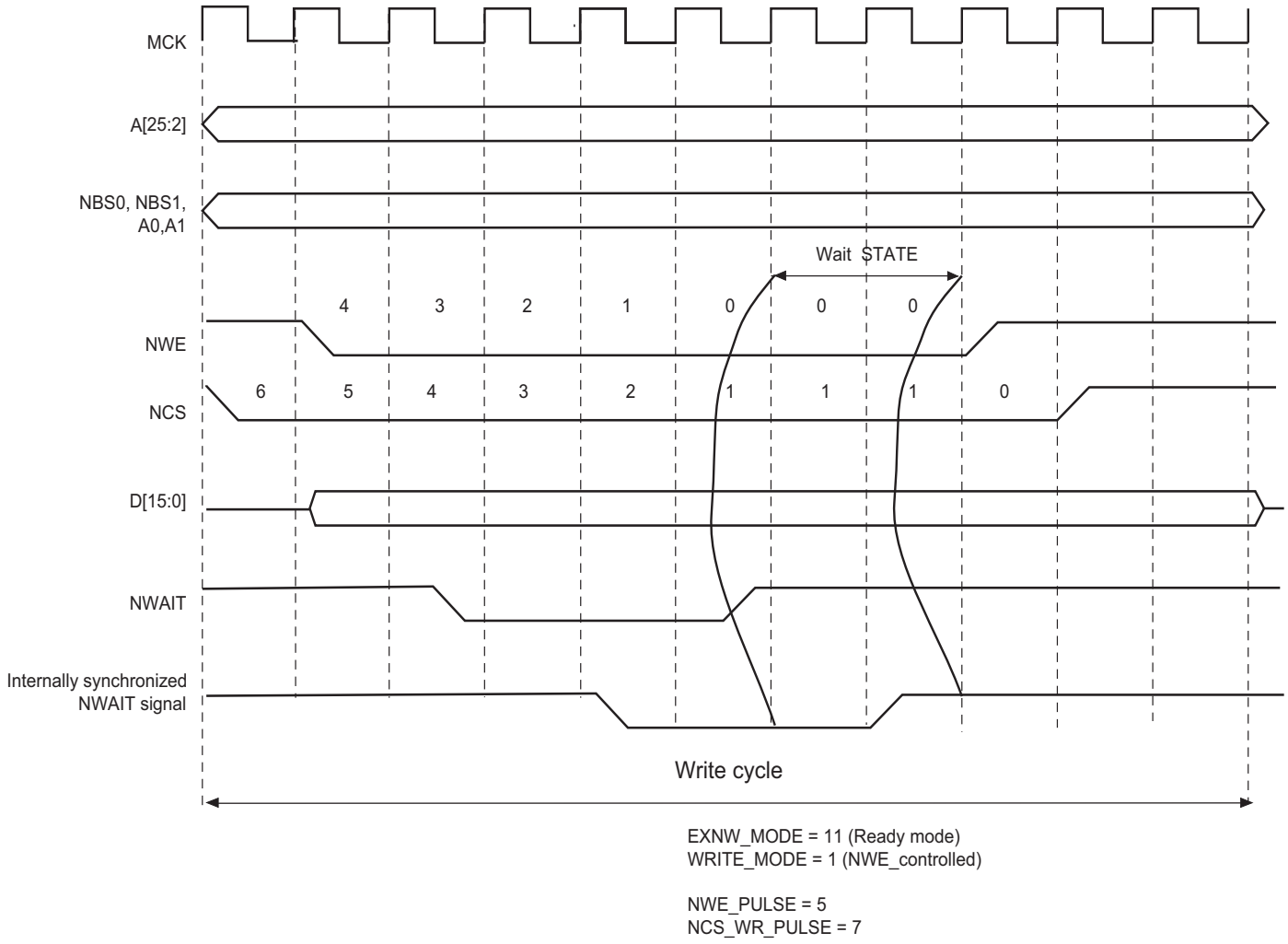
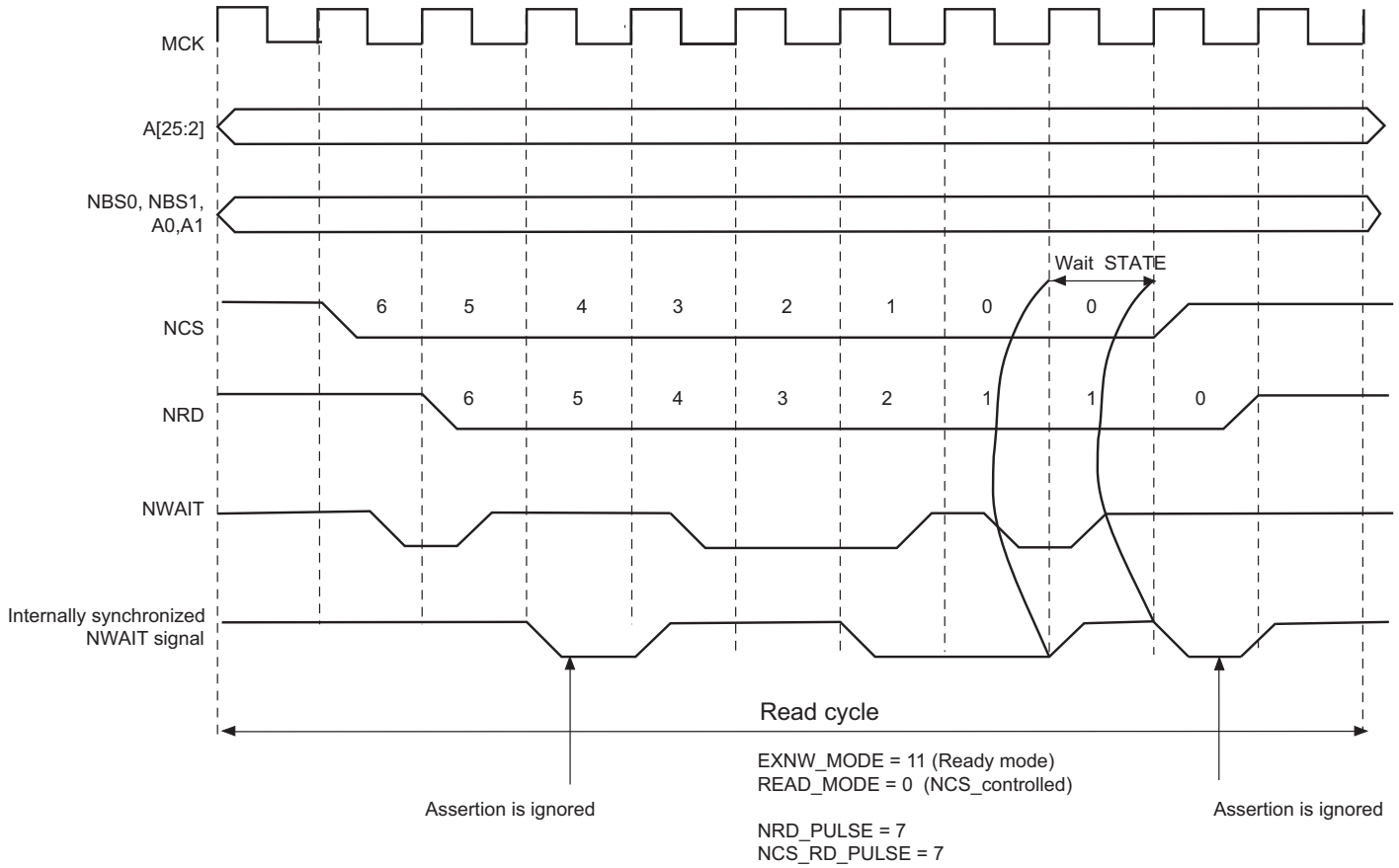


Figure 30-26.NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)



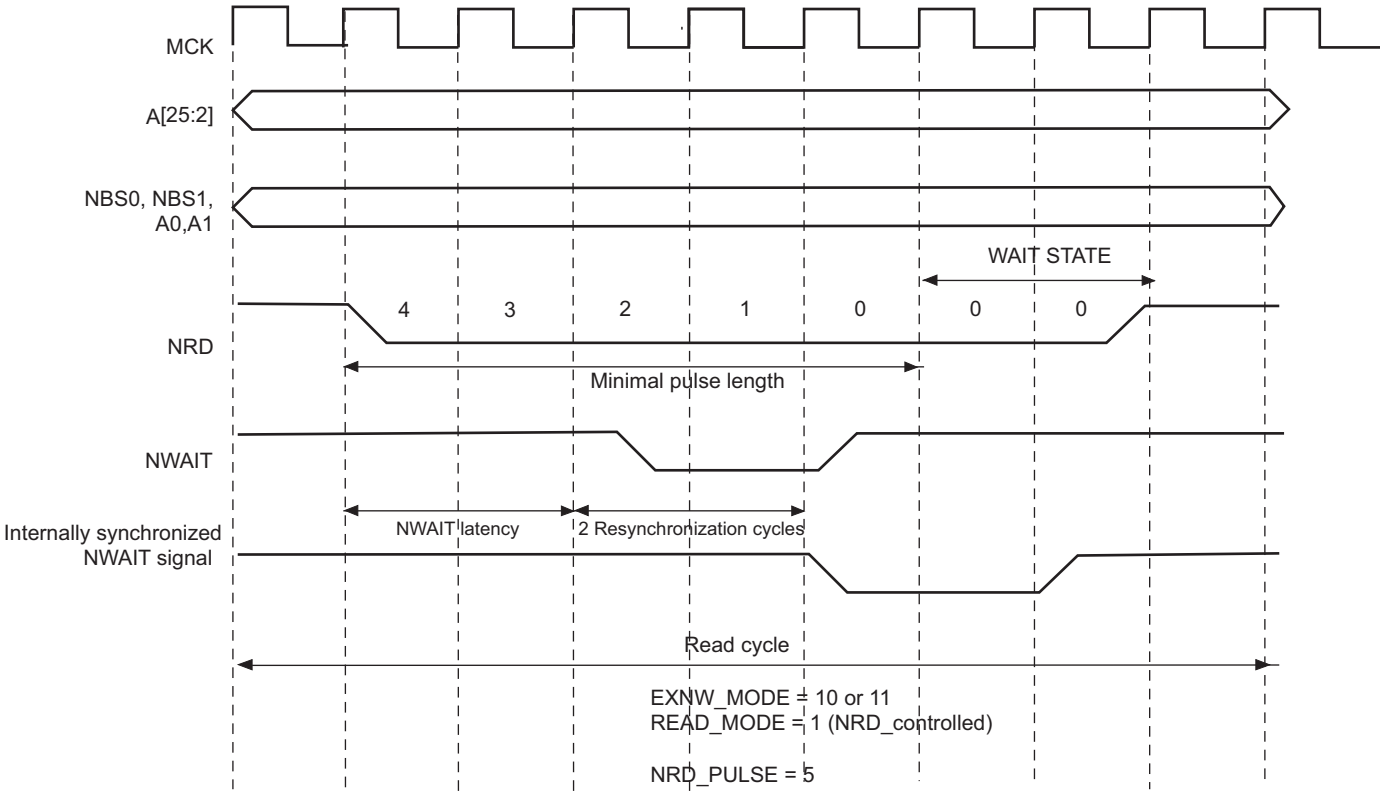
30.14.4 NWAIT Latency and Read/Write Timings

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 30-27.

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 30-27.NWAIT Latency



30.15 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.

30.15.1 Slow Clock Mode Waveforms

Figure 30-28 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 30-7 indicates the value of read and write parameters in slow clock mode.

Figure 30-28. Write/Read Cycles in Slow Clock Mode

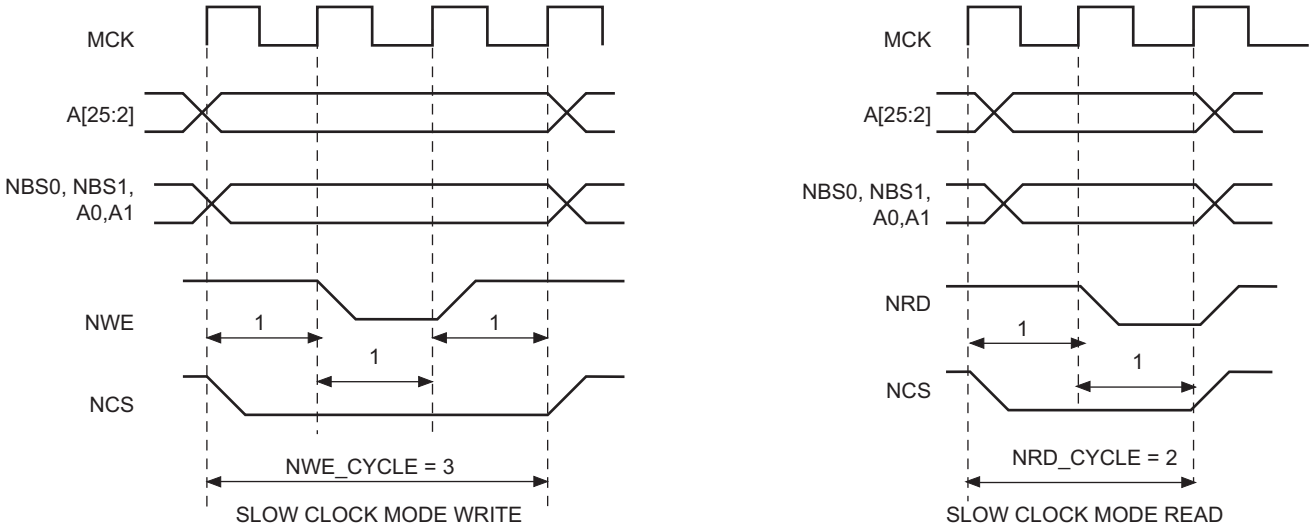


Table 30-7. Read and Write Timing Parameters in Slow Clock Mode

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

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

When switching from slow clock mode to normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters. See Figure 30-29. The external device may not be fast enough to support such timings.

Figure 30-30 illustrates the recommended procedure to properly switch from one mode to the other.

Figure 30-29.Clock Rate Transition occurs while the SMC is performing a Write Operation

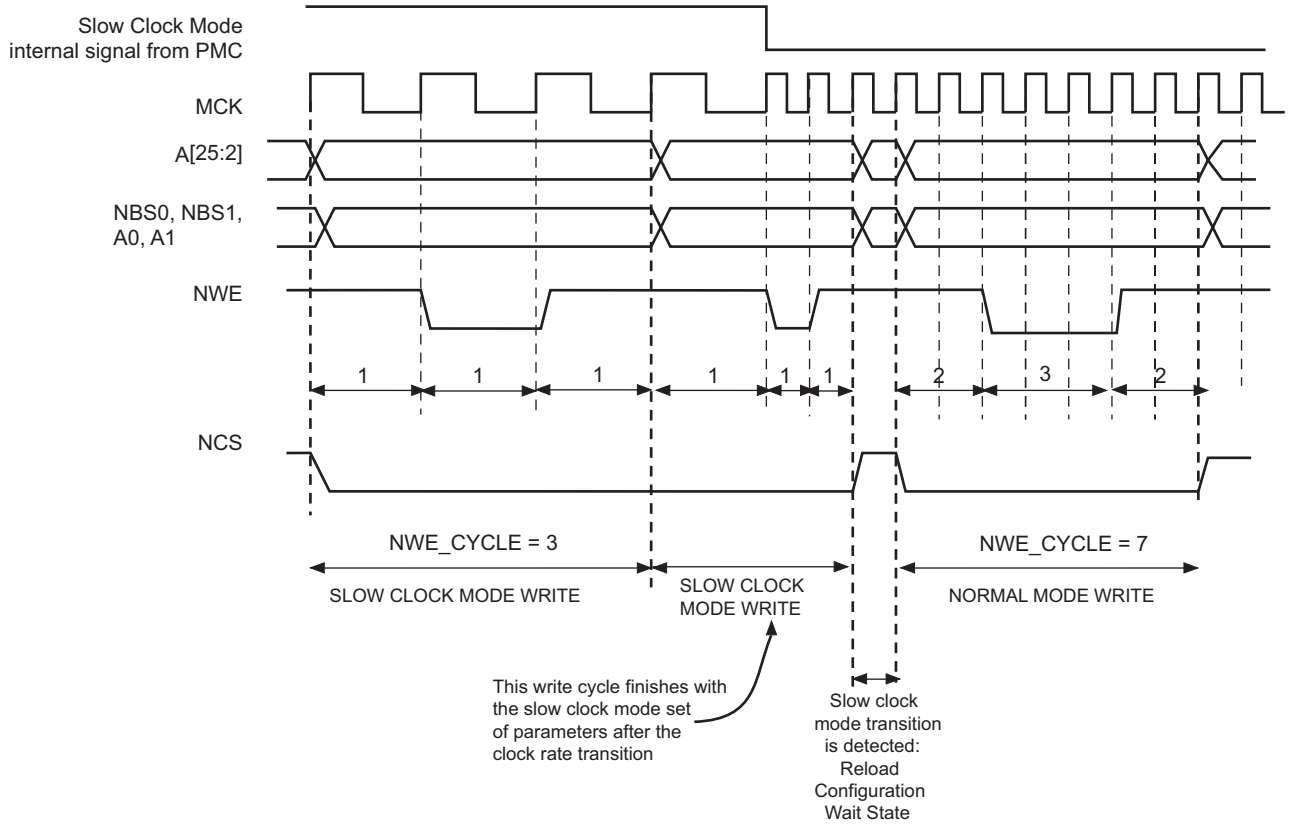
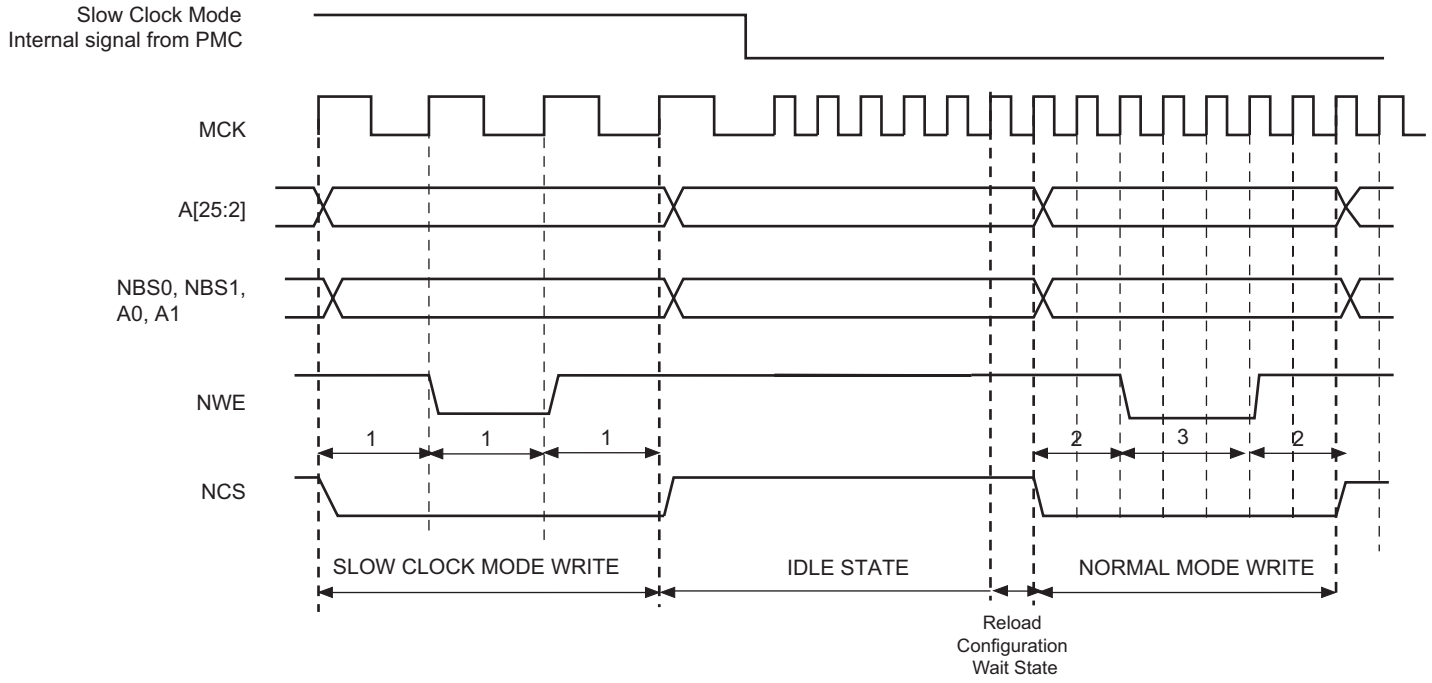


Figure 30-30.Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



30.16 Register Write Protection

To prevent any single software error that may corrupt SMC behavior, selected registers can be write-protected by setting the WPEN bit in the HSMC Write Protection Mode Register (HSMC_WPMR).

If a write access in a write-protected register is detected, then the WPVS flag in the HSMC Write Protection Status Register (HSMC_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 HSMC_WPSR.

The following registers can be write-protected:

- HSMC Setup Register
- HSMC Pulse Register
- HSMC Cycle Register
- HSMC Timings Register
- HSMC Mode Register

30.17 NAND Flash Controller Operations

30.17.1 NFC Overview

The NAND Flash Controller (NFC) handles all the command, address and data sequences of the NAND low level protocol. An SRAM is used as an internal read/write buffer when data is transferred from or to the NAND.

30.17.2 NFC Control Registers

NAND Flash Read and NAND Flash Program operations can be performed through the NFC Command Registers. In order to minimize CPU intervention and latency, commands are posted in a command buffer. This buffer provides zero wait state latency. The detailed description of the command encoding scheme is explained below.

The NFC handles an automatic transfer between the external NAND Flash and the chip via the NFC SRAM. It is done via NFC Command Registers.

The NFC Command Registers are very efficient to use. When writing to these registers:

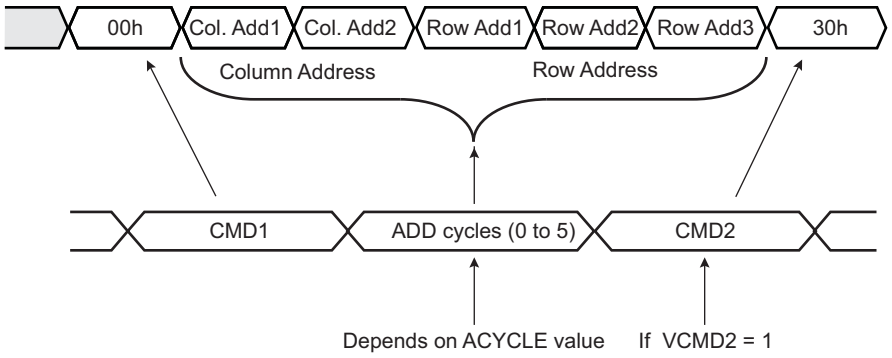
- the address of the register (NFCADDR_CMD) is the command used
- the data of the register (NFCDATA_ADDT) is the address to be sent to the NAND Flash

So, in one single access the command is sent and immediately executed by the NFC. Two commands can even be programmed within a single access (CMD1, CMD2) depending on the VCMD2 value.

The NFC can send up to five address cycles.

Figure 30-31 below shows a typical NAND Flash Page Read Command of a NAND Flash Memory and correspondence with NFC Address Command Register.

Figure 30-31.NFC/NAND Flash Access Example



For more details refer to Section 30.17.2.2 “NFC Address Command” on page 436.

Reading the NFC command register (to any address) will give the status of the NFC. This is especially useful to know if the NFC is busy, for example.

30.17.2.1 Building NFC Address Command Example

The base address is made of HOST_ADDR address.

Page read operation example:

```
// Build the Address Command (NFCADDR_CMD)
AddressCommand = (HOST_ADDR |
                  NFCCMD=1 | // NFC Command Enable
                  NFCWR=0   |// NFC Read Data from NAND Flash
                  DATAEN=1    |    // NFC Data phase Enable.
                  CSID=1        |        // Chip Select ID = 1
                 ACYCLE= 5 |     // Number of address cycle.
                 VCMD2=1    |               // CMD2 is sent after Address Cycles
                 CMD2=0x30  |               // CMD2 = 30h
                 CMD1=0x0)    // CMD1 = Read Command = 00h

  // Set the Address for Cycle 0
  HSMC_ADDR = Col. Add1

  // Write command with the Address Command built above
*AddressCommand = (Col. Add2   |// ADDR_CYCLE1
           Row Add1  | // ADDR_CYCLE2
           Row Add2  |// ADDR_CYCLE3
           Row Add3  )// ADDR_CYCLE4
```

30.17.2.2 NFC Address Command

Name: NFCADDR_CMD

Access: Read/Write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	NFCWR	DATAEN	CSID

23	22	21	20	19	18	17	16
CSID		ACYCLE			VCMD2	CMD2	

15	14	13	12	11	10	9	8
CMD2						CMD1	

7	6	5	4	3	2	1	0
CMD1						–	–

- **CMD1: Command Register Value for Cycle 1**

If NFCCMD is set, when a read or write access occurs, the NFC sends this command.

- **CMD2: Command Register Value for Cycle 2**

If NFCCMD and VCMD2 field are set to one, the NFC sends this command after CMD1.

- **VCMD2: Valid Cycle 2 Command**

When set to true, the CMD2 field is issued after the address cycle.

- **ACYCLE: Number of Address required for the current command**

When ACYCLE field is different from zero, ACYCLE Address cycles are performed after Command Cycle 1. The maximum number of cycles is 5.

- **CSID: Chip Select Identifier**

Chip select used

- **DATAEN: NFC data phase enable**

When set to true, the NFC will automatically read or write data after the command.

- **NFCWR: NFC Write Enable**

0: NFC reads data from the NAND Flash.

1: NFC writes data into the NAND Flash.

30.17.2.3 NFC Data Address

Name: NFCDATA_ADDT

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
ADDR_CYCLE4							
23	22	21	20	19	18	17	16
ADDR_CYCLE3							
15	14	13	12	11	10	9	8
ADDR_CYCLE2							
7	6	5	4	3	2	1	0
ADDR_CYCLE1							

- **ADDR_CYCLE1: NAND Flash Array Address Cycle 1**

When less than five address cycles are used, ADDR_CYCLE1 is the first byte written to the NAND Flash.

When five address cycles are used, ADDR_CYCLE1 is the second byte written to NAND Flash.

- **ADDR_CYCLE2: NAND Flash Array Address Cycle 2**

When less than five address cycles are used, ADDR_CYCLE2 is the second byte written to the NAND Flash.

When five address cycles are used, ADDR_CYCLE2 is the third byte written to the NAND Flash.

- **ADDR_CYCLE3: NAND Flash Array Address Cycle 3**

When less than five address cycles are used, ADDR_CYCLE3 is the third byte written to the NAND Flash.

When five address cycles are used, ADDR_CYCLE3 is the fourth byte written to the NAND Flash.

- **ADDR_CYCLE4: NAND Flash Array Address Cycle 4**

When less than five address cycles are used, ADDR_CYCLE4 is the fourth byte written to the NAND Flash.

When five address cycles are used, ADDR_CYCLE4 is the fifth byte written to the NAND Flash.

Note: If five address cycles are used, the first address cycle is ADDR_CYCLE0. Refer to HSMC_ADDR register.

30.17.2.4 NFC DATA Status

Name: NFCDATA_STATUS

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	NFCBUSY	NFCWR	DATAEN	CSID

23	22	21	20	19	18	17	16
CSID		ACYCLE			VCMD2	CMD2	

15	14	13	12	11	10	9	8
CMD2						CMD1	

7	6	5	4	3	2	1	0
CMD1						–	–

- **CMD1: Command Register Value for Cycle 1**

When a Read or Write Access occurs, the Physical Memory Interface drives the IO bus with CMD1 field during the Command Latch cycle 1.

- **CMD2: Command Register Value for Cycle 2**

When VCMD2 field is set to true, the Physical Memory Interface drives the IO bus with CMD2 field during the Command Latch cycle 2.

- **VCMD2: Valid Cycle 2 Command**

When set to true, the CMD2 field is issued after addressing cycle.

- **ACYCLE: Number of Address required for the current command**

When ACYCLE field is different from zero, ACYCLE Address cycles are performed after Command Cycle 1.

- **CSID: Chip Select Identifier**

Chip select used

- **DATAEN: NFC Data phase enable**

When set to true, the NFC data phase is enabled.

- **NFCWR: NFC Write Enable**

0: NFC is in read mode.

1: NFC is in write mode.

- **NFCBUSY: NFC Busy Status Flag**

If set to true, it indicates that the NFC is busy.

30.17.3 NFC Initialization

Prior to any Command and Data Transfer, the SMC User Interface must be configured to meet the device timing requirements.

- Write enable Configuration

Use NWE_SETUP, NWE_PULSE and NWE_CYCLE to define the write enable waveform according to the device datasheet.

Use TADL field in the HSMC_TIMINGS register to configure the timing between the last address latch cycle and the first rising edge of WEN for data input.

Figure 30-32.Write Enable Timing Configuration

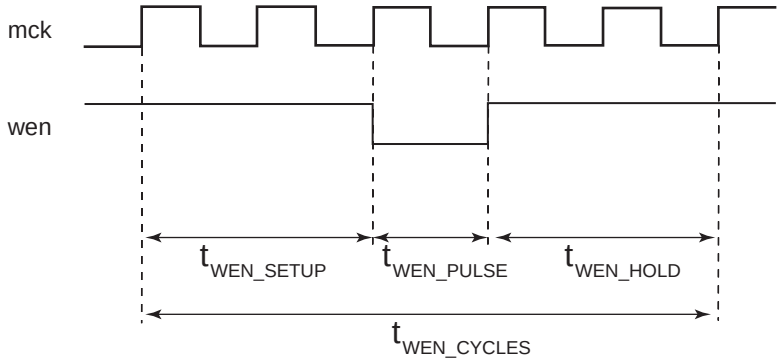
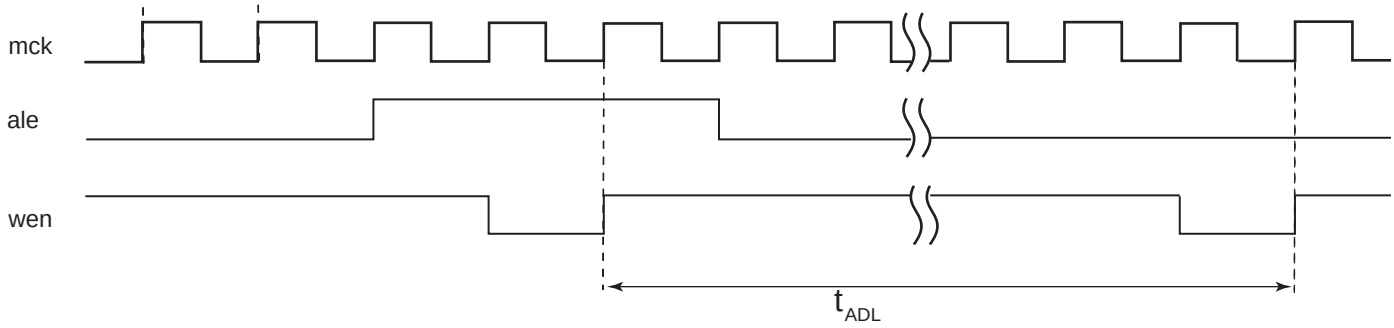


Figure 30-33.Write Enable Timing for NAND Flash Device Data Input Mode



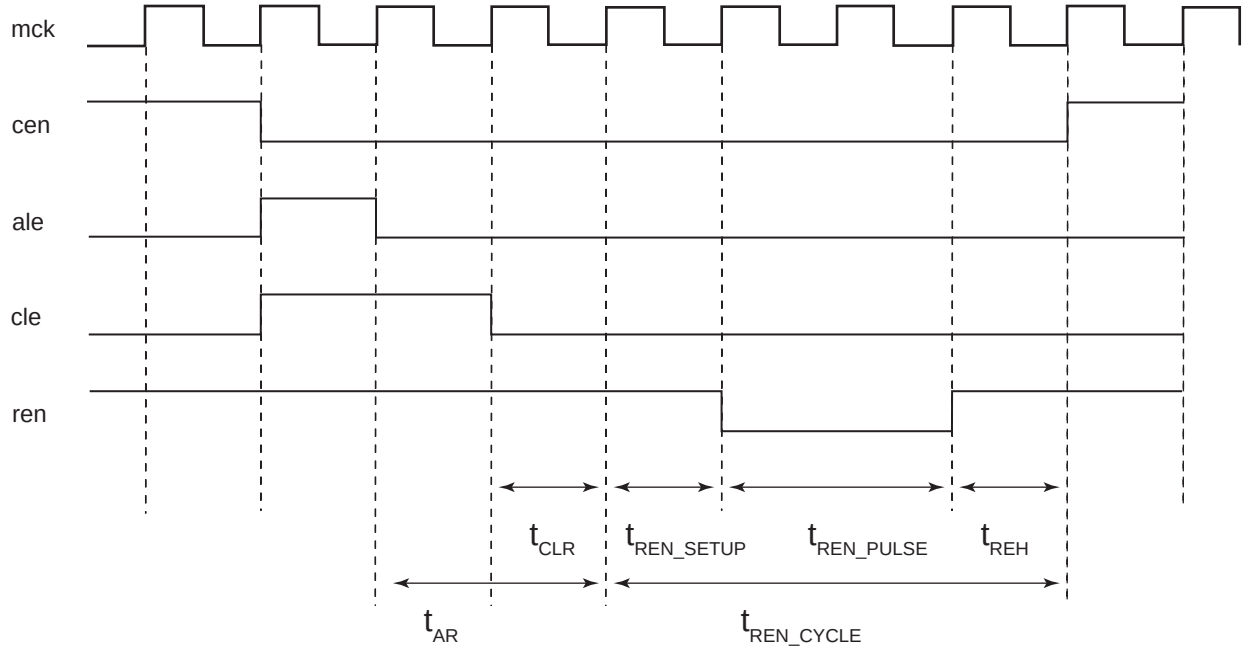
- Read Enable Configuration

Use NRD_SETUP, NRD_PULSE and NRD_CYCLE to define the read enable waveform according to the device datasheet.

Use TAR field in the HSMC_TIMINGS register to configure the timings between the address latch enable falling edge to read the enable falling edge.

Use TCLR field in the HSMC_TIMINGS register to configure the timings between the command latch enable falling edge to read the enable falling edge.

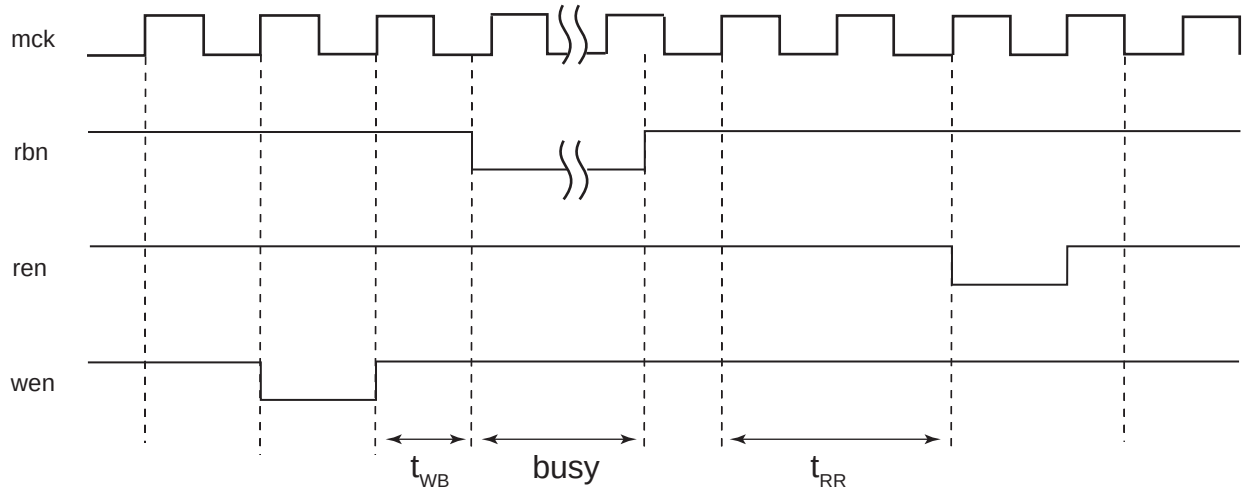
Figure 30-34.Read Enable Timing Configuration Working with NAND Flash Device



- Ready/Busy Signal Timing configuration working with a NAND Flash device

Use TWB field in HSMC_TIMINGS register to configure the maximum elapsed time between the rising edge of the wen signal and the falling edge of the rbn signal. Use TRR field in the HSMC_TIMINGS register to program the number of clock cycles between the rising edge of the rbn signal and the falling edge of the ren signal.

Figure 30-35.Ready/Busy Timing Configuration

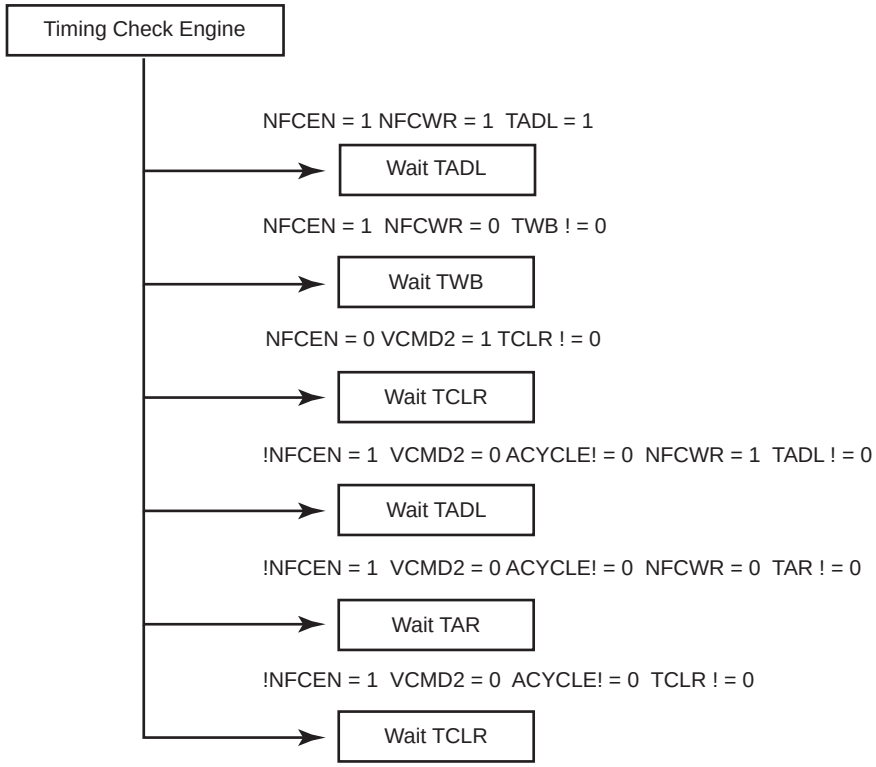


30.17.3.1 NAND Flash Controller Timing Engine

When the NFC Command register is written, the NFC issues a NAND Flash Command and optionally performs a data transfer between the NFC SRAM and the NAND Flash device. The NAND Flash Controller Timing Engine guarantees valid NAND Flash timings, depending on the set of parameters decoded from the address bus. These timings are defined in the HSMC_TIMINGS register.

For information on the timing used depending on the command, see Figure 30-36.

Figure 30-36.NAND Flash Controller Timing Engine



See the NFC Address Command register description and the HSMC Timings Register.

30.17.4 NFC SRAM

30.17.4.1 NFC SRAM Mapping

If the NFC is used to read and write Data from and to the NAND Flash, the configuration depends on the page size, the relevant field is PAGESIZE in HSMC_CFG register. See Table 30-8 to Table 30-12 for detailed mapping.

The NFC SRAM size is 8 Kbytes. The NFC can handle the NAND Flash with a page size of 8 Kbytes or of a lower size (such as 2 Kbytes for example). In case of a 4-Kbyte or lower page size, the NFC SRAM can be split into two banks. The field BANK in the HSMC_BANK register is used to select where NAND flash data are written or read. For 8 Kbytes page size this field is not relevant.

Note that a “ping-pong” mode (write or read to a bank while the NFC writes or reads to another bank) is accessible with the NFC (using two different banks).

If the NFC is not used, the NFC SRAM can be used for a general purpose by the application.

Table 30-8. NFC SRAM Bank Mapping for 512 bytes

Offset	Use	Access
0x00000000–0x000001FF	Main Area Bank 0	Read/Write
0x00000200–0x000003FF	Spare Area Bank 0	Read/Write
0x00001200–0x000013FF	Main Area Bank 1	Read/Write
0x00001400–0x000015FF	Spare Area Bank 1	Read/Write

Table 30-9. NFC SRAM Bank Mapping for 1 Kbyte

Offset	Use	Access
0x00000000–0x000003FF	Main Area Bank 0	Read/Write
0x00000400–0x000005FF	Spare Area Bank 0	Read/Write
0x00001200–0x000015FF	Main Area Bank 1	Read/Write
0x00001600–0x000017FF	Spare Area Bank 1	Read/Write

Table 30-10. NFC SRAM Bank Mapping for 2 Kbytes

Offset	Use	Access
0x00000000–0x000007FF	Main Area Bank 0	Read/Write
0x00000800–0x000009FF	Spare Area Bank 0	Read/Write
0x00001200–0x000019FF	Main Area Bank 1	Read/Write
0x00001A00–0x00001BFF	Spare Area Bank 1	Read/Write

Table 30-11. NFC SRAM Bank Mapping for 4 Kbytes

Offset	Use	Access
0x00000000–0x00000FFF	Main Area Bank 0	Read/Write
0x00001000–0x000011FF	Spare Area Bank 0	Read/Write
0x00001200–0x000021FF	Main Area Bank 1	Read/Write
0x00002200–0x000023FF	Spare Area Bank 1	Read/Write

Table 30-12. NFC SRAM Bank Mapping for 8 Kbytes, only one bank is available

Offset	Use	Access
0x00000000–0x00001FFF	Main Area Bank 0	Read/Write
0x00002000–0x000023FF	Spare Area Bank 0	Read/Write

30.17.4.2 NFC SRAM Access Prioritization Algorithm

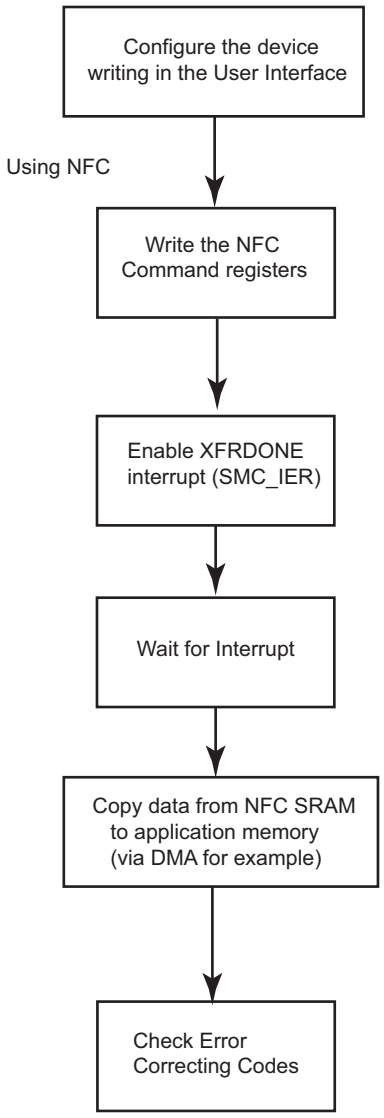
When the NAND Flash Controller (NFC) is reading from or writing to an NFC SRAM bank, the other bank is available. If an NFC SRAM access occurs when the NFC performs a read or write operation in the same bank, then the access is discarded. The write operation is not performed. The read operation returns undefined data. If this situation is encountered, the AWB status flag located in the NFC Status Register is raised and indicates that a shared resource access violation has occurred.

30.17.5 NAND Flash Operations

This section describes the software operations needed to issue commands to the NAND Flash device and to perform data transfers using the NFC.

30.17.5.1 Page Read

Figure 30-37.Page Read Flow Chart

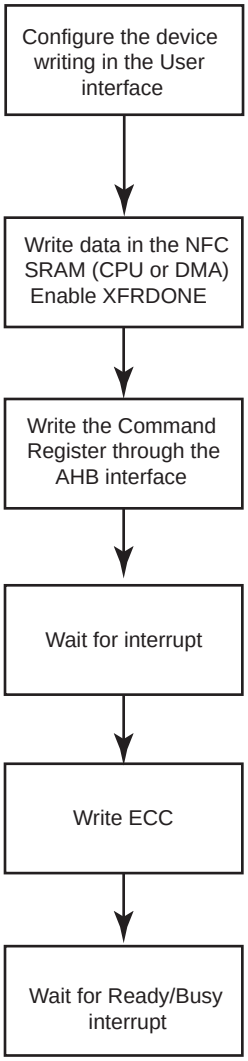


Note that, instead of using the interrupt, one can poll the NFCBUSY flag.

For more information on the NFC Control Register, see Section 30.17.2.2 “NFC Address Command”.

30.17.5.2 Program Page

Figure 30-38.Program Page Flow Chart



Writing the ECC cannot be done using the NFC; it needs to be done “manually”.

Note that, instead of using the interrupt, one can poll the NFCBUSY flag.

For more information on the NFC Control Register, see Section 30.17.2.2 “NFC Address Command”.

30.18 PMECC Controller Functional Description

The Programmable Multibit Error Correcting Code (PMECC) controller is a programmable binary BCH (Bose, Chaudhuri and Hocquenghem) encoder/decoder. This controller can be used to generate redundancy information for both SLC and MLC NAND devices. It supports redundancy for correction of 2, 4, 8, 12, or 24 errors per sector of data. The 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 (see Table 30-13 on page 448). This operation shall be executed for each sector. At decoding time, the PMECC module generates the remainders of the received codeword by the minimal polynomials. When all remainders for a given sector are set to zero, no error occurred. When the remainders are different from 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 PMECC Interrupt Status Register (HSMC_PMECCISR). This register indicates which sector is corrupted.

The processor must execute the following decoding steps to find the error location within a sector:

1. Syndrome computation.
2. Finding the error location polynomial.
3. Finding the roots of the error location polynomial.

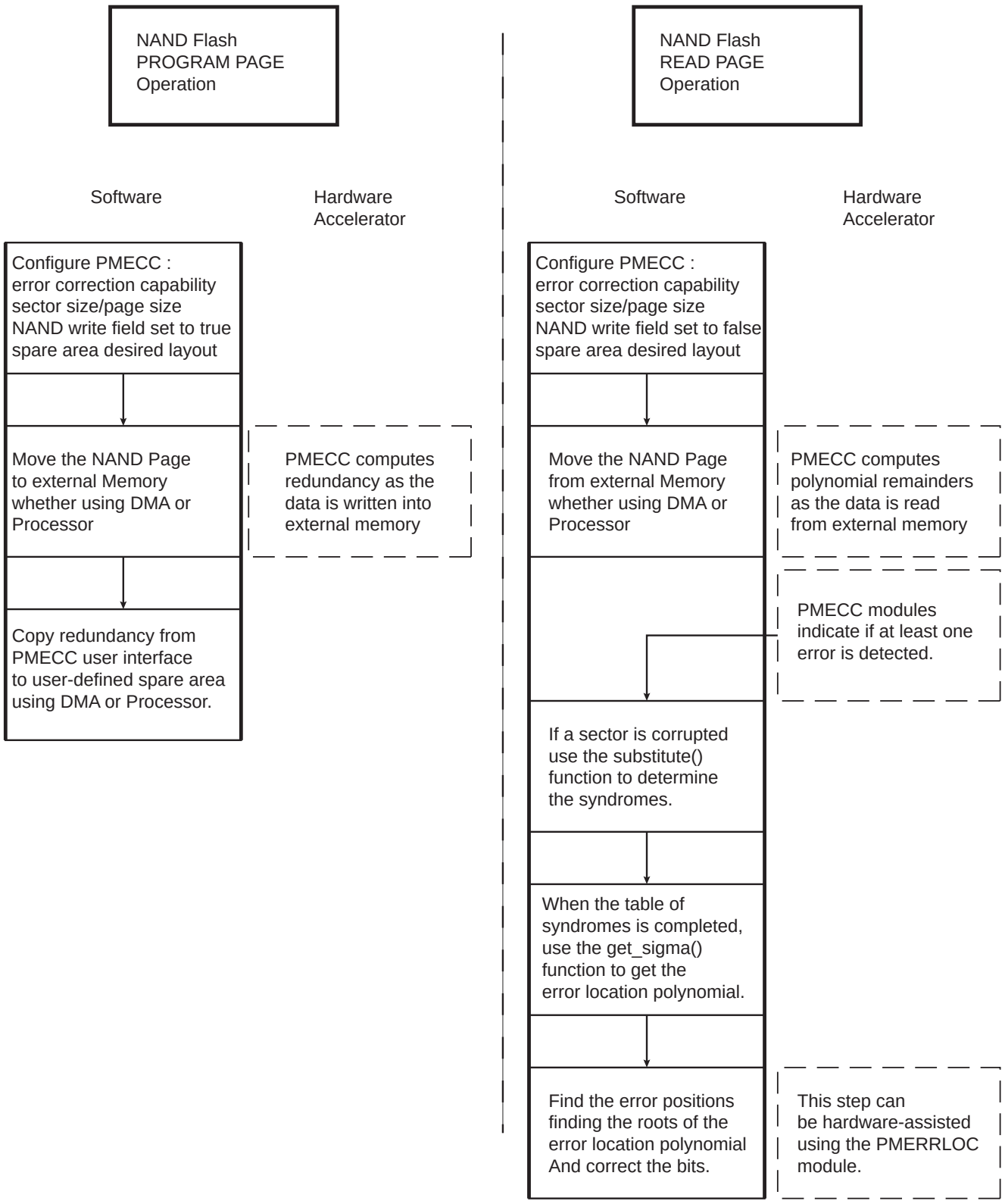
All decoding steps involve finite field computation. It means that a library of finite field arithmetic must be available to perform addition, multiplication and inversion. These arithmetic operations can be performed through the use of a memory mapped look-up table, or direct software implementation. The software implementation presented is based on look-up 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. Assuming that beta = alpha ^ index, then beta belongs to the field, and gf_log(beta) = gf_log(alpha ^ index) = index. The gf_antilog table provides exponent inverse of the element; if beta = alpha ^ index, then gf_antilog(index) = beta.

The first step consists in the syndrome computation. The PMECC module computes the remainders and the software must substitute the power of the primitive element. The procedure implementation is given in Section 30.19.1 “Remainder Substitution Procedure” on page 452.

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 30.19.2 “Find the Error Location Polynomial Sigma(x)” on page 453.

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 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 PMERRLOC module provides this kind of hardware acceleration.

Figure 30-39.Software Hardware Multibit Error Correction Dataflow



30.18.1 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 30-13. Relevant Redundancy Registers

BCH_ERR Field	Sector Size Set to 512 Bytes	Sector Size Set to 1024 Bytes
0	PMECC0	PMECC0
1	PMECC0, PMECC1	PMECC0, PMECC1
2	PMECC0, PMECC1, PMECC2, PMECC3	PMECC0, PMECC1, PMECC2, PMECC3
3	PMECC0, PMECC1, PMECC2, PMECC3, PMECC4, PMECC5, PMECC6	PMECC0, PMECC1, PMECC2, PMECC3, PMECC4, PMECC5, PMECC6
4	PMECC0, PMECC1, PMECC2, PMECC3, PMECC4, PMECC5, PMECC6, PMECC7, PMECC8, PMECC9	PMECC0, PMECC1, PMECC2, PMECC3, PMECC4, PMECC5, PMECC6, PMECC7, PMECC8, PMECC9, PMECC10

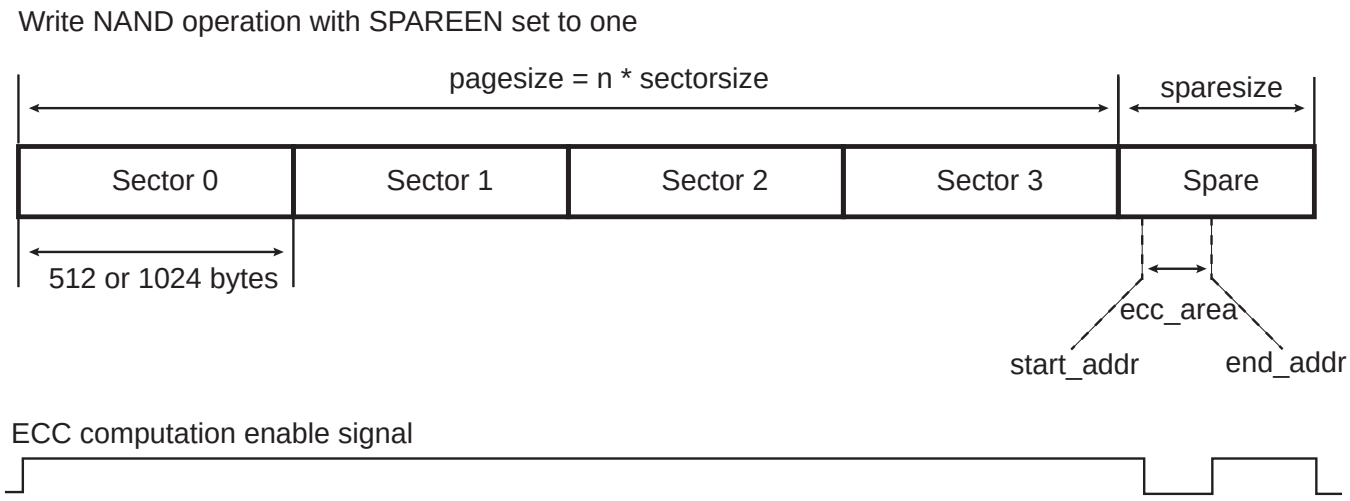
Table 30-14. Number of Relevant ECC Bytes per Sector, Copied from LSByte to MSByte

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

30.18.1.1 SLC/MLC Write Operation with Spare Enable Bit Set

When the SPAREEN field of the PMECCFG 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 1 in the DATA field of the PMECCTRL register. When the encoding process is over, the redundancy shall be written to the spare area in user mode. The USER field of the PMECCTRL register must be set to one.

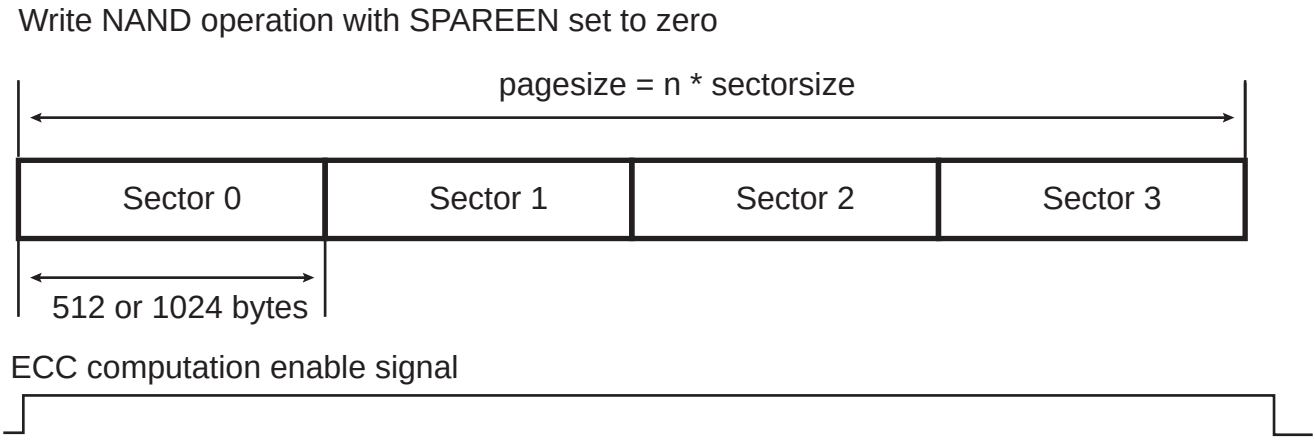
Figure 30-40.NAND Write Operation with Spare Encoding



30.18.1.2 SLC/MLC Write Operation with Spare Disable

When the SPAREEN field of PMECCFG is set to zero, the spare area is not encoded with the stream of data. This mode is entered by writing 1 to the DATA field of the PMECCTRL register.

Figure 30-41.NAND Write Operation



30.18.2 MLC/SLC Read Page Operation Using PMECC

Table 30-15. Relevant Remainder Registers

BCH_ERR Field	Sector Size Set to 512 Bytes	Sector Size Set to 1024 Bytes
0	PMECCREM0	PMECCREM0
1	PMECCREM0, PMECCREM1	PMECCREM0, PMECCREM1
2	PMECCREM0, PMECCREM1, PMECCREM2, PMECCREM3,	PMECCREM0, PMECCREM1, PMECCREM2, PMECCREM3
3	PMECCREM0, PMECCREM1, PMECCREM2, PMECCREM3, PMECCREM4, PMECCREM5, PMECCREM6, PMECCREM7	PMECCREM0, PMECCREM1, PMECCREM2, PMECCREM3, PMECCREM4, PMECCREM5, PMECCREM6, PMECCREM7

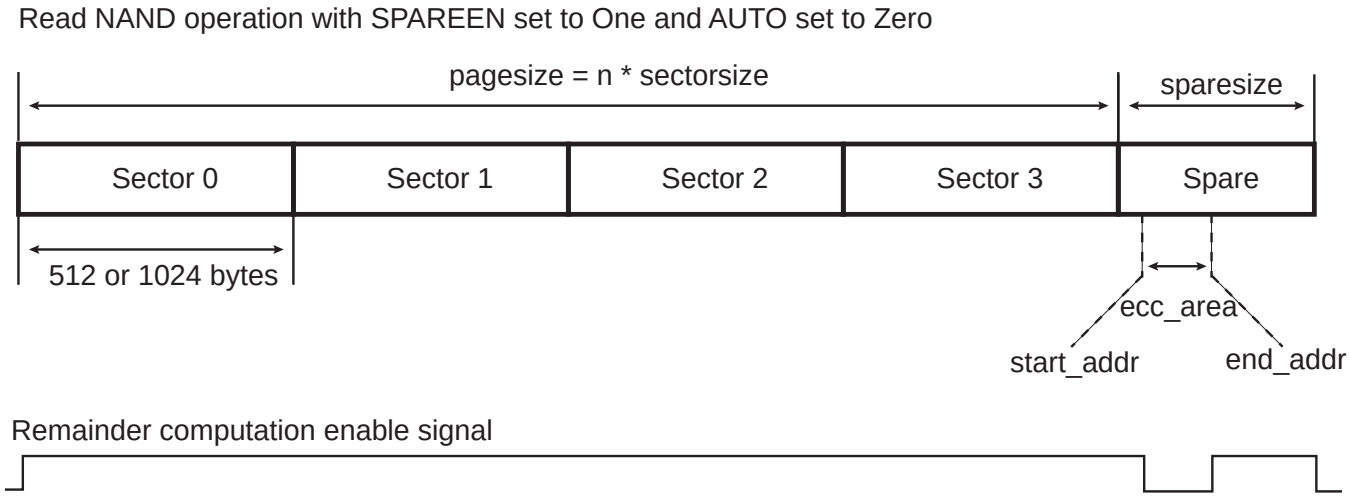
Table 30-15. Relevant Remainder Registers (Continued)

BCH_ERR Field	Sector Size Set to 512 Bytes	Sector Size Set to 1024 Bytes
4	PMECCREM0, PMECCREM1, PMECCREM2, PMECCREM3, PMECCREM4, PMECCREM5, PMECCREM6, PMECCREM7, PMECCREM8, PMECCREM9, PMECCREM10, PMECCREM11	PMECCREM0, PMECCREM1, PMECCREM2, PMECCREM3, PMECCREM4, PMECCREM5, PMECCREM6, PMECCREM7, PMECCREM8, PMECCREM9, PMECCREM10, PMECCREM11

30.18.2.1 MLC/SLC Read Operation with Spare Decoding

When the spare area is protected, it contains valid data. As the redundancy may be included in the middle of the information stream, the user shall program the start address and the end address of the ECC area. The controller will automatically skip the ECC area. This mode is entered writing 1 in the DATA field of the PMECCTRL register. When the page has been fully retrieved from the NAND, the ECC area shall be read using the user mode, writing 1 to the USER field of the PMECCTRL register.

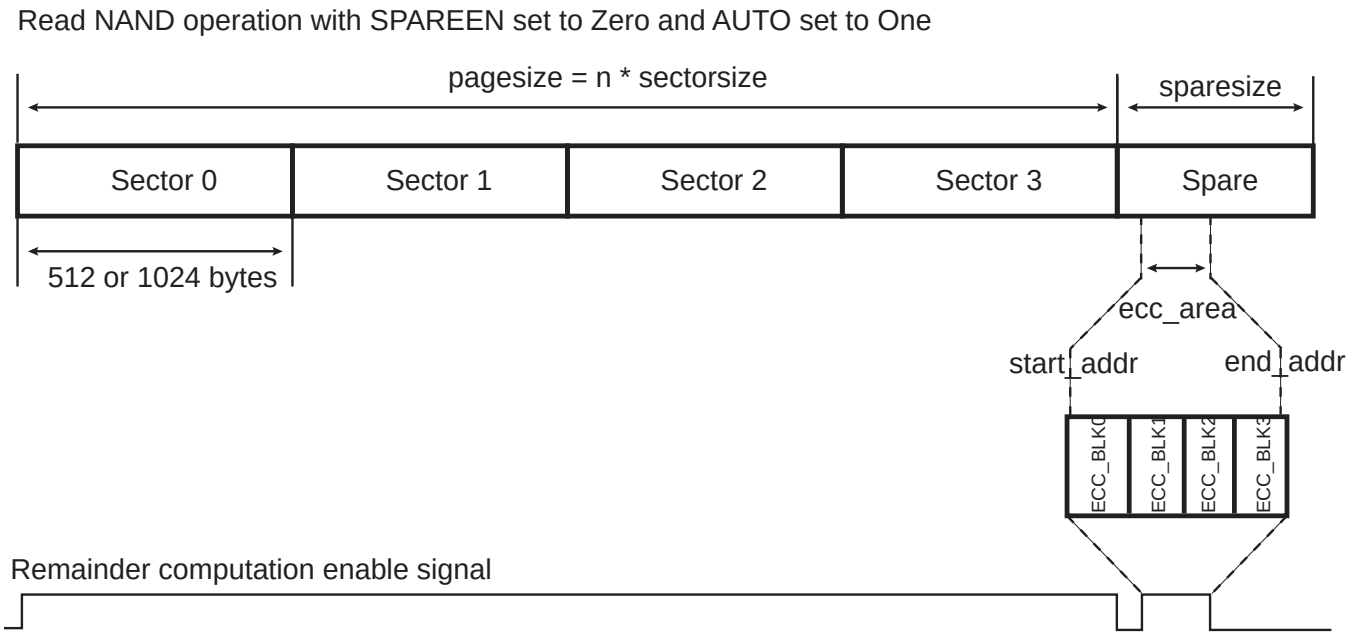
Figure 30-42.Read Operation with Spare Decoding



30.18.2.2 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 writing 1 in the DATA field of the PMECCTRL register. When AUTO field is set to one, the ECC is retrieved automatically; otherwise, the ECC must be read using the user mode.

Figure 30-43.Read Operation



30.18.2.3 MLC/SLC User Read ECC Area

This mode allows a manual retrieve of the ECC. It is entered writing 1 in the USER field of the PMECCTRL register.

Figure 30-44.Read User Mode

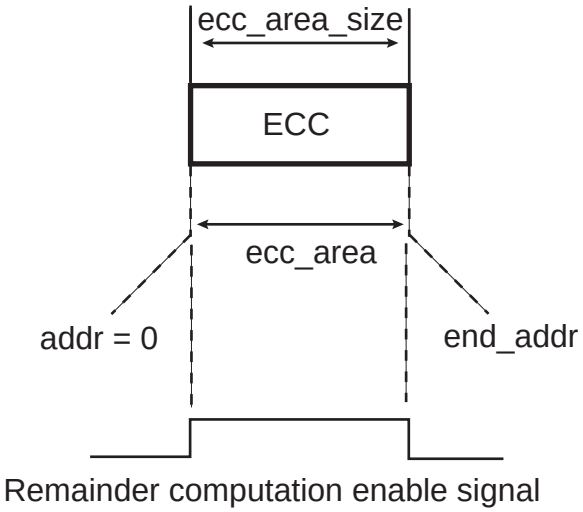


Table 30-16. MLC Controller Configuration when the Host Controller is Used

Transfer Type	NFC		PMECC		
	RSPARE	WSPARE	SPAREEN	AUTO	User Mode
Program Page main area is protected, spare is not protected, spare is written manually	0	0	0	0	Not used
Program Page main area is protected, spare is protected, spare is written by NFC	0	1	1	0	Not used
Read Page main area is protected, spare is not protected, spare is not retrieved by NFC	0	0	0	0	Used
Read Page main area is protected, spare is not protected, spare is retrieved by NFC	1	0	0	1	Not used
Read Page main area is protected, spare is protected, spare is retrieved by NFC	1	0	1	0	Used

30.19 Software Implementation

30.19.1 Remainder Substitution Procedure

The substitute function evaluates the remainder polynomial, with different values of the field primitive element. The addition arithmetic operation is performed with the exclusive OR. The multiplication arithmetic operation is performed through the gf_log and gf_antilog look-up tables.

The REM2NP1 and REMN2NP3 fields of the PMECCREMN registers contain only odd remainders. Each bit indicates whether the coefficient of the remainder polynomial 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;
}
```

30.19.2 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;
}
```

30.19.3 Find the Error Position

The output of the get_sigma() procedure is a polynomial stored in the smu[NB_ERROR+1][] table. The error positions are the roots of that polynomial. The degree of that polynomial is a very important information, as it gives the number of errors. PMERRLOC module provides hardware accelerator for that step.

30.19.3.1 Error Location

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 32 fully programmable coefficients. These coefficients belong to GF(2^13) or GF(2^14). The coefficient programmed in the PMERRLOC{*i*} is the coefficient of X ^ *i* in the polynomial.

The search operation is started as soon as a write access is detected in the ELEN register and can be disabled 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 30-17. 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 30-18. 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 PMECC engine is searching for roots, the BUSY field of the ELSR register remains asserted. An interrupt is asserted at the end of the computation, and the DONE bit of the PMECC Error Location Interrupt Status Register (HSMC_ELSIR) is set. The ERR_CNT field of the HSMC_ELISR indicates the number of errors. The error position can be read in the PMERRLOCX registers.

30.20 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 30-19. For each chip select, a set of four registers is used to program the parameters of the external device. In Table 30-19, “CS_number” denotes the chip select number. Sixteen bytes per chip select are required.

Table 30-19. Register Mapping

Offset	Register	Name	Access	Reset
0x000	HSMC NFC Configuration Register	HSMC_CFG	Read/Write	0x0
0x004	HSMC NFC Control Register	HSMC_CTRL	Write-only	0x0
0x008	HSMC NFC Status Register	HSMC_SR	Read-only	0x0
0x00C	HSMC NFC Interrupt Enable Register	HSMC_IER	Write-only	0x0
0x010	HSMC NFC Interrupt Disable Register	HSMC_IDR	Write-only	0x0
0x014	HSMC NFC Interrupt Mask Register	HSMC_IMR	Read-only	0x0
0x018	HSMC NFC Address Cycle Zero Register	HSMC_ADDR	Read/Write	0x0
0x01C	HSMC Bank Address Register	HSMC_BANK	Read/Write	0x0
0x020–0x06C	Reserved	–	–	–
0x70	PMECC Configuration Register	HSMC_PMECCFG	Read/Write	0x0
0x74	PMECC Spare Area Size Register	HSMC_PMECCSAREA	Read/Write	0x0
0x78	PMECC Start Address Register	HSMC_PMECCSADDR	Read/Write	0x0
0x7C	PMECC End Address Register	HSMC_PMECCEADDR	Read/Write	0x0
0x80	Reserved	–	–	–
0x84	PMECC Control Register	HSMC_PMECCTRL	Write-only	0x0
0x88	PMECC Status Register	HSMC_PMECCSR	Read-only	0x0
0x8C	PMECC Interrupt Enable register	HSMC_PMECCIER	Write-only	0x0
0x90	PMECC Interrupt Disable Register	HSMC_PMECCIDR	Write-only	–
0x94	PMECC Interrupt Mask Register	HSMC_PMECCIMR	Read-only	0x0
0x98	PMECC Interrupt Status Register	HSMC_PMECCISR	Read-only	0x0
0x9C–AC	Reserved	–	–	–
0x0B0+sec_num*(0x40)+0x00	PMECC Redundancy 0 Register	HSMC_PMECC0	Read-only	0x0
0x0B0+sec_num*(0x40)+0x04	PMECC Redundancy 1 Register	HSMC_PMECC1	Read-only	0x0
0x0B0+sec_num*(0x40)+0x08	PMECC Redundancy 2 Register	HSMC_PMECC2	Read-only	0x0
0x0B0+sec_num*(0x40)+0x0C	PMECC Redundancy 3 Register	HSMC_PMECC3	Read-only	0x0
0x0B0+sec_num*(0x40)+0x10	PMECC Redundancy 4 Register	HSMC_PMECC4	Read-only	0x0
0x0B0+sec_num*(0x40)+0x14	PMECC Redundancy 5 Register	HSMC_PMECC5	Read-only	0x0
0x0B0+sec_num*(0x40)+0x18	PMECC Redundancy 6 Register	HSMC_PMECC6	Read-only	0x0
0x0B0+sec_num*(0x40)+0x1C	PMECC Redundancy 7 Register	HSMC_PMECC7	Read-only	0x0
0x0B0+sec_num*(0x40)+0x20	PMECC Redundancy 8 Register	HSMC_PMECC8	Read-only	0x0
0x0B0+sec_num*(0x40)+0x24	PMECC Redundancy 9 Register	HSMC_PMECC9	Read-only	0x0
0x0B0+sec_num*(0x40)+0x28	PMECC Redundancy 10 Register	HSMC_PMECC10	Read-only	0x0

Table 30-19. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x2B0+sec_num*(0x40)+0x00	PMECC Remainder 0 Register	HSMC_REM0	Read-only	0x0
0x2B0+sec_num*(0x40)+0x04	PMECC Remainder 1 Register	HSMC_REM1	Read-only	0x0
0x2B0+sec_num*(0x40)+0x08	PMECC Remainder 2 Register	HSMC_REM2	Read-only	0x0
0x2B0+sec_num*(0x40)+0x0C	PMECC Remainder 3 Register	HSMC_REM3	Read-only	0x0
0x2B0+sec_num*(0x40)+0x10	PMECC Remainder 4 Register	HSMC_REM4	Read-only	0x0
0x2B0+sec_num*(0x40)+0x14	PMECC Remainder 5 Register	HSMC_REM5	Read-only	0x0
0x2B0+sec_num*(0x40)+0x18	PMECC Remainder 6 Register	HSMC_REM6	Read-only	0x0
0x2B0+sec_num*(0x40)+0x1C	PMECC Remainder 7 Register	HSMC_REM7	Read-only	0x0
0x2B0+sec_num*(0x40)+0x20	PMECC Remainder 8 Register	HSMC_REM8	Read-only	0x0
0x2B0+sec_num*(0x40)+0x24	PMECC Remainder 9 Register	HSMC_REM9	Read-only	0x0
0x2B0+sec_num*(0x40)+0x28	PMECC Remainder 10 Register	HSMC_REM10	Read-only	0x0
0x2B0+sec_num*(0x40)+0x2C	PMECC Remainder 11 Register	HSMC_REM11	Read-only	0x0
0x4A0–0x4FC	Reserved	–	–	–
0x500	PMECC Error Location Configuration Register	HSMC_ELCFG	Read/Write	0x0
0x504	PMECC Error Location Primitive Register	HSMC_ELPRIM	Read-only	0x401A
0x508	PMECC Error Location Enable Register	HSMC_ELEN	Write-only	0x0
0x50C	PMECC Error Location Disable Register	HSMC_ELDIS	Write-only	0x0
0x510	PMECC Error Location Status Register	HSMC_ELSR	Read	0x0
0x514	PMECC Error Location Interrupt Enable register	HSMC_ELIER	Write-only	0x0
0x518	PMECC Error Location Interrupt Disable Register	HSMC_ELIDR	Write-only	0x0
0x51C	PMECC Error Location Interrupt Mask Register	HSMC_ELIMR	Read	0x0
0x520	PMECC Error Location Interrupt Status Register	HSMC_ELISR	Read	0x0
0x524–0x52C	Reserved	–	–	–
0x528	PMECC Error Location SIGMA 0 Register	HSMC_SIGMA0	Read/Write	0x1
...	...	...	...	...
0x588	PMECC Error Location SIGMA 24 Register	HSMC_SIGMA24	Read/Write	0x0
0x58C	PMECC Error Location 0 Register	HSMC_ERRLOC0	Read-only	0x0
...	...	...	...	...
0x5E8	PMECC Error Location 23 Register	HSMC_ERRLOC23	Read-only	0x0
0x5EC–0x5FC	Reserved	–	–	–
0x14*CS_number+0x600	HSMC Setup Register	HSMC_SETUP	Read/Write	–

Table 30-19. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x14*CS_number+0x604	HSMC Pulse Register	HSMC_PULSE	Read/Write	–
0x14*CS_number+0x608	HSMC Cycle Register	HSMC_CYCLE	Read/Write	–
0x14*CS_number+0x60C	HSMC Timings Register	HSMC_TIMINGS	Read/Write	–
0x14*CS_number+0x610	HSMC Mode Register	HSMC_MODE	Read/Write	–
0x6A0	HSMC Off Chip Memory Scrambling Register	HSMC_OCMS	Read/Write	0x0
0x6A4	HSMC Off Chip Memory Scrambling KEY1 Register	HSMC_KEY1	Write-once	0x0
0x6A8	HSMC Off Chip Memory Scrambling KEY2 Register	HSMC_KEY2	Write-once	0x0
0x6AC–0x6E0	Reserved	–	–	–
0x6E4	HSMC Write Protection Mode Register	HSMC_WPMR	Read/Write	0x0
0x6E8	HSMC Write Protection Status Register	HSMC_WPSR	Read-only	0x0
0x6FC	Reserved	–	–	–

30.20.1 HSMC NFC Configuration Register

Name: HSMC_CFG

Address: 0xFFFFC000

Access: Read/Write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	NFCSPARESIZE						

23	22	21	20	19	18	17	16
–	DTOMUL			DTOCYC			

15	14	13	12	11	10	9	8
–	–	RBEDGE	EDGECTRL	–	–	RSPARE	WSPARE

7	6	5	4	3	2	1	0
–	–	–	–	–	PAGESIZE		

- **PAGESIZE: Page Size of the NAND Flash Device**

Value	Name	Description
0	PS512	Main area 512 bytes
1	PS1024	Main area 1024 bytes
2	PS2048	Main area 2048 bytes
3	PS4096	Main area 4096 bytes
4	PS8192	Main area 8192 bytes

- **WSPARE: Write Spare Area**

0: The NFC skips the spare area in write mode.

1: The NFC writes both main area and spare area in write mode.

- **RSPARE: Read Spare Area**

0: The NFC skips the spare area in read mode.

1: The NFC reads both main area and spare area in read mode.

- **EDGECTRL: Rising/Falling Edge Detection Control**

0: Rising edge is detected

1: Falling edge is detected

- **RBEDGE: Ready/Busy Signal Edge Detection**

0: When set to zero, RB_EDGE fields indicate the level of the Ready/Busy lines.

1: When set to one, RB_EDGE fields indicate only transition on Ready/Busy lines.

- **DTOCYC: Data Timeout Cycle Number**

- **DTOMUL: Data Timeout Multiplier**

These fields determine the maximum number of Master Clock cycles that the SMC waits until the detection of a rising edge on Ready/Busy signal.

Data Timeout Multiplier is defined by DTOMUL as shown in the following table:

Value	Name	Description
0	X1	DTOCYC
1	X16	DTOCYC x 16
2	X128	DTOCYC x 128
3	X256	DTOCYC x 256
4	X1024	DTOCYC x 1024
5	X4096	DTOCYC x 4096
6	X65536	DTOCYC x 65536
7	X1048576	DTOCYC x 1048576

If the data timeout set by DTOCYC and DTOMUL has been exceeded, the Data Timeout Error flag (DTOE) in the NFC Status Register (NFC_SR) raises.

- **NFCSPARESIZE: NAND Flash Spare Area Size Retrieved by the Host Controller**

The spare size is set to (NFCSPARESIZE+1) * 4 bytes. The spare area is only retrieved when RSPARE or WSPARE is activated.

30.20.2 HSMC NFC Control Register

Name: HSMC_CTRL

Address: 0xFFFFC004

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
–	–	–	–	–	–	NFCDIS	NFCEN

- **NFCEN: NAND Flash Controller Enable**

0: No effect

1: Enable the NAND Flash controller.

- **NFCDIS: NAND Flash Controller Disable**

0: No effect

1: Disable the NAND Flash controller.

30.20.3 HSMC NFC Status Register

Name: HSMC_SR

Address: 0xFFFFC008

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	RB_EDGE0
23	22	21	20	19	18	17	16
NFCASE	AWB	UNDEF	DTOE	–	–	CMDDONE	XFRDONE
15	14	13	12	11	10	9	8
–	NFCSID			NFCWR	–	–	NFCBUSY
7	6	5	4	3	2	1	0
–	–	RB_FALL	RB_RISE	–	–	–	SMCSTS

- **SMCSTS: NAND Flash Controller Status (this field cannot be reset)**

0: NAND Flash Controller disabled

1: NAND Flash Controller enabled

- **RB_RISE: Selected Ready Busy Rising Edge Detected**

When set to one, this flag indicates that a rising edge on the Ready/Busy Line has been detected. This flag is reset after a status read operation. The Ready/Busy line selected is the decoding of the set NFCCSID, RBNSEL fields.

- **RB_FALL: Selected Ready Busy Falling Edge Detected**

When set to one, this flag indicates that a falling edge on the Ready/Busy Line has been detected. This flag is reset after a status read operation. The Ready/Busy line is selected through the decoding of the set NFCSID, RBNSEL fields.

- **NFCBUSY: NFC Busy (this field cannot be reset)**

When set to one, this flag indicates that the Controller is activated and accesses the memory device.

- **NFCWR: NFC Write/Read Operation (this field cannot be reset)**

When a command is issued, this field indicates the current Read or Write Operation.

- **NFCSID: NFC Chip Select ID (this field cannot be reset)**

When a command is issued, this field indicates the value of the targeted chip select.

- **XFRDONE: NFC Data Transfer Terminated**

When set to one, this flag indicates that the NFC has terminated the Data Transfer. This flag is reset after a status read operation.

- **CMDDONE: Command Done**

When set to one, this flag indicates that the NFC has terminated the Command. This flag is reset after a status read operation.

- **DTOE: Data Timeout Error**

When set to one, this flag indicates that the Data timeout set be by DTOMUL and DTOCYC has been exceeded. This flag is reset after a status read operation.

- **UNDEF: Undefined Area Error**

When set to one, this flag indicates that the processor performed an access in an undefined memory area. This flag is reset after a status read operation.

- **AWB: Accessing While Busy**

If set to one, this flag indicates that an AHB master has performed an access during the busy phase. This flag is reset after a status read operation.

- **NFCASE: NFC Access Size Error**

If set to one, this flag indicates that an illegal access has been detected in the NFC Memory Area. Only Word Access is allowed within the NFC memory area. This flag is reset after a status read operation.

- **RB_EDGEx: Ready/Busy Line x Edge Detected**

If set to one, this flag indicates that an edge has been detected on the Ready/Busy Line x. Depending on the EDGE CTRL field located in the HSMC_CFG register, only rising or falling edge is detected. This flag is reset after a status read operation.

30.20.4 HSMC NFC Interrupt Enable Register

Name: HSMC_IER

Address: 0xFFFFC00C

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	RB_EDGE0
23	22	21	20	19	18	17	16
NFCASE	AWB	UNDEF	DTOE	–	–	CMDDONE	XFRDONE
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	RB_FALL	RB_RISE	–	–	–	–

- **RB_RISE: Ready Busy Rising Edge Detection Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **RB_FALL: Ready Busy Falling Edge Detection Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **XFRDONE: Transfer Done Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **CMDDONE: Command Done Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **DTOE: Data Timeout Error Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **UNDEF: Undefined Area Access Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **AWB: Accessing While Busy Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **NFCASE: NFC Access Size Error Interrupt Enable**

0: No effect

1: Interrupt source enabled

- **RB_EDGEx: Ready/Busy Line x Interrupt Enable**

0: No effect

1: Interrupt source enabled

30.20.5 HSMC NFC Interrupt Disable Register

Name: HSMC_IDR

Address: 0xFFFFC010

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	RB_EDGE0
23	22	21	20	19	18	17	16
NFCASE	AWB	UNDEF	DTOE	–	–	CMDDONE	XFRDONE
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	RB_FALL	RB_RISE	–	–	–	–

- **RB_RISE: Ready Busy Rising Edge Detection Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **RB_FALL: Ready Busy Falling Edge Detection Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **XFRDONE: Transfer Done Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **CMDDONE: Command Done Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **DTOE: Data Timeout Error Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **UNDEF: Undefined Area Access Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **AWB: Accessing While Busy Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **NFCASE: NFC Access Size Error Interrupt Disable**

0: No effect

1: Interrupt source disabled

- **RB_EDGEx: Ready/Busy Line x Interrupt Disable**

0: No effect

1: Interrupt source disabled

30.20.6 HSMC NFC Interrupt Mask Register

Name: HSMC_IMR

Address: 0xFFFFC014

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	RB_EDGE0
23	22	21	20	19	18	17	16
NFCASE	AWB	UNDEF	DTOE	–	–	CMDDONE	XFRDONE
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	RB_FALL	RB_RISE	–	–	–	–

- **RB_RISE: Ready Busy Rising Edge Detection Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **RB_FALL: Ready Busy Falling Edge Detection Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **XFRDONE: Transfer Done Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **CMDDONE: Command Done Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **DTOE: Data Timeout Error Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **UNDEF: Undefined Area Access Interrupt Mask5**

0: Interrupt source disabled

1: Interrupt source enabled

- **AWB: Accessing While Busy Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **NFCASE: NFC Access Size Error Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

- **RB_EDGEx: Ready/Busy Line x Interrupt Mask**

0: Interrupt source disabled

1: Interrupt source enabled

30.20.7 HSMC NFC Address Cycle Zero Register

Name: HSMC_ADDR

Address: 0xFFFFC018

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
ADDR_CYCLE0							

- **ADDR_CYCLE0: NAND Flash Array Address Cycle 0**

When five address cycles are used, ADDR_CYCLE0 is the first byte written to the NAND Flash (used by the NFC).

30.20.8 HSMC NFC Bank Register

Name: HSMC_BANK

Address: 0xFFFFC01C

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
–	–	–	–	–	–	–	BANK

- **BANK: Bank Identifier**

0: Bank 0 is used.

1: Bank 1 is used.

30.20.9 PMECC Configuration Register

Name: HSMC_PMECCFG

Address: 0xFFFFC070

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 Correcting 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 sectors 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 area 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.

30.20.10 PMECC Spare Area Size Register

Name: HSMC_PMECCSAREA

Address: 0xFFFFC074

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**

Number of bytes in the spare area. The spare area size is equal to (SPARESIZE + 1) bytes.

30.20.11 PMECC Start Address Register

Name: HSMC_PMECCSADDR

Address: 0xFFFFC078

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**

This register is programmed with the start ECC start address. When STARTADDR is equal to 0, then the first ECC byte is located at the first byte of the spare area.

30.20.12 PMECC End Address Register

Name: HSMC_PMECCEADDR

Address: 0xFFFFC07C

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**

This register is programmed with the start ECC end address. When ENDADDR is equal to *N*, then the first ECC byte is located at byte *N* of the spare area.

30.20.13 PMECC Control Register

Name: HSMC_PMECCTRL

Address: 0xFFFFC084

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**

0: No effect

1: Reset the PMECC controller.

- **DATA: Start a Data Phase**

0: No effect

1: The PMECC controller enters a Data phase.

- **USER: Start a User Mode Phase**

0: No effect

1: The PMECC controller enters a User mode phase.

- **ENABLE: PMECC Enable**

0: No effect

1: Enable the PMECC controller.

- **DISABLE: PMECC Enable**

0: No effect

1: Disable the PMECC controller.

30.20.14 PMECC Status Register

Name: HSMC_PMECCSR

Address: 0xFFFFC088

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**

0: PMECC controller finite state machine reached idle state

1: PMECC controller finite state machine is processing the incoming byte stream

- **ENABLE: PMECC Enable bit**

0: PMECC controller disabled

1: PMECC controller enabled

30.20.15 PMECC Interrupt Enable Register

Name: HSMC_PMECCIER

Address: 0xFFFFC08C

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**

0: No effect

1: The Multibit Error interrupt is enabled. An interrupt will be raised if at least one error is detected in at least one sector.

30.20.16 PMECC Interrupt Disable Register

Name: HSMC_PMECCIDR

Address: 0xFFFFC090

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
–	–	–	–	–	–	–	ERRID

- **ERRID: Error Interrupt Disable**

0: No effect

1: Multibit Error interrupt disabled

30.20.17 PMECC Interrupt Mask Register

Name: HSMC_PMECCIMR

Address: 0xFFFFC094

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 Mask**

0: Multibit Error disabled

1: Multibit Error enabled

30.20.18 PMECC Interrupt Status Register

Name: HSMC_PMECCISR

Address: 0xFFFFC098

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 HSMC_PMECCISR indicates that sector *i* is corrupted.

30.20.19 PMECC Redundancy x Register

Name: HSMC_PMECCx [x=0..10] [sec_num=0..7]

Address: 0xFFFFC0B0 [0][0] .. 0xFFFFC0D8 [10][0]
0xFFFFC0F0 [0][1] .. 0xFFFFC118 [10][1]
0xFFFFC130 [0][2] .. 0xFFFFC158 [10][2]
0xFFFFC170 [0][3] .. 0xFFFFC198 [10][3]
0xFFFFC1B0 [0][4] .. 0xFFFFC1D8 [10][4]
0xFFFFC1F0 [0][5] .. 0xFFFFC218 [10][5]
0xFFFFC230 [0][6] .. 0xFFFFC258 [10][6]
0xFFFFC270 [0][7] .. 0xFFFFC298 [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.

30.20.20 PMECC Remainder x Register

Name: HSMC_REMx [x=0..11] [sec_num=0..7]

Address: 0xFFFFC2B0 [0][0] .. 0xFFFFC2DC [11][0]
0xFFFFC2F0 [0][1] .. 0xFFFFC31C [11][1]
0xFFFFC330 [0][2] .. 0xFFFFC35C [11][2]
0xFFFFC370 [0][3] .. 0xFFFFC39C [11][3]
0xFFFFC3B0 [0][4] .. 0xFFFFC3DC [11][4]
0xFFFFC3F0 [0][5] .. 0xFFFFC41C [11][5]
0xFFFFC430 [0][6] .. 0xFFFFC45C [11][6]
0xFFFFC470 [0][7] .. 0xFFFFC49C [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.

30.20.21 PMECC Error Location Configuration Register

Name: HSMC_ELCFG

Address: 0xFFFFC500

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 bytes sector.

1: The ECC computation is based on a 1024 bytes sector.

30.20.22 PMECC Error Location Primitive Register

Name: HSMC_ELPRIM

Address: 0xFFFFC504

Access: Read-only

Reset: 0x401A

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**

This field indicates the Primitive Polynomial used in the ECC computation.

30.20.23 PMECC Error Location Enable Register

Name: HSMC_ELEN

Address: 0xFFFFC508

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
–	–	ENINIT					
7	6	5	4	3	2	1	0
ENINIT							

- **ENINIT: Error Location Enable**

Initial bit number in the codeword.

30.20.24 PMECC Error Location Disable Register

Name: HSMC_ELDIS

Address: 0xFFFFC50C

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
–	–	–	–	–	–	–	DIS

- **DIS: Disable Error Location Engine**

0: No effect

1: Disable the Error location engine.

30.20.25 PMECC Error Location Status Register

Name: HSMC_ELSR

Address: 0xFFFFC510

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
–	–	–	–	–	–	–	BUSY

- **BUSY: Error Location Engine Busy**

0: Error location engine is disabled.

1: Error location engine is enabled and is finding roots of the polynomial.

30.20.26 PMECC Error Location Interrupt Enable Register

Name: HSMC_ELIER

Address: 0xFFFFC514

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
–	–	–	–	–	–	–	DONE

- **DONE: Computation Terminated Interrupt Enable**

0: No effect

1: Interrupt Enable.

30.20.27 PMECC Error Location Interrupt Disable Register

Name: HSMC_ELIDR

Address: 0xFFFFC518

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
–	–	–	–	–	–	–	DONE

- **DONE: Computation Terminated Interrupt Disable**

0: No effect

1: Interrupt disable.

30.20.28 PMECC Error Location Interrupt Mask Register

Name: HSMC_ELIMR

Address: 0xFFFFC51C

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**

0: Computation Terminated interrupt disabled

1: Computation Terminated interrupt enabled

30.20.29 PMECC Error Location Interrupt Status Register

Name: HSMC_ELISR

Address: 0xFFFFC520

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**

When set to one, this indicates that the error location engine has completed the root finding algorithm.

- **ERR_CNT: Error Counter value**

This field indicates the number of roots of the polynomial.

30.20.30 PMECC Error Location SIGMAx Register

Name: HSMC_SIGMAx [x=0..24]

Address: 0xFFFFC528 [0] .. 0xFFFFC588 [24]

Access: Read/Write

Reset: 0x1

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.

30.20.31 PMECC Error Location x Register

Name: HSMC_ERRLOCx [x=0..23]

Address: 0xFFFFC58C

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 1 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.

30.20.32 HSMC Setup Register

Name: HSMC_SETUPx [x=0..3]

Address: 0xFFFFC600 [0], 0xFFFFC614 [1], 0xFFFFC628 [2], 0xFFFFC63C [3]

Access: Write-only

Reset: –

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					

This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Mode Register.

- **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 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.

30.20.33 HSMC Pulse Register

Name: HSMC_PULSEx [x=0..3]

Address: 0xFFFFC604 [0], 0xFFFFC618 [1], 0xFFFFC62C [2], 0xFFFFC640 [3]

Access: Write-only

Reset: –

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					

This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Mode Register.

- **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 must be at least one 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 must be at least one clock cycle.

- **NRD_PULSE: NRD Pulse Length**

The NRD signal pulse length is defined as:

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

The NRD pulse width must be as least 1 clock cycle.

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

In READ mode, The NCS signal pulse length is defined as:

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

30.20.34 HSMC Cycle Register

Name: HSMC_CYCLEx [x=0..3]

Address: 0xFFFFC608 [0], 0xFFFFC61C [1], 0xFFFFC630 [2], 0xFFFFC644 [3]

Access: Read/Write

Reset: –

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							

This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Mode Register.

- **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.

30.20.35 HSMC Timings Register

Name: HSMC_TIMINGSx [x=0..3]

Address: 0xFFFFC60C [0], 0xFFFFC620 [1], 0xFFFFC634 [2], 0xFFFFC648 [3]

Access: Read/Write

Reset: –

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

This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Mode Register.

- **TCLR: CLE to REN Low Delay**

Command Latch Enable falling edge to Read Enable falling edge timing.

Latch Enable Falling to Read Enable Falling = (TCLR[3] * 64) + TCLR[2:0] clock cycles.

- **TADL: ALE to Data Start**

Last address latch cycle to the first rising edge of WEN for data input.

Last address latch to first rising edge of WEN = (TADL[3] * 64) + TADL[2:0] clock cycles.

- **TAR: ALE to REN Low Delay**

Address Latch Enable falling edge to Read Enable falling edge timing.

Address Latch Enable to Read Enable = (TAR[3] * 64) + TAR[2:0] clock cycles.

- **OCMS: Off Chip Memory Scrambling Enable**

When set to one, the memory scrambling is activated.

- **TRR: Ready to REN Low Delay**

Ready/Busy signal to Read Enable falling edge timing.

Read to REN = (TRR[3] * 64) + TRR[2:0] clock cycles.

- **TWB: WEN High to REN to Busy**

Write Enable rising edge to Ready/Busy falling edge timing.

Write Enable to Read/Busy = (TWB[3] * 64) + TWB[2:0] clock cycles.

- **RBNSEL: Ready/Busy Line Selection**

This field indicates the selected Ready/Busy Line from the RBN bundle.

- **NFSEL: NAND Flash Selection**

If this bit is set to one, the chip select is assigned to NAND Flash write enable and read enable lines drive the Error Correcting Code module.

30.20.36 HSMC Mode Register

Name: HSMC_MODEx [x=0..3]

Address: 0xFFFFC610 [0], 0xFFFFC624 [1], 0xFFFFC638 [2], 0xFFFFC64C [3]

Access: Read/Write

Reset: –

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

This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Mode Register.

- **READ_MODE: Selection of the Control Signal for Read Operation**

1 (NRD_CTRL): The Read operation is controlled by the NRD signal.

0 (NCS_CTRL): The Read operation is controlled by the NCS signal.

- **WRITE_MODE: Selection of the Control Signal for Write Operation**

1 (NWE_CTRL): The Write operation is controlled by the NWE signal.

0 (NCS_CTRL): The Write operation is controller by the NCS signal.

- **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 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.

Value	Name	Description
0	DISABLED	Disabled
1	—	Reserved
2	FROZEN	Frozen Mode
3	READY	Ready Mode

- Disabled: 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-bit data bus.

1 (BYTE_WRITE): Byte write access type:

– Write operation is controlled using NCS, NWR0, NWR1.

– Read operation is controlled using NCS and NRD.

0 (BYTE_SELECT): Byte select access type:

– Write operation is controlled using NCS, NWE, NBS0, NBS1.

– Read operation is controlled using NCS, NRD, NBS0, NBS1.

- **DBW: Data Bus Width**

Value	Name	Description
0	BIT_8	8-bit bus
1	BIT_16	16-bit bus

- **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 enabled

– The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization disabled

– The number of TDF wait states is inserted before the next access begins.

30.20.37 HSMC Off Chip Memory Scrambling Register

Name: HSMC_OCMS

Address: 0xFFFFC6A0

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
–	–	–	–	–	–	SRSE	SMSE

- **SMSE: Static Memory Controller Scrambling Enable**

0: Disable “Off Chip” Scrambling for SMC access.

1: Enable “Off Chip” Scrambling for SMC access. (If OCMS field is set to 1 in the relevant HSMC_TIMINGS register.)

- **SRSE: SRAM Scrambling Enable**

0: Disable SRAM Scrambling for SRAM access.

1: Enable SRAM Scrambling for SRAM access. (If OCMS field is set to 1 in the relevant HSMC_TIMINGS register.)

30.20.38 HSMC Off Chip Memory Scrambling Key1 Register

Name: HSMC_KEY1

Address: 0xFFFFC6A4

Access: Write-once

Reset: 0x00000000

31	30	29	28	27	26	25	24
KEY1							

23	22	21	20	19	18	17	16
KEY1							

15	14	13	12	11	10	9	8
KEY1							

7	6	5	4	3	2	1	0
KEY1							

- **KEY1: Off Chip Memory Scrambling (OCMS) Key Part 1**

When Off Chip Memory Scrambling is enabled by setting the HSMC_OMCS and HSMC_TIMINGS registers in accordance, the data scrambling depends on KEY1 and KEY2 values.

30.20.39 HSMC Off Chip Memory Scrambling Key2 Register

Name: HSMC_KEY2

Address: 0xFFFFC6A8

Access: Write-once

Reset: 0x00000000

31	30	29	28	27	26	25	24
KEY2							

23	22	21	20	19	18	17	16
KEY2							

15	14	13	12	11	10	9	8
KEY2							

7	6	5	4	3	2	1	0
KEY2							

- **KEY2: Off Chip Memory Scrambling (OCMS) Key Part 2**

When Off Chip Memory Scrambling is enabled by setting the HSMC_OMCS and HSMC_TIMINGS registers in accordance, the data scrambling depends on KEY2 and KEY1 values.

30.20.40 HSMC Write Protection Mode Register

Name: HSMC_WPMR

Address: 0xFFFFC6E4

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 Write Protection if WPKEY value corresponds to 0x534D43 (“SMC” in ASCII)

1: Enables Write Protection if WPKEY value corresponds to 0x534D43 (“SMC” in ASCII)

See Section 30.16 “Register Write Protection” for list of write-protected registers.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x534D43	PASSWD	Writing any other value in this field aborts the write operation of bit WPEN. Always reads as 0.

30.20.41 HSMC Write Protection Status Register

Name: HSMC_WPSR

Address: 0xFFFFC6E8

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 Protect Violation has occurred since the last read of the HSMC_WPSR.

1: A Write Protect Violation has occurred since the last read of the HSMC_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.

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.

31.2 Embedded Characteristics

- 3 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 on DMAC0
- 8 DMA Channels on DMAC1
- 16 External Request Lines on DMAC0
- 22 External Request Lines on DMAC1
- Embedded FIFO
- Channel Locking and Bus Locking Capability

31.3 DMA Controller Peripheral Connections

The DMA Controller handles the transfer between peripherals and memory and receives triggers from the peripherals listed in tables that follow.

For each listed DMA Channel Number, the SIF and/or DIF bitfields in the DMAC_CTRLBx register must be programmed with a value compatible to the MATRIX “Master to Slave Access” definition provided in the “Bus Matrix (MATRIX)” section of the product datasheet. See Section 31.8.17 ”DMAC Channel x [x = 0..7] Control B Register” (where x is the DMA Channel Number).

Depending on transfer descriptor location, the DSCR_IF bitfield must be programmed with a value compatible to the MATRIX “Master to Slave Access” definition provided in the “Bus Matrix (MATRIX)” section of the product datasheet. See Section 31.8.15 ”DMAC Channel x [x = 0..7] Descriptor Address Register” (where x is the DMA Channel Number).

31.3.1 DMA Controller 0

The DMA Controller 0 handles the transfer between peripherals and memory and receives triggers from the peripherals connected on APB0 (see Table 31-1).

Table 31-1. DMA Channels Definition (DMAC0)

Instance name	Channel T/R	Interface number
HSMCI0	Receive/transmit	0
SPI0	Transmit	1
SPI0	Receive	2
USART0	Transmit	3
USART0	Receive	4
USART1	Transmit	5
USART1	Receive	6
TWI0	Transmit	7
TWI0	Receive	8
TWI1	Transmit	9
TWI1	Receive	10
UART0	Transmit	11
UART0	Receive	12
SSC0	Transmit	13
SSC0	Receive	14
SMD	Transmit	15
SMD	Receive	16

31.3.2 DMA Controller 1

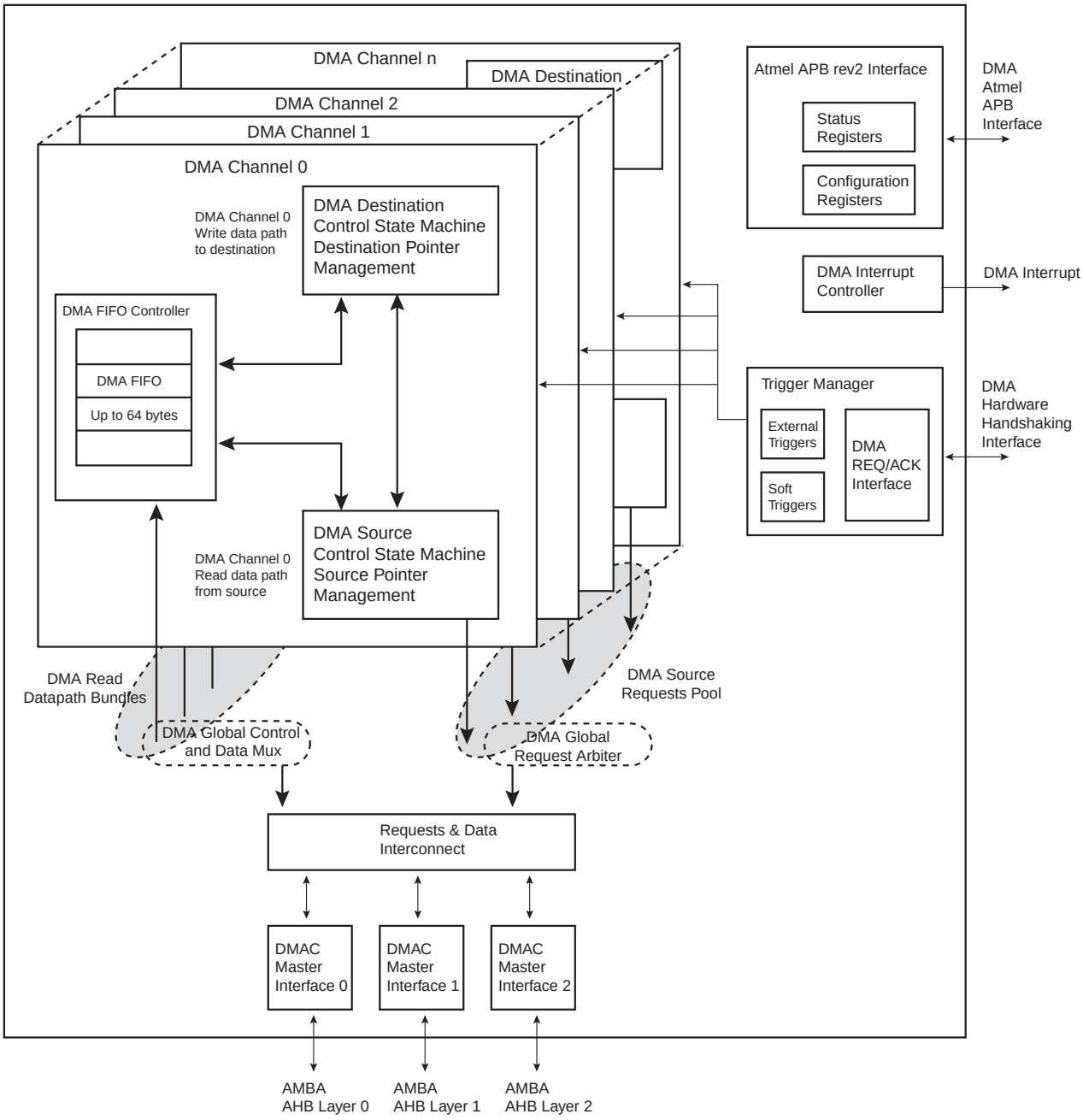
The DMA Controller 1 handles the transfer between peripherals and memory and receives triggers from the peripherals connected on APB1 (see Table 31-2).

Table 31-2. DMA Channels Definition (DMAC1)

Instance name	Channel T/R	Interface Number
HSMCI1	Receive/transmit	0
HSMCI2	Receive/transmit	1
ADC	Receive	2
SSC1	Transmit	3
SSC1	Receive	4
UART1	Transmit	5
UART1	Receive	6
USART2	Transmit	7
USART2	Receive	8
USART3	Transmit	9
USART3	Receive	10
TWI2	Transmit	11
TWI2	Receive	12
DBGU	Transmit	13
DBGU	Receive	14
SPI1	Transmit	15
SPI1	Receive	16
SHA	Transmit	17
AES	Transmit	18
AES	Receive	19
TDES	Transmit	20
TDES	Receive	21

31.4 Block Diagram

Figure 31-1. DMA Controller (DMAC) Block Diagram



31.5 Functional Description

31.5.1 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 Global Configuration Register”). 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.

Transfer hierarchy: Figure 31-2 on page 514 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 514 shows the transfer hierarchy for memory.

Figure 31-2. DMAC Transfer Hierarchy for Non-Memory Peripheral

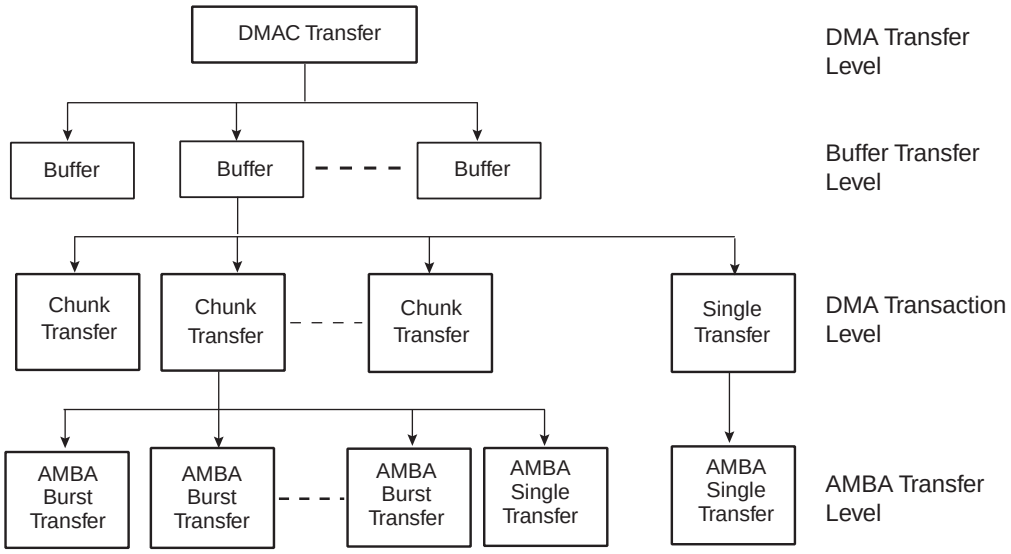
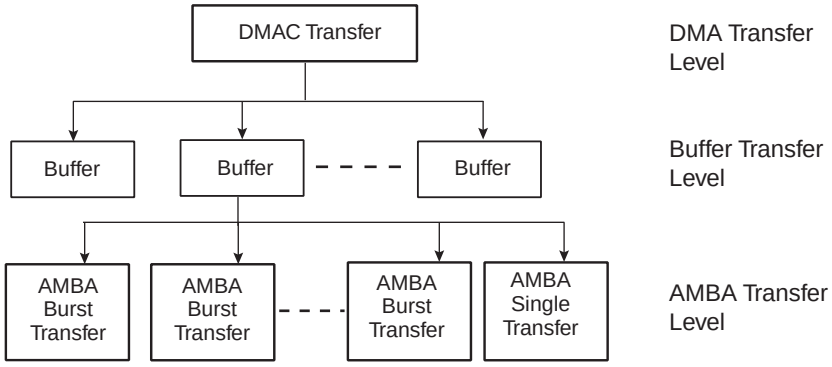


Figure 31-3. DMAC Transfer Hierarchy for Memory



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.

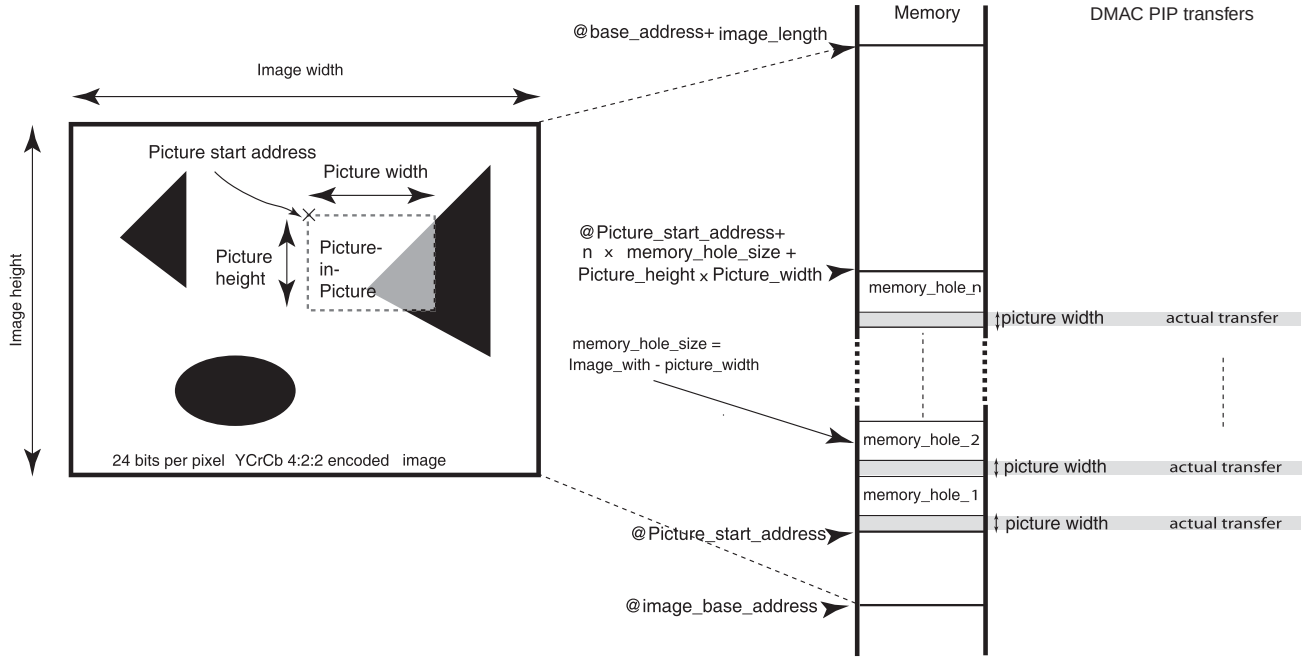
Multi-buffer DMAC transfer: A DMAC transfer may consist of multiple DMAC buffers. Multi-buffer DMAC transfers are supported through buffer chaining (linked list pointers), auto-reloading of channel registers, and contiguous buffers. The source and destination can independently select which method to 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 515 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.

31.5.2 Memory Peripherals

Figure 31-3 on page 514 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.

31.5.3 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.

31.5.3.1 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.

31.5.4 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.

31.5.4.1 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).

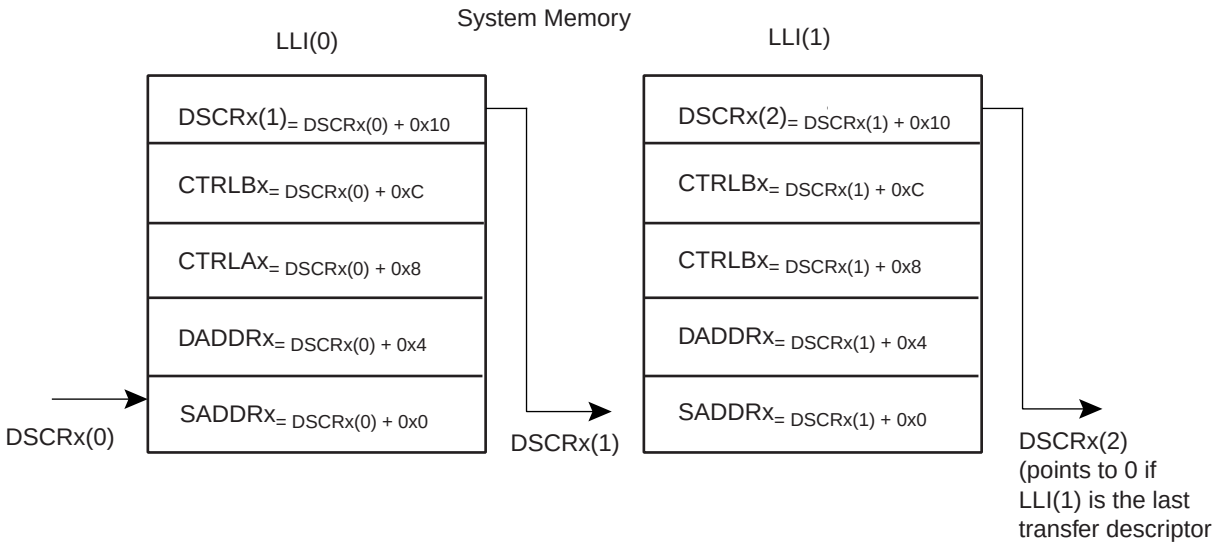
To set up buffer chaining, a sequence of linked lists must be programmed in memory.

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 517 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.

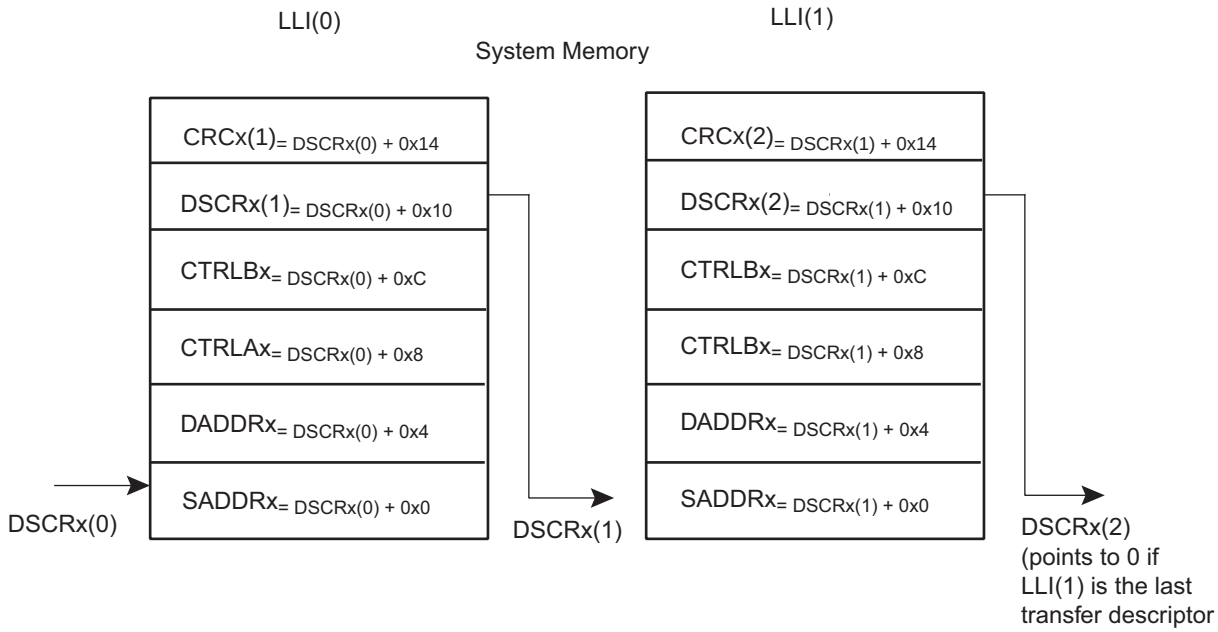
Figure 31-5. Multi Buffer Transfer Using Linked List



Descriptor Integrity Check

When the Descriptor Integrity Check is enabled, a cyclic redundancy check information is attached to the descriptor. When fetched from the memory, the descriptor is verified through the use of a CRC16-CCIT (0x1021 polynom) by the DMAC channel. If a CRC error is detected, then the DICERR flag is set in the DMAC_EBCISR register. The CRC16 is computed from MSB to LSB. The BTSIZE and DONE fields of the DMAC_CTRLAx register are ignored and set to zero.

Figure 31-6. Linked List with CRC16 Attached



31.5.4.2 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 519, 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.

31.5.4.3 Ending Multi-buffer Transfers

All multi-buffer transfers must end as shown in Row 1 of Table 31-3 on page 519. 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 519, (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.

31.5.5 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 519.

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.

31.5.5.1 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 519. 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 31-7 on page 523) 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 519. 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 517 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 519.
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).

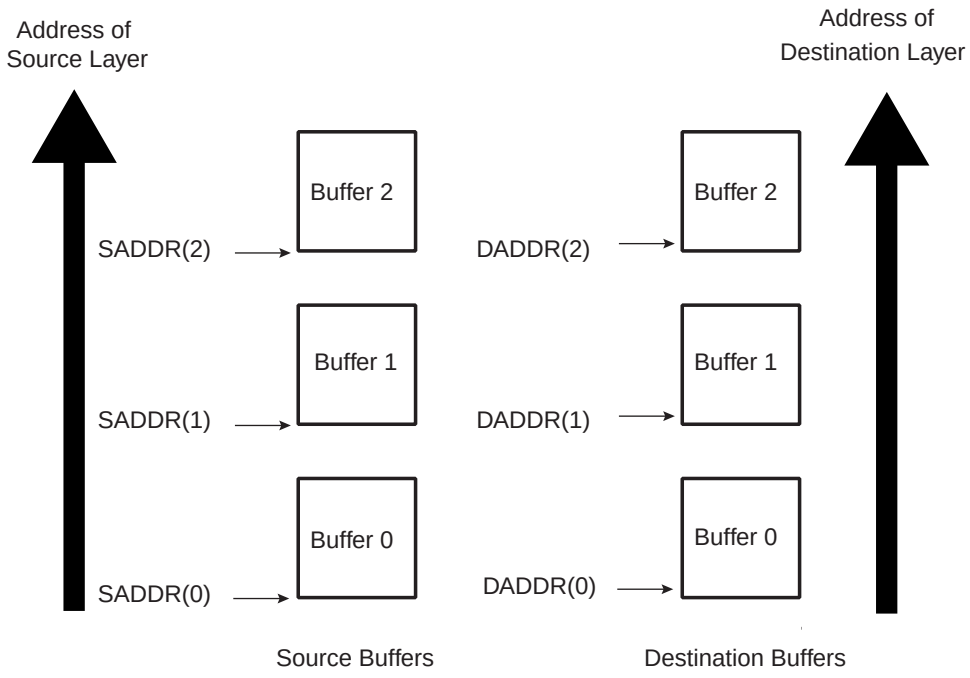
Note: The LLI.DMAC_SADDRx, LLI. DMAC_DADDRx, LLI.DMAC_DSCRx, LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The DMAC automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLBx and DMAC_CTRLAx channel registers from the DMAC_DSCRx(0).

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.

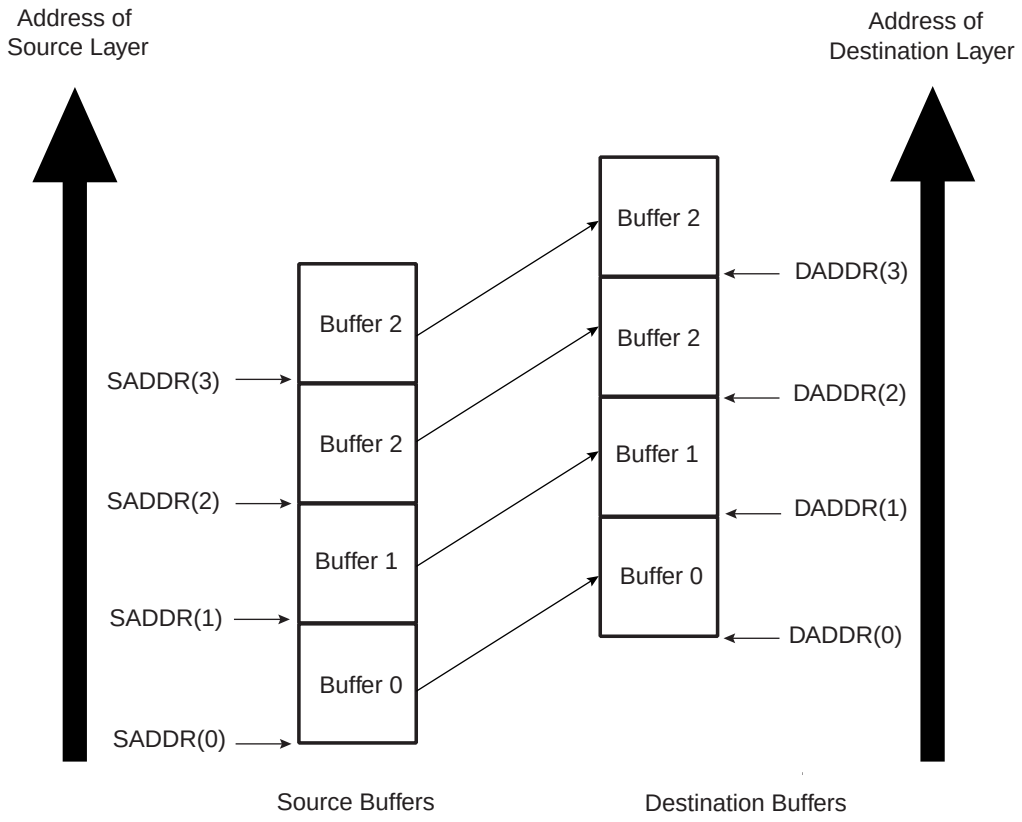
17. The DMAC does not wait for the buffer interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current DMAC_DSCRx register and automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx channel registers. The DMAC transfer 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 519. 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 31-7 on page 523.

Figure 31-7. Multi-buffer with Linked List Address for Source and Destination



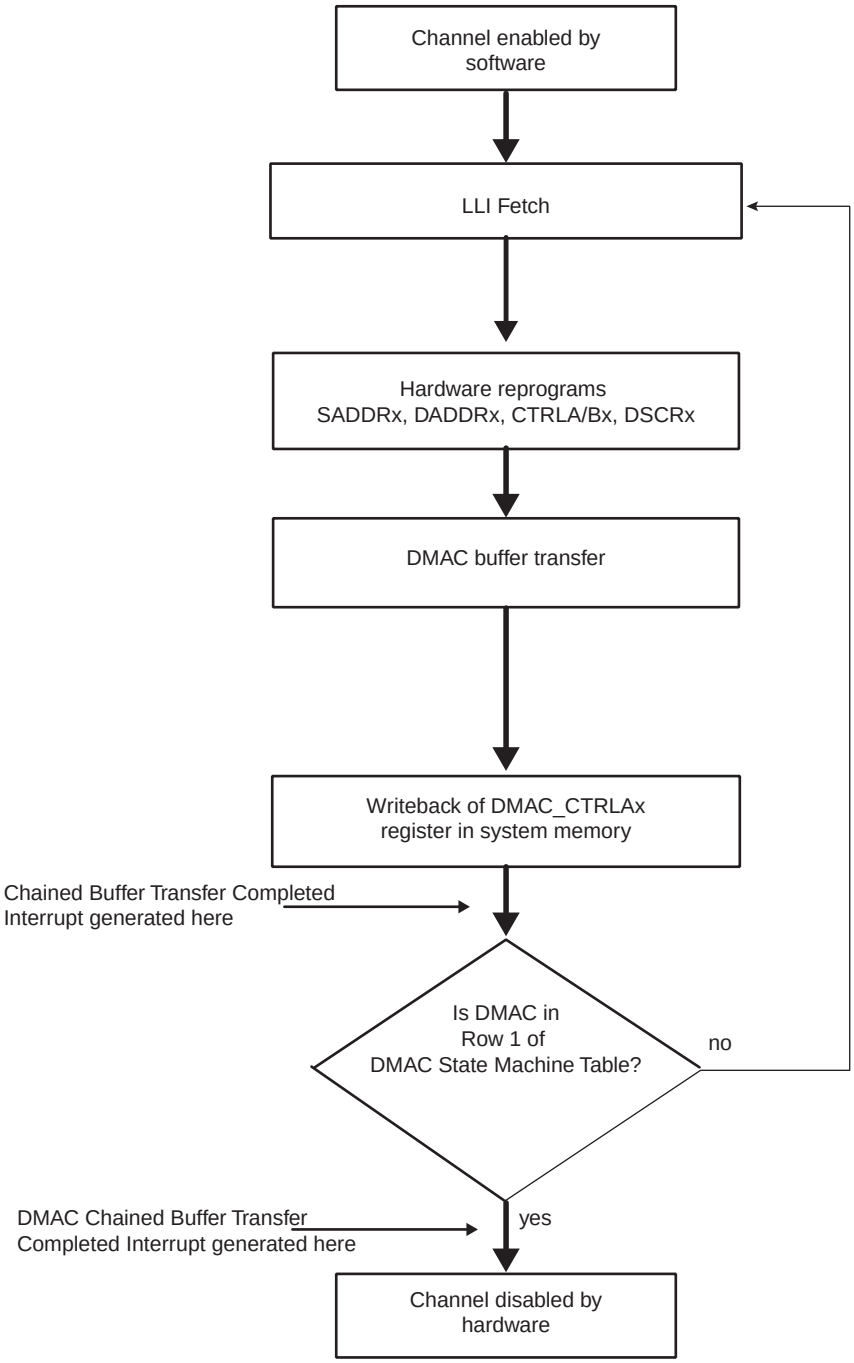
If the user needs to execute a DMAC transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum buffer size DMAC_CTRLAx.BTSIZE, then this can be achieved using the type of multi-buffer transfer as shown in Figure 31-8 on page 524.

Figure 31-8. Multi-buffer with Linked Address for Source and Destination Buffers are Contiguous



The DMAC transfer flow is shown in Figure 31-9 on page 525.

Figure 31-9. 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 519. 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 519. 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 519. 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 519 before the last buffer of the DMAC transfer has completed. The transfer is similar to that shown in Figure 31-10 on page 527. The DMAC transfer flow is shown in Figure 31-11 on page 528.

Figure 31-10.Multi-buffer DMAC Transfer with Source and Destination Address Auto-reloaded

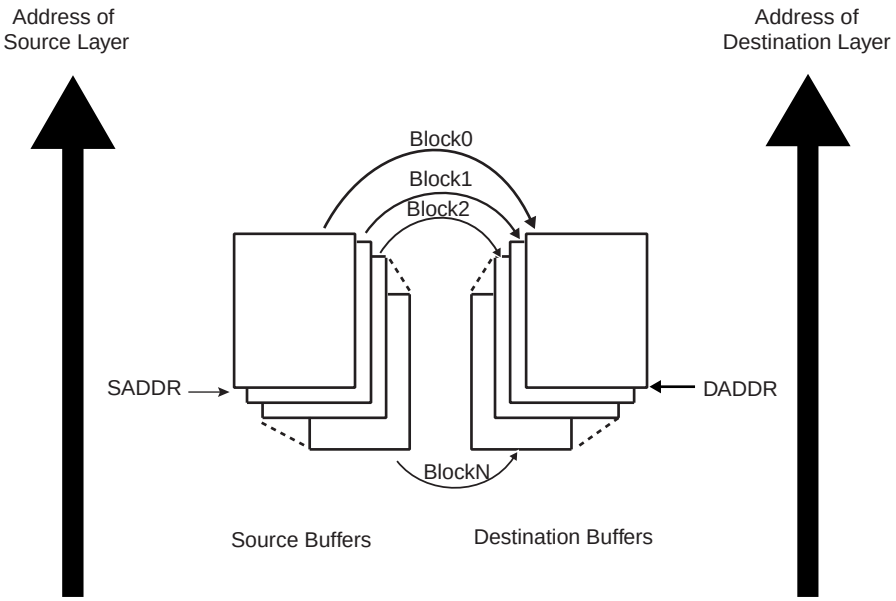
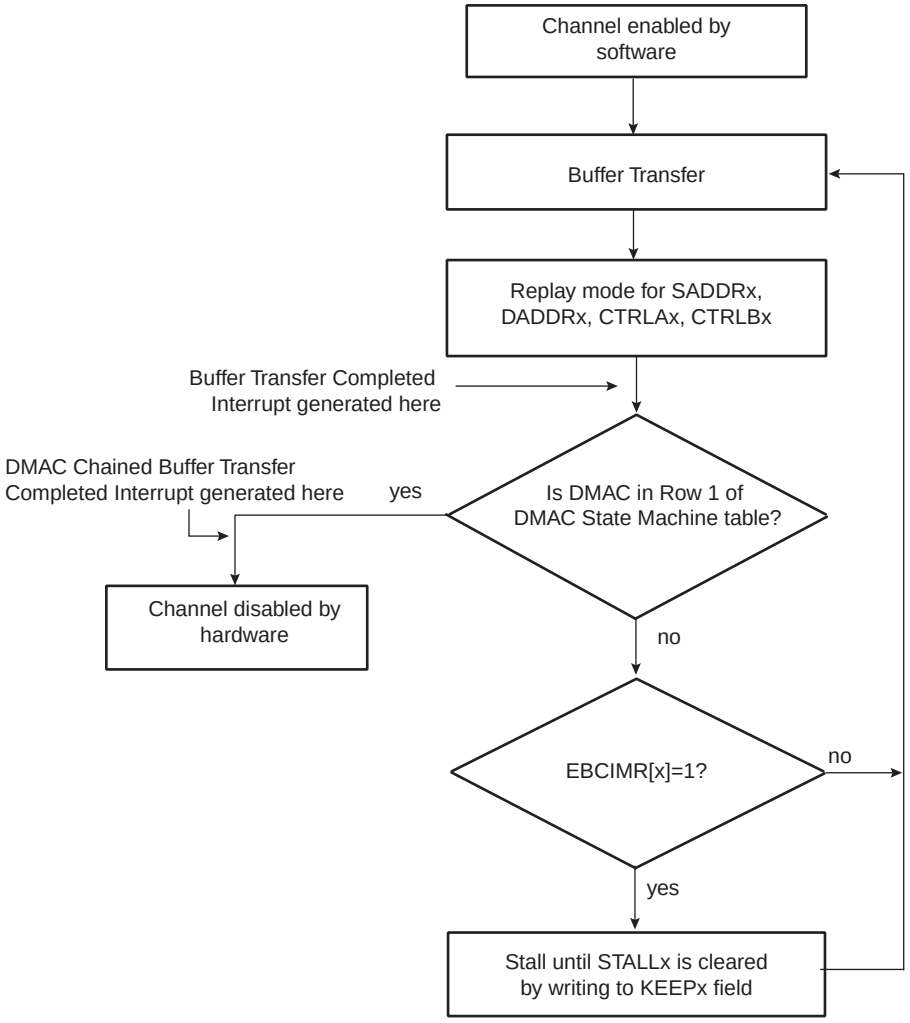


Figure 31-11.DMAC Transfer Flow for Source and Destination Address Auto-reloaded



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 519 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 517 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 519.
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.

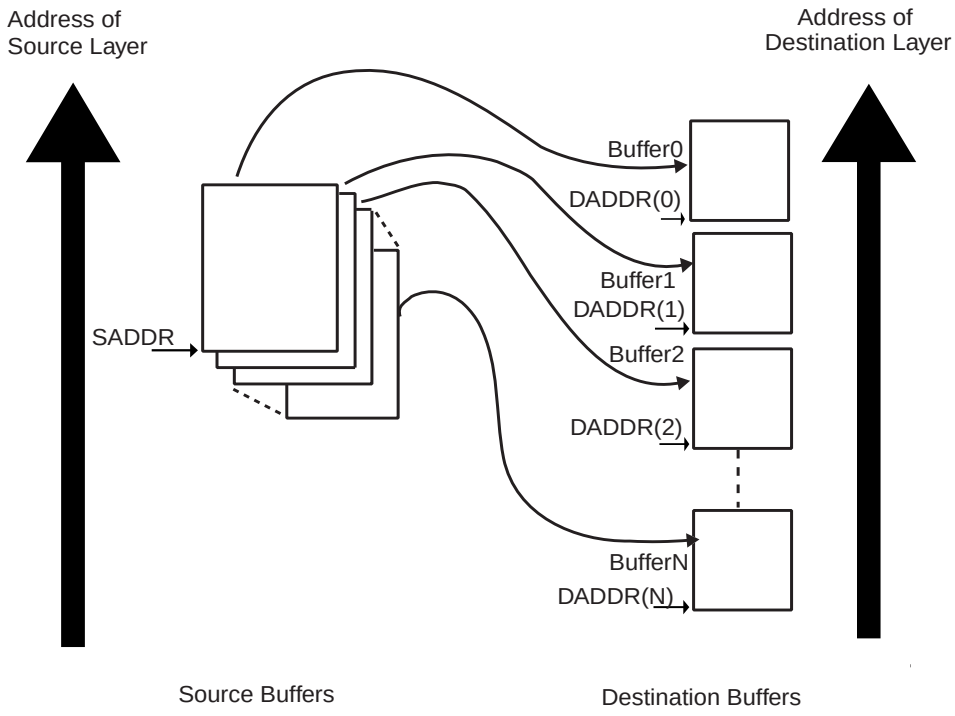
Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the polled LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLA.DONE bit was cleared at the start of the transfer.

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 519. 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 519, the following step is performed.

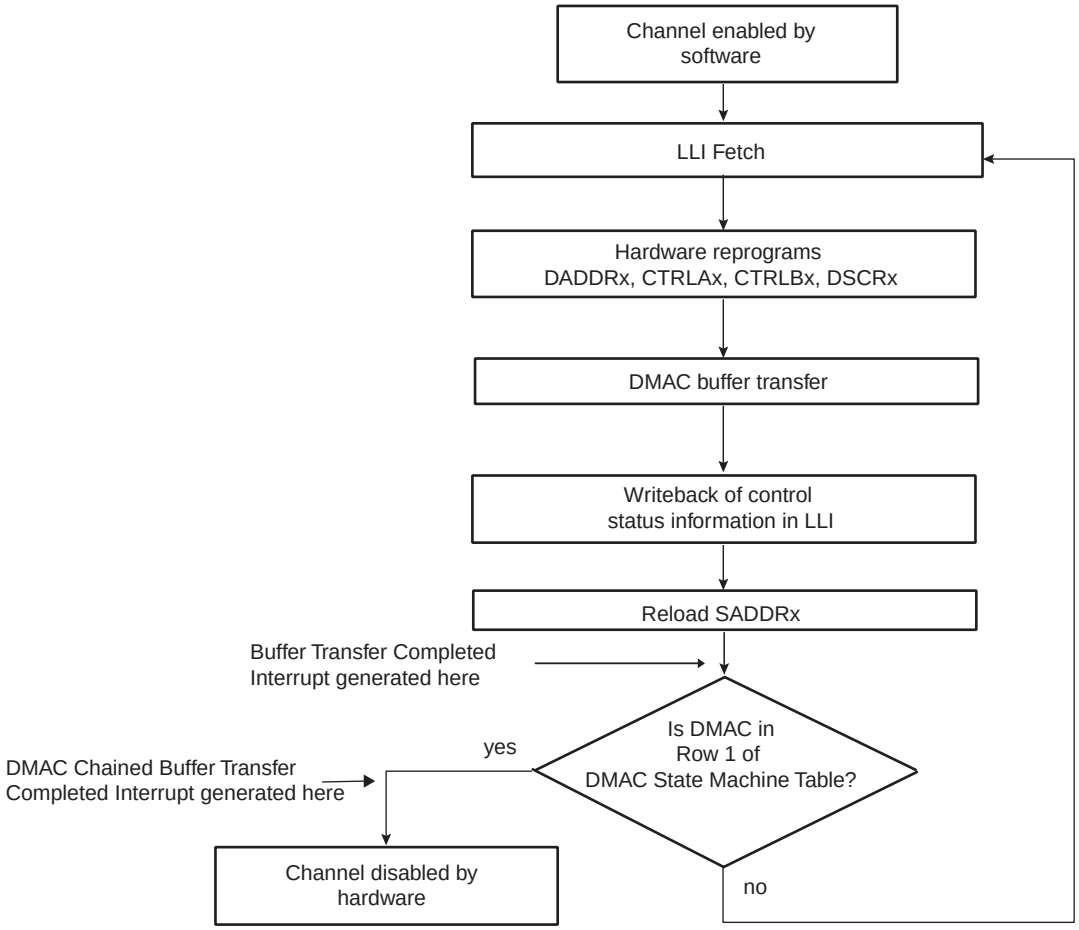
19. The DMAC fetches the next LLI from the memory location pointed to by the current DMAC_DSCRx register, and automatically reprograms the DMAC_DADDRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx channel registers. Note that the DMAC_SADDRx is not re-programmed as the reloaded value is used for the next 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 519. The DMAC transfer might look like that shown in Figure 31-12 on page 530.

Figure 31-12.Multi-buffer DMAC Transfer with Source Address Auto-reloaded and Linked List Destination Address



The DMAC Transfer flow is shown in Figure 31-13 on page 531.

Figure 31-13.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 519. 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 519. 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 519. 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 519.
    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 519 before the last buffer of the DMAC transfer has completed.

The transfer is similar to that shown in Figure 31-14 on page 533.

The DMAC Transfer flow is shown in Figure 31-15 on page 534.

Figure 31-14.Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address

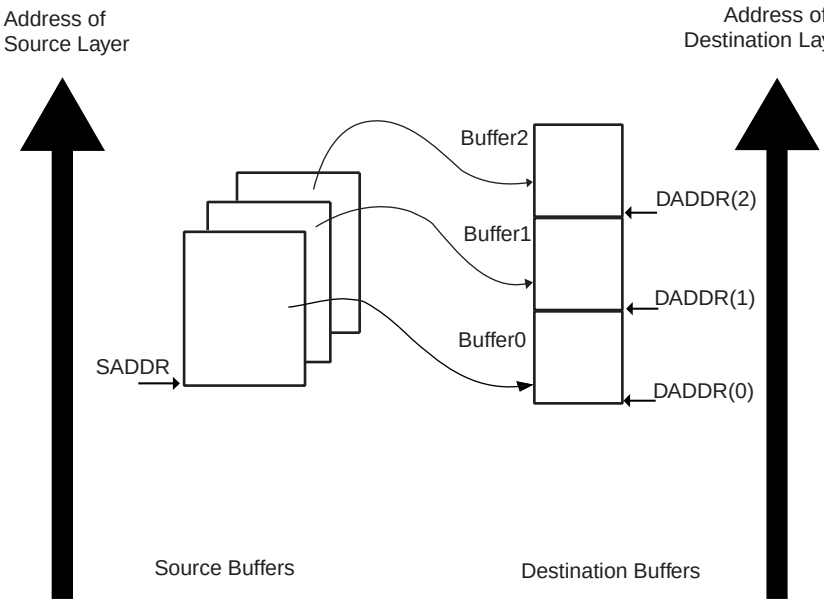
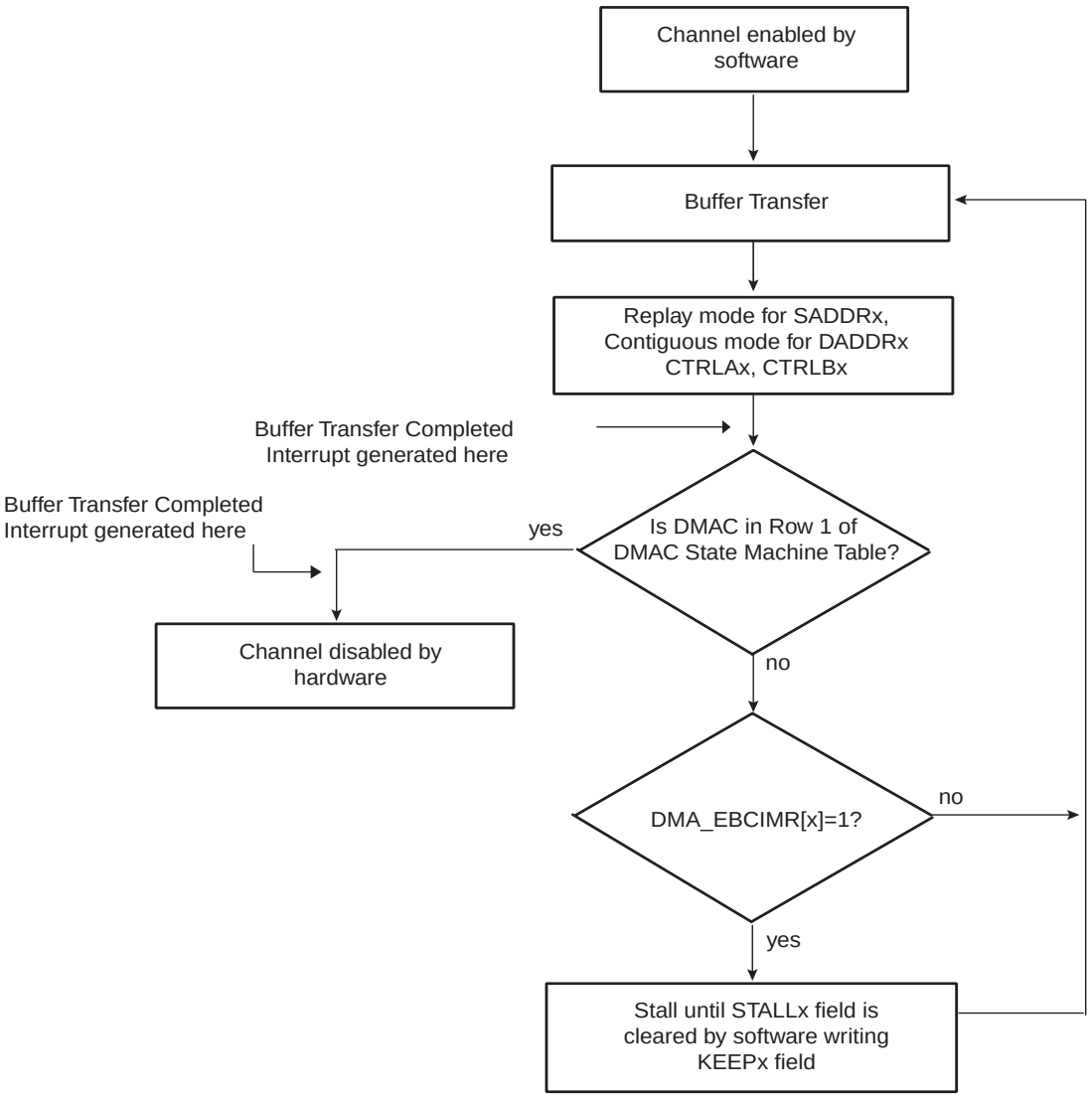


Figure 31-15.DMAC Transfer Replay Mode is Enabled for the Source and Contiguous Destination Address



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 519, 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 517 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 519
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.

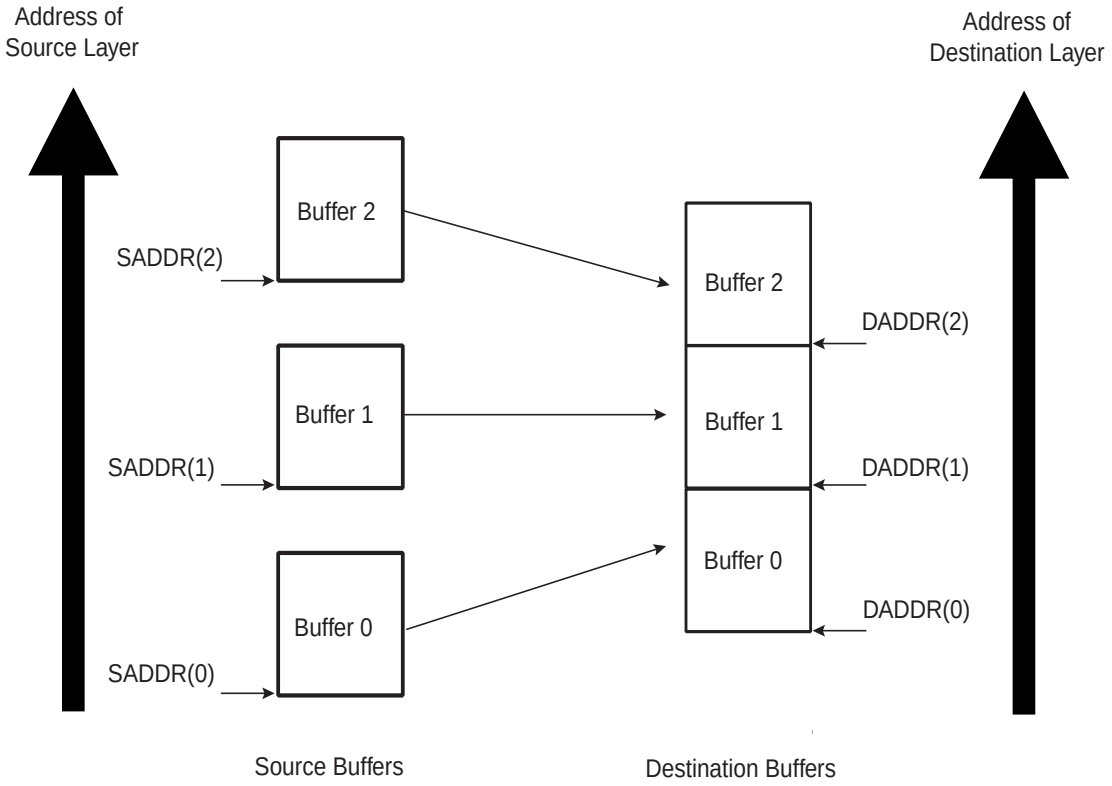
Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLAx.DONE bit was cleared at the start of the transfer.

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 519. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer.

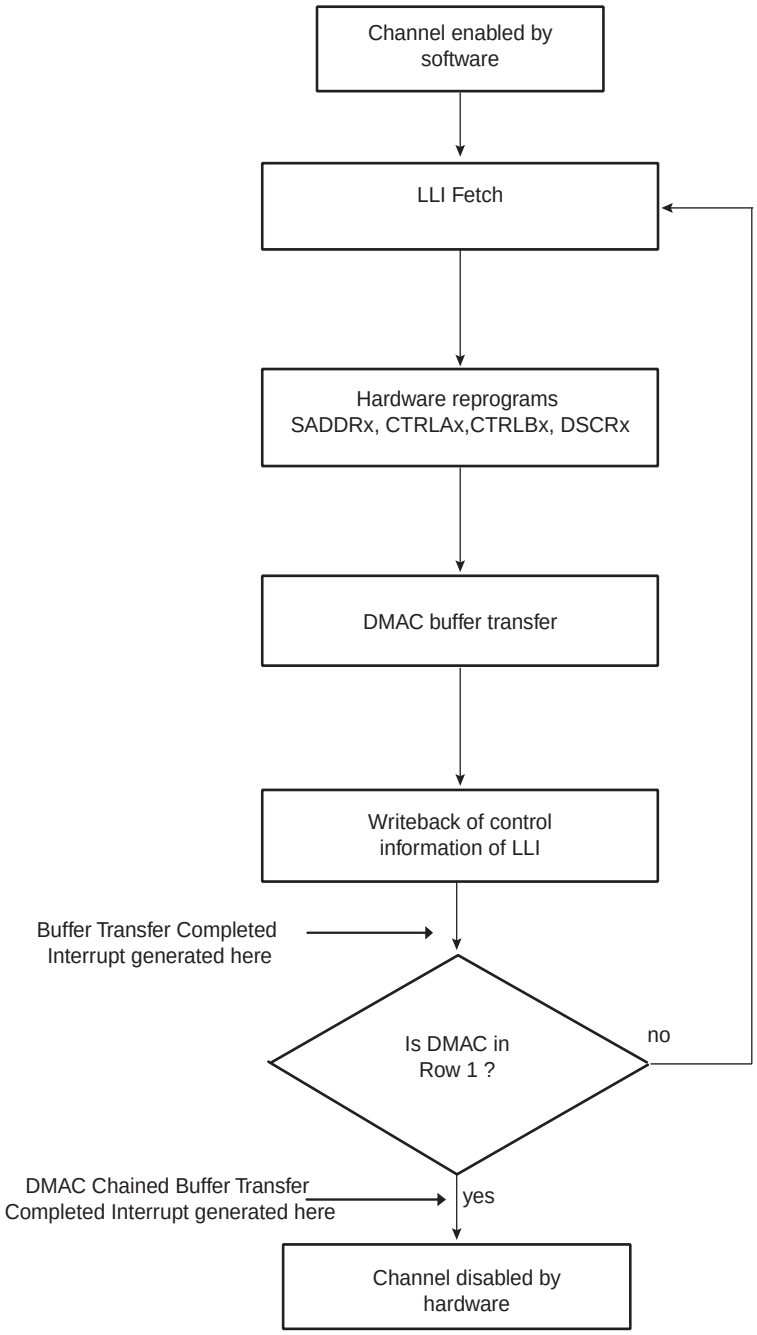
The DMAC transfer might look like that shown in Figure 31-16 on page 536. Note that the destination address is decrementing.

Figure 31-16.DMAC Transfer with Linked List Source Address and Contiguous Destination Address



The DMAC transfer flow is shown in Figure 31-17 on page 537.

Figure 31-17.DMAC Transfer Flow for Linked List Source Address and Contiguous Destination Address



31.5.6 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.

31.5.6.1 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.

31.6 DMAC Software Requirements

- There must not be any write operation to Channel registers in an active channel after the channel enable is made HIGH. If any channel parameters must be reprogrammed, this can only be done after disabling the DMAC channel.
- You must program the DMAC_SADDRx and DMAC_DADDRx channel registers with a byte, half-word and word aligned address depending on the source width and destination width.
- After the software disables a channel by writing into the channel disable register, it must re-enable the channel only after it has polled a 0 in the corresponding channel enable status register. This is because the current AHB Burst must terminate properly.
- If you program the BTSIZE field in the DMAC_CTRLA as zero, and the DMAC 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.

31.7 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 541
- "DMAC Enable Register" on page 542
- "DMAC Channel x [x = 0..7] Source Address Register" on page 553
- "DMAC Channel x [x = 0..7] Destination Address Register" on page 554
- "DMAC Channel x [x = 0..7] Descriptor Address Register" on page 555
- "DMAC Channel x [x = 0..7] Control A Register" on page 556
- "DMAC Channel x [x = 0..7] Control B Register" on page 558
- "DMAC Channel x [x = 0..7] Configuration Register" on page 560

31.8 DMA Controller (DMAC) User Interface

Table 31-5. 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	–	–	–

31.8.1 DMAC Global Configuration Register

Name: DMAC_GCFG

Address: 0xFFFFE600 (0), 0xFFFFE800 (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
–	–	–	–	–	–	–	DICEN
7	6	5	4	3	2	1	0
–	–	–	ARB_CFG	–	–	–	–

Note: Bit fields 0, 1, 2, and 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**

Value	Name	Description
0	FIXED	Fixed priority arbiter (see “Basic Definitions”)
1	ROUND_ROBIN	Modified round robin arbiter.

- **DICEN: Descriptor Integrity Check**

0: Descriptor Integrity Check Interface is Disabled.

1: Descriptor Integrity Check Interface is Enabled.

31.8.2 DMAC Enable Register

Name: DMAC_EN

Address: 0xFFFFE604 (0), 0xFFFFE804 (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.

31.8.3 DMAC Software Single Request Register

Name: DMAC_SREQ

Address: 0xFFFFE608 (0), 0xFFFFE808 (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.

31.8.4 DMAC Software Chunk Transfer Request Register

Name: DMAC_CREQ

Address: 0xFFFFE60C (0), 0xFFFFE80C (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.

31.8.5 DMAC Software Last Transfer Flag Register

Name: DMAC_LAST

Address: 0xFFFFE610 (0), 0xFFFFE810 (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.

31.8.6 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Enable Register

Name: DMAC_EBCIER

Address: 0xFFFFE618 (0), 0xFFFFE818 (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
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.

- **DICERRx: Descriptor Integrity Check Error [7:0]**

Descriptor Integrity Check Error Interrupt Enable Register. Set the relevant bit in the DICERR field to enable the interrupt for channel i.

31.8.7 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Disable Register

Name: DMAC_EBCIDR

Address: 0xFFFFE61C (0), 0xFFFFE81C (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
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.

- **DICERRx: Descriptor Integrity Check Error [7:0]**

Descriptor Integrity Check Error Interrupt Disable Register, When set, a bit of the DICERR field disables the interrupt from the relevant DMAC channel.

31.8.8 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Mask Register

Name: DMAC_EBCIMR

Address: 0xFFFFE620 (0), 0xFFFFE820 (1)

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
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.

- **DICERRx: Descriptor Integrity Check Error [7:0]**

0: Descriptor Integrity Check Error Interrupt is disabled for channel i.

1: Descriptor Integrity Check Error Interrupt is enabled for channel i.

31.8.9 DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register

Name: DMAC_EBCISR

Address: 0xFFFFE624 (0), 0xFFFFE824 (1)

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
DICERR7	DICERR6	DICERR5	DICERR4	DICERR3	DICERR2	DICERR1	DICERR0
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.

- **DICERRx: Descriptor Integrity Check Error [7:0]**

When DICERR[*i*] is set, Channel *i* has detected a Descriptor Integrity Check Error.

31.8.10 DMAC Channel Handler Enable Register

Name: DMAC_CHER

Address: 0xFFFFE628 (0), 0xFFFFE828 (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.

31.8.11 DMAC Channel Handler Disable Register

Name: DMAC_CHDR

Address: 0xFFFFE62C (0), 0xFFFFE82C (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.

31.8.12 DMAC Channel Handler Status Register

Name: DMAC_CHSR

Address: 0xFFFFE630 (0), 0xFFFFE830 (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.

31.8.13 DMAC Channel x [x = 0..7] Source Address Register

Name: DMAC_SADDRx [x = 0..7]

Address: 0xFFFFE63C (0)[0], 0xFFFFE664 (0)[1], 0xFFFFE68C (0)[2], 0xFFFFE6B4 (0)[3], 0xFFFFE6DC (0)[4], 0xFFFFE704 (0)[5], 0xFFFFE72C (0)[6], 0xFFFFE754 (0)[7], 0xFFFFE83C (1)[0], 0xFFFFE864 (1)[1], 0xFFFFE88C (1)[2], 0xFFFFE8B4 (1)[3], 0xFFFFE8DC (1)[4], 0xFFFFE904 (1)[5], 0xFFFFE92C (1)[6], 0xFFFFE954 (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.

31.8.14 DMAC Channel x [x = 0..7] Destination Address Register

Name: DMAC_DADDRx [x = 0..7]

Address: 0xFFFFE640 (0)[0], 0xFFFFE668 (0)[1], 0xFFFFE690 (0)[2], 0xFFFFE6B8 (0)[3], 0xFFFFE6E0 (0)[4], 0xFFFFE708 (0)[5], 0xFFFFE730 (0)[6], 0xFFFFE758 (0)[7], 0xFFFFE840 (1)[0], 0xFFFFE868 (1)[1], 0xFFFFE890 (1)[2], 0xFFFFE8B8 (1)[3], 0xFFFFE8E0 (1)[4], 0xFFFFE908 (1)[5], 0xFFFFE930 (1)[6], 0xFFFFE958 (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.

31.8.15 DMAC Channel x [x = 0..7] Descriptor Address Register

Name: DMAC_DSCRx [x = 0..7]

Address: 0xFFFFE644 (0)[0], 0xFFFFE66C (0)[1], 0xFFFFE694 (0)[2], 0xFFFFE6BC (0)[3], 0xFFFFE6E4 (0)[4], 0xFFFFE70C (0)[5], 0xFFFFE734 (0)[6], 0xFFFFE75C (0)[7], 0xFFFFE844 (1)[0], 0xFFFFE86C (1)[1], 0xFFFFE894 (1)[2], 0xFFFFE8BC (1)[3], 0xFFFFE8E4 (1)[4], 0xFFFFE90C (1)[5], 0xFFFFE934 (1)[6], 0xFFFFE95C (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 (first DMA Master Interface)
01	AHB_IF1	The buffer transfer descriptor is fetched via AHB-Lite Interface 1 (second DMA Master Interface)
10	AHB_IF2	The buffer transfer descriptor is fetched via AHB-Lite Interface 2 (third DMA Master Interface)

- **DSCR: Buffer Transfer Descriptor Address**

This address is word aligned.

31.8.16 DMAC Channel x [x = 0..7] Control A Register

Name: DMAC_CTRLAx [x = 0..7]

Address: 0xFFFFE648 (0)[0], 0xFFFFE670 (0)[1], 0xFFFFE698 (0)[2], 0xFFFFE6C0 (0)[3], 0xFFFFE6E8 (0)[4], 0xFFFFE710 (0)[5], 0xFFFFE738 (0)[6], 0xFFFFE760 (0)[7], 0xFFFFE848 (1)[0], 0xFFFFE870 (1)[1], 0xFFFFE898 (1)[2], 0xFFFFE8C0 (1)[3], 0xFFFFE8E8 (1)[4], 0xFFFFE910 (1)[5], 0xFFFFE938 (1)[6], 0xFFFFE960 (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 564

- **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
10	WORD	the transfer size is set to 32-bit width
11	DWORD	the transfer size is set to 64-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
10	WORD	the transfer size is set to 32-bit width
11	DWORD	the transfer size is set to 64-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.

31.8.17 DMAC Channel x [x = 0..7] Control B Register

Name: DMAC_CTRLBx [x = 0..7]

Address: 0xFFFFE64C (0)[0], 0xFFFFE674 (0)[1], 0xFFFFE69C (0)[2], 0xFFFFE6C4 (0)[3], 0xFFFFE6EC (0)[4], 0xFFFFE714 (0)[5], 0xFFFFE73C (0)[6], 0xFFFFE764 (0)[7], 0xFFFFE84C (1)[0], 0xFFFFE874 (1)[1], 0xFFFFE89C (1)[2], 0xFFFFE8C4 (1)[3], 0xFFFFE8EC (1)[4], 0xFFFFE914 (1)[5], 0xFFFFE93C (1)[6], 0xFFFFE964 (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 (first DMA Master Interface)
01	AHB_IF1	The source transfer is done via AHB_Lite Interface 1 (second DMA Master Interface)
10	AHB_IF2	The source transfer is done via AHB_Lite Interface 2 (third DMA Master Interface)

- **DIF: Destination Interface Selection Field**

Value	Name	Description
00	AHB_IF0	The destination transfer is done via AHB_Lite Interface 0 (first DMA Master Interface)
01	AHB_IF1	The destination transfer is done via AHB_Lite Interface 1 (second DMA Master Interface)
10	AHB_IF2	The destination transfer is done via AHB_Lite Interface 2 (third DMA Master Interface)

- **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
00	MEM2MEM_DMA_FC	Memory-to-Memory Transfer DMAC is flow controller
01	MEM2PER_DMA_FC	Memory-to-Peripheral Transfer DMAC is flow controller
10	PER2MEM_DMA_FC	Peripheral-to-Memory Transfer DMAC is flow controller
11	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.

31.8.18 DMAC Channel x [x = 0..7] Configuration Register

Name: DMAC_CFGx [x = 0..7]

Address: 0xFFFFE650 (0)[0], 0xFFFFE678 (0)[1], 0xFFFFE6A0 (0)[2], 0xFFFFE6C8 (0)[3], 0xFFFFE6F0 (0)[4], 0xFFFFE718 (0)[5], 0xFFFFE740 (0)[6], 0xFFFFE768 (0)[7], 0xFFFFE850 (1)[0], 0xFFFFE878 (1)[1], 0xFFFFE8A0 (1)[2], 0xFFFFE8C8 (1)[3], 0xFFFFE8F0 (1)[4], 0xFFFFE918 (1)[5], 0xFFFFE940 (1)[6], 0xFFFFE968 (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_PER_MSB		DST_H2SEL	DST_REP	SRC_PER_MSB		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 564

- **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.

- **SRC_PER_MSB: SRC_PER Most Significant Bits**

This field indicates the Most Significant bits of the SRC_PER field.

- **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.

- **DST_PER_MSB: DST_PER Most Significant Bits**

This field indicates the Most Significant bits of the DST_PER field.

- **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.

31.8.19 DMAC Channel x [x = 0..7] Source Picture-in-Picture Configuration Register

Name: DMAC_SPIPx [x = 0..7]

Address: 0xFFFFE654 (0)[0], 0xFFFFE67C (0)[1], 0xFFFFE6A4 (0)[2], 0xFFFFE6CC (0)[3], 0xFFFFE6F4 (0)[4], 0xFFFFE71C (0)[5], 0xFFFFE744 (0)[6], 0xFFFFE76C (0)[7], 0xFFFFE854 (1)[0], 0xFFFFE87C (1)[1], 0xFFFFE8A4 (1)[2], 0xFFFFE8CC (1)[3], 0xFFFFE8F4 (1)[4], 0xFFFFE91C (1)[5], 0xFFFFE944 (1)[6], 0xFFFFE96C (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							

- **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.

31.8.20 DMAC Channel x [x = 0..7] Destination Picture-in-Picture Configuration Register

Name: DMAC_DPIPx [x = 0..7]

Address: 0xFFFFE658 (0)[0], 0xFFFFE680 (0)[1], 0xFFFFE6A8 (0)[2], 0xFFFFE6D0 (0)[3], 0xFFFFE6F8 (0)[4], 0xFFFFE720 (0)[5], 0xFFFFE748 (0)[6], 0xFFFFE770 (0)[7], 0xFFFFE858 (1)[0], 0xFFFFE880 (1)[1], 0xFFFFE8A8 (1)[2], 0xFFFFE8D0 (1)[3], 0xFFFFE8F8 (1)[4], 0xFFFFE920 (1)[5], 0xFFFFE948 (1)[6], 0xFFFFE970 (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							

- **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.

31.8.21 DMAC Write Protect Mode Register

Name: DMAC_WPMR

Address: 0xFFFFE7E4 (0), 0xFFFFE9E4 (1)

Access: Read-write

Reset: See Table 31-5

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 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 541
- “DMAC Enable Register” on page 542
- “DMAC Channel x [x = 0..7] Source Address Register” on page 553
- “DMAC Channel x [x = 0..7] Destination Address Register” on page 554
- “DMAC Channel x [x = 0..7] Descriptor Address Register” on page 555
- “DMAC Channel x [x = 0..7] Control A Register” on page 556
- “DMAC Channel x [x = 0..7] Control B Register” on page 558
- “DMAC Channel x [x = 0..7] Configuration Register” on page 560

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x444D41	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

31.8.22 DMAC Write Protect Status Register

Name: DMAC_WPSR

Address: 0xFFFFE7E8 (0), 0xFFFFE9E8 (1)

Access: Read-only

Reset: See Table 31-5

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 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. AHB LCD Controller (LCDC)

32.1 Description

The LCD controller (LCDC) consists of logic for transferring LCD image data from an external display buffer to an LCD module. The LCD has one display input buffer per overlay that fetches pixels through the dual AHB master interface and a lookup table to allow palletized display configurations. The LCD controller is programmable on a per overlay basis, and supports different LCD resolutions, window sizes, image formats and pixel depths.

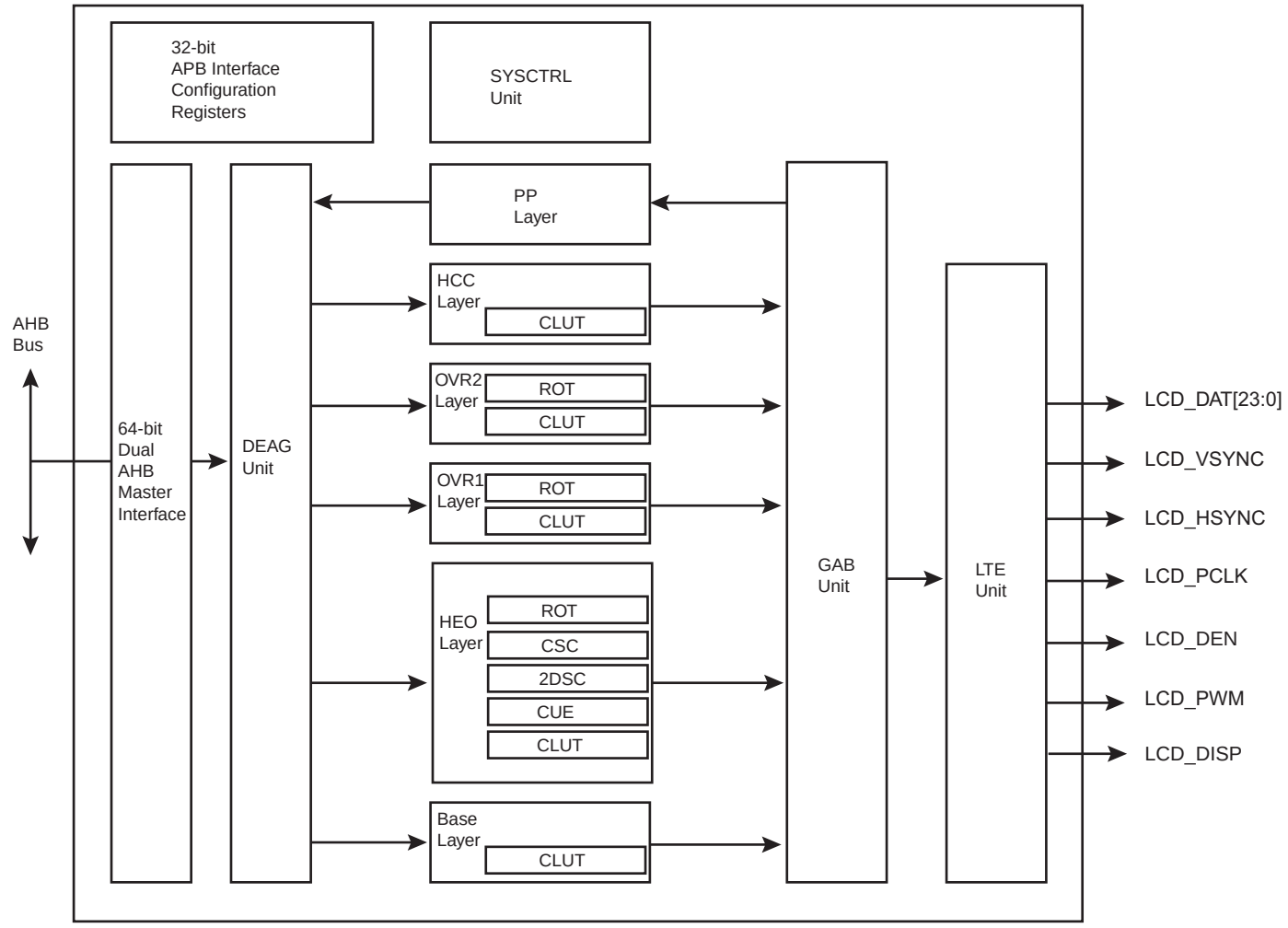
The LCD is connected to the ARM Advanced High Performance Bus (AHB) as a master for reading pixel data. It also integrates an APB interface to configure its registers.

32.2 Embedded Characteristics

- Dual AHB Master Interface
- Supports Single Scan Active TFT Display
- Supports 12-, 16-, 18- and 24-bit Output Mode through the Spatial Dithering Unit
- Asynchronous Output Mode Supported (at synthesis time)
- 1, 2, 4, 8 bits per pixel (palletized)
- 12, 16, 18, 19, 24, 25 and 32 bits per pixel (non palletized)
- Supports One Base Layer (background)
- Supports Two Overlay Layer Windows
- Supports One High End Overlay (HEO) Window
- Supports One Hardware Cursor, Fixed or Free Size
- Hardware Cursor Fixed Size on the following patterns: 32x32, 64x64 and 128x128
- Little Endian Memory Organization
- Programmable Timing Engine, with Integer Clock Divider
- Programmable Polarity for Data, Line Synchro and Frame Synchro.
- Display Size up to 2048x2048, or up to 720p in video format
- Color Lookup Table with up to 256 entries and Predefined 8-bit Alpha
- Programmable Negative and Positive Row Striding for all Layers
- Programmable Negative and Positive Pixel Striding for all Overlay1, Overlay2 and HEO layers
- High End Overlay supports 4:2:0 Planar Mode and Semiplanar Mode
- High End Overlay supports 4:2:2 Planar Mode, Semiplanar Mode and Packed
- High End Overlay includes Chroma Upsampling Unit
- Horizontal and Vertical Rescaling unit with Edge Interpolation and Independent Non Integer Ratio
- Hidden Layer Removal supported.
- Integrates Fully Programmable Color Space Conversion
- Overlay1, Overlay2 and High End Overlay Integrate Rotation Engine: 90, 180, 270
- Blender Function Supports Arbitrary 8-bit Alpha Value and Chroma Keying
- DMA User interface uses Linked List Structure and Add-to-queue Structure

32.3 Block Diagram

Figure 32-1. Block Diagram



HEO : High End Overlay

CUE : Chroma Upsampling Engine

CSC : Color Space Conversion

2DSC : Two Dimension Scaler

DEAG : DMA Engine Address Generation

HCC: Hardware Cursor Channel

GAB : Global Alpha Blender

LTE: LCD Timing Engine

ROT : Hardware Rotation

32.4 I/O Lines Description

Name	Description	Type
LCD_PWM	Contrast control signal, using Pulse Width Modulation	**Output**
LCD_HSYNC	Horizontal Synchronization Pulse	Output
LCD_VSYNC	Vertical Synchronization Pulse	Output
LCD_DAT[23:0]	LCD 24-bit data bus	Output
LCD_DEN	Data Enable	Output
LCD_DISP	Display Enable signal	Output
LCD_PCLK	Pixel Clock	Output

32.5 Product Dependencies

32.5.1 I/O Lines

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

Table 32-1. I/O Lines

Instance	Signal	I/O Line	Peripheral
LCDC	LCDDAT0	PA0	A
LCDC	LCDDAT1	PA1	A
LCDC	LCDDAT2	PA2	A
LCDC	LCDDAT3	PA3	A
LCDC	LCDDAT4	PA4	A
LCDC	LCDDAT5	PA5	A
LCDC	LCDDAT6	PA6	A
LCDC	LCDDAT7	PA7	A
LCDC	LCDDAT8	PA8	A
LCDC	LCDDAT9	PA9	A
LCDC	LCDDAT10	PA10	A
LCDC	LCDDAT11	PA11	A
LCDC	LCDDAT12	PA12	A
LCDC	LCDDAT13	PA13	A
LCDC	LCDDAT14	PA14	A
LCDC	LCDDAT15	PA15	A
LCDC	LCDDAT16	PA16	A
LCDC	LCDDAT16	PC14	C
LCDC	LCDDAT17	PA17	A
LCDC	LCDDAT17	PC13	C

Table 32-1. I/O Lines

LCDC	LCDDAT18	PA18	A
LCDC	LCDDAT18	PC12	C
LCDC	LCDDAT19	PA19	A
LCDC	LCDDAT19	PC11	C
LCDC	LCDDAT20	PA20	A
LCDC	LCDDAT20	PC10	C
LCDC	LCDDAT21	PA21	A
LCDC	LCDDAT21	PC15	C
LCDC	LCDDAT22	PA22	A
LCDC	LCDDAT22	PE27	C
LCDC	LCDDAT23	PA23	A
LCDC	LCDDAT23	PE28	C
LCDC	LCDDEN	PA29	A
LCDC	LCDDISP	PA25	A
LCDC	LCDHSYNC	PA27	A
LCDC	LCDPCK	PA28	A
LCDC	LCDPWM	PA24	A
LCDC	LCDVSYNC	PA26	A

32.5.2 Power Management

The LCD Controller is not continuously clocked. The user must first enable the LCD Controller clock in the Power Management Controller before using it (PMC_PCER).

32.5.3 Interrupt Sources

The LCD Controller interrupt line is connected to one of the internal sources of the Advanced Interrupt Controller. Using the LCD Controller interrupt requires prior programming of the AIC.

Table 32-2. Peripheral IDs

Instance	ID
LCDC	36

32.6 Functional Description

The LCD module integrates the following digital blocks:

- DMA Engine Address Generation (DEAG). This block performs data prefetch and requests access to the AHB interface.
- Input Overlay FIFO stores the stream of pixels.
- Color Lookup Table (CLUT). These 256 RAM-based lookup table entries are selected when the color depth is set to 1, 2, 4 or 8 bpp.
- Chroma Upsampling Engine (CUE). This block is selected when the input image sampling format is YUV (Y'CbCr) 4:2:0 and converts it to higher quality 4:4:4 image.
- Color Space Conversion (CSC) changes the color spare from YUV to RGB.
- Two Dimension Scaler (2DSC) resizes the image.
- Global Alpha Blender (GAB) performs programmable 256 level alpha blending.
- Output FIFO stores the blended pixel prior to display.
- LCD Timing Engine provides a fully programmable HSYNC-VSYNC interface.

The DMA controller reads the image through the AHB master interface. The LCD controller engine formats the display data, then the GAB performs alpha blending if required, and writes the final pixel into the output FIFO. The programmable timing engine drives a valid pixel onto the LCD_DAT[23:0] display bus.

32.6.1 Timing Engine Configuration

32.6.1.1 Pixel Clock Period Configuration

The pixel clock (PCLK) generated by the timing engine is the source clock (SCLK) divided by the field CLKDIV in the LCDC_LCDCFG0 register. The source clock can be selected between the system clock and the 2x system clock with the field CLKSEL located in the LCDC_LCDCFG0 register. The Pixel Clock period formula is given below:

$$PCLK = \frac{SCLK}{CLKDIV + 2}$$

The Pixel Clock polarity is also programmable.

32.6.1.2 Horizontal and Vertical Synchronization Configuration

The following fields are used to configure the timing engine:

- HSPW field
- VSPW field
- VFPW field
- VBPW field
- HFPW field
- HBPW field
- PPL field
- RPF field

The polarity of output signals is also programmable.

32.6.1.3 Timing Engine Power Up Software Operation

The following sequence is used to enable the display:

1. Configure LCD timing parameters, signal polarity and clock period.
2. Enable the Pixel Clock by writing one to the CLKEN field of the LCDC_LCDEN register.
3. Poll CLKSTS field of the LCDC_LCDSR register to check that the clock is running.
4. Enable Horizontal and Vertical Synchronization by writing one to the SYNCEN field of the LCDC_LCDEN register.
5. Poll LCDSTS field of the LCDC_LCDSR register to check that the synchronization is up.

6. Enable the display power signal by writing one to the DISPEN field of the LCDC_LCDEN register.
7. Poll DISPSTS field of the LCDC_LCDSR register to check that the power signal is activated.

The GUARDTIME field of the LCDC_LCDCFG5 register is used to configure the number of frames before the assertion of the DISP signal.

32.6.1.4 Timing Engine Power Down Software Operation

The following sequence is used to disable the display:

1. Disable the DISP signal by writing DISPDIS field of the LCDC_LCDDIS register.
2. Poll DISPSTS field of the LCDC_LCDSR register to verify that the DISP is no longer activated.
3. Disable the HSYNC and VSYNC signals by writing one to SYNCDIS field of the LCDC_LCDDIS register.
4. Poll LCDSTS field of the LCDC_LCDSR register to check that the synchronization is off.
5. Disable the Pixel clock by writing one in the CLKDIS field of the LCDC_LCDDIS register.

32.6.2 DMA Software Operations

32.6.2.1 DMA Channel Descriptor (DSCR) Alignment and Structure

The DMA Channel Descriptor (DSCR) must be aligned on a 64-bit boundary.

The DMA Channel Descriptor structure contains three fields:

- DSCR.CHXADDR: Frame Buffer base address register
- DSCR.CHXCTRL: Transfer Control register.
- DSCR.CHXNEXT: Next Descriptor Address register.

Table 32-3. DMA Channel Descriptor Structure

System Memory	Structure Field for Channel CHX
DSCR + 0x0	ADDR
DSCR + 0x4	CTRL
DSCR + 0x8	NEXT

32.6.2.2 Programming a DMA Channel

1. Check the status of the channel by reading the CHXCHSR register.
2. Write the channel descriptor (DSCR) structure in the system memory by writing DSCR.CHXADDR Frame base address, DSCR.CHXCTRL channel control and DSCR.CHXNEXT next descriptor location.
3. If more than one descriptor is expected, the field DFETCH of DSCR.CHXCTRL is set to one to enable the descriptor fetch operation.
4. Write the DSCR.CHXNEXT register with the address location of the descriptor structure and set DFETCH field of the DSCR.CHXCTRL register to one.
5. Enable the relevant channel by writing one to the CHEN field of the CHXCHER register.
6. An interrupt may be raised if unmasked when the descriptor has been loaded.

32.6.2.3 Disabling a DMA channel

1. Clearing the DFETCH bit in the DSCR.CHXCTRL field of the DSCR structure will disable the channel at the end of the frame.
2. Setting the DSCR.CHXNEXT field of the DSCR structure will disable the channel at the end of the frame.
3. Writing one to the CHDIS field of the CHXCHDR register will disable the channel at the end of the frame.
4. Writing one to the CHRST field of the CHXCHDR register will disable the channel immediately. This may occur in the middle of the image.
5. Polling CHSR field in the CHXCHSR register until the channel is successfully disabled.

32.6.2.4 DMA Dynamic Linking of a New Transfer Descriptor

1. Write the new descriptor structure in the system memory.
2. Write the address of the new structure in the CHXHEAD register.
3. Add the new structure to the queue of descriptors by writing one to the A2QEN field of the CHXCHER register.
4. The new descriptor will be added to the queue on the next frame.
5. An interrupt will be raised if unmasked, when the head descriptor structure has been loaded by the DMA channel.

32.6.2.5 DMA Interrupt Generation

The DMA controller operation sets the following interrupt flags in the interrupt status register CHXISR:

- DMA field indicates that the DMA transfer is completed.
- DSCR field indicates that the descriptor structure is loaded in the DMA controller.
- ADD field indicates that a descriptor has been added to the descriptor queue.
- DONE field indicates that the channel transfer has terminated and the channel is automatically disabled.

32.6.2.6 DMA Address Alignment Requirements

When programming the DSCR.CHXADDR field of the DSCR structure the following requirement must be met.

Table 32-4. DMA Address Alignment when CLUT Mode is Selected

CLUT Mode	DMA Address Alignment
1 bpp	8 bits
2 bpp	8 bits
4 bpp	8 bits
8 bpp	8 bits

Table 32-5. DMA Address Alignment when RGB Mode is Selected

RGB Mode	DMA Address Alignment
12 bpp RGB 444	16 bits
16 bpp ARGB 4444	16 bits
16 bpp RGBA 4444	16 bits
16 bpp RGB 565	16 bits
16 bpp TRGB 1555	16 bits
18 bpp RGB 666	32 bits
18 bpp RGB 666 PACKED	8 bits
19 bpp TRGB 1666	32 bits
19 bpp TRGB 1666	8 bits
24 bpp RGB 888	32 bits
24 bpp RGB 888 PACKED	8 bits
25 bpp TRGB 1888	32 bits
32 bpp ARGB 8888	32 bits
32 bpp RGBA 8888	32 bits

Table 32-6. DMA Address Alignment when YUV Mode is Selected

YUV Mode	DMA Address Alignment
32 bpp AYCrCb	32 bits
16 bpp YCrCb 4:2:2	32 bits
16 bpp semiplanar YCrCb 4:2:2	Y 8 bits
	CrCb 16 bits
16 bpp planar YCrCb 4:2:2	Y 8 bits
	Cr 8 bits
	Cb 8 bits
12 bpp YCrCb 4:2:0	Y 8 bits
	CrCb 16 bits

Table 32-6. DMA Address Alignment when YUV Mode is Selected

YUV Mode	DMA Address Alignment
12 bpp YCrCb 4:2:0	Y 8 bits
	Cr 8 bits
	Cb 8 bits

32.6.3 Overlay Software Configuration

32.6.3.1 System Bus Access Attributes

These attributes are defined to improve bandwidth of the overlay.

- LOCKDIS field: when set to one the AHB lock signal is not asserted when the PSTRIDE value is different from zero (rotation in progress).
- ROTDIS field: when set to one the Pixel Striding optimization is disabled.
- DLBO field: when set to one only defined burst lengths are performed when the DMA channel retrieves the data from the memory.
- BLEN field: defines the maximum burst length of the DMA channel.
- SIF field: defines the targeted DMA interface.

32.6.3.2 Color Attributes

- CLUTMODE field: selects one color lookup mode
- RGBMODE field: selects the RGB mode.
- YUVMODE field: selects the Luminance Chrominance mode.

32.6.3.3 Window Position, Size, Scaling and Striding Attributes

- XPOS: YPOS fields define the position of the overlay window.
- XSIZE: YSIZE fields define the size of the displayed window.
- XMEMSIZE: YMEMSIZE fields define the size of the image frame buffer.
- XSTRIDE: PSTRIDE fields define the line and pixel striding.
- XFACTOR: YFACTOR fields define the scaling ratio.

The position and size attributes are to be programmed to keep the window within the display area.

When the color lookup mode is enabled the following restrictions apply on the horizontal and vertical window size:

Table 32-7. Color Lookup Mode and Window Size

CLUT MODE	x-y Size Requirement
1 bpp	multiple of 8 pixels
2 bpp	multiple of 4 pixels
4 bpp	multiple of 2 pixels
8 bpp	free size

Pixel striding is disabled when CLUT mode is enabled.

When YUV mode is enabled the following restrictions apply on the window size:

Table 32-8. YUV Mode and Window Size

YUV MODE	x-y Requirement, Scaling Turned Off	x-y Requirement, Scaling Turned On
AYUV	free size	x-y size is greater than 5
YUV 4:2:2 packed	xsize is greater than 2 pixels	x-y size is greater than 5
YUV 4:2:2 semiplanar	xsize is greater than 2 pixels	x-y size is greater than 5
YUV 4:2:2 planar	xsize is greater than 2 pixels	x-y size is greater than 5
YUV 4:2:0 semiplanar	xsize is greater that 2 pixels	x-y size is greater than 5
YUV 4:2:0 planar	xsize is greater than 2 pixels	x-y size is greater than 5

In RGB mode, there is no restriction on the line length.

32.6.3.4 Overlay Blender Attributes

When two or more video layers are used, alpha blending is performed to define the final image displayed. Each window has its own blending attributes.

- CRKEY Field: enables the chroma keying and match logic.
- INV Field: performs bit inversion at pixel level.
- ITER2BL Field: when set the iterated data path is selected.
- ITER Field.
- REVALPHA Field: uses the reverse alpha value.
- GAEN Field: enables the global alpha value in the data path.
- LAEN Field: enables the local alpha value from the pixel.
- OVR Field: when set the overlay is selected as an input of the blender.
- DMA Field: the DMA data path is activated.
- REP Field: enables the bit replication to fill the 24-bit internal data path.
- DSTKEY Field: when set, Destination keying is enabled.
- GA Field: defines the global alpha value.

32.6.3.5 Overlay Attributes Software Operation

1. When required, write the overlay attributes configuration registers.
2. Set UPDATEEN field of the CHXCHER register.
3. Poll UPDATESR field in the CHXCHSR, the update applies when that field is reset.

32.6.4 RGB Frame Buffer Memory Bitmap

32.6.4.1 1 bpp Through Color Lookup Table

Table 32-9. 1 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 1 bpp	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0

32.6.4.2 2 bpp Through Color Lookup Table

Table 32-10. 2 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 2 bpp	p15		p14		p13		p12		p11		p10		p9		p8		p7		p6		p5		p4		p3		p2		p1		p0	

32.6.4.3 4 bpp Through Color Lookup Table

Table 32-11. 4 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 4 bpp	p7				p6				p5				p4				p3				p2				p1				p0			

32.6.4.4 8 bpp Through Color Lookup Table

Table 32-12. 8 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 8 bpp	p3								p2								p1								p0							

32.6.4.5 12 bpp Memory Mapping, RGB 4:4:4

Table 32-13. 12 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	–				R1[3:0]				G1[3:0]				B1[3:0]				–				R0[3:0]				G0[3:0]				B0[3:0]			

32.6.4.6 16 bpp Memory Mapping with Alpha Channel, ARGB 4:4:4:4

Table 32-14. 16 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	A1[3:0]				R1[3:0]				G1[3:0]				B1[3:0]				A0[3:0]				R0[3:0]				G0[3:0]				B0[3:0]			

32.6.4.7 16 bpp Memory Mapping with Alpha Channel, RGBA 4:4:4:4

Table 32-15. 16 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	R1[3:0]				G13:0]				B1[3:0]				A1[3:0]				R0[3:0]				G0[3:0]				B0[3:0]				A0[3:0]			

32.6.4.8 16 bpp Memory Mapping with Alpha Channel, RGB 5:6:5

Table 32-16. 16 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16bpp	R1[4:0]								G1[5:0]								B1[4:0]								R0[4:0]							
	G0[5:0]								B0[4:0]																							

32.6.4.9 16 bpp Memory Mapping with Transparency Bit, ARGB 1:5:5:5

Table 32-17. 16 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 4 bpp	A1	R1[4:0]							G1[4:0]								B1[4:0]								A0	R0[4:0]						
	G0[4:0]								B0[4:0]																							

32.6.4.10 18 bpp Unpacked Memory Mapping with Transparency Bit, RGB 6:6:6

Table 32-18. 18 bpp Unpacked Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp															R0[5:0]						G0[5:0]						B0[5:0]					

32.6.4.11 18 bpp Packed Memory Mapping with Transparency Bit, RGB 6:6:6

Table 32-19. 18 bpp Packed Memory Mapping, Little Endian Organization at Address 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp	G1[1:0]		B1[5:0]												R0[5:0]						G0[5:0]						B0[5:0]					

Table 32-20. 18 bpp Packed Memory Mapping, Little Endian Organization at Address 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp	R2[3:0]				G2[5:0]				B2[5:0]														R1[5:2]				G1[5:2]					

Table 32-21. 18 bpp Packed Memory Mapping, Little Endian Organization at Address 0x8, 0x9, 0xA, 0xB

Mem addr	0xB								0xA								0x9								0x8							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 18 bpp	G4[1:0]		B4[5:0]										R3[5:0]				G3[5:0]								B3[3:0]						R2[5:4]	

32.6.4.12 19 bpp Unpacked Memory Mapping with Transparency Bit, RGB 1:6:6:6

Table 32-22. 19 bpp Unpacked Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp														A0	R0[5:0]						G0[5:0]								B0[5:0]			

32.6.4.13 19 bpp Packed Memory Mapping with Transparency Bit, ARGB 1:6:6:6

Table 32-23. 19 bpp Packed Memory Mapping, Little Endian Organization at Address 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp	G1[1:0]		B1[5:0]											A0	R0[5:0]						G0[5:0]								B0[5:0]			

Table 32-24. 19 bpp Packed Memory Mapping, Little Endian Organization at Address 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp	R2[3:0]				G2[5:0]						B2[5:0]											A1	R1[5:2]						G1[5:2]			

Table 32-25. 18 bpp Packed Memory Mapping, Little Endian Organization at Address 0x8, 0x9, 0xA, 0xB

Mem addr	0xB								0xA								0x9								0x8							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 19 bpp	G4[1:0]		B4[5:0]									A3	R3[5:0]				G3[5:0]								B3[3:0]						R2[5:4]	

32.6.4.14 24 bpp Unpacked Memory Mapping, RGB 8:8:8

Table 32-26. 24 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 24 bpp									R0[7:0]								G0[7:0]								B0[7:0]							

32.6.4.15 24 bpp Packed Memory Mapping, RGB 8:8:8

Table 32-27. 24 bpp Packed Memory Mapping, Little Endian Organization at Address 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 24 bpp	B1[7:0]								R0[7:0]								G0[7:0]								B0[7:0]							

Table 32-28. 24 bpp Packed Memory Mapping, Little Endian Organization at Address 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 24 bpp	G2[7:0]								B2[7:0]								R1[7:0]								G1[7:0]							

32.6.4.16 25 bpp Memory Mapping, ARGB 1:8:8:8

Table 32-29. 25 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 25 bpp								A0	R0[7:0]								G0[7:0]								B0[7:0]							

32.6.4.17 32 bpp Memory Mapping, ARGB 8:8:8:8

Table 32-30. 32 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 32 bpp	A0[7:0]								R0[7:0]								G0[7:0]								B0[7:0]							

32.6.4.18 32 bpp Memory Mapping, RGBA 8:8:8:8

Table 32-31. 32 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 32 bpp	R0[7:0]								G0[7:0]								B0[7:0]								A0[7:0]							

32.6.5 YUV Frame Buffer Memory Mapping

32.6.5.1 AYCbCr 4:4:4 Interleaved Frame Buffer Memory Mapping

Table 32-32. 32 bpp Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	A0[7:0]								Y0[7:0]								Cb0[7:0]								Cr0[7:0]							

32.6.5.2 4:2:2 Interleaved Mode Frame Buffer Memory Mapping

Table 32-33. 16 bpp 4:2:2 interleaved Mode 0

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Cr0[7:0]								Y1[7:0]								Cb0[7:0]								Y0[7:0]							

Table 32-34. 16 bpp 4:2:2 interleaved Mode 1

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Y1[7:0]								Cr0[7:0]								Y0[7:0]								Cb0[7:0]							

Table 32-35. 16 bpp 4:2:2 interleaved Mode 2

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Cb0[7:0]								Y1[7:0]								Cr0[7:0]								Y0[7:0]							

Table 32-36. 16 bpp 4:2:2 interleaved Mode 3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Y1[7:0]								Cb0[7:0]								Y0[7:0]								Cr0[7:0]							

32.6.5.3 4:2:2 Semiplanar Mode Frame Buffer Memory Mapping

Table 32-37. 4:2:2 Semiplanar Luminance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

Table 32-38. 4:2:2 Semiplanar Chrominance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Cb2[7:0]								Cr2[7:0]								Cb0[7:0]								Cr0[7:0]							

32.6.5.4 4:2:2 Planar Mode Frame Buffer Memory Mapping

Table 32-39. 4:2:2 Planar Mode Luminance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

Table 32-40. 4:2:2 Planar Mode Chrominance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 16 bpp	C3[7:0]								C2[7:0]								C1[7:0]								C0[7:0]							

32.6.5.5 4:2:0 Planar Mode Frame Buffer Memory Mapping

In Planar Mode, the three video components Y, Cr and Cb are split into 3 memory areas and stored in a raster-scan order. These three memory planes are contiguous and always aligned on a 32-bit boundary.

Table 32-41. 4:2:0 Planar Mode Luminance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

Table 32-42. 4:2:0 Planar Mode Luminance Memory Mapping, Little Endian Organization for Byte 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	Y7[7:0]								Y6[7:0]								Y5[7:0]								Y4[7:0]							

Table 32-43. 4:2:0 Planar Mode Chrominance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	C3[7:0]								C2[7:0]								C1[7:0]								C0[7:0]							

Table 32-44. 4:2:0 Planar Mode Chrominance Memory Mapping, Little Endian Organization for Byte 0x4, 0x5, 0x6, 0x7

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	C7[7:0]								C6:[7:0]								C5[7:0]								C4[7:0]							

32.6.5.6 4:2:0 Semiplanar Frame Buffer Memory Mapping

Table 32-45. 4:2:0 Semiplanar Mode Luminance Memory Mapping, Little Endian Organization

Mem addr	0x7								0x6								0x5								0x4							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

Table 32-46. 4:2:0 Semiplanar Mode Chrominance Memory Mapping, Little Endian Organization

Mem addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 12 bpp	Cb1[7:0]								Cr1[7:0]								Cb0[7:0]								Cr0[7:0]							

32.6.6 Chrominance Upsampling Unit

Both 4:2:2 and 4:2:0 input formats are supported by the LCD module. In 4:2:2, the two chrominance components are sampled at half the sample rate of the luminance. The horizontal chrominance resolution is halved. When this input format is selected, the chrominance upsampling unit uses two chrominances to interpolate the missing component.

In 4:2:0, Cr and Cb components are subsampled at a factor of two vertically and horizontally. When this input mode is selected, the chrominance upsampling unit uses two and four chroma components to generate the missing horizontal and vertical components.

Figure 32-2. 4:2:2 Upsampling Algorithm

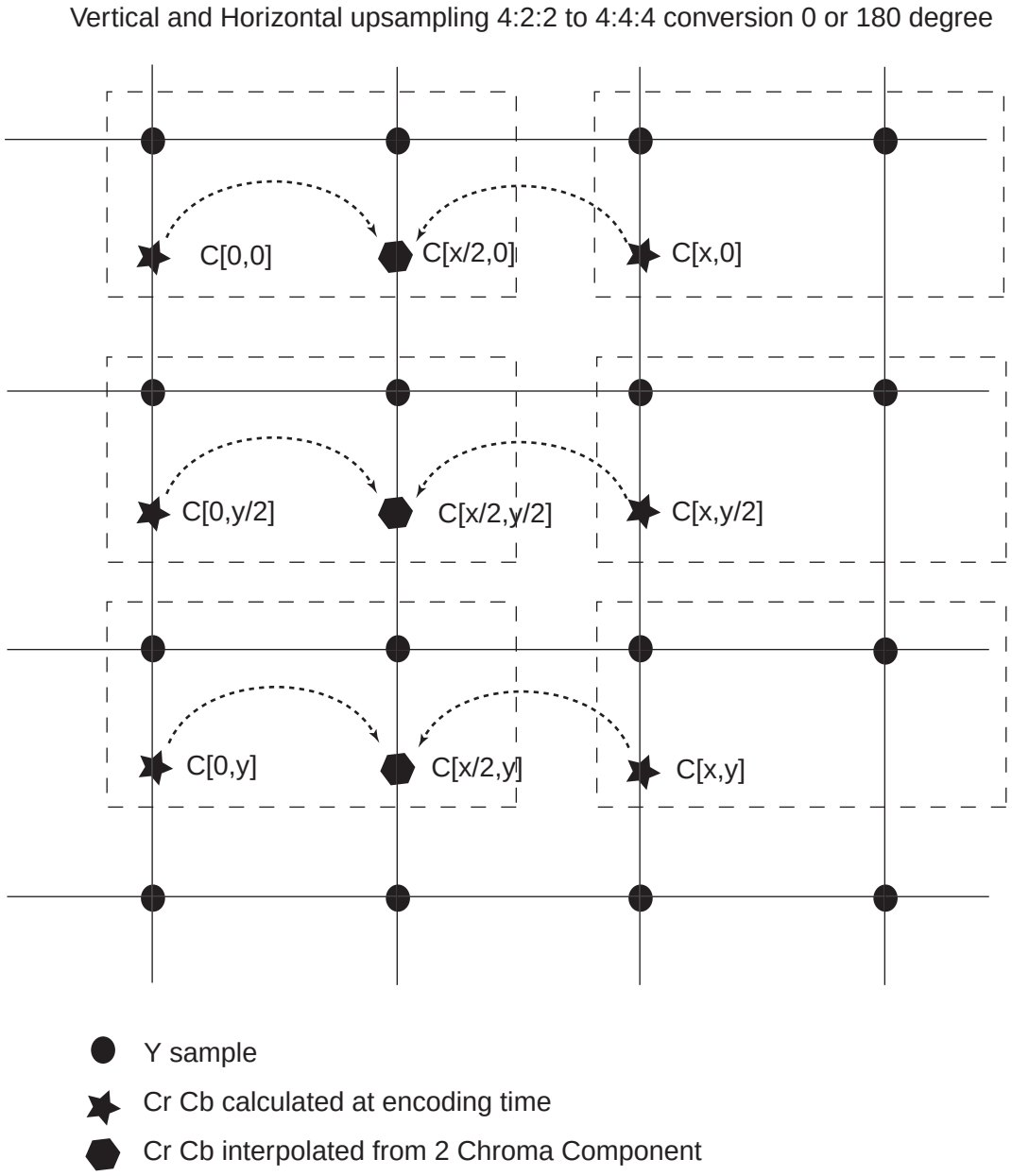


Figure 32-3. 4:2:2 Packed Upsampling Algorithm

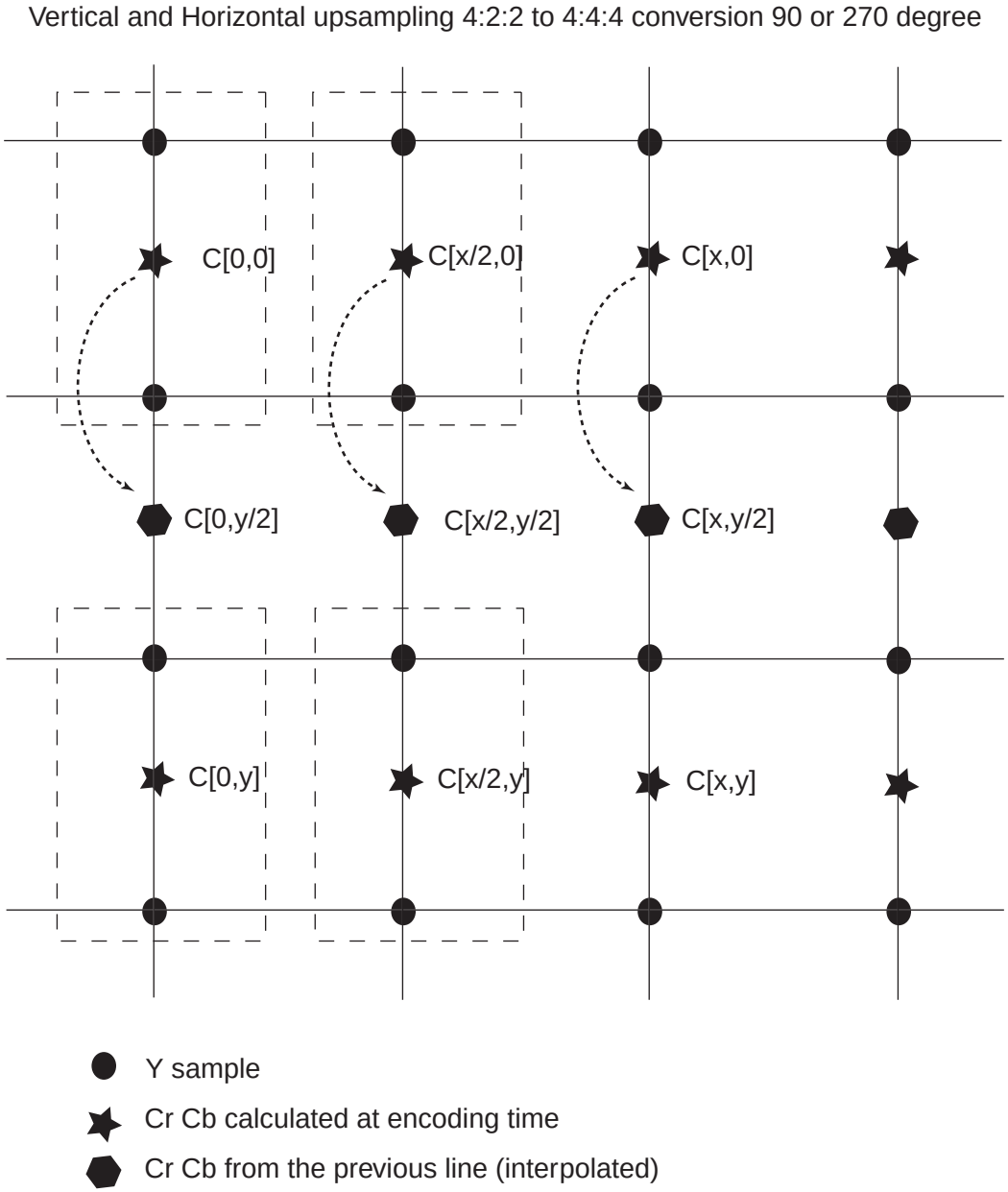


Figure 32-4. 4:2:2 Semiplanar and Planar Upsampling Algorithm - 90 or 270 Degree Rotation Activated

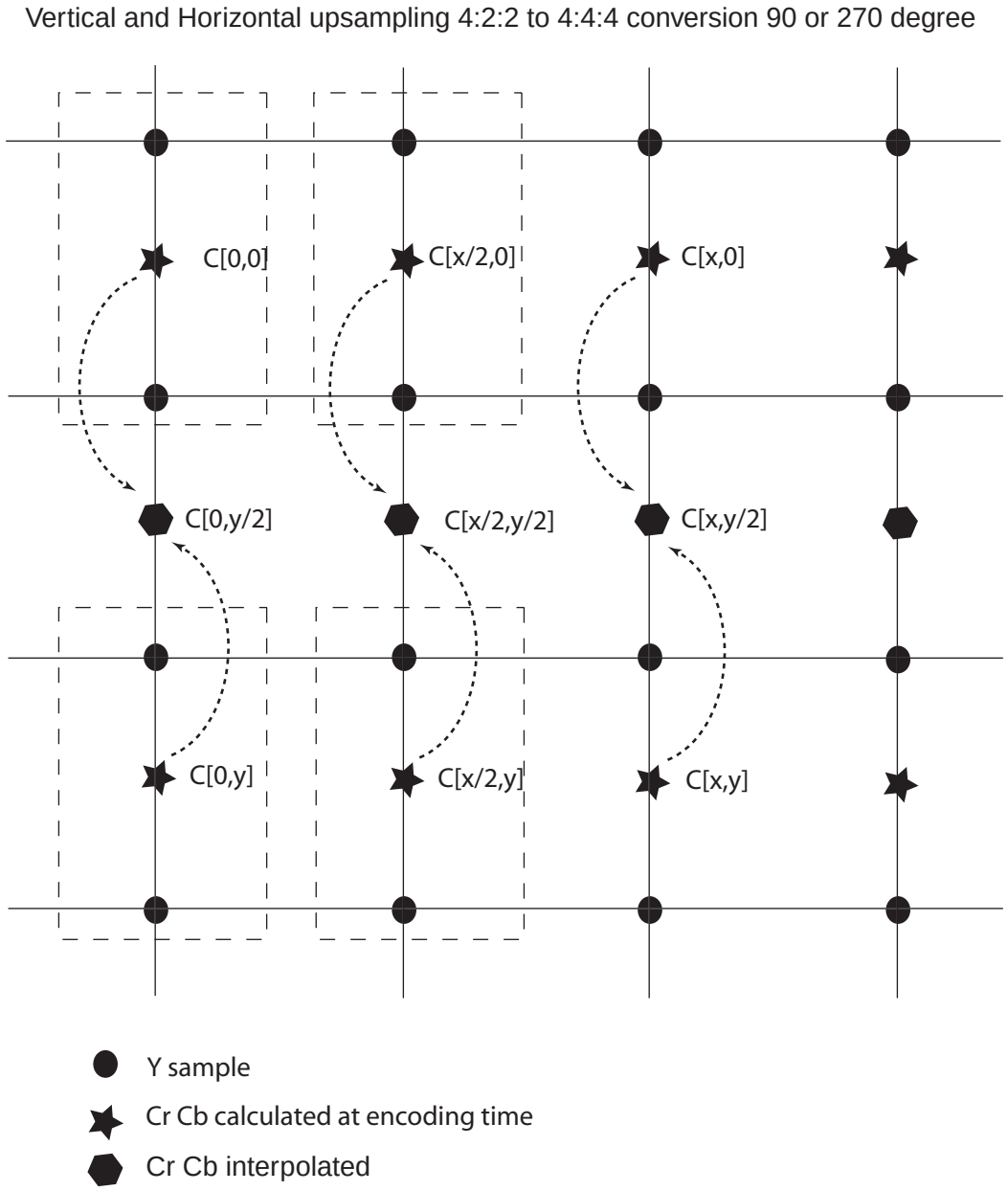
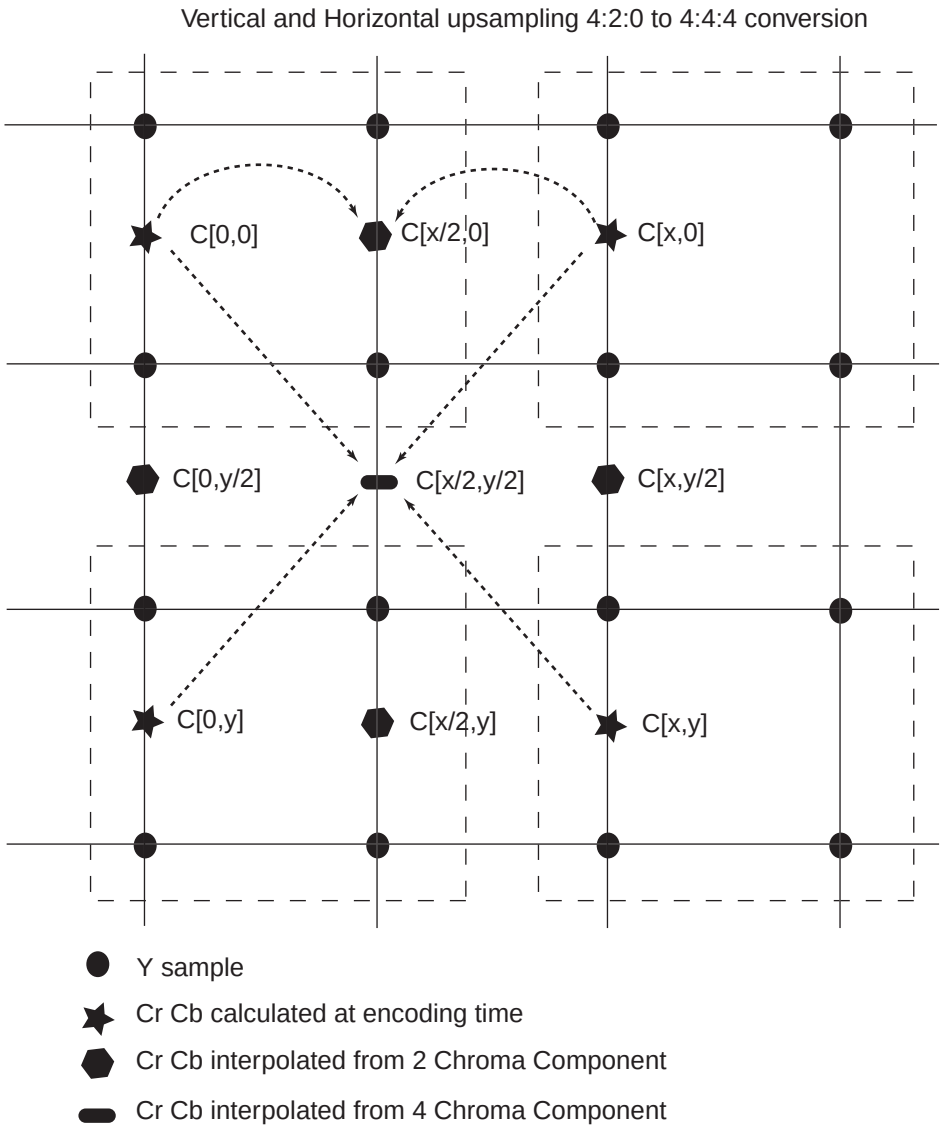


Figure 32-5. 4:2:0 Upsampling Algorithm



$$Chroma\left[\frac{x}{2}, 0\right] = \frac{Cr[0,0] + Cr[0,x]}{2}$$

$$Chroma\left[0, \frac{y}{2}\right] = \frac{Cr[0,0] + C[0,y]}{2}$$

$$Chroma\left[\frac{x}{2}, \frac{y}{2}\right] = \frac{Cr[0,0] + Cr[x,0] + Cr[y,0] + Cr[x,y]}{4}$$

$$Chroma\left[x, \frac{y}{2}\right] = \frac{Cr[x,0] + Cr[x,y]}{2}$$

$$Chroma\left[\frac{x}{2}, y\right] = \frac{Cr[0,y] + Cr[x,y]}{2}$$

32.6.6.1 Chrominance Upsampling Algorithm

1. Read line n from chrominance cache and interpolate [x/2,0] chrominance component filling the 1 x 2 kernel with line n. If the chrominance cache is empty, then fetch the first line from external memory and interpolate from the external memory. Duplicate the last chrominance at the end of line.
2. Fetch line n+1 from external memory, write line n + 1 to chrominance cache, read line n from the chrominance cache. interpolate [0,y/2], [x/2,y/2] and [x, y/2] filling the 2x2 kernel with line n and n+1. Duplicate the last chrominance line to generate the last interpolated line.
3. Repeat step 1 and step 2.

32.6.7 Line and Pixel Striding

The LCD module includes a mechanism to increment the memory address by a programmable amount when the end of line has been reached, this offset is referred to as XSTRIDE and is defined on a per overlay basis. It also contains a PSTRIDE field that allows a programmable jump at the pixel level. Pixel stride is the value from one pixel to the next.

32.6.7.1 Line Striding

When the end of line has been reached, the DMA address counter points to the next pixel address. The channel DMA address register is added to the XSTRIDE field, and then updated. If XSTRIDE is set to zero, the DMA address register remains unchanged. The XSTRIDE field of the channel configuration register is aligned to the pixel size boundary. The XSTRIDE field is a two's complement number. The following formula applies at the line boundary and indicates how the DMA controller computes the next pixel address. The function Sizeof() returns the number of bytes required to store a pixel.

$$NextPixelAddress = CurrentPixelAddress + Sizeof(pixel) + XSTRIDE$$

32.6.7.2 Pixel Striding

The DMA channel engine may optionally fetch non contiguous pixels. The channel DMA address register is added to the PSTRIDE field and then updated. If PSTRIDE is set to zero, the DMA address register remains unchanged and pixels are contiguous. The PSTRIDE field of the channel configuration register is aligned to the pixel size boundary. The PSTRIDE is a two's complement number. The following formula applies at the pixel boundary and indicates how the DMA controller computes the next pixel address. The function Sizeof() returns the number of bytes required to store a pixel.

$$NextPixelAddress = CurrentPixelAddress + Sizeof(pixel) + PSTRIDE$$

32.6.8 Color Space Conversion Unit

The color space conversion unit converts Luminance Chrominance color space into the Red Green Blue color space. The conversion matrix is defined below and is fully programmable through the LCD user interface

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} CSCRY & CSCRU & CSCRV \\ CSCGY & CSCGU & CSCGV \\ CSCBY & CSCBU & CSCBV \end{bmatrix} \bullet \begin{bmatrix} Y - Yoff \\ Cb - Cboff \\ Cr - Croff \end{bmatrix}$$

Color space conversion coefficients are defined with the following equation:

$$CSC_{ij} = \frac{1}{2^7} \cdot \left[-2^9 \cdot c_9 + \sum_{n=0}^{8} c_n \cdot 2^n \right]$$

Color space conversion coefficients are defined with one sign bit, 2 integer bits and 7 fractional bits. The range of the CSCij coefficients is defined below with a step of 1/128.

$$-4 \leq CSC_{ij} \leq 3{,}9921875$$

Additionally a set scaling factor {Yoff, Cboff, Croff} can be applied.

32.6.9 Two Dimension Scaler

The High End Overlay (HEO) data path includes a hardware scaler that allows an image resize in both horizontal and vertical directions.

32.6.9.1 Video Scaler Description

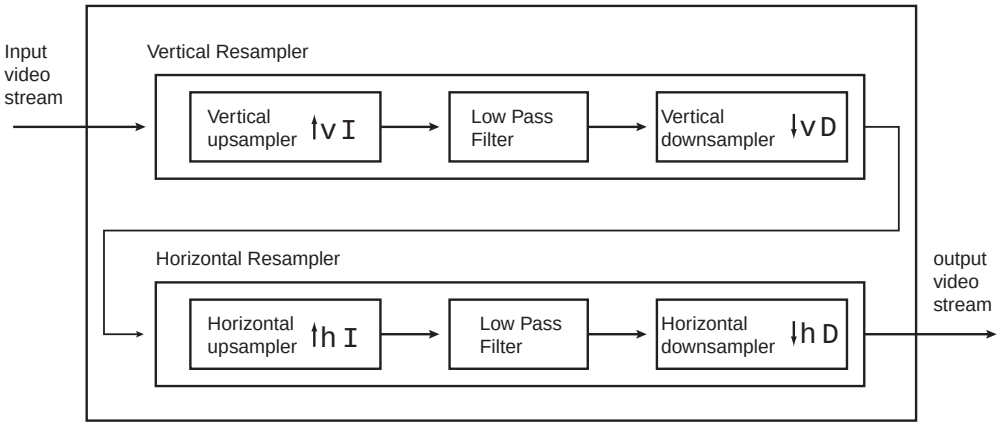
The scaling operation is based on a vertical and horizontal resampling algorithm. The sampling rate of the original image is increased when the video is upscaled, and decreased when the video is downscaled. A Vertical resampler is used to perform a vertical interpolation by a factor of *vI,* and a decimation by a factor of *vD*. A Horizontal resampler is used to perform a vertical interpolation by a factor of *hI,* and a decimation by a factor of *hD.* Both horizontal and vertical low pass filters are designed to minimize the aliasing effect. The frequency response of the low pass filter has the following characteristics:

$$H(\omega) = \begin{cases} I & \text{when } 0 \leq |\omega| \leq min(\frac{\pi}{I}, \frac{\pi}{D}) \\ 0 & \text{otherwise} \end{cases}$$

Taking into account the linear phase condition and anticipating the filter length M, the desired frequency response is modified.

$$H(\omega) = \begin{cases} Ie^{-j\omega\frac{M}{2}} & \text{when } 0 \leq |\omega| \leq min(\frac{\pi}{I}, \frac{\pi}{D}) \\ 0 & \text{otherwise} \end{cases}$$

Figure 32-6. Video Resampler Architecture



The impulse response of the low pass filter defined is:

$$h(n) = \begin{cases} I \times \frac{\omega_c}{\pi} \text{ when } n = 0 \\ I \times \frac{\omega_c}{\pi} \times \frac{\sin(\omega_c n)}{\omega_c n} \text{otherwise} \end{cases}$$

Or, for the filter of length M:

$$h(n) = \begin{cases} I \times \frac{\omega_c}{\pi} \text{ when } n = \frac{M}{2} \\ I \times \frac{\omega_c}{\pi} \times \frac{\sin\left(\omega_c\left(n - \frac{M}{2}\right)\right)}{\omega_c\left(n - \frac{M}{2}\right)} \text{otherwise} \end{cases}$$

This ideal filter is non-causal and cannot be realized. The unit sample response *h(n)* is infinite in duration and must be truncated depending on the expected length *M* of the filter. This truncation is equivalent to the multiplication of the impulse response by a window function *w(n)*.

Table 32-47. Window Function for a Filter Length M

Name of the Window Function	Time Domain Sequence *w(n)*
Barlett	$1 - \frac{2 \times \left\lvert n - \frac{M-1}{2} \right\rvert}{M-1}$
Blackman	$0{,}42 - 0{,}5 \times \cos\frac{2\pi n}{M-1} + 0{,}08 \times \cos\frac{4\pi n}{M-1}$
Hamming	$0{,}54 - 0{,}46 \times \cos\frac{2\pi n}{M-1}$
Hanning	$0{,}5 - 0{,}5 \times \cos\frac{2\pi n}{M-1}$

The horizontal resampler includes an 8-phase 5-tap filter equivalent to a 40-tap FIR described in Figure 32-8.

The vertical resampler includes an 8-phase 3-tap filter equivalent to a 24-tap FIR described in Figure 32-8.

Figure 32-7. Horizontal Resampler Filter Architecture

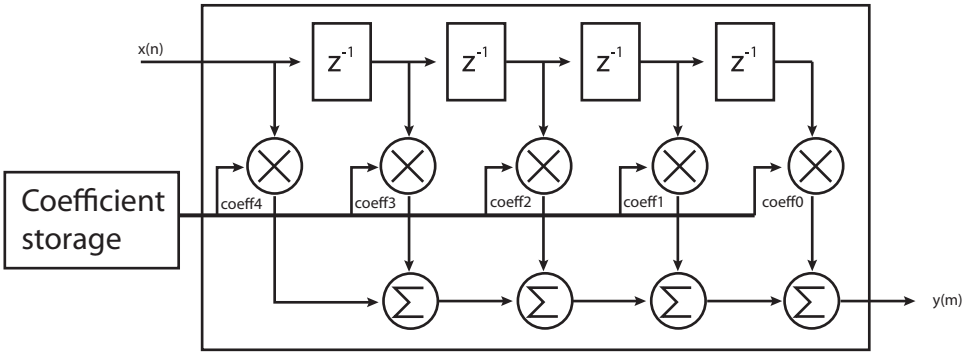
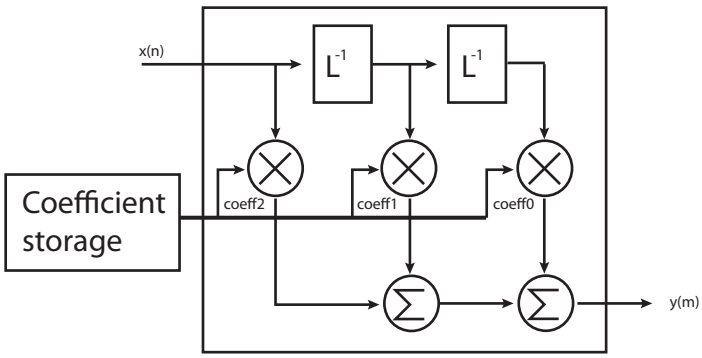


Figure 32-8. Vertical Resampler Filter Architecture



32.6.9.2 Horizontal Scaler

The XMEMSIZE field of the LCDC_HEOCFG4 register indicates the horizontal size minus one of the image in the system memory. The XSIZE field of the LCDC_HEOCFG3 register contains the horizontal size minus one of the window.The SCALEN field of the LCDC_HEOCFG13 register is set to one. The scaling factor is programmed in the XFACTOR field of the LCDC_HEOCFG13 register. Use the following algorithm to find the XFACTOR value.

$$XFACTOR_{1st} = floor\left(\frac{8 \times 256 \times XMEMSIZE - 256 \times XPHIDEF}{XSIZE}\right)$$

$$XFACTOR_{1st} = XFACTOR_{1st} + 1$$

$$XMEMSIZE_{max} = floor\left(\frac{XFACTOR_{1st} \times XSIZE + 256 \times XPHIDEF}{2048}\right)$$

$$XFACTOR = XFACTOR_{1st} - 1 \qquad when(XMEMSIZE_{max} > XMEMSIZE)$$

$$XFACTOR = XFACTOR_{1st} \qquad otherwise$$

32.6.9.3 Vertical Scaler

The YMEMSIZE field of the LCDC_HEOCFG4 register indicates the vertical size minus one of the image in the system memory. The YSIZE field of the LCDC_HEOCFG3 register contains the vertical size minus one of the window. The SCALEN field of the LCDC_HEOCFG13 register is set to one. The scaling factor is programmed in the YFACTOR field of the LCDC_HEOCFG13 register.

$$YFACTOR_{1st} = floor\left(\frac{8 \times 256 \times YMEMSIZE - 256 \times YPHIDEF}{YSIZE}\right)$$

$$YFACTOR_{1st} = YFACTOR_{1st} + 1$$

$$YMEMSIZE_{max} = floor\left(\frac{YFACTOR_{1st} \times YSIZE + 256 \times YPHIDEF}{2048}\right)$$

$$YFACTOR = YFACTOR_{1st} - 1 \qquad when(YMEMSIZE_{max} > YMEMSIZE)$$
$$YFACTOR = YFACTOR_{1st} \qquad otherwise$$

32.6.10 Hardware Cursor

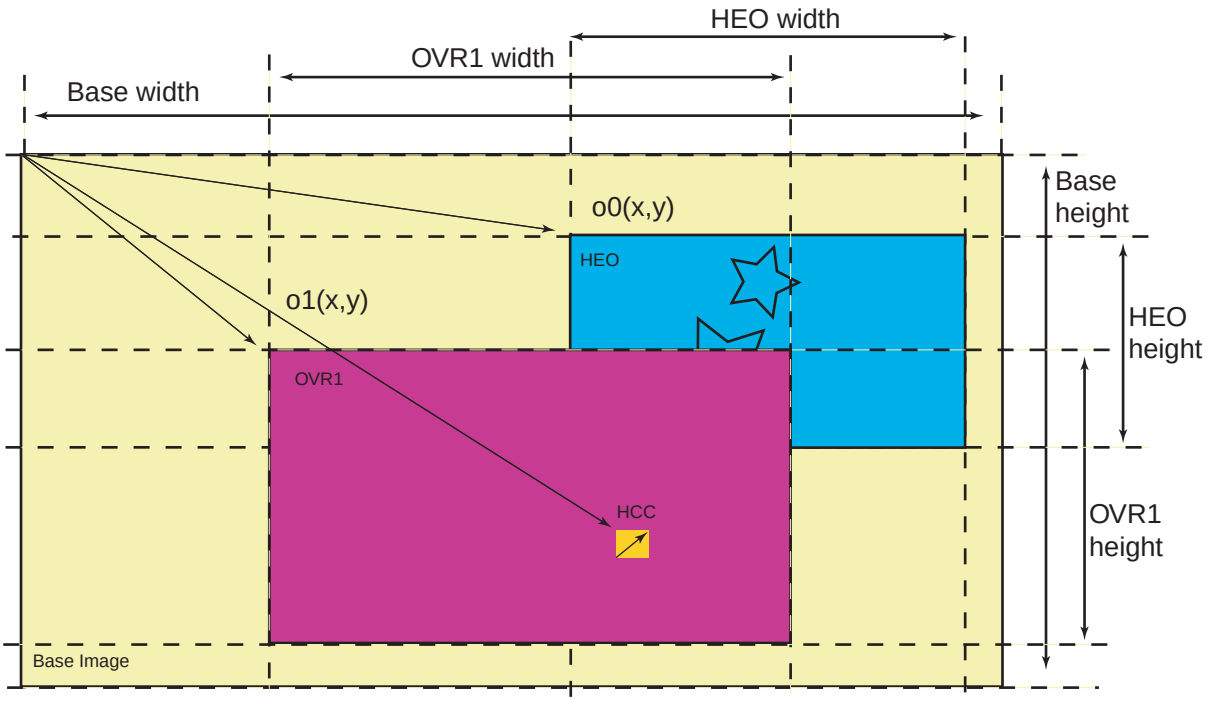
The LCD module integrates a hardware cursor database. This layer features only a minimal set of color among 1, 2, 4 and 8 bpp palletized and 16 bpp to 32 bpp true color. The cursor size is limited to 128 x 128 pixels.

32.6.11 Color Combine Unit

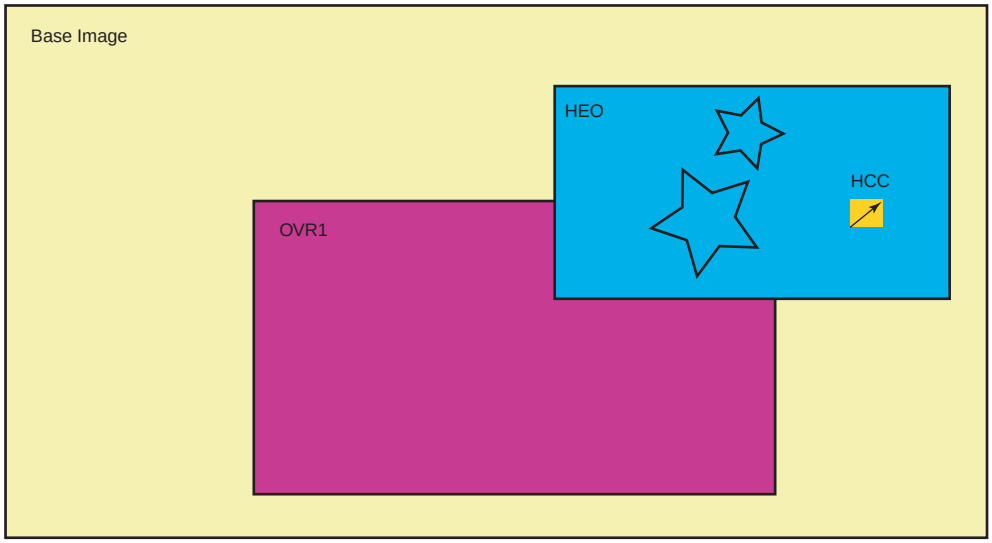
32.6.11.1 Window Overlay

The LCD module provides hardware support for multiple “overlay plane” that can be used to display windows on top of the image without destroying the image located below. The overlay image can use any color depth. Using the overlay alleviates the need to re-render the occluded portion of the image. When pixels are combined together through the alpha blending unit, a new color is created. This new pixel is called an iterated pixel and is passed to the next blending stage. Then, this pixel may be combined again with another pixel. The VIDPRI field located in the LCDC_HEOCFG12 register configures the video priority algorithm used to display the layers. When VIDPRI field is set to zero, the OVR1 layer is located above the HEO layer. When VIDPRI field is set to one, OVR1 is located below the HEO layer.

Figure 32-9. Overlay Example with Two Different Video Prioritization Algorithms



Video Prioritization Algorithm 1 : HCC > OVR1 > HEO > BASE



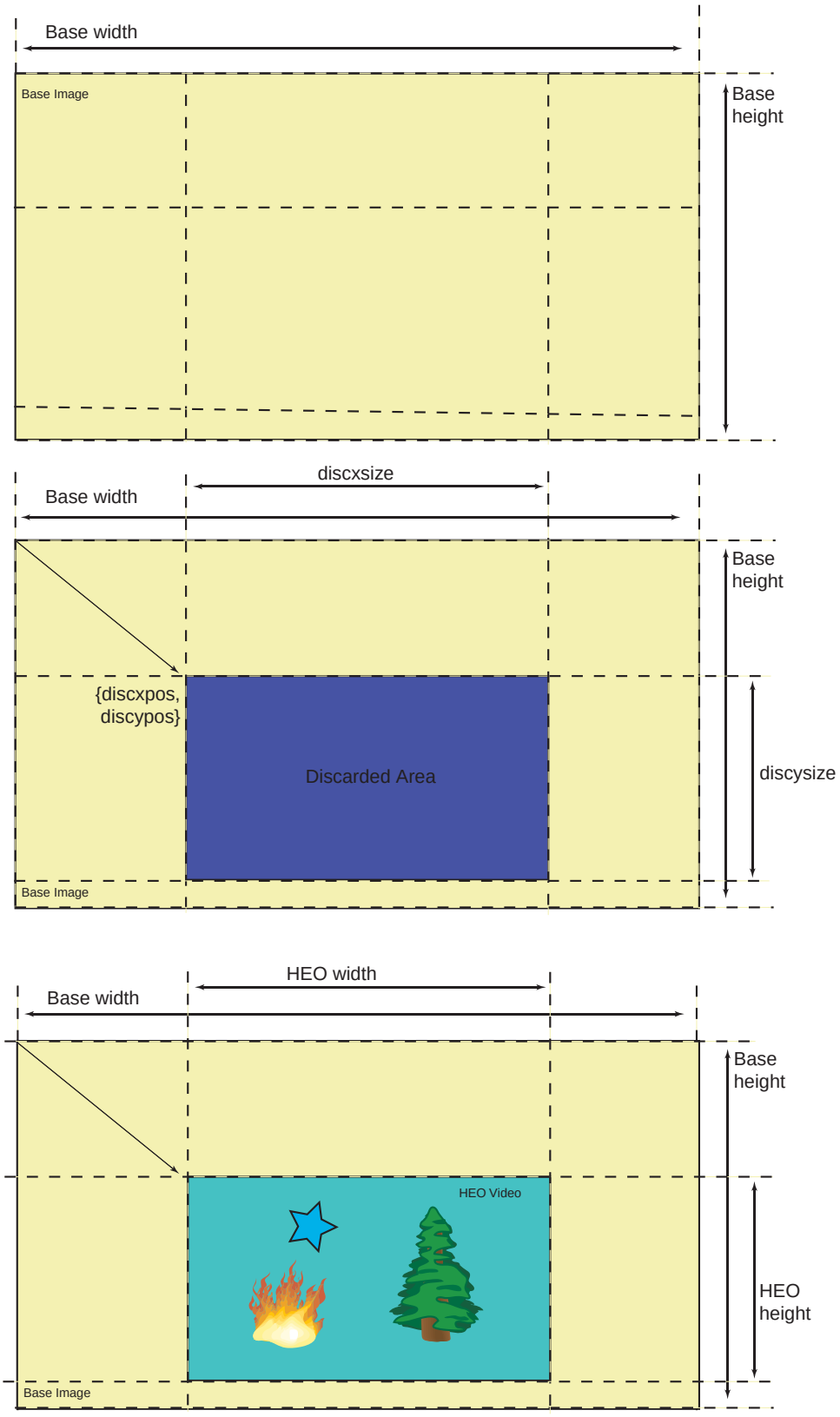
Video Prioritization Algorithm 2 : HCC > HEO > OVR1 > BASE

32.6.11.2 Base Layer, with Window Overlay Optimization

When the base layer is combined with at least one active overlay, the whole base layer frame is retrieved from the memory though it is not visible. A set of registers is used to disable the Base DMA when this condition is met. These registers are listed below:

- LCDC_CFG5: DISCXPOS field discard area horizontal position
- LCDC_CFG5: DISCYPOS field discard area vertical position
- LCDC_CFG6: DISCXSIZE field discard area horizontal size
- LCDC_CFG6: DISCYSIZE field discard area vertical size
- LCDC_CFG4: DISCEN field discard area enable

Figure 32-10.Base Layer Discard Area



32.6.11.3 Overlay Blending

The blending function requires two pixels (one iterated from the previous blending stage and one from the current overlay color) and a set of blending configuration parameters. These parameters define the color operation.

Figure 32-11.Alpha Blender Function

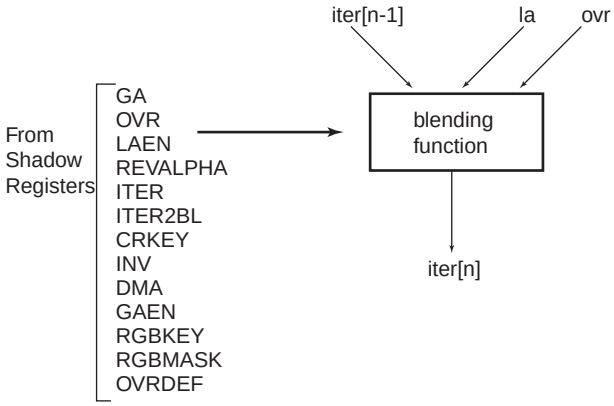
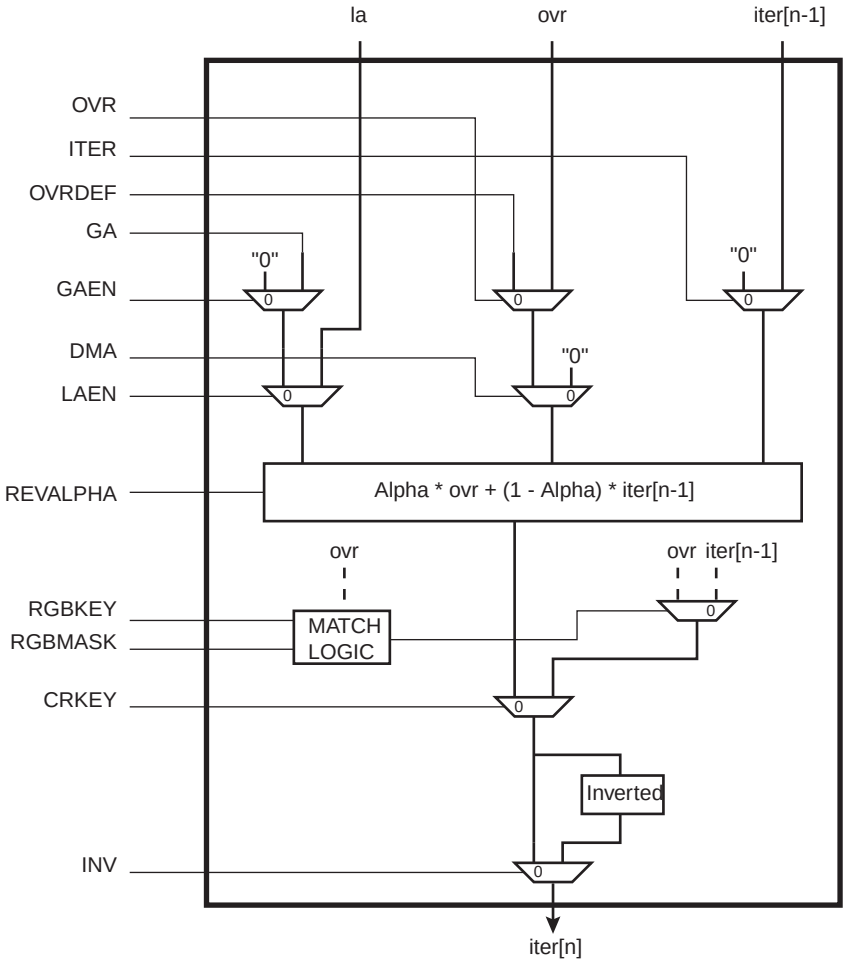
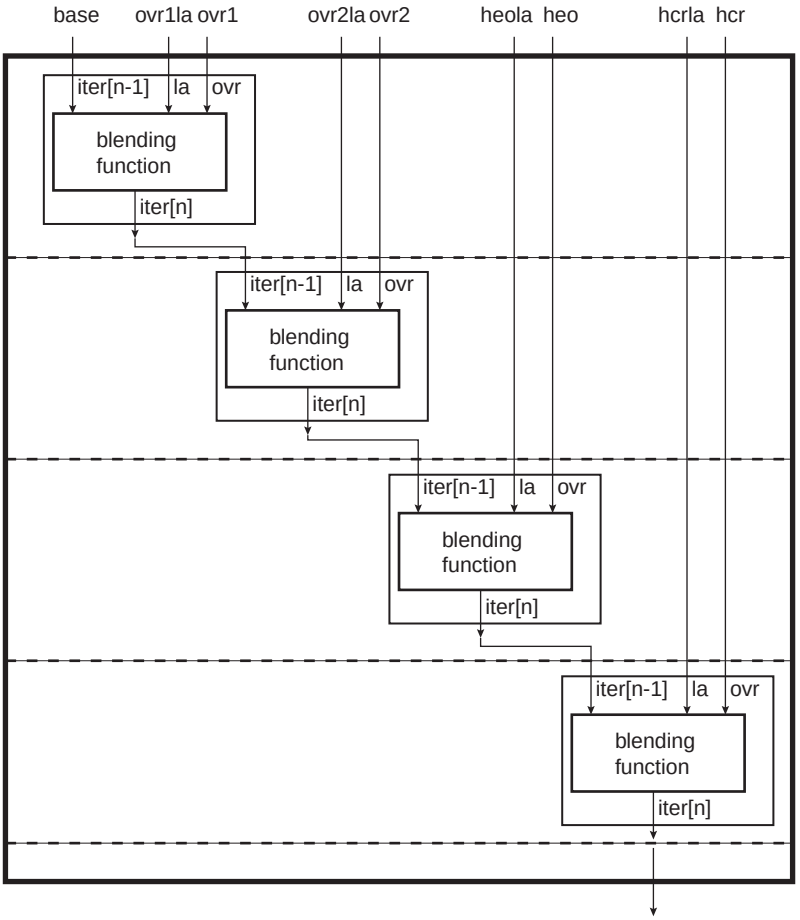


Figure 32-12.Alpha Blender Database



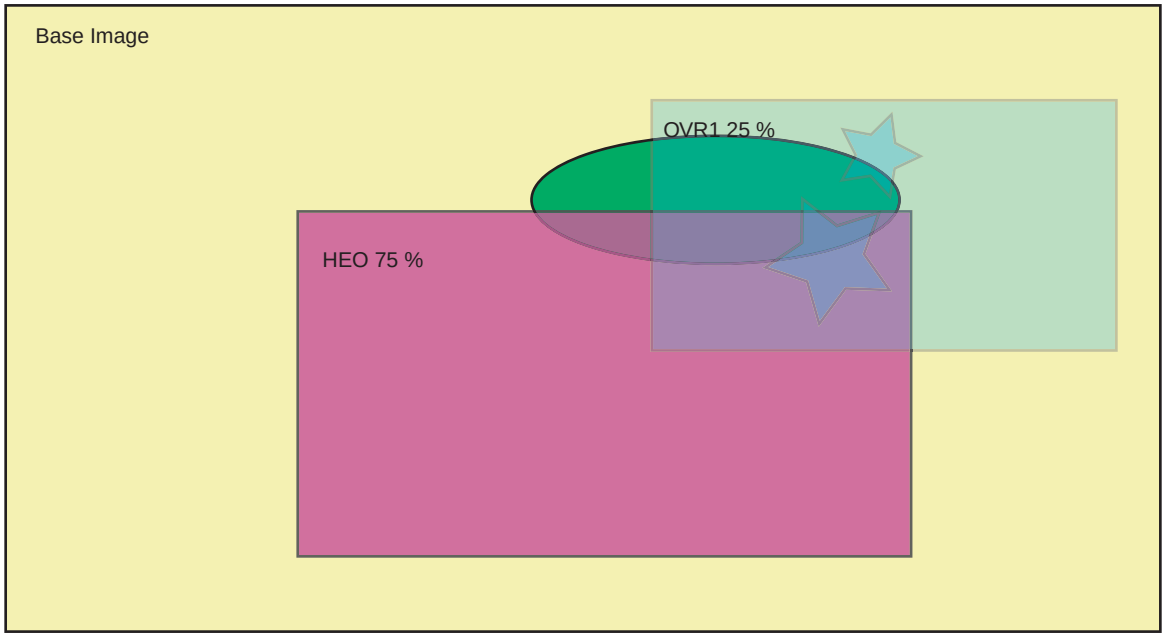
32.6.11.4 Global Alpha Blender

Figure 32-13.Global Alpha Blender



32.6.11.5 Window Blending

Figure 32-14.256-level Alpha Blending



Video Prioritization Algorithm 1 : OVR1 > HEO > BASE

32.6.11.6 Color Keying

Color keying involves a method of bit-block image transfer (Blit). This entails blitting one image onto another where not all the pixels are copied. Blitting usually involves two bitmaps, a source bitmap and a destination bitmap. A raster operation (ROP) is performed to define whether the iterated color or the overlay color is to be visible or not.

Source Color Keying

If the masked overlay color matches the color key then the iterated color is selected, Source Color Keying is activated using the following configuration.

- Select the Overlay to Blit
- Set DSTKEY field to zero
- Activate Color Keying setting CRKEY field to 1
- Program Color Key writing RKEY, GKEY and BKEY fields
- Program Color Mask writing RKEY, GKEY and BKEY fields

When the Mask register is set to zero, the comparison is disabled and the raster operation is activated.

Destination Color Keying

If the iterated masked color matches the color key then the overlay color is selected, Destination Color Keying is activated using the following configuration:

- Select the Overlay to Blit
- Set DSTKEY field to one
- Activate Color Keying setting CRKEY field to 1
- Program Color Key writing RKEY, GKEY and BKEY fields
- Program Color Mask writing RKEY, GKEY and BKEY fields

When the Mask register is set to zero, the comparison is disabled and the raster operation is activated.

32.6.12 LCDC PWM Controller

This block generates the LCD contrast control signal (LCD_PWM) to make possible the control of the display's contrast by software. This is an 8-bit PWM (Pulse Width Modulation) signal that can be converted to an analog voltage with a simple passive filter.

The PWM module has a free-running counter whose value is compared against a compare register (PWMCVAL field of the LCDC_LCDCFG6 register). If the value in the counter is less than that in the register, the output brings the value of the polarity (PWMPOL field) bit in the PWM control register: LCDC_LCDCFG6. Otherwise, the opposite value is output. Thus, a periodic waveform with a pulse width proportional to the value in the compare register is generated.

Due to the comparison mechanism, the output pulse has a width between zero and 255 PWM counter cycles. Thus by adding a simple passive filter outside the chip, an analog voltage between 0 and (255/256) × VDD can be obtained (for the positive polarity case, or between (1/256) × VDD and VDD for the negative polarity case). Other voltage values can be obtained by adding active external circuitry.

For PWM mode, the counter frequency can be adjusted to four different values using the PWMPS field of the LCDC_LCDCFG6 register.

The PWM module can be fed with the slow clock or the system clock, depending on the CLKPWMSEL field of the LCDC_CFG0 register.

32.6.13 Post Processing Controller

The output stream of pixels can be either displayed on the screen or written to the memory using the Post Processing Controller (PPC). When the PPC is used, the screen display is disabled, but synchronization signals remain active (if enabled). The stream of pixel can be written in RGB mode or encoded in YCbCr 422 mode. A programmable color space conversion stage is available.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} CSCYR & CSCYG & CSCYB \\ CSCUR & CSCUG & CSCUB \\ CSCVR & CSCVG & CSCUB \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Yoff \\ Uoff \\ Voff \end{bmatrix}$$

32.6.14 LCD Overall Performance

32.6.14.1 Color Lookup Table (CLUT)

Table 32-48. CLUT Pixel Performance

CLUT MODE	Pixels/Cycle	ROTATION	SCALING
1 bpp	64	Not supported	Supported
2 bpp	32	Not supported	Supported
3 bpp	16	Not supported	Supported
4 bpp	8	Not supported	Supported

32.6.14.2 RGB Mode Fetch Performance

Table 32-49. RGB Mode Performance

RGB Mode	Pixels/Cycle Memory Burst Mode	Rotation Peak Random Memory Access (pixels/cycle)		SCALING Burst Mode or Rotation Optimization Available
		Rotation Optimization[1]	**Normal Mode**	
12 bpp	4	1	0.2	Supported
16 bpp	4	1	0.2	Supported
18 bpp	2	1	0.2	Supported
18 bpp RGB PACKED	2.666	Not supported	0.2	Supported
19 bpp	2	1	0.2	Supported
19 bpp PACKED	2.666	Not Supported	0.2	Supported
24 bpp	2	1	0.2	Supported
24 bpp PACKED	2.666	Not Supported	0.2	Supported
25 bpp	2	1	0.2	Supported
32 bpp	2	1	0.2	Supported

Note: 1. Rotation optimization = AHB lock asserted on consecutive single access.

32.6.14.3 YUV Mode Fetch Performance

Table 32-50. Single Stream for 0 Wait State Memory

YUV Mode	Pixels/Cycle memory burst mode	Rotation Peak Random Memory Access (pixels/cycle)		SCALING burst mode or rotation optimization is available
		Rotation Optimization	**Normal Mode**	
32 bpp AYUV	2	1	0.2	Supported
16 bpp 422	4	Not Supported	Not Supported	Supported

Note: rotation optimization = AHB lock asserted on consecutive single access

Table 32-51. Multiple Stream for 0 Wait State Memory

YUV Mode	Comp/Cycle memory burst mode	Rotation Peak Random Memory Access (pixels/cycle)		SCALING burst mode or rotation optimization is available
		Rotation Optimization	**Normal Mode**	
16 bpp 422 semiplanar	8 Y, 4 UV	1 Y, 1 UV (2 streams)	0.2 Y 0.2 UV (2 streams)	Supported
16 bpp 422 planar	8 Y, 8 U, 8 V	1 Y, 1 U, 1 V (3 streams)	0.2 Y, 0.2 U, 0.2 V (3 streams)	Supported
12 bpp 4:2:0 semiplanar	8 Y, 4 UV	1 Y, 1 UV (2 streams)	0.2 Y 0.2 UV (2 streams)	Supported
12 bpp 4:2:0 planar	8 Y, 8 U, 8 V	1 Y, 1 U, 1 V (3 streams)	0.2 Y, 0.2 U, 0.2 V (3 streams)	Supported

Note: In order to provide more bandwidth, when multiple streams are used to transfer Y, UV, U or V components, two AHB interfaces are recommended or multiple AXI ID are required.

Table 32-52. YUV Planar Overall Performance 1 AHB Interface for 0 Wait State Memory

YUV Mode	Pix/Cycle memory burst mode	Rotation Peak Random Memory Access (pixels/cycle)		SCALING burst mode or rotation optimization is available
		Rotation Optimization	**Normal Mode**	
16 bpp 422 semiplanar	4	0.66	0.132	Supported
16 bpp 422 planar	4	0.5	0.1	Supported
12 bpp 4:2:0 semiplanar	5.32	0.8	0.16	Supported
12 bpp 4:2:0 planar	5.32	0.66	0.132	Supported

In order to provide more bandwidth, when multiple streams are used to transfer Y, UV, U or V components, two AHB interfaces are recommended or multiple AXI ID are required.

32.6.15 Input FIFO

The LCD module includes one input FIFO per overlay. These input FIFOs are used to buffer the AHB burst and serialize the stream of pixels.

32.6.16 Output FIFO

The LCD module includes one output FIFO that stores the blended pixel.

32.6.17 Output Timing Generation

32.6.17.1 Active Display Timing Mode

Figure 32-15.Active Display Timing

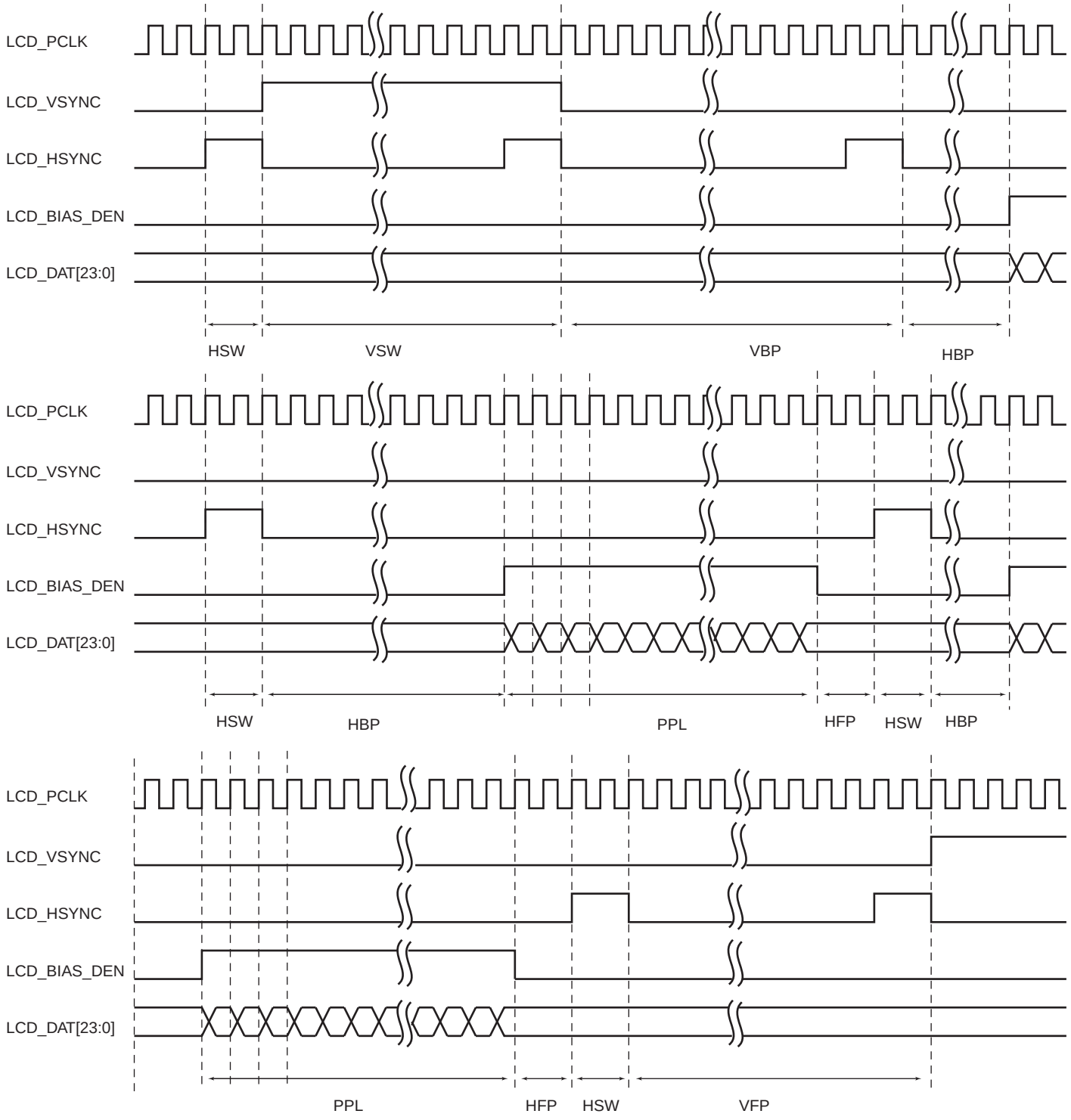


Figure 32-16.Vertical Synchronization Timing (part 1)

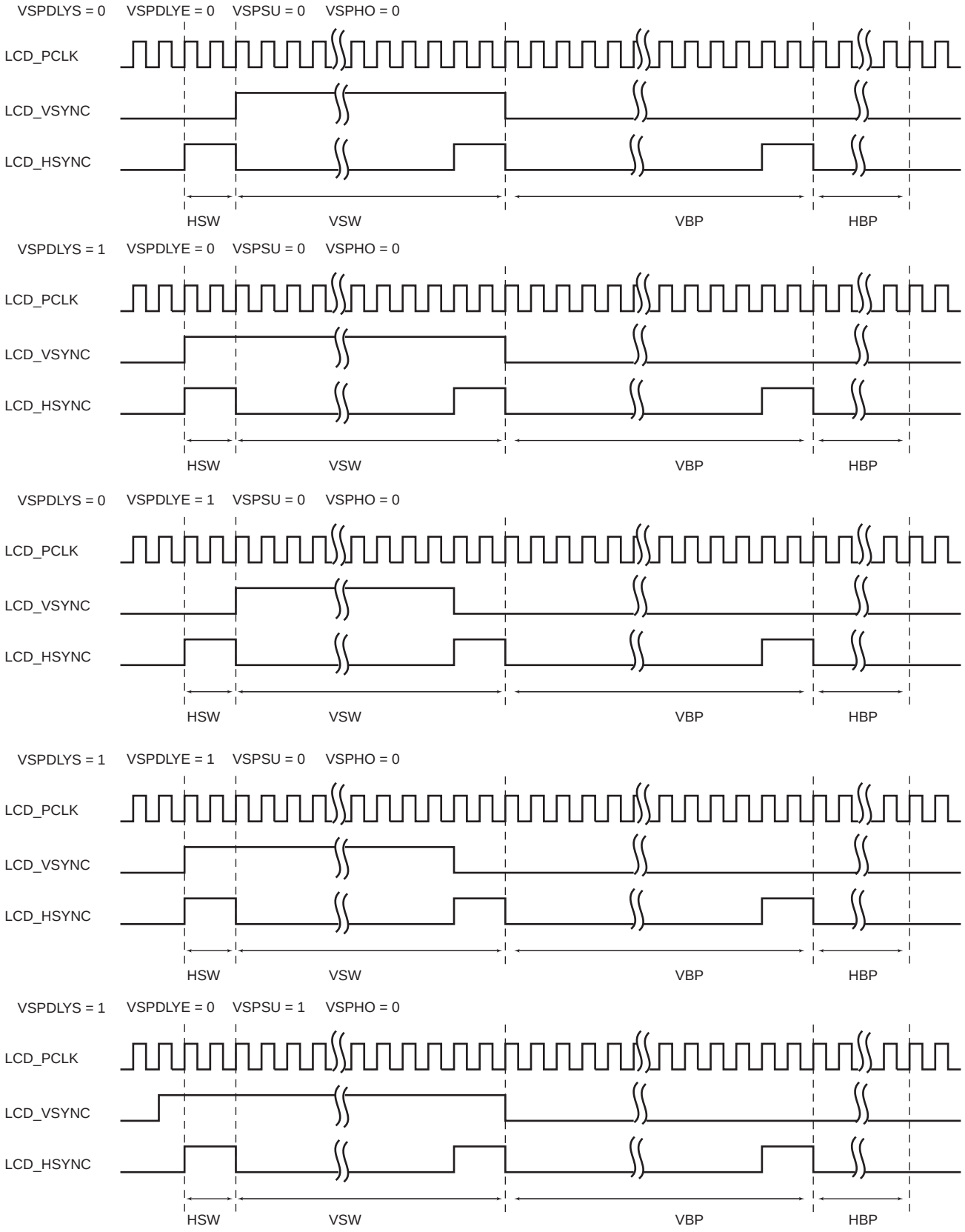


Figure 32-17.Vertical Synchronization Timing (part 2)

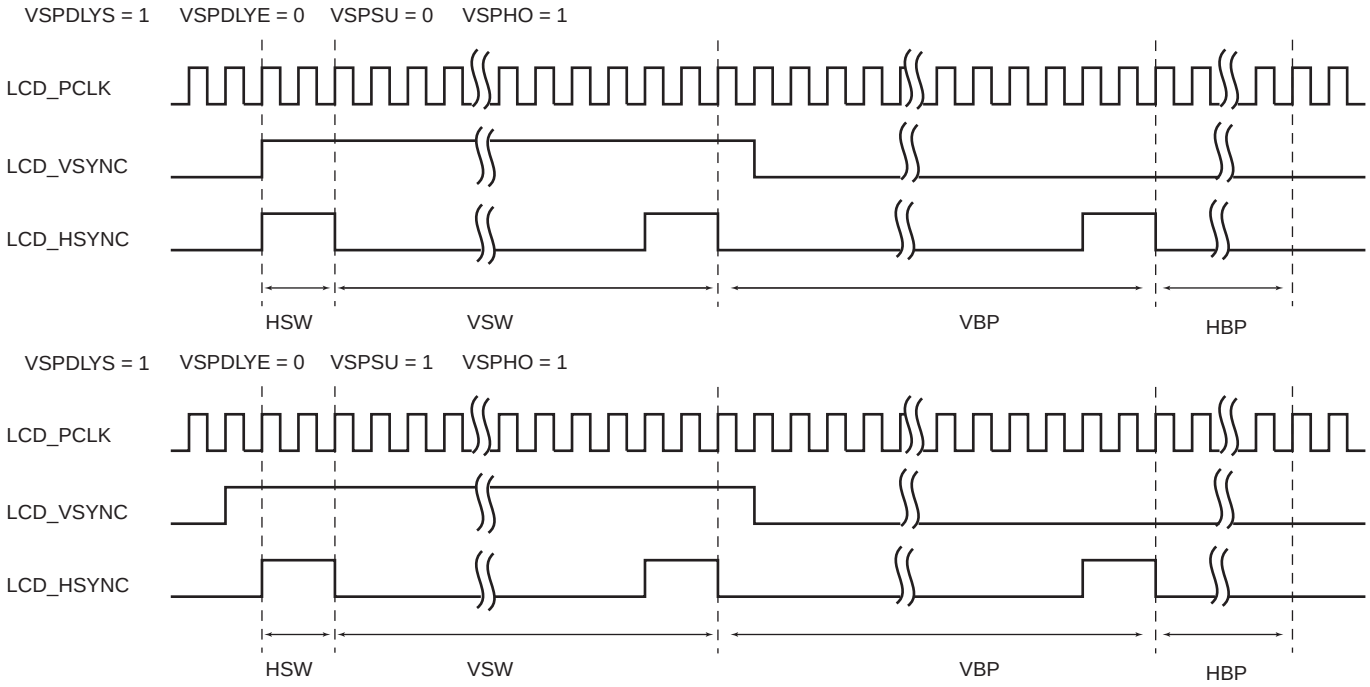
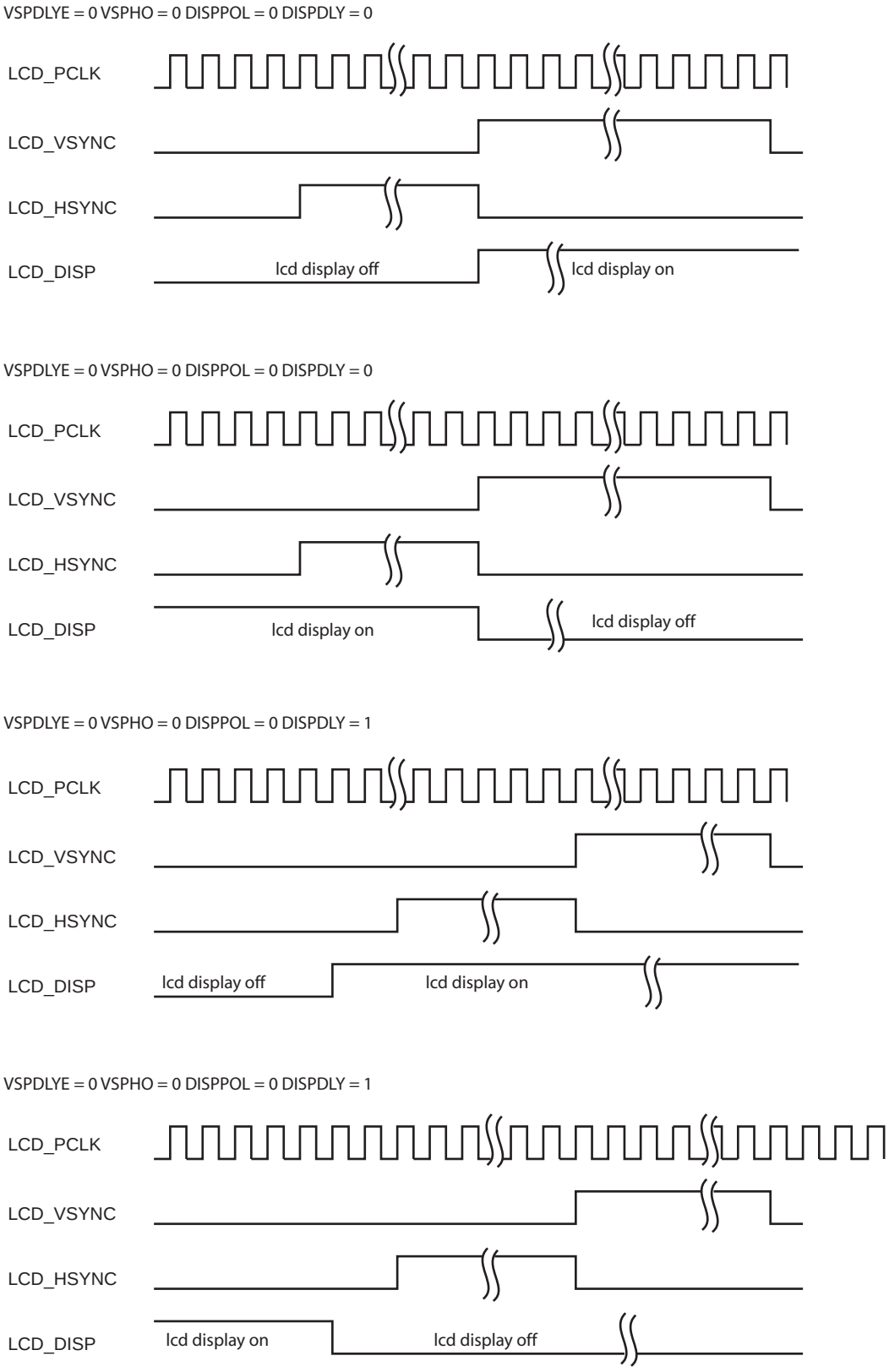


Figure 32-18.DISP Signal Timing Diagram



32.6.18 Output Format

32.6.18.1 Active Mode Output Pin Assignment

Table 32-53. Active Mode Output with 24 bits Bus Interface Configuration

Pin ID	TFT 24 bits	TFT 18 bits	TFT 16 bits	TFT 12 bits
LCD_DAT[23]	R[7]	–	–	–
LCD_DAT[22]	R[6]	–	–	–
LCD_DAT[21]	R[5]	–	–	–
LCD_DAT[20]	R[4]	–	–	–
LCD_DAT[19]	R[3]	–	–	–
LCD_DAT[18]	R[2]	–	–	–
LCD_DAT[17]	R[1]	R[5]	–	–
LCD_DAT[16]	R[0]	R[4]	–	–
LCD_DAT[15]	G[7]	R[3]	R[4]	–
LCD_DAT[14]	G[6]	R[2]	R[3]	–
LCD_DAT[13]	G[5]	R[1]	R[2]	–
LCD_DAT[12]	G[4]	R[0]	R[1]	–
LCD_DAT[11]	G[3]	G[5]	R[0]	R[3]
LCD_DAT[10]	G[2]	G[4]	G[5]	R[2]
LCD_DAT[9]	G[1]	G[3]	G[4]	R[1]
LCD_DAT[8]	G[0]	G[2]	G[3]	R[0]
LCD_DAT[7]	B[7]	G[1]	G[2]	G[3]
LCD_DAT[6]	B[6]	G[0]	G[1]	G[2]
LCD_DAT[5]	B[5]	B[5]	G[0]	G[1]
LCD_DAT[4]	B[4]	B[4]	B[4]	G[0]
LCD_DAT[3]	B[3]	B[3]	B[3]	B[3]
LCD_DAT[2]	B[2]	B[2]	B[2]	B[2]
LCD_DAT[1]	B[1]	B[1]	B[1]	B[1]
LCD_DAT[0]	B[0]	B[0]	B[0]	B[0]

32.7 LCD Controller (LCDC) User Interface

Table 32-54. Register Mapping

Offset	Register	Name	Access	Reset
0x00000000	LCD Controller Configuration Register 0	LCDC_LCDCFG0	Read-write	0x00000000
0x00000004	LCD Controller Configuration Register 1	LCDC_LCDCFG1	Read-write	0x00000000
0x00000008	LCD Controller Configuration Register 2	LCDC_LCDCFG2	Read-write	0x00000000
0x0000000C	LCD Controller Configuration Register 3	LCDC_LCDCFG3	Read-write	0x00000000
0x00000010	LCD Controller Configuration Register 4	LCDC_LCDCFG4	Read-write	0x00000000
0x00000014	LCD Controller Configuration Register 5	LCDC_LCDCFG5	Read-write	0x00000000
0x00000018	LCD Controller Configuration Register 6	LCDC_LCDCFG6	Read-write	0x00000000
0x0000001C	Reserved	–	–	–
0x00000020	LCD Controller Enable Register	LCDC_LCDEN	Write-only	–
0x00000024	LCD Controller Disable Register	LCDC_LCDDIS	Write-only	–
0x00000028	LCD Controller Status Register	LCDC_LCDSR	Read-only	0x00000000
0x0000002C	LCD Controller Interrupt Enable Register	LCDC_LCDIER	Write-only	–
0x00000030	LCD Controller Interrupt Disable Register	LCDC_LCDIDR	Write-only	–
0x00000034	LCD Controller Interrupt Mask Register	LCDC_LCDIMR	Read-only	0x00000000
0x00000038	LCD Controller Interrupt Status Register	LCDC_LCDISR	Read-only	0x00000000
0x0000003C	LCD Controller Attribute Register	LCDC_ATTR	Write-only	–
0x00000040	Base Layer Channel Enable Register	LCDC_BASECHER	Write-only	0x00000000
0x00000044	Base Layer Channel Disable Register	LCDC_BASECHDR	Write-only	0x00000000
0x00000048	Base Layer Channel Status Register	LCDC_BASECHSR	Read-only	0x00000000
0x0000004C	Base Layer Interrupt Enable Register	LCDC_BASEIER	Write-only	0x00000000
0x00000050	Base Layer Interrupt Disabled Register	LCDC_BASEIDR	Write-only	0x00000000
0x00000054	Base Layer Interrupt Mask Register	LCDC_BASEIMR	Read-only	0x00000000
0x00000058	Base Layer Interrupt status Register	LCDC_BASEISR	Read-only	0x00000000
0x0000005C	Base DMA Head Register	LCDC_BASEHEAD	Read-write	0x00000000
0x00000060	Base DMA Address Register	LCDC_BASEADDR	Read-write	0x00000000
0x00000064	Base DMA Control Register	LCDC_BASECTRL	Read-write	0x00000000
0x00000068	Base DMA Next Register	LCDC_BASENEXT	Read-write	0x00000000
0x0000006C	Base Configuration register 0	LCDC_BASECFG0	Read-write	0x00000000
0x00000070	Base Configuration register 1	LCDC_BASECFG1	Read-write	0x00000000
0x00000074	Base Configuration register 2	LCDC_BASECFG2	Read-write	0x00000000
0x00000078	Base Configuration register 3	LCDC_BASECFG3	Read-write	0x00000000
0x0000007C	Base Configuration register 4	LCDC_BASECFG4	Read-write	0x00000000
0x00000080	Base Configuration register 5	LCDC_BASECFG5	Read-write	0x00000000
0x00000084	Base Configuration register 6	LCDC_BASECFG6	Read-write	0x00000000
0x88-0x13C	Reserved	–	–	–
0x00000140	Overlay 1 Channel Enable Register	LCDC_OVRCHER1	Write-only	0x00000000
0x00000144	Overlay 1 Channel Disable Register	LCDC_OVRCHDR1	Write-only	0x00000000
0x00000148	Overlay 1 Channel Status Register	LCDC_OVRCHSR1	Read-only	0x00000000

Table 32-54. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0000014C	Overlay 1 Interrupt Enable Register	LCDC_OVRIER1	Write-only	0x00000000
0x00000150	Overlay 1 Interrupt Disable Register	LCDC_OVRIDR1	Write-only	0x00000000
0x00000154	Overlay 1 Interrupt Mask Register	LCDC_OVRIMR1	Read-only	0x00000000
0x00000158	Overlay 1 Interrupt Status Register	LCDC_OVRISR1	Read-only	0x00000000
0x0000015C	Overlay 1 DMA Head Register	LCDC_OVRHEAD1	Read-write	0x00000000
0x00000160	Overlay 1 DMA Address Register	LCDC_OVRADDR1	Read-write	0x00000000
0x00000164	Overlay 1 DMA Control Register	LCDC_OVRCTRL1	Read-write	0x00000000
0x00000168	Overlay 1 DMA Next Register	LCDC_OVRNEXT1	Read-write	0x00000000
0x0000016C	Overlay 1 Configuration 0 Register	LCDC_OVR1CFG0	Read-write	0x00000000
0x00000170	Overlay 1 Configuration 1 Register	LCDC_OVR1CFG1	Read-write	0x00000000
0x00000174	Overlay 1 Configuration 2 Register	LCDC_OVR1CFG2	Read-write	0x00000000
0x00000178	Overlay 1 Configuration 3 Register	LCDC_OVR1CFG3	Read-write	0x00000000
0x0000017C	Overlay 1 Configuration 4 Register	LCDC_OVR1CFG4	Read-write	0x00000000
0x00000180	Overlay 1 Configuration 5 Register	LCDC_OVR1CFG5	Read-write	0x00000000
0x00000184	Overlay 1 Configuration 6 Register	LCDC_OVR1CFG6	Read-write	0x00000000
0x00000188	Overlay 1 Configuration 7 Register	LCDC_OVR1CFG7	Read-write	0x00000000
0x0000018C	Overlay 1 Configuration 8Register	LCDC_OVR1CFG8	Read-write	0x00000000
0x00000190	Overlay 1 Configuration 9 Register	LCDC_OVR1CFG9	Read-write	0x00000000
0x194-0x23C	Reserved	–	–	–
0x00000240	Overlay 2 Channel Enable Register	LCDC_OVRCHER2	Write-only	0x00000000
0x00000244	Overlay 2 Channel Disable Register	LCDC_OVRCHDR2	Write-only	0x00000000
0x00000248	Overlay 2 Channel Status Register	LCDC_OVRCHSR2	Read-only	0x00000000
0x0000024C	Overlay 2 Interrupt Enable Register	LCDC_OVRIER2	Write-only	0x00000000
0x00000250	Overlay 2 Interrupt Disable Register	LCDC_OVRIDR2	Write-only	0x00000000
0x00000254	Overlay 2 Interrupt Mask Register	LCDC_OVRIMR2	Read-only	0x00000000
0x00000258	Overlay 2 Interrupt status Register	LCDC_OVRISR2	Read-only	0x00000000
0x0000025C	Overlay 2 DMA Head Register	LCDC_OVRHEAD2	Read-write	0x00000000
0x00000260	Overlay 2 DMA Address Register	LCDC_OVRADDR2	Read-write	0x00000000
0x00000264	Overlay 2 DMA Control Register	LCDC_OVRCTRL2	Read-write	0x00000000
0x00000268	Overlay 2 DMA Next Register	LCDC_OVRNEXT2	Read-write	0x00000000
0x0000026C	Overlay 2 Configuration 0 Register	LCDC_OVR2CFG0	Read-write	0x00000000
0x00000270	Overlay 2 Configuration 1 Register	LCDC_OVR2CFG1	Read-write	0x00000000
0x00000274	Overlay 2 Configuration 2 Register	LCDC_OVR2CFG2	Read-write	0x00000000
0x00000278	Overlay 2 Configuration 3 Register	LCDC_OVR2CFG3	Read-write	0x00000000
0x0000027C	Overlay 2 Configuration 4 Register	LCDC_OVR2CFG4	Read-write	0x00000000
0x00000280	Overlay 2 Configuration 5 Register	LCDC_OVR2CFG5	Read-write	0x00000000
0x00000284	Overlay 2 Configuration 6 Register	LCDC_OVR2CFG6	Read-write	0x00000000
0x00000288	Overlay 2 Configuration 7 Register	LCDC_OVR2CFG7	Read-write	0x00000000
0x0000028C	Overlay 2 Configuration 8 Register	LCDC_OVR2CFG8	Read-write	0x00000000

Table 32-54. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x00000290	Overlay 2 Configuration 9 Register	LCDC_OVR2CFG9	Read-write	0x00000000
0x294-33C	Reserved	–	–	–
0x00000340	High-End Overlay Channel Enable Register	LCDC_HEOCHER	Write-only	0x00000000
0x00000344	High-End Overlay Channel Disable Register	LCDC_HEOCHDR	Write-only	0x00000000
0x00000348	High-End Overlay Channel Status Register	LCDC_HEOCHSR	Read-only	0x00000000
0x0000034C	High-End Overlay Interrupt Enable Register	LCDC_HEOIER	Write-only	0x00000000
0x00000350	High-End Overlay Interrupt Disable Register	LCDC_HEOIDR	Write-only	0x00000000
0x00000354	High-End Overlay Interrupt Mask Register	LCDC_HEOIMR	Read-only	0x00000000
0x00000358	High-End Overlay Interrupt Status Register	LCDC_HEOISR	Read-only	0x00000000
0x0000035C	High-End Overlay DMA Head Register	LCDC_HEOHEAD	Read-write	0x00000000
0x00000360	High-End Overlay DMA Address Register	LCDC_HEOADDR	Read-write	0x00000000
0x00000364	High-End Overlay DMA Control Register	LCDC_HEOCTRL	Read-write	0x00000000
0x00000368	High-End Overlay DMA Next Register	LCDC_HEONEXT	Read-write	0x00000000
0x0000036C	High-End Overlay U DMA Head Register	LCDC_HEOUHEAD	Read-write	0x00000000
0x00000370	High-End Overlay U DMA Address Register	LCDC_HEOUADDR	Read-write	0x00000000
0x00000374	High-End Overlay U DMA control Register	LCDC_HEOUCTRL	Read-write	0x00000000
0x00000378	High-End Overlay U DMA Next Register	LCDC_HEOUNEXT	Read-write	0x00000000
0x0000037C	High-End Overlay V DMA Head Register	LCDC_HEOVHEAD	Read-write	0x00000000
0x00000380	High-End Overlay V DMA Address Register	LCDC_HEOVADDR	Read-write	0x00000000
0x00000384	High-End Overlay V DMA control Register	LCDC_HEOVCTRL	Read-write	0x00000000
0x00000388	High-End Overlay VDMA Next Register	LCDC_HEOVNEXT	Read-write	0x00000000
0x0000038C	High-End Overlay Configuration Register 0	LCDC_HEOCFG0	Read-write	0x00000000
0x00000390	High-End Overlay Configuration Register 1	LCDC_HEOCFG1	Read-write	0x00000000
0x00000394	High-End Overlay Configuration Register 2	LCDC_HEOCFG2	Read-write	0x00000000
0x00000398	High-End Overlay Configuration Register 3	LCDC_HEOCFG3	Read-write	0x00000000
0x0000039C	High-End Overlay Configuration Register 4	LCDC_HEOCFG4	Read-write	0x00000000
0x000003A0	High-End Overlay Configuration Register 5	LCDC_HEOCFG5	Read-write	0x00000000
0x000003A4	High-End Overlay Configuration Register 6	LCDC_HEOCFG6	Read-write	0x00000000
0x000003A8	High-End Overlay Configuration Register 7	LCDC_HEOCFG7	Read-write	0x00000000
0x000003AC	High-End Overlay Configuration Register 8	LCDC_HEOCFG8	Read-write	0x00000000
0x000003B0	High-End Overlay Configuration Register 9	LCDC_HEOCFG9	Read-write	0x00000000
0x000003B4	High-End Overlay Configuration Register 10	LCDC_HEOCFG10	Read-write	0x00000000
0x000003B8	High-End Overlay Configuration Register 11	LCDC_HEOCFG11	Read-write	0x00000000
0x000003BC	High-End Overlay Configuration Register 12	LCDC_HEOCFG12	Read-write	0x00000000
0x000003C0	High-End Overlay Configuration Register 13	LCDC_HEOCFG13	Read-write	0x00000000
0x000003C4	High-End Overlay Configuration Register 14	LCDC_HEOCFG14	Read-write	0x00000000
0x000003C8	High-End Overlay Configuration Register 15	LCDC_HEOCFG15	Read-write	0x00000000
0x000003CC	High-End Overlay Configuration Register 16	LCDC_HEOCFG16	Read-write	0x00000000
0x000003D0	High-End Overlay Configuration Register 17	LCDC_HEOCFG17	Read-write	0x00000000

Table 32-54. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x000003D4	High-End Overlay Configuration Register 18	LCDC_HEOCFG18	Read-write	0x00000000
0x000003D8	High-End Overlay Configuration Register 19	LCDC_HEOCFG19	Read-write	0x00000000
0x000003DC	High-End Overlay Configuration Register 20	LCDC_HEOCFG20	Read-write	0x00000000
0x000003E0	High-End Overlay Configuration Register 21	LCDC_HEOCFG21	Read-write	0x00000000
0x000003E4	High-End Overlay Configuration Register 22	LCDC_HEOCFG22	Read-write	0x00000000
0x000003E8	High-End Overlay Configuration Register 23	LCDC_HEOCFG23	Read-write	0x00000000
0x000003EC	High-End Overlay Configuration Register 24	LCDC_HEOCFG24	Read-write	0x00000000
0x000003F0	High-End Overlay Configuration Register 25	LCDC_HEOCFG25	Read-write	0x00000000
0x000003F4	High-End Overlay Configuration Register 26	LCDC_HEOCFG26	Read-write	0x00000000
0x000003F8	High-End Overlay Configuration Register 27	LCDC_HEOCFG27	Read-write	0x00000000
0x000003FC	High-End Overlay Configuration Register 28	LCDC_HEOCFG28	Read-write	0x00000000
0x00000400	High-End Overlay Configuration Register 29	LCDC_HEOCFG29	Read-write	0x00000000
0x00000404	High-End Overlay Configuration Register 30	LCDC_HEOCFG30	Read-write	0x00000000
0x00000408	High-End Overlay Configuration Register 31	LCDC_HEOCFG31	Read-write	0x00000000
0x0000040C	High-End Overlay Configuration Register 32	LCDC_HEOCFG32	Read-write	0x00000000
0x00000410	High-End Overlay Configuration Register 33	LCDC_HEOCFG33	Read-write	0x00000000
0x00000414	High-End Overlay Configuration Register 34	LCDC_HEOCFG34	Read-write	0x00000000
0x00000418	High-End Overlay Configuration Register 35	LCDC_HEOCFG35	Read-write	0x00000000
0x0000041C	High-End Overlay Configuration Register 36	LCDC_HEOCFG36	Read-write	0x00000000
0x00000420	High-End Overlay Configuration Register 37	LCDC_HEOCFG37	Read-write	0x00000000
0x00000424	High-End Overlay Configuration Register 38	LCDC_HEOCFG38	Read-write	0x00000000
0x00000428	High-End Overlay Configuration Register 39	LCDC_HEOCFG39	Read-write	0x00000000
0x0000042C	High-End Overlay Configuration Register 40	LCDC_HEOCFG40	Read-write	0x00000000
0x00000430	High-End Overlay Configuration Register 41	LCDC_HEOCFG41	Read-write	0x00000000
0x434-0x43C	Reserved	–	–	–
0x00000440	Hardware Cursor Channel Enable Register	LCDC_HCRCHER	Write-only	0x00000000
0x00000444	Hardware Cursor Channel disable Register	LCDC_HCRCHDR	Write-only	0x00000000
0x00000448	Hardware Cursor Channel Status Register	LCDC_HCRCHSR	Read-only	0x00000000
0x0000044C	Hardware Cursor Interrupt Enable Register	LCDC_HCRIER	Write-only	0x00000000
0x00000450	Hardware Cursor Interrupt Disable Register	LCDC_HCRIDR	Write-only	0x00000000
0x00000454	Hardware Cursor Interrupt Mask Register	LCDC_HCRIMR	Read-only	0x00000000
0x00000458	Hardware Cursor Interrupt Status Register	LCDC_HCRISR	Read-only	0x00000000
0x0000045C	Hardware Cursor DMA Head Register	LCDC_HCRHEAD	Read-write	0x00000000
0x00000460	Hardware cursor DMA Address Register	LCDC_HCRADDR	Read-write	0x00000000
0x00000464	Hardware Cursor DMA Control Register	LCDC_HCRCTRL	Read-write	0x00000000
0x00000468	Hardware Cursor DMA NExt Register	LCDC_HCRNEXT	Read-write	0x00000000
0x0000046C	Hardware Cursor Configuration 0 Register	LCDC_HCRCFG0	Read-write	0x00000000
0x00000470	Hardware Cursor Configuration 1 Register	LCDC_HCRCFG1	Read-write	0x00000000
0x00000474	Hardware Cursor Configuration 2 Register	LCDC_HCRCFG2	Read-write	0x00000000

Table 32-54. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x00000478	Hardware Cursor Configuration 3 Register	LCDC_HCRCFG3	Read-write	0x00000000
0x0000047C	Hardware Cursor Configuration 4 Register	LCDC_HCRCFG4	Read-write	0x00000000
0x00000480	Reserved	–	–	–
0x00000484	Hardware Cursor Configuration 6 Register	LCDC_HCRCFG6	Read-write	0x00000000
0x00000488	Hardware Cursor Configuration 7 Register	LCDC_HCRCFG7	Read-write	0x00000000
0x0000048C	Hardware Cursor Configuration 8 Register	LCDC_HCRCFG8	Read-write	0x00000000
0x00000490	Hardware Cursor Configuration 9 Register	LCDC_HCRCFG9	Read-write	0x00000000
0x494-0x53C	Reserved	–	–	–
0x00000540	Post Processing Channel Enable Register	LCDC_PPCHER	Write-only	0x00000000
0x00000544	Post Processing Channel Disable Register	LCDC_PPCHDR	Write-only	0x00000000
0x00000548	Post Processing Channel Status Register	LCDC_PPCHSR	Read-only	0x00000000
0x0000054C	Post Processing Interrupt Enable Register	LCDC_PPIER	Write-only	0x00000000
0x00000550	Post Processing Interrupt Disable Register	LCDC_PPIDR	Write-only	0x00000000
0x00000554	Post Processing Interrupt Mask Register	LCDC_PPIMR	Read-only	0x00000000
0x00000558	Post Processing Interrupt Status Register	LCDC_PPISR	Read-only	0x00000000
0x0000055C	Post Processing Head Register	LCDC_PPHEAD	Read-write	0x00000000
0x00000560	Post Processing Address Register	LCDC_PPADDR	Read-write	0x00000000
0x00000564	Post Processing Control Register	LCDC_PPCTRL	Read-write	0x00000000
0x00000568	Post Processing Next Register	LCDC_PPNEXT	Read-write	0x00000000
0x0000056C	Post Processing Configuration Register 0	LCDC_PPCFG0	Read-write	0x00000000
0x00000570	Post Processing Configuration Register 1	LCDC_PPCFG1	Read-write	0x00000000
0x00000574	Post Processing Configuration Register 2	LCDC_PPCFG2	Read-write	0x00000000
0x00000578	Post Processing Configuration Register 3	LCDC_PPCFG3	Read-write	0x00000000
0x0000057C	Post Processing Configuration Register 4	LCDC_PPCFG4	Read-write	0x00000000
0x00000580	Post Processing Configuration Register 5	LCDC_PPCFG5	Read-write	0x00000000
0x584-0x5FC	Reserved	–	–	–
0x600	Base CLUT Register 0	LCDC_BASECLUT0	Read-write	0x00000000
...	...	...	...	...
0x8FC	Base CLUT Register 255	LCDC_BASECLUT255	Read-write	0x00000000
0xA00	Overlay 1 CLUT Register 0	LCDC_OVR1CLUT0	Read-write	0x00000000
...	...	...	...	...
0xDFC	Overlay 1 CLUT Register 255	LCDC_OVR1CLUT255	Read-write	0x00000000
0xE00	Overlay 2 CLUT Register 0	LCDC_OVR2CLUT0	Read-write	0x00000000
...	...	...	...	...
0x11FC	Overlay 2 CLUT Register 255	LCDC_OVR2CLUT255	Read-write	0x00000000
0x1200	High End Overlay CLUT Register 0	LCDC_HEOCLUT0	Read-write	0x00000000
...	...	...	...	...
0x15FC	High End Overlay CLUT Register 255	LCDC_HEOCLUT255	Read-write	0x00000000
0x1600	Hardware Cursor CLUT Register 0	LCDC_HCRCLUT0	Read-write	0x00000000

Table 32-54. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
...	...	...	...	...
0x19FC	Hardware Cursor CLUT Register 255	LCDC_HCRCLUT255	Read-write	0x00000000
0x1A00-0x1FE4	Reserved	–	–	–

Note: 1. The CLUT registers are located in the RAM.

32.7.1 LCD Controller Configuration Register 0

Name: LCDC_LCDCFG0

Address: 0xF0030000

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
–	–	CGDISPP	CGDISHCR	CGDISHEO	CGDISOVR2	CGDISOVR1	CGDISBASE
7	6	5	4	3	2	1	0
–	–	–	–	CLKPWMSEL	CLKSEL	–	CLKPOL

- **CLKPOL: LCD Controller Clock Polarity**

0: Data/Control signals are launched on the rising edge of the Pixel Clock.

1: Data/Control signals are launched on the falling edge of the Pixel Clock.

- **CLKSEL: LCD Controller Clock Source Selection**

0: The Asynchronous output stage of the LCD controller is fed by the System Clock.

1: The Asynchronous output state of the LCD controller is fed by the 2x System Clock.

- **CLKPWMSEL: LCD Controller PWM Clock Source Selection**

0: The slow clock is selected and feeds the PWM module.

1: The system clock is selected and feeds the PWM module.

- **CGDISBASE: Clock Gating Disable Control for the Base Layer**

0: Automatic Clock Gating is enabled for the Base Layer.

1: Clock is running continuously.

- **CGDISOVR1: Clock Gating Disable Control for the Overlay 1 Layer**

0: Automatic Clock Gating is enabled for the Overlay 1 Layer.

1: Clock is running continuously.

- **CGDISOVR2: Clock Gating Disable Control for the Overlay 2 Layer**

0: Automatic Clock Gating is enabled for the Overlay 2 Layer.

1: Clock is running continuously.

- **CGDISHEO: Clock Gating Disable Control for the High End Overlay**

0: Automatic Clock Gating is enabled for the High End Overlay Layer.

1: Clock is running continuously.

- **CGDISHCR: Clock Gating Disable Control for the Hardware Cursor Layer**

0: Automatic Clock Gating is enabled for the Hardware Cursor Layer.

1: Clock is running continuously.

- **CGDISPP: Clock Gating Disable Control for the Post Processing Layer**

0: Automatic Clock Gating is enabled for the Post Processing Layer.

1: Clock is running continuously.

- **CLKDIV: LCD Controller Clock Divider**

8-bit width clock divider for pixel clock LCD_PCLK.

pixel_clock = selected_clock / (CLKDIV+2)

where

selected_clock is equal to system_clock when CLKSEL field is set to 0 and system_clock2x when CLKSEL is set to 1.

32.7.2 LCD Controller Configuration Register 1

Name: LCDC_LCDCFG1

Address: 0xF0030004

Access: Read-write

Reset: 0x00000000

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

- **HSPW: Horizontal Synchronization Pulse Width**

Width of the LCD_HSYNC pulse, given in pixel clock cycles. Width is (HSPW+1) LCD_PCLK cycles.

- **VSPW: Vertical Synchronization Pulse Width**

Width of the LCD_VSYNC pulse, given in number of lines. Width is (VSPW+1) lines.

32.7.3 LCD Controller Configuration Register 2

Name: LCDC_LCDCFG2

Address: 0xF0030008

Access: Read-write

Reset: 0x00000000

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

- **VFPW: Vertical Front Porch Width**

This field indicates the number of lines at the end of the Frame. The blanking interval is equal to (VFPW+1) lines.

- **VBPW: Vertical Back Porch Width**

This field indicates the number of lines at the beginning of the Frame. The blanking interval is equal to VBPW lines.

32.7.4 LCD Controller Configuration Register 3

Name: LCDC_LCDCFG3

Address: 0xF003000C

Access: Read-write

Reset: 0x00000000

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

- **HFPW: Horizontal Front Porch Width**

Number of pixel clock cycles inserted at the end of the active line. The interval is equal to (HFPW+1) LCD_PCLK cycles.

- **HBPW: Horizontal Back Porch Width**

Number of pixel clock cycles inserted at the beginning of the line. The interval is equal to (HBPW+1) LCD_PCLK cycles.

32.7.5 LCD Controller Configuration Register 4

Name: LCDC_LCDCFG4

Address: 0xF0030010

Access: Read-write

Reset: 0x00000000

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

- **RPF: Number of Active Row Per Frame**

Number of active lines in the frame. The frame height is equal to (RPF+1) lines.

- **PPL: Number of Pixels Per Line**

Number of pixel in the frame. The number of active pixels in the frame is equal to (PPL+1) pixels.

32.7.6 LCD Controller Configuration Register 5

Name: LCDC_LCDCFG5

Address: 0xF0030014

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	GUARDTIME				
15	14	13	12	11	10	9	8
–	–	VSPHO	VSPSU	–	PP	MODE	
7	6	5	4	3	2	1	0
DISPDLY	DITHER	–	DISPPOL	VSPDLYE	VSPDLYS	VSPOL	HSPOL

- **HSPOL: Horizontal Synchronization Pulse Polarity**

0: Active High

1: Active Low

- **VSPOL: Vertical Synchronization Pulse Polarity**

0: Active High

1: Active Low

- **VSPDLYS: Vertical Synchronization Pulse Start**

0: The first active edge of the Vertical synchronization pulse is synchronous with the second edge of the horizontal pulse.

1: The first active edge of the Vertical synchronization pulse is synchronous with the first edge of the horizontal pulse.

- **VSPDLYE: Vertical Synchronization Pulse End**

0: The second active edge of the Vertical synchronization pulse is synchronous with the second edge of the horizontal pulse.

1: The second active edge of the Vertical synchronization pulse is synchronous with the first edge of the horizontal pulse.

- **DISPPOL: Display Signal Polarity**

0: Active High

1: Active Low

- **DITHER: LCD Controller Dithering**

0: Dithering logical unit is disabled.

1: Dithering logical unit is activated.

- **DISPDLY: LCD Controller Display Power Signal Synchronization**

0: The LCD_DISP signal is asserted synchronously with the second active edge of the horizontal pulse.

1: The LCD_DISP signal is asserted asynchronously with both edges of the horizontal pulse.

- **MODE: LCD Controller Output Mode**

Value	Name	Description
0	OUTPUT_12BPP	LCD output mode is set to 12 bits per pixel
1	OUTPUT_16BPP	LCD output mode is set to 16 bits per pixel
2	OUTPUT_18BPP	LCD output mode is set to 18 bits per pixel
3	OUTPUT_24BPP	LCD output mode is set to 24 bits per pixel

- **PP: Post Processing Enable**

0: The Blended pixel is pushed into the output FIFO.

1: The Blended pixel is written back to memory, the post processing stage is enabled.

- **VSPSU: LCD Controller Vertical synchronization Pulse Setup Configuration**

0: The vertical synchronization pulse is asserted synchronously with horizontal pulse edge.

1: The vertical synchronization pulse is asserted one pixel clock cycle before the horizontal pulse.

- **VSPHO: LCD Controller Vertical synchronization Pulse Hold Configuration**

0: The vertical synchronization pulse is asserted synchronously with horizontal pulse edge.

1: The vertical synchronization pulse is held active one pixel clock cycle after the horizontal pulse.

- **GUARDTIME: LCD DISPLAY Guard Time**

Number of frames inserted during start up before LCD_DISP assertion.

Number of frames inserted after LCD_DISP reset.

32.7.7 LCD Controller Configuration Register 6

Name: LCDC_LCDCFG6

Address: 0xF0030018

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
PWMCVAL							
7	6	5	4	3	2	1	0
–	–	–	PWMPOL	–	PWMPS		

- **PWMPS: PWM Clock Prescaler**

3-bit value. Selects the configuration of the counter prescaler module. The PWMPS field decoding is listed below.

Value	Name	Description
000	DIV_1	The counter advances at a rate of fCOUNTER = fPWM_SELECTED_CLOCK
001	DIV_2	The counter advances at a rate of fCOUNTER = fPWM_SELECTED_CLOCK/2
010	DIV_4	The counter advances at a rate of fCOUNTER = fPWM_SELECTED_CLOCK/4
011	DIV_8	The counter advances at a rate of fCOUNTER = fPWM_SELECTED_CLOCK/8
100	DIV_16	The counter advances at a rate of fCOUNTER = fPWM_SELECTED_CLOCK/16
101	DIV_32	The counter advances at a of rate fCOUNTER = fPWM_SELECTED_CLOCK/32
110	DIV_64	The counter advances at a of rate fCOUNTER = fPWM_SELECTED_CLOCK/64

- **PWMPOL: LCD Controller PWM Signal Polarity**

This bit defines the polarity of the PWM output signal. If set to one, the output pulses are high level (the output will be high whenever the value in the counter is less than the value CVAL) If set to zero, the output pulses are low level.

- **PWMCVAL: LCD Controller PWM Compare Value**

PWM compare value. Used to adjust the analog value obtained after an external filter to control the contrast of the display.

32.7.8 LCD Controller Enable Register

Name: LCDC_LCDEN

Address: 0xF0030020

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
–	–	–	–	PWMEN	DISPEN	SYNCEN	CLKEN

- **CLKEN: LCD Controller Pixel Clock Enable**

0: Writing this field to zero has no effect.

1: When set to one the pixel clock logical unit is activated.

- **SYNCEN: LCD Controller Horizontal and Vertical Synchronization Enable**

0: Writing this field to zero has no effect.

1: When set to one, both horizontal and vertical synchronization (LCD_VSYNC and LCD_HSYNC) signals are generated.

- **DISPEN: LCD Controller DISP Signal Enable**

0: Writing this field to zero has no effect.

1: When set to one, LCD_DISP signal is generated.

- **PWMEN: LCD Controller Pulse Width Modulation Enable**

0: Writing this field to zero has no effect.

1: When set to one, the pwm is enabled.

32.7.9 LCD Controller Disable Register

Name: LCDC_LCDDIS

Address: 0xF0030024

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
–	–	–	–	PWMRST	DISPRST	SYNCRST	CLKRST
7	6	5	4	3	2	1	0
–	–	–	–	PWMDIS	DISPDIS	SYNCDIS	CLKDIS

- **CLKDIS: LCD Controller Pixel Clock Disable**

0: No effect.

1: Disables the pixel clock.

- **SYNCDIS: LCD Controller Horizontal and Vertical Synchronization Disable**

0: No effect.

1: Disables the synchronization signals after the end of the frame.

- **DISPDIS: LCD Controller DISP Signal Disable**

0: No effect

1: Disables the DISP signal.

- **PWMDIS: LCD Controller Pulse Width Modulation Disable**

0: No effect

1: Disables the pulse width modulation signal.

- **CLKRST: LCD Controller Clock Reset**

0: No effect.

1: Resets the pixel clock generator module. The pixel clock duty cycle may be violated.

- **SYNCRST: LCD Controller Horizontal and Vertical Synchronization Reset**

0: No effect.

1: Resets the timing engine. Both Horizontal and vertical pulse width are violated.

- **DISPRST: LCD Controller DISP Signal Reset**

0: No effect.

1: Resets the DISP signal.

- **PWMRST: LCD Controller PWM Reset**

0: No effect.

1: Resets the PWM module, the duty cycle may be violated.

32.7.10 LCD Controller Status Register

Name: LCDC_LCDSR

Address: 0xF0030028

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
–	–	–	SIPSTS	PWMSTS	DISPSTS	LCDSTS	CLKSTS

- **CLKSTS: Clock Status**

0: Pixel Clock is disabled.

1: Pixel Clock is running.

- **LCDSTS: LCD Controller Synchronization status**

0: Timing Engine is disabled.

1: Timing Engine is running.

- **DISPSTS: LCD Controller DISP Signal Status**

0: DISP is disabled.

1: DISP signal is activated.

- **PWMSTS: LCD Controller PWM Signal Status**

0: PWM is disabled.

1: PWM signal is activated.

- **SIPSTS: Synchronization In Progress**

0: Clock domain synchronization is terminated.

1: Synchronization is in progress. Access to the registers LCDC_LCDCCFG[0..6], LCDC_LCDEN and LCDC_LCDDIS has no effect.

32.7.11 LCD Controller Interrupt Enable Register

Name: LCDC_LCDIER

Address: 0xF003002C

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
–	–	PPIE	HCRIE	HEOIE	OVR2IE	OVR1IE	BASEIE
7	6	5	4	3	2	1	0
–	–	–	FIFOERRIE	–	DISPIE	DISIE	SOFIE

- **SOFIE: Start of Frame Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **DISIE: LCD Disable Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **DISPIE: Power UP/Down Sequence Terminated Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **FIFOERRIE: Output FIFO Error Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **BASEIE: Base Layer Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **OVR1IE: Overlay 1 Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **OVR2IE: Overlay 2 Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **HEOIE: High End Overlay Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **HCRIE: Hardware Cursor Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

- **PPIE: Post Processing Interrupt Enable Register**

0: No effect

1: Enables the interrupt.

32.7.12 LCD Controller Interrupt Disable Register

Name: LCDC_LCDIDR

Address: 0xF0030030

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
–	–	PPID	HCRID	HEOID	OVR2ID	OVR1ID	BASEID
7	6	5	4	3	2	1	0
–	–	–	FIFOERRID	–	DISPID	DISID	SOFID

- **SOFID: Start of Frame Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **DISID: LCD Disable Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **DISPID: Power UP/Down Sequence Terminated Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **FIFOERRID: Output FIFO Error Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **BASEID: Base Layer Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **OVR1ID: Overlay 1 Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **OVR2ID: Overlay 2 Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **HEOID: High End Overlay Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **HCRID: Hardware Cursor Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

- **PPID: Post Processing Interrupt Disable Register**

0: No effect

1: Disables the interrupt.

32.7.13 LCD Controller Interrupt Mask Register

Name: LCDC_LCDIMR

Address: 0xF0030034

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
–	–	PPIM	HCRIM	HEOIM	OVR2IM	OVR1IM	BASEIM
7	6	5	4	3	2	1	0
–	–	–	FIFOERRIM	–	DISPIM	DISIM	SOFIM

- **SOFIM: Start of Frame Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DISIM: LCD Disable Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DISPIM: Power UP/Down Sequence Terminated Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **FIFOERRIM: Output FIFO Error Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **BASEIM: Base Layer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR1IM: Overlay 1 Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR2IM: Overlay 2 Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **HEOIM: High End Overlay Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **HCRIM: Hardware Cursor Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **PPIM: Post Processing Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.14 LCD Controller Interrupt Status Register

Name: LCDC_LCDISR

Address: 0xF0030038

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
–	–	PP	HCR	HEO	OVR2	OVR1	BASE
7	6	5	4	3	2	1	0
–	–	–	FIFOERR	–	DISP	DIS	SOF

- **SOF: Start of Frame Interrupt Status Register**

When set to one, this flag indicates that a start of frame event has been detected. This flag is reset after a read operation.

- **DIS: LCD Disable Interrupt Status Register**

When set to one, this flag indicates that the horizontal and vertical timing generator has been successfully disabled. This flag is reset after a read operation.

- **DISP: Power-up/Power-down Sequence Terminated Interrupt Status Register**

When set to one, this flag indicates whether the power-up sequence or power-down sequence has terminated. This flag is reset after a read operation.

- **FIFOERR: Output FIFO Error**

When set to one, this flag indicates that an underflow occurs in the output FIFO. This flag is reset after a read operation.

- **BASE: Base Layer Raw Interrupt Status Register**

When set to one, this flag indicates that a base layer interrupt is pending. This flag is reset as soon as the BASEISR register is read.

- **OVR1: Overlay 1 Raw Interrupt Status Register**

When set to one, this flag indicates that an Overlay 1 layer interrupt is pending. This flag is reset as soon as the OVR1ISR register is read.

- **OVR2: Overlay 2 Raw Interrupt Status Register**

When set to one this flag indicates that an Overlay 1 layer interrupt is pending. This flag is reset as soon as the OVR1ISR register is read.

- **HEO: High End Overlay Raw Interrupt Status Register**

When set to one, this flag indicates that a Hi End layer interrupt is pending. This flag is reset as soon as the HEOISR register is read.

- **HCR: Hardware Cursor Raw Interrupt Status Register**

When set to one, this flag indicates that a Hardware Cursor layer interrupt is pending. This flag is reset as soon as the HCRISR register is read.

- **PP: Post Processing Raw Interrupt Status Register**

When set to one, this flag indicates that Post Processing interrupt is pending. This flag is reset as soon as the PPISR register is read.

32.7.15 LCD Controller Attribute Register

Name: LCDC_ATTR

Address: 0xF003003C

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
–	–	PPA2Q	HCRA2Q	HEOA2Q	OVR2A2Q	OVR1A2Q	BASEA2Q
7	6	5	4	3	2	1	0
–	–	PP	HCR	HEO	OVR2	OVR1	BASE

- **BASE: Base Layer Update Attribute Register**

0: No effect

1: Update the BASE window attributes.

- **OVR1: Overlay 1 Update Attribute Register**

0: No effect

1: Update the OVR1 window attribute

- **OVR2: Overlay 2 Update Attribute Register**

0: No effect

1: Update the OVR2 window attribute

- **HEO: High-End Overlay Update Attribute Register**

0: No effect

1: Update the HEO window attribute

- **HCR: Hardware Cursor Update Attribute Register**

0: No effect

1: Update the BCH window attribute

- **PP: Post-Processing Update Attribute Register**

0: No effect

1: Update the PP window attribute

- **BASEA2Q: Base Layer Update Attribute Register**

0: No effect

1: Add the descriptor pointed out by the BASEHEAD register to the descriptor list.

- **OVR1A2Q: Overlay 1 Update Attribute Register**

0: No effect

1: Add the descriptor pointed out by the OVR1HEAD register to the descriptor list.

- **OVR2A2Q: Overlay 2 Update Attribute Register**

0: No effect

1: Add the descriptor pointed out by the OVR2HEAD register to the descriptor list.

- **HEOA2Q: High-End Overlay Update Attribute Register**

0: No effect

1: Add the descriptor pointed out by the HEOHEAD register to the descriptor list.

- **HCRA2Q: Hardware Cursor Update Attribute Register**

0: No effect

1: Add the descriptor pointed out by the HCRHEAD register to the descriptor list.

- **PPA2Q: Post-Processing Update Attribute Register**

0: No effect

1: Add the descriptor pointed out by the PPHEAD register to the descriptor list.

32.7.16 Base Layer Channel Enable Register

Name: LCDC_BASECHER

Address: 0xF0030040

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
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

- **CHEN: Channel Enable Register**

0: No effect.

1: Enables the DMA channel.

- **UPDATEEN: Update Overlay Attributes Enable Register**

0: No effect.

1: Updates windows attributes on the next start of frame.

- **A2QEN: Add to Queue Enable Register**

When set to one, it indicates that a valid descriptor has been written to memory, its memory location should be written to the DMA head pointer. The A2QSR status bit is set to one, and it is reset by hardware as soon as the descriptor pointed out by the DMA head pointer is added to the list.

32.7.17 Base Layer Channel Disable Register

Name: LCDC_BASECHDR

Address: 0xF0030044

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
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

- **CHDIS: Channel Disable Register**

When set to one this field disables the layer at the end of the current frame. The frame is completed.

- **CHRST: Channel Reset Register**

When set to one this field resets the layer immediately. The frame is aborted.

32.7.18 Base Layer Channel Status Register

Name: LCDC_BASECHSR

Address: 0xF0030048

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
–	–	–	–	–	A2QSR	UPDATESR	CHSR

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will be updated on the next frame.

- **A2QSR: Add To Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

32.7.19 Base Layer Interrupt Enable Register

Name: LCDC_BASEIER

Address: 0xF003004C

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

32.7.20 Base Layer Interrupt Disable Register

Name: LCDC_BASEIDR

Address: 0xF0030050

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DSCR: Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DONE: End of List Interrupt Disable Register**

0: No effect.

1: interrupt source is disabled.

- **OVR: Overflow Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

32.7.21 Base Layer Interrupt Mask Register

Name: LCDC_BASEIMR

Address: 0xF0030054

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.22 Base Layer Interrupt Status Register

Name: LCDC_BASEISR

Address: 0xF0030058

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **DSCR: DMA Descriptor Loaded**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **ADD: Head Descriptor Loaded**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **DONE: End of List Detected**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **OVR: Overflow Detected**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

32.7.23 Base Layer Head Register

Name: LCDC_BASEHEAD

Address: 0xF003005C

Access: Read-write

Reset: 0x00000000

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

- **HEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.24 Base Layer Address Register

Name: LCDC_BASEADDR

Address: 0xF0030060

Access: Read-write

Reset: 0x00000000

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: DMA Transfer Start Address**

Frame buffer base address.

32.7.25 Base Layer Control Register

Name: LCDC_BASECTRL

Address: 0xF0030064

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
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	LFETCH	DFETCH

- **DFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **LFETCH: Lookup Table Fetch Enable**

0: Lookup Table DMA fetch is disabled.

1: Lookup Table DMA fetch is enabled.

- **DMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **DSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **ADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **DONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.26 Base Layer Next Register

Name: LCDC_BASENEXT

Address: 0xF0030068

Access: Read-write

Reset: 0x00000000

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

- **NEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.27 Base Layer Configuration 0 Register

Name: LCDC_BASECFG0

Address: 0xF003006C

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
–	–	–	–	–	–	–	DLBO
7	6	5	4	3	2	1	0
–	–	BLEN		–	–	–	SIF

- **SIF: Source Interface**

0: Base Layer data is retrieved through AHB interface 0.

1: Base Layer data is retrieved through AHB interface 1.

- **BLEN: AHB Burst Length**

Value	Name	Description
0	AHB_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **DLBO: Defined Length Burst Only For Channel Bus Transaction.**

0: Undefined length INCR burst is used for a burst of 2 and 3 beats.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

32.7.28 Base Layer Configuration 1 Register

Name: LCDC_BASECFG1

Address: 0xF0030070

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
–				–	–	CLUTMODE	
7	6	5	4	3	2	1	0
RGBMODE				–	–	–	CLUTEN

- **CLUTEN: Color Lookup Table Enable**

0: RGB mode is selected.

1: Color lookup table is selected.

- **RGBMODE: RGB Input Mode Selection**

Value	Name	Description
0	12BPP_RGB_444	12 bpp RGB 444
1	16BPP_ARGB_4444	16 bpp ARGB 4444
2	16BPP_RGBA_4444	16 bpp RGBA 4444
3	16BPP_RGB_565	16 bpp RGB 565
4	16BPP_TRGB_1555	16 bpp TRGB 1555
5	18BPP_RGB_666	18 bpp RGB 666
6	18BPP_RGB_666PACKED	18 bpp RGB 666 PACKED
7	19BPP_TRGB_1666	19 bpp TRGB 1666
8	19BPP_TRGB_PACKED	19 bpp TRGB 1666 PACKED
9	24BPP_RGB_888	24 bpp RGB 888
10	24BPP_RGB_888_PACKED	24 bpp RGB 888 PACKED
11	25BPP_TRGB_1888	25 bpp TRGB 1888
12	32BPP_ARGB_8888	32 bpp ARGB 8888
13	32BPP_RGBA_8888	32 bpp RGBA 8888

- **CLUTMODE: Color Lookup Table Input Mode Selection**

Value	Name	Description
0	CLUT_1BPP	color lookup table mode set to 1 bit per pixel
1	CLUT_2BPP	color lookup table mode set to 2 bits per pixel
2	CLUT_4BPP	color lookup table mode set to 4 bits per pixel
3	CLUT_8BPP	color lookup table mode set to 8 bits per pixel

32.7.29 Base Layer Configuration 2 Register

Name: LCDC_BASECFG2

Address: 0xF0030074

Access: Read-write

Reset: 0x00000000

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

- **XSTRIDE: Horizontal Stride**

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.30 Base Layer Configuration 3 Register

Name: LCDC_BASECFG3

Address: 0xF0030078

Access: Read-write

Reset: 0x00000000

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

- **RDEF: Red Default**

Default Red color when the Base DMA channel is disabled.

- **GDEF: Green Default**

Default Green color when the Base DMA channel is disabled.

- **BDEF: Blue Default**

Default Blue color when the Base DMA channel is disabled.

32.7.31 Base Layer Configuration 4 Register

Name: LCDC_BASECFG4

Address: 0xF003007C

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
–	–	–	–	DISCEN	–	REP	DMA
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **DMA: Use DMA Data Path**

0: The default color is used on the Base Layer.

1: The DMA channel retrieves the pixels stream from the memory.

- **REP: Use Replication logic to expand RGB color to 24 bits**

0: When the selected pixel depth is less than 24 bpp the pixel is shifted and least significant bits are set to 0.

1: When the selected pixel depth is less than 24 bpp the pixel is shifted and the least significant bit replicates the msb.

- **DISCEN: Discard Area Enable**

0: The whole frame is retrieved from memory.

1:When set to one the DMA channel discards the area located at screen coordinate {DISCXPOS, DISCYPOS}.

32.7.32 Base Layer Configuration 5 Register

Name: LCDC_BASECFG5

Address: 0xF0030080

Access: Read-write

Reset: 0x00000000

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

- **DISCXPOS: Discard Area horizontal coordinate**

Horizontal Position of the Discard Area.

- **DISCYPOS: Discard Area Vertical coordinate**

Vertical Position of the Discard Area.

32.7.33 Base Layer Configuration 6 Register

Name: LCDC_BASECFG6

Address: 0xF0030084

Access: Read-write

Reset: 0x00000000

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

- **DISCXSIZE: Discard Area Horizontal Size**

Discard Horizontal size in pixels. The Discard size is set to (DISCXSIZE+1) pixels in horizontal.

- **DISCYSIZE: Discard Area Vertical Size**

Discard Vertical size in pixels. The Discard size is set to (DISCYSIZE+1) pixels in vertical.

32.7.34 Overlay 1 Layer Channel Enable Register

Name: LCDC_OVRCHER1

Address: 0xF0030140

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
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

- **CHEN: Channel Enable Register**

0: No effect.

1: Enables the DMA channel.

- **UPDATEEN: Update Overlay Attributes Enable Register**

0: No effect.

1: Updates window attributes (size, alpha-blending, etc.) on the next start of frame.

- **A2QEN: Add to Queue Enable Register**

When set to one, it indicates that a valid descriptor has been written to memory, its memory location should be written to the DMA head pointer. The A2QSR status bit is set to one, and it is reset by hardware as soon as the descriptor pointed out by the DMA head pointer is added to the list.

32.7.35 Overlay 1 Layer Channel Disable Register

Name: LCDC_OVRCHDR1

Address: 0xF0030144

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
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

- **CHDIS: Channel Disable Register**

When set to one this field disables the layer at the end of the current frame. The frame is completed.

- **CHRST: Channel Reset Register**

When set to one this field resets the layer immediately. The frame is aborted.

32.7.36 Overlay 1 Layer Channel Status Register

Name: LCDC_OVRCHSR1

Address: 0xF0030148

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
–	–	–	–	–	A2QSR	UPDATESR	CHSR

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will be updated on the next frame.

- **A2QSR: Add to Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

32.7.37 Overlay 1 Layer Interrupt Enable Register

Name: LCDC_OVRIER1

Address: 0xF003014C

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

32.7.38 Overlay 1 Layer Interrupt Disable Register

Name: LCDC_OVRIDR1

Address: 0xF0030150

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DSCR: Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DONE: End of List Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **OVR: Overflow Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

32.7.39 Overlay 1 Layer Interrupt Mask Register

Name: LCDC_OVRIMR1

Address: 0xF0030154

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.40 Overlay 1 Layer Interrupt Status Register

Name: LCDC_OVRISR1

Address: 0xF0030158

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **DSCR: DMA Descriptor Loaded**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **ADD: Head Descriptor Loaded**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **DONE: End of List Detected Register**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **OVR: Overflow Detected**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

32.7.41 Overlay 1 Layer Head Register

Name: LCDC_OVRHEAD1

Address: 0xF003015C

Access: Read-write

Reset: 0x00000000

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

- **HEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.42 Overlay 1 Layer Address Register

Name: LCDC_OVRADDR1

Address: 0xF0030160

Access: Read-write

Reset: 0x00000000

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: DMA Transfer Overlay 1 Address**

Overlay 1 frame buffer base address.

32.7.43 Overlay 1 Layer Control Register

Name: LCDC_OVRCTRL1

Address: 0xF0030164

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
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	LFETCH	DFETCH

- **DFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **LFETCH: Lookup Table Fetch Enable**

0: Lookup Table DMA fetch is disabled.

1: Lookup Table DMA fetch is enabled.

- **DMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **DSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **ADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **DONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.44 Overlay 1 Layer Next Register

Name: LCDC_OVRNEXT1

Address: 0xF0030168

Access: Read-write

Reset: 0x00000000

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

- **NEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.45 Overlay 1 Layer Configuration 0 Register

Name: LCDC_OVR1CFG0

Address: 0xF003016C

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
–	–	LOCKDIS	ROTDIS	–	–	–	DLBO
7	6	5	4	3	2	1	0
–	–	BLEN		–	–	–	SIF

- **SIF: Source Interface**

0: Base Layer data is retrieved through AHB interface 0.

1: Base Layer data is retrieved through AHB interface 1.

- **BLEN: AHB Burst Length**

Value	Name	Description
0	AHB_BLEN_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_BLEN_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_BLEN_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_BLEN_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **DLBO: Defined Length Burst Only for Channel Bus Transaction.**

0: Undefined length INCR burst is used for a burst of 2 and 3 beats.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

- **ROTDIS: Hardware Rotation Optimization Disable**

0: Rotation optimization is enabled.

1: Rotation optimization is disabled.

- **LOCKDIS: Hardware Rotation Lock Disable**

0: AHB lock signal is asserted when a rotation is performed.

1: AHB lock signal is cleared when a rotation is performed.

32.7.46 Overlay 1 Layer Configuration 1 Register

Name: LCDC_OVR1CFG1

Address: 0xF0030170

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
–	–	–	–	–	–	CLUTMODE	
7	6	5	4	3	2	1	0
RGBMODE				–	–	–	CLUTEN

- **CLUTEN: Color Lookup Table Enable**

0: RGB mode is selected.

1: Color lookup table is selected.

- **RGBMODE: RGB Input Mode Selection**

Value	Name	Description
0	12BPP_RGB_444	12 bpp RGB 444
1	16BPP_ARGB_4444	16 bpp ARGB 4444
2	16BPP_RGBA_4444	16 bpp RGBA 4444
3	16BPP_RGB_565	16 bpp RGB 565
4	16BPP_TRGB_1555	16 bpp TRGB 1555
5	18BPP_RGB_666	18 bpp RGB 666
6	18BPP_RGB_666PACKED	18 bpp RGB 666 PACKED
7	19BPP_TRGB_1666	19 bpp TRGB 1666
8	19BPP_TRGB_PACKED	19 bpp TRGB 1666 PACKED
9	24BPP_RGB_888	24 bpp RGB 888
10	24BPP_RGB_888_PACKED	24 bpp RGB 888 PACKED
11	25BPP_TRGB_1888	25 bpp TRGB 1888
12	32BPP_ARGB_8888	32 bpp ARGB 8888
13	32BPP_RGBA_8888	32 bpp RGBA 8888

- **CLUTMODE: Color Lookup table input mode selection**

Value	Name	Description
0	CLUT_1BPP	color lookup table mode set to 1 bit per pixel
1	CLUT_2BPP	color lookup table mode set to 2 bits per pixel
2	CLUT_4BPP	color lookup table mode set to 4 bits per pixel
3	CLUT_8BPP	color lookup table mode set to 8 bits per pixel

32.7.47 Overlay 1 Layer Configuration 2 Register

Name: LCDC_OVR1CFG2

Address: 0xF0030174

Access: Read-write

Reset: 0x00000000

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

- **XPOS: Horizontal Window Position**

Overlay 1 Horizontal window position.

- **YPOS: Vertical Window Position**

Overlay 1 Vertical window position.

32.7.48 Overlay 1 Layer Configuration 3 Register

Name: LCDC_OVR1CFG3

Address: 0xF0030178

Access: Read-write

Reset: 0x00000000

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

- **XSIZE: Horizontal Window Size**

Overlay 1 window width in pixels. The window width is set to (XSIZE+1).

The following constraint must be met:

$$XPOS + XSIZE \leq PPL$$

- **YSIZE: Vertical Window Size**

Overlay 1 window height in pixels. The window height is set to (YSIZE+1).

The following constrain must be met:

$$YPOS + YSIZE \leq RPF$$

32.7.49 Overlay 1 Layer Configuration 4 Register

Name: LCDC_OVR1CFG4

Address: 0xF003017C

Access: Read-write

Reset: 0x00000000

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

- **XSTRIDE: Horizontal Stride**

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.50 Overlay 1 Layer Configuration 5 Register

Name: LCDC_OVR1CFG5

Address: 0xF0030180

Access: Read-write

Reset: 0x00000000

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

- **PSTRIDE: Pixel Stride**

PSTRIDE represents the memory offset, in bytes, between two pixels of the image.

32.7.51 Overlay 1 Layer Configuration 6 Register

Name: LCDC_OVR1CFG6

Address: 0xF0030184

Access: Read-write

Reset: 0x00000000

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

- **RDEF: Red Default**

Default Red color when the Overlay 1 DMA channel is disabled.

- **GDEF: Green Default**

Default Green color when the Overlay 1 DMA channel is disabled.

- **BDEF: Blue Default**

Default Blue color when the Overlay 1 DMA channel is disabled.

32.7.52 Overlay 1 Layer Configuration 7 Register

Name: LCDC_OVR1CFG7

Address: 0xF0030188

Access: Read-write

Reset: 0x00000000

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

- **RKEY: Red Color Component Chroma Key**

Reference Red chroma key used to match the Red color of the current overlay.

- **GKEY: Green Color Component Chroma Key**

Reference Green chroma key used to match the Green color of the current overlay.

- **BKEY: Blue Color Component Chroma Key**

Reference Blue chroma key used to match the Blue color of the current overlay.

32.7.53 Overlay 1 Layer Configuration 8 Register

Name: LCDC_OVR1CFG8

Address: 0xF003018C

Access: Read-write

Reset: 0x00000000

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

- **RMASK: Red Color Component Chroma Key Mask**

Red Mask used when the compare function is used. If a bit is set then this bit is compared.

- **GMASK: Green Color Component Chroma Key Mask**

Green Mask used when the compare function is used. If a bit is set then this bit is compared.

- **BMASK: Blue Color Component Chroma Key Mask**

Blue Mask used when the compare function is used. If a bit is set then this bit is compared.

32.7.54 Overlay1 Layer Configuration 9 Register

Name: LCDC_OVR1CFG9

Address: 0xF0030190

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
GA							
15	14	13	12	11	10	9	8
–	–	–	–	–	DSTKEY	REP	DMA
7	6	5	4	3	2	1	0
OVR	LAEN	GAEN	REVALPHA	ITER	ITER2BL	INV	CRKEY

- **CRKEY: Blender Chroma Key Enable**

0: Chroma key matching is disabled.

1: Chroma key matching is enabled.

- **INV: Blender Inverted Blender Output Enable**

0: Iterated pixel is the blended pixel.

1: Iterated pixel is the inverted pixel.

- **ITER2BL: Blender Iterated Color Enable**

0: Final adder stage operand is set to 0.

1: Final adder stage operand is set to the iterated pixel value.

- **ITER: Blender Use Iterated Color**

0: Pixel difference is set to 0.

1: Pixel difference is set to the iterated pixel value.

- **REVALPHA: Blender Reverse Alpha**

0: Pixel difference is multiplied by alpha.

1: Pixel difference is multiplied by 1 - alpha.

- **GAEN: Blender Global Alpha Enable**

0: Global alpha blending coefficient is disabled.

1: Global alpha blending coefficient is enabled.

- **LAEN: Blender Local Alpha Enable**

0: Local alpha blending coefficient is disabled.

1: Local alpha blending coefficient is enabled.

- **OVR: Blender Overlay Layer Enable**

0: Overlay pixel color is set to the default overlay pixel color.

1: Overlay pixel color is set to the DMA channel pixel color.

- **DMA: Blender DMA Layer Enable**

0: The default color is used on the Overlay 1 Layer.

1: The DMA channel retrieves the pixels stream from the memory.

- **REP: Use Replication logic to expand RGB color to 24 bits**

0: When the selected pixel depth is less than 24 bpp the pixel is shifted and least significant bits are set to 0.

1: When the selected pixel depth is less than 24 bpp the pixel is shifted and the least significant bit replicates the msb.

- **DSTKEY: Destination Chroma Keying**

0: Source Chroma keying is enabled.

1: Destination Chroma keying is used.

- **GA: Blender Global Alpha**

Global alpha blender for the current layer.

32.7.55 Overlay 2 Layer Channel Enable Register

Name: LCDC_OVRCHER2

Address: 0xF0030240

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
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

- **CHEN: Channel Enable Register**

0: No effect.

1: Enables the DMA channel.

- **UPDATEEN: Update Overlay Attributes Enable Register**

0: No effect.

1: Updates windows attributes on the next start of frame.

- **A2QEN: Add to Queue Enable Register**

When set to one, it indicates that a valid descriptor has been written to memory, its memory location should be written to the DMA head pointer. The A2QSR status bit is set to one, and it is reset by hardware as soon as the descriptor pointed out by the DMA head pointer is added to the list.

32.7.56 Overlay 2 Layer Channel Disable Register

Name: LCDC_OVRCHDR2

Address: 0xF0030244

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
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

- **CHDIS: Channel Disable Register**

When set to one this field disables the layer at the end of the current frame.

- **CHRST: Channel Reset Register**

When set to one this field disables the layer at the end of the current frame.

32.7.57 Overlay 2 Layer Channel Status Register

Name: LCDC_OVRCHSR2

Address: 0xF0030248

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
–	–	–	–	–	A2QSR	UPDATESR	CHSR

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will update on the next Frame.

- **A2QSR: Add To Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

32.7.58 Overlay 2 Layer Interrupt Enable Register

Name: LCDC_OVRIER2

Address: 0xF003024C

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

32.7.59 Overlay 2 Layer Interrupt Disable Register

Name: LCDC_OVRIDR2

Address: 0xF0030250

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DSCR: Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DONE: End of List Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **OVR: Overflow Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

32.7.60 Overlay 2 Layer Interrupt Mask Register

Name: LCDC_OVRIMR2

Address: 0xF0030254

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.61 Overlay 2 Layer Interrupt Status Register

Name: LCDC_OVRISR2

Address: 0xF0030258

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **DSCR: DMA Descriptor Loaded**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **DONE: End Of List Interrupt Disable Register**

When set to one this flag indicates that a End of List condition has occurred. This flag is reset after a read operation.

- **OVR: Overflow Detected**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

32.7.62 Overlay 2 Layer Head Register

Name: LCDC_OVRHEAD2

Address: 0xF003025C

Access: Read-Write

Reset: 0x00000000

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

- **HEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.63 Overlay 2 Layer Address Register

Name: LCDC_OVRADDR2

Address: 0xF0030260

Access: Read-Write

Reset: 0x00000000

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: DMA Transfer Overlay 2 Address**

Overlay 2 frame buffer base address.

32.7.64 Overlay 2 Layer Control Register

Name: LCDC_OVRCTRL2

Address: 0xF0030264

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
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	LFETCH	DFETCH

- **DFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **LFETCH: Lookup Table Fetch Enable**

0: Lookup Table DMA fetch is disabled.

1: Lookup Table DMA fetch is enabled.

- **DMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **DSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **ADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **DONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.65 Overlay 2 Layer Next Register

Name: LCDC_OVRNEXT2

Address: 0xF0030268

Access: Read-Write

Reset: 0x00000000

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

- **NEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.66 Overlay 2 Layer Configuration 0 Register

Name: LCDC_OVR2CFG0

Address: 0xF003026C

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
–	–	LOCKDIS	ROTDIS	–	–	–	DLBO
7	6	5	4	3	2	1	0
–	–	BLEN		–	–	–	–

- **BLEN: AHB Burst Length**

Value	Name	Description
0	AHB_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **DLBO: Defined Length Burst Only For Channel Bus Transaction.**

0: Undefined length INCR burst is used for 2 and 3 beats burst.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

- **ROTDIS: Hardware Rotation Optimization Disable**

0: Rotation optimization is enabled.

1: Rotation optimization is disabled.

- **LOCKDIS: Hardware Rotation Lock Disable**

0: AHB lock signal is asserted when a rotation is performed.

1: AHB lock signal is cleared when a rotation is performed.

32.7.67 Overlay 2 Layer Configuration 1 Register

Name: LCDC_OVR2CFG1

Address: 0xF0030270

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
–	–	–	–	–	–	CLUTMODE	
7	6	5	4	3	2	1	0
RGBMODE				–	–	–	CLUTEN

- **CLUTEN: Color Lookup Table Enable**

0: RGB mode is selected

1: Color lookup table is selected

- **RGBMODE: RGB Input Mode Selection**

Value	Name	Description
0	12BPP_RGB_444	12 bpp RGB 444
1	16BPP_ARGB_4444	16 bpp ARGB 4444
2	16BPP_RGBA_4444	16 bpp RGBA 4444
3	16BPP_RGB_565	16 bpp RGB 565
4	16BPP_TRGB_1555	16 bpp TRGB 1555
5	18BPP_RGB_666	18 bpp RGB 666
6	18BPP_RGB_666PACKED	18 bpp RGB 666 PACKED
7	19BPP_TRGB_1666	19 bpp TRGB 1666
8	19BPP_TRGB_PACKED	19 bpp TRGB 1666 PACKED
9	24BPP_RGB_888	24 bpp RGB 888
10	24BPP_RGB_888_PACKED	24 bpp RGB 888 PACKED
11	25BPP_TRGB_1888	25 bpp TRGB 1888
12	32BPP_ARGB_8888	32 bpp ARGB 8888
13	32BPP_RGBA_8888	32 bpp RGBA 8888

- **CLUTMODE: Color Lookup table input mode selection**

Value	Name	Description
0	CLUT_1BPP	color lookup table mode set to 1 bit per pixel
1	CLUT_2BPP	color lookup table mode set to 2 bits per pixel
2	CLUT_4BPP	color lookup table mode set to 4 bits per pixel
3	CLUT_8BPP	color lookup table mode set to 8 bits per pixel

32.7.68 Overlay 2 Layer Configuration 2 Register

Name: LCDC_OVR2CFG2

Address: 0xF0030274

Access: Read-Write

Reset: 0x00000000

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

- **XPOS: Horizontal Window Position**

Overlay 2 Horizontal window position.

- **YPOS: Vertical Window Position**

Overlay 2 Vertical window position.

32.7.69 Overlay 2 Layer Configuration 3 Register

Name: LCDC_OVR2CFG3

Address: 0xF0030278

Access: Read-Write

Reset: 0x00000000

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

- **XSIZE: Horizontal Window Size**

Overlay 2 window width in pixels. The window width is set to (XSIZE+1).

The following constraint must be met:

$$XPOS + XSIZE \leq PPL$$

- **YSIZE: Vertical Window Size**

Overlay 2 window height in pixels. The window height is set to (YSIZE+1).

The following constrain must be met:

$$YPOS + YSIZE \leq RPF$$

32.7.70 Overlay 2 Layer Configuration 4 Register

Name: LCDC_OVR2CFG4

Address: 0xF003027C

Access: Read-Write

Reset: 0x00000000

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

- **XSTRIDE: Horizontal Stride**

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.71 Overlay 2 Layer Configuration 5 Register

Name: LCDC_OVR2CFG5

Address: 0xF0030280

Access: Read-Write

Reset: 0x00000000

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

- **PSTRIDE: Pixel Stride**

PSTRIDE represents the memory offset, in bytes, between two pixels of the image memory.

32.7.72 Overlay 2 Layer Configuration 6 Register

Name: LCDC_OVR2CFG6

Address: 0xF0030284

Access: Read-Write

Reset: 0x00000000

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

- **RDEF: Red Default**

Default Red color when the Overlay 1 DMA channel is disabled.

- **GDEF: Green Default**

Default Green color when the Overlay 1 DMA channel is disabled.

- **BDEF: Blue Default**

Default Blue color when the Overlay 1 DMA channel is disabled.

32.7.73 Overlay 2 Layer Configuration 7 Register

Name: LCDC_OVR2CFG7

Address: 0xF0030288

Access: Read-Write

Reset: 0x00000000

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

- **RKEY: Red Color Component Chroma Key**

Reference Red chroma key used to match the Red color of the current overlay.

- **GKEY: Green Color Component Chroma Key**

Reference Green chroma key used to match the Green color of the current overlay.

- **BKEY: Blue Color Component Chroma Key**

Reference Blue chroma key used to match the Blue color of the current overlay.

32.7.74 Overlay 2 Layer Configuration 8 Register

Name: LCDC_OVR2CFG8

Address: 0xF003028C

Access: Read-Write

Reset: 0x00000000

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

- **RMASK: Red Color Component Chroma Key Mask**

Red Mask used when the compare function is used. If a bit is set then this bit is compared.

- **GMASK: Green Color Component Chroma Key Mask**

Green Mask used when the compare function is used. If a bit is set then this bit is compared.

- **BMASK: Blue Color Component Chroma Key Mask**

Blue Mask used when the compare function is used. If a bit is set then this bit is compared.

32.7.75 Overlay 2 Layer Configuration 9 Register

Name: LCDC_OVR2CFG9

Address: 0xF0030290

Access: Read-Write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
GA							
15	14	13	12	11	10	9	8
–	–	–	–	–	DSTKEY	REP	DMA
7	6	5	4	3	2	1	0
OVR	LAEN	GAEN	REVALPHA	ITER	ITER2BL	INV	CRKEY

- **CRKEY: Blender Chroma Key Enable**

0: Chroma key matching is disabled.

1: Chroma key matching is enabled.

- **INV: Blender Inverted Blender Output Enable**

0: Iterated pixel is the blended pixel.

1: Iterated pixel is the inverted pixel.

- **ITER2BL: Blender Iterated Color Enable**

0: Final adder stage operand is set to 0.

1: Final adder stage operand is set to the iterated pixel value.

- **ITER: Blender Use Iterated Color**

0: Pixel difference is set to 0.

1: Pixel difference is set to the iterated pixel value.

- **REVALPHA: Blender Reverse Alpha**

0: Pixel difference is multiplied by alpha.

1: Pixel difference is multiplied by 1 - alpha.

- **GAEN: Blender Global Alpha Enable**

0: Global alpha blending coefficient is disabled.

1: Global alpha blending coefficient is enabled.

- **LAEN: Blender Local Alpha Enable**

0: Local alpha blending coefficient is disabled.

1: Local alpha blending coefficient is enabled.

- **OVR: Blender Overlay Layer Enable**

0: Overlay pixel color is set to the default overlay pixel color.

1: Overlay pixel color is set to the DMA channel pixel color.

- **DMA: Blender DMA Layer Enable**

0: The default color is used on the Overlay 1 Layer.

1: The DMA channel retrieves the pixels stream from the memory.

- **REP: Use Replication logic to expand RGB color to 24 bits**

0: When the selected pixel depth is less than 24 bpp the pixel is shifted and least significant bits are set to 0.

1: When the selected pixel depth is less than 24 bpp the pixel is shifted and the least significant bit replicates the msb.

- **DSTKEY: Destination Chroma Keying**

0: Source Chroma keying is enabled.

1: Destination Chroma keying is used.

- **GA: Blender Global Alpha**

Global alpha blender for the current layer.

32.7.76 High End Overlay Layer Channel Enable Register

Name: LCDC_HEOCHER

Address: 0xF0030340

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
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

- **CHEN: Channel Enable Register**

0: No effect.

1: Enables the DMA channel.

- **UPDATEEN: Update Overlay Attributes Enable Register**

0: No effect.

1: Updates windows attributes on the next start of frame.

- **A2QEN: Add to Queue Enable Register**

When set to one, it indicates that a valid descriptor has been written to memory, its memory location should be written to the DMA head pointer. The A2QSR status bit is set to one, and it is reset by hardware as soon as the descriptor pointed out by the DMA head pointer is added to the list.

32.7.77 High End Overlay Layer Channel Disable Register

Name: LCDC_HEOCHDR

Address: 0xF0030344

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
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

- **CHDIS: Channel Disable Register**

When set to one this field disables the layer at the end of the current frame. The frame is completed.

- **CHRST: Channel Reset Register**

When set to one this field resets the layer immediately. The frame is aborted.

32.7.78 High End Overlay Layer Channel Status Register

Name: LCDC_HEOCHSR

Address: 0xF0030348

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
–	–	–	–	–	A2QSR	UPDATESR	CHSR

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will be updated on the next frame.

- **A2QSR: Add To Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

32.7.79 High End Overlay Layer Interrupt Enable Register

Name: LCDC_HEOIER

Address: 0xF003034C

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	VOVR	VDONE	VADD	VDSCR	VDMA	–	–
15	14	13	12	11	10	9	8
–	UOVR	UDONE	UADD	UDSCR	UDMA	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **UDMA: End of DMA Transfer for U or UV Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **UDSCR: Descriptor Loaded for U or UV Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **UADD: Head Descriptor Loaded for U or UV Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **UDONE: End of List for U or UV Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **UOVR: Overflow for U or UV Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **VDMA: End of DMA for V Chrominance Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **VDSCR: Descriptor Loaded for V Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **VADD: Head Descriptor Loaded for V Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **VDONE: End of List for V Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **VOVR: Overflow for V Chrominance Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

32.7.80 High End Overlay Layer Interrupt Disable Register

Name: LCDC_HEOIDR

Address: 0xF0030350

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	VOVR	VDONE	VADD	VDSCR	VDMA	–	–
15	14	13	12	11	10	9	8
–	UOVR	UDONE	UADD	UDSCR	UDMA	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DSCR: Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DONE: End of List Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **OVR: Overflow Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **UDMA: End of DMA Transfer for U or UV Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **UDSCR: Descriptor Loaded for U or UV Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **UADD: Head Descriptor Loaded for U or UV Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **UDONE: End of List Interrupt for U or UV Chrominance Component Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **UOVR: Overflow Interrupt for U or UV Chrominance Component Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **VDMA: End of DMA Transfer for V Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **VDSCR: Descriptor Loaded for V Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **VADD: Head Descriptor Loaded for V Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **VDONE: End of List for V Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **VOVR: Overflow for V Chrominance Component Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

32.7.81 High End Overlay Layer Interrupt Mask Register

Name: LCDC_HEOIMR

Address: 0xF0030354

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	VOVR	VDONE	VADD	VDSCR	VDMA	–	–
15	14	13	12	11	10	9	8
–	UOVR	UDONE	UADD	UDSCR	UDMA	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **UDMA: End of DMA Transfer for U or UV Chrominance Component Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **UDSCR: Descriptor Loaded for U or UV Chrominance Component Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **UADD: Head Descriptor Loaded for U or UV Chrominance Component Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **UDONE: End of List for U or UV Chrominance Component Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **UOVR: Overflow for U Chrominance Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **VDMA: End of DMA Transfer for V Chrominance Component Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **VDSCR: Descriptor Loaded for V Chrominance Component Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **VADD: Head Descriptor Loaded for V Chrominance Component Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **VDONE: End of List for V Chrominance Component Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **VOVR: Overflow for V Chrominance Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.82 High End Overlay Layer Interrupt Status Register

Name: LCDC_HEOISR

Address: 0xF0030358

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	VOVR	VDONE	VADD	VDSCR	VDMA	–	–
15	14	13	12	11	10	9	8
–	UOVR	UDONE	UADD	UDSCR	UDMA	–	–
7	6	5	4	3	2	1	0
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **DSCR: DMA Descriptor Loaded**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **ADD: Head Descriptor Loaded**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **DONE: End of List Detected**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **OVR: Overflow Detected**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

- **UDMA: End of DMA Transfer for U component**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **UDSCR: DMA Descriptor Loaded for U component**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **UADD: Head Descriptor Loaded for U component**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **UDONE: End of List Detected for U component**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **UOVR: Overflow Detected for U component**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

- **VDMA: End of DMA Transfer for V component**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **VDSCR: DMA Descriptor Loaded for V component**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **VADD: Head Descriptor Loaded for V component**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **VDONE: End of List Detected for V component**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **VOVR: Overflow Detected for V component**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

32.7.83 High End Overlay Layer Head Register

Name: LCDC_HEOHEAD

Address: 0xF003035C

Access: Read-write

Reset: 0x00000000

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

- **HEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.84 High End Overlay Layer Address Register

Name: LCDC_HEOADDR

Address: 0xF0030360

Access: Read-write

Reset: 0x00000000

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: DMA Transfer start Address**

Frame Buffer Base Address.

32.7.85 High End Overlay Layer Control Register

Name: LCDC_HEOCTRL

Address: 0xF0030364

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
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	LFETCH	DFETCH

- **DFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **LFETCH: Lookup Table Fetch Enable**

0: Lookup Table DMA fetch is disabled.

1: Lookup Table DMA fetch is enabled.

- **DMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **DSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **ADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **DONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.86 High End Overlay Layer Next Register

Name: LCDC_HEONEXT

Address: 0xF0030368

Access: Read-write

Reset: 0x00000000

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

- **NEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.87 High End Overlay Layer U-UV Head Register

Name: LCDC_HEOUHEAD

Address: 0xF003036C

Access: Read-write

Reset: 0x00000000

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

- **UHEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.88 High End Overlay Layer U-UV Address Register

Name: LCDC_HEOUADDR

Address: 0xF0030370

Access: Read-write

Reset: 0x00000000

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

- **UADDR: DMA Transfer Start Address for U or UV Chrominance**

U or UV frame buffer address.

32.7.89 High End Overlay Layer U-UV Control Register

Name: LCDC_HEOUCTRL

Address: 0xF0030374

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
–	–	UDONEIEN	UADDIEN	UDSCRIEN	UDMAIEN	–	UDFETCH

- **UDFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **UDMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **UDSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **UADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **UDONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.90 High End Overlay Layer U-UV Next Register

Name: LCDC_HEOUNEXT

Address: 0xF0030378

Access: Read-write

Reset: 0x00000000

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

- **UNEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.91 High End Overlay Layer V Head Register

Name: LCDC_HEOVHEAD

Address: 0xF003037C

Access: Read-write

Reset: 0x00000000

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

- **VHEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.92 High End Overlay Layer V Address Register

Name: LCDC_HEOVADDR

Address: 0xF0030380

Access: Read-write

Reset: 0x00000000

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

- **VADDR: DMA Transfer Start Address for V Chrominance**

Frame Buffer Base Address.

32.7.93 High End Overlay Layer V Control Register

Name: LCDC_HEOVCTRL

Address: 0xF0030384

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
–	–	VDONEIEN	VADDIEN	VDSCRIEN	VDMAIEN	–	VDFETCH

- **VDFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **VDMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **VDSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **VADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **VDONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.94 High End Overlay Layer V Next Register

Name: LCDC_HEOVNEXT

Address: 0xF0030388

Access: Read-write

Reset: 0x00000000

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

- **VNEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.95 High End Overlay Layer Configuration 0 Register

Name: LCDC_HEOCFG0

Address: 0xF003038C

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
–	–	LOCKDIS	ROTDIS	–	–	–	DLBO
7	6	5	4	3	2	1	0
BLENUV		BLEN		–	–	–	SIF

- **SIF: Source Interface**

0: Base Layer data is retrieved through AHB interface 0.

1: Base Layer data is retrieved through AHB interface 1.

- **BLEN: AHB Burst Length**

Value	Name	Description
0	AHB_BLEN_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_BLEN_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_BLEN_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_BLEN_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **BLENUV: AHB Burst Length for U-V channel**

Value	Name	Description
0	AHB_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **DLBO: Defined Length Burst Only For Channel Bus Transaction.**

0: Undefined length INCR burst is used for a burst of 2 and 3 beats.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

- **ROTDIS: Hardware Rotation Optimization Disable**

0: Rotation optimization is enabled.

1: Rotation optimization is disabled.

- **LOCKDIS: Hardware Rotation Lock Disable**

0: AHB lock signal is asserted when a rotation is performed.

1: AHB lock signal is cleared when a rotation is performed.

32.7.96 High End Overlay Layer Configuration 1 Register

Name: LCDC_HEOCFG1

Address: 0xF0030390

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	DSCALEOPT	–	–	YUV422SWP	YUV422ROT
15	14	13	12	11	10	9	8
YUVMODE				–	–	CLUTMODE	
7	6	5	4	3	2	1	0
RGBMODE				–	–	YUVEN	CLUTEN

- **CLUTEN: Color Lookup Table Enable**

0: RGB mode is selected.

1: Color Lookup table is selected.

- **YUVEN: YUV Color Space Enable**

0: Color space is RGB

1: Color Space is YUV

- **RGBMODE: RGB input mode selection**

Value	Name	Description
0	12BPP_RGB_444	12 bpp RGB 444
1	16BPP_ARGB_4444	16 bpp ARGB 4444
2	16BPP_RGBA_4444	16 bpp RGBA 4444
3	16BPP_RGB_565	16 bpp RGB 565
4	16BPP_TRGB_1555	16 bpp TRGB 1555
5	18BPP_RGB_666	18 bpp RGB 666
6	18BPP_RGB_666PACKED	18 bpp RGB 666 PACKED
7	19BPP_TRGB_1666	19 bpp TRGB 1666
8	19BPP_TRGB_PACKED	19 bpp TRGB 1666 PACKED
9	24BPP_RGB_888	24 bpp RGB 888
10	24BPP_RGB_888_PACKED	24 bpp RGB 888 PACKED
11	25BPP_TRGB_1888	25 bpp TRGB 1888
12	32BPP_ARGB_8888	32 bpp ARGB 8888
13	32BPP_RGBA_8888	32 bpp RGBA 8888

- **CLUTMODE: Color Lookup table input mode selection**

Value	Name	Description
0	CLUT_1BPP	color lookup table mode set to 1 bit per pixel
1	CLUT_2BPP	color lookup table mode set to 2 bits per pixel
2	CLUT_4BPP	color lookup table mode set to 4 bits per pixel
3	CLUT_8BPP	color lookup table mode set to 8 bits per pixel

- **YUVMODE: YUV input mode selection**

Value	Name	Description
0	32BPP_AYCBCR	32 bpp AYCbCr 444
1	16BPP_YCBCR_MODE0	16 bpp Cr(n)Y(n+1)Cb(n)Y(n) 422
2	16BPP_YCBCR_MODE1	16 bpp Y(n+1)Cr(n)Y(n)Cb(n) 422
3	16BPP_YCBCR_MODE2	16 bpp Cb(n)Y(+1)Cr(n)Y(n) 422
4	16BPP_YCBCR_MODE3	16 bpp Y(n+1)Cb(n)Y(n)Cr(n) 422
5	16BPP_YCBCR_SEMIPLANAR	16 bpp Semiplanar 422 YCbCr
6	16BPP_YCBCR_PLANAR	16 bpp Planar 422 YCbCr
7	12BPP_YCBCR_SEMIPLANAR	12 bpp Semiplanar 420 YCbCr
8	12BPP_YCBCR_PLANAR	12 bpp Planar 420 YCbCr

- **YUV422ROT: YUV 4:2:2 Rotation**

When set to one, this bit indicates that the Chroma Upsampling kernel is configured to use the 4:2:2 Rotation Algorithm. This field is relevant only when a rotation angle of 90 degrees or 270 degrees is used.

- **YUV422SWP: YUV 4:2:2 SWAP**

When set to one, the Y component of the YUV 4:2:2 packed data stream are swapped.

- **DSCALEOPT: Down Scaling Bandwidth Optimization**

0: Scaler Optimization is disabled.

1: Scaler Optimization is enabled, only relevant pixels are retrieved from memory to fill the scaler filter.

32.7.97 High End Overlay Layer Configuration 2 Register

Name: LCDC_HEOCFG2

Address: 0xF0030394

Access: Read-write

Reset: 0x00000000

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

- **XPOS: Horizontal Window Position**

High End Overlay Horizontal window position.

- **YPOS: Vertical Window Position**

High End Overlay Vertical window position.

32.7.98 High End Overlay Layer Configuration 3 Register

Name: LCDC_HEOCFG3

Address: 0xF0030398

Access: Read-write

Reset: 0x00000000

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

- **XSIZE: Horizontal Window Size**

High End Overlay window width in pixels. The window width is set to (XSIZE+1).

The following constraint must be met:

$$XPOS + XSIZE \leq PPL$$

- **YSIZE: Vertical Window Size**

High End Overlay window height in pixels. The window height is set to (YSIZE+1).

The following constrain must be met:

$$YPOS + YSIZE \leq RPF$$

32.7.99 High End Overlay Layer Configuration 4 Register

Name: LCDC_HEOCFG4

Address: 0xF003039C

Access: Read-write

Reset: 0x00000000

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

- **XMEMSIZE: Horizontal image Size in Memory**

High End Overlay image width in pixels. The image width is set to (XMEMSIZE+1).

- **YMEMSIZE: Vertical image Size in Memory**

High End Overlay image height in pixels. The image height is set to (YMEMSIZE+1).

32.7.100High End Overlay Layer Configuration 5 Register

Name: LCDC_HEOCFG5

Address: 0xF00303A0

Access: Read-write

Reset: 0x00000000

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

- **XSTRIDE: Horizontal Stride**

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.101High End Overlay Layer Configuration 6 Register

Name: LCDC_HEOCFG6

Address: 0xF00303A4

Access: Read-write

Reset: 0x00000000

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

- **PSTRIDE: Pixel Stride**

PSTRIDE represents the memory offset, in bytes, between two pixels of the image memory.

32.7.102High End Overlay Layer Configuration 7 Register

Name: LCDC_HEOCFG7

Address: 0xF00303A8

Access: Read-write

Reset: 0x00000000

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

- **UVXSTRIDE: UV Horizontal Stride**

UVXSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.103High End Overlay Layer Configuration 8 Register

Name: LCDC_HEOCFG8

Address: 0xF00303AC

Access: Read-write

Reset: 0x00000000

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

- **UVPSTRIDE: UV Pixel Stride**

UVPSTRIDE represents the memory offset, in bytes, between two pixels of the image memory.

32.7.104High End Overlay Layer Configuration 9 Register

Name: LCDC_HEOCFG9

Address: 0xF00303B0

Access: Read-write

Reset: 0x00000000

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

- **RDEF: Red Default**

Default Red color when the High End Overlay DMA channel is disabled.

- **GDEF: Green Default**

Default Green color when the High End Overlay DMA channel is disabled.

- **BDEF: Blue Default**

Default Blue color when the High End Overlay DMA channel is disabled.

32.7.105High End Overlay Layer Configuration 10 Register

Name: LCDC_HEOCFG10

Address: 0xF00303B4

Access: Read-write

Reset: 0x00000000

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

- **RKEY: Red Color Component Chroma Key**

Reference Red chroma key used to match the Red color of the current overlay.

- **GKEY: Green Color Component Chroma Key**

Reference Green chroma key used to match the Green color of the current overlay.

- **BKEY: Blue Color Component Chroma Key**

Reference Blue chroma key used to match the Blue color of the current overlay.

32.7.106High End Overlay Layer Configuration 11 Register

Name: LCDC_HEOCFG11

Address: 0xF00303B8

Access: Read-write

Reset: 0x00000000

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

- **RMASK: Red Color Component Chroma Key Mask**

Red Mask used when the compare function is used. If a bit is set then this bit is compared.

- **GMASK: Green Color Component Chroma Key Mask**

Green Mask used when the compare function is used. If a bit is set then this bit is compared.

- **BMASK: Blue Color Component Chroma Key Mask**

Blue Mask used when the compare function is used. If a bit is set then this bit is compared.

32.7.107High End Overlay Layer Configuration 12 Register

Name: LCDC_HEOCFG12

Address: 0xF00303BC

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
GA							
15	14	13	12	11	10	9	8
–	–	–	VIDPRI	–	DSTKEY	REP	DMA
7	6	5	4	3	2	1	0
OVR	LAEN	GAEN	REVALPHA	ITER	ITER2BL	INV	CRKEY

- **CRKEY: Blender Chroma Key Enable**

0: Chroma key matching is disabled.

1: Chroma key matching is enabled.

- **INV: Blender Inverted Blender Output Enable**

0: Iterated pixel is the blended pixel.

1: Iterated pixel is the inverted pixel.

- **ITER2BL: Blender Iterated Color Enable**

0: Final adder stage operand is set to 0.

1: Final adder stage operand is set to the iterated pixel value.

- **ITER: Blender Use Iterated Color**

0: Pixel difference is set to 0.

1: Pixel difference is set to the iterated pixel value.

- **REVALPHA: Blender Reverse Alpha**

0: Pixel difference is multiplied by alpha.

1: Pixel difference is multiplied by 1 - alpha.

- **GAEN: Blender Global Alpha Enable**

0: Global alpha blending coefficient is disabled.

1: Global alpha blending coefficient is enabled.

- **LAEN: Blender Local Alpha Enable**

0: Local alpha blending coefficient is disabled.

1: Local alpha blending coefficient is enabled.

- **OVR: Blender Overlay Layer Enable**

0: Overlay pixel color is set to the default overlay pixel color.

1: Overlay pixel color is set to the DMA channel pixel color.

- **DMA: Blender DMA Layer Enable**

0: The default color is used on the Overlay 1 Layer.

1: The DMA channel retrieves the pixels stream from the memory.

- **REP: Use Replication logic to expand RGB color to 24 bits**

0: When the selected pixel depth is less than 24 bpp the pixel is shifted and least significant bits are set to 0.

1: When the selected pixel depth is less than 24 bpp the pixel is shifted and the least significant bit replicates the msb.

- **DSTKEY: Destination Chroma Keying**

0: Source Chroma keying is enabled.

1: Destination Chroma keying is used.

- **VIDPRI: Video Priority**

0: OVR1 layer is above HEO layer.

1: OVR1 layer is below HEO layer.

- **GA: Blender Global Alpha**

Global alpha blender for the current layer.

32.7.108High End Overlay Layer Configuration 13 Register

Name: LCDC_HEOCFG13

Address: 0xF00303C0

Access: Read-write

Reset: 0x00000000

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

- **SCALEN: Hardware Scaler Enable**

0: Scaler is disabled

1: Scaler is enabled.

- **YFACTOR: Vertical Scaling Factor**

Scaler Vertical Factor.

- **XFACTOR: Horizontal Scaling Factor**

Scaler Horizontal Factor.

32.7.109High End Overlay Layer Configuration 14 Register

Name: LCDC_HEOCFG14

Address: 0xF00303C4

Access: Read-write

Reset: 0x00000000

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

- **CSCRY: Color Space Conversion Y coefficient for Red Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCRU: Color Space Conversion U coefficient for Red Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCRV: Color Space Conversion V coefficient for Red Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCYOFF: Color Space Conversion Offset**

0: Offset is set to 0

1: Offset is set to 16

32.7.110High End Overlay Layer Configuration 15 Register

Name: LCDC_HEOCFG15

Address: 0xF00303C8

Access: Read-write

Reset: 0x00000000

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

- **CSCGY: Color Space Conversion Y coefficient for Green Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCGU: Color Space Conversion U coefficient for Green Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCGV: Color Space Conversion V coefficient for Green Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCUOFF: Color Space Conversion Offset**

0: Offset is set to 0

1: Offset is set to 128

32.7.111High End Overlay Layer Configuration 16 Register

Name: LCDC_HEOCFG16

Address: 0xF00303CC

Access: Read-write

Reset: 0x00000000

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

- **CSCBY: Color Space Conversion Y coefficient for Blue Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCBU: Color Space Conversion U coefficient for Blue Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCBV: Color Space Conversion V coefficient for Blue Component 1:2:7 format**

Color Space Conversion coefficient format is 1 sign bit, 2 magnitude bits and 7 fractional bits.

- **CSCVOFF: Color Space Conversion Offset**

0: Offset is set to 0

1: Offset is set to 128

32.7.112High End Overlay Layer Configuration 17 Register

Name: LCDC_HEOCFG17

Address: 0xF00303D0

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI0COEFF3							
23	22	21	20	19	18	17	16
XPHI0COEFF2							
15	14	13	12	11	10	9	8
XPHI0COEFF1							
7	6	5	4	3	2	1	0
XPHI0COEFF0							

- **XPHI0COEFF0: Horizontal Coefficient for phase 0 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI0COEFF1: Horizontal Coefficient for phase 0 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI0COEFF2: Horizontal Coefficient for phase 0 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI0COEFF3: Horizontal Coefficient for phase 0 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.113High End Overlay Layer Configuration 18 Register

Name: LCDC_HEOCFG18

Address: 0xF00303D4

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
XPHI0COEFF4							

- **XPHI0COEFF4: Horizontal Coefficient for phase 0 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.114High End Overlay Layer Configuration 19 Register

Name: LCDC_HEOCFG19

Address: 0xF00303D8

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI1COEFF3							
23	22	21	20	19	18	17	16
XPHI1COEFF2							
15	14	13	12	11	10	9	8
XPHI1COEFF1							
7	6	5	4	3	2	1	0
XPHI1COEFF0							

- **XPHI1COEFF0: Horizontal Coefficient for phase 1 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI1COEFF1: Horizontal Coefficient for phase 1 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI1COEFF2: Horizontal Coefficient for phase 1 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI1COEFF3: Horizontal Coefficient for phase 1 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.115High End Overlay Layer Configuration 20 Register

Name: LCDC_HEOCFG20

Address: 0xF00303DC

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
XPHI1COEFF4							

- **XPHI1COEFF4: Horizontal Coefficient for phase 1 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.116High End Overlay Layer Configuration 21 Register

Name: LCDC_HEOCFG21

Address: 0xF00303E0

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI2COEFF3							
23	22	21	20	19	18	17	16
XPHI2COEFF2							
15	14	13	12	11	10	9	8
XPHI2COEFF1							
7	6	5	4	3	2	1	0
XPHI2COEFF0							

- **XPHI2COEFF0: Horizontal Coefficient for phase 2 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI2COEFF1: Horizontal Coefficient for phase 2 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI2COEFF2: Horizontal Coefficient for phase 2 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI2COEFF3: Horizontal Coefficient for phase 2 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.117High End Overlay Layer Configuration 22 Register

Name: LCDC_HEOCFG22

Address: 0xF00303E4

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
XPHI2COEFF4							

- **XPHI2COEFF4: Horizontal Coefficient for phase 2 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.118High End Overlay Layer Configuration 23 Register

Name: LCDC_HEOCFG23

Address: 0xF00303E8

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI3COEFF3							
23	22	21	20	19	18	17	16
XPHI3COEFF2							
15	14	13	12	11	10	9	8
XPHI3COEFF1							
7	6	5	4	3	2	1	0
XPHI3COEFF0							

- **XPHI3COEFF0: Horizontal Coefficient for phase 3 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI3COEFF1: Horizontal Coefficient for phase 3 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI3COEFF2: Horizontal Coefficient for phase 3 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI3COEFF3: Horizontal Coefficient for phase 3 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.119High End Overlay Layer Configuration 24 Register

Name: LCDC_HEOCFG24

Address: 0xF00303EC

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
XPHI3COEFF4							

- **XPHI3COEFF4: Horizontal Coefficient for phase 3 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.120High End Overlay Layer Configuration 25 Register

Name: LCDC_HEOCFG25

Address: 0xF00303F0

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI4COEFF3							
23	22	21	20	19	18	17	16
XPHI4COEFF2							
15	14	13	12	11	10	9	8
XPHI4COEFF1							
7	6	5	4	3	2	1	0
XPHI4COEFF0							

- **XPHI4COEFF0: Horizontal Coefficient for phase 4 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI4COEFF1: Horizontal Coefficient for phase 4 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI4COEFF2: Horizontal Coefficient for phase 4 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI4COEFF3: Horizontal Coefficient for phase 4 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.121High End Overlay Layer Configuration 26 Register

Name: LCDC_HEOCFG26

Address: 0xF00303F4

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
XPHI4COEFF4							

- **XPHI4COEFF4: Horizontal Coefficient for phase 4 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.122High End Overlay Layer Configuration 27 Register

Name: LCDC_HEOCFG27

Address: 0xF00303F8

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI5COEFF3							
23	22	21	20	19	18	17	16
XPHI5COEFF2							
15	14	13	12	11	10	9	8
XPHI5COEFF1							
7	6	5	4	3	2	1	0
XPHI5COEFF0							

- **XPHI5COEFF0: Horizontal Coefficient for phase 5 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI5COEFF1: Horizontal Coefficient for phase 5 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI5COEFF2: Horizontal Coefficient for phase 5 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI5COEFF3: Horizontal Coefficient for phase 5 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.123High End Overlay Layer Configuration 28 Register

Name: LCDC_HEOCFG28

Address: 0xF00303FC

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
XPHI5COEFF4							

- **XPHI5COEFF4: Horizontal Coefficient for phase 5 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.124High End Overlay Layer Configuration 29 Register

Name: LCDC_HEOCFG29

Address: 0xF0030400

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI6COEFF3							
23	22	21	20	19	18	17	16
XPHI6COEFF2							
15	14	13	12	11	10	9	8
XPHI6COEFF1							
7	6	5	4	3	2	1	0
XPHI6COEFF0							

- **XPHI6COEFF0: Horizontal Coefficient for phase 6 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI6COEFF1: Horizontal Coefficient for phase 6 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI6COEFF2: Horizontal Coefficient for phase 6 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI6COEFF3: Horizontal Coefficient for phase 6 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.125High End Overlay Layer Configuration 30 Register

Name: LCDC_HEOCFG30

Address: 0xF0030404

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
XPHI6COEFF4							

- **XPHI6COEFF4: Horizontal Coefficient for phase 6 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.126High End Overlay Layer Configuration 31 Register

Name: LCDC_HEOCFG31

Address: 0xF0030408

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
XPHI7COEFF3							
23	22	21	20	19	18	17	16
XPHI7COEFF2							
15	14	13	12	11	10	9	8
XPHI7COEFF1							
7	6	5	4	3	2	1	0
XPHI7COEFF0							

- **XPHI7COEFF0: Horizontal Coefficient for phase 7 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI7COEFF1: Horizontal Coefficient for phase 7 tap 1**

Coefficient format is 1 sign bit and 7 fractional bits.

- **XPHI7COEFF2: Horizontal Coefficient for phase 7 tap 2**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **XPHI7COEFF3: Horizontal Coefficient for phase 7 tap 3**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.127High End Overlay Layer Configuration 32 Register

Name: LCDC_HEOCFG32

Address: 0xF003040C

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
XPHI7COEFF4							

- **XPHI7COEFF4: Horizontal Coefficient for phase 7 tap 4**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.128High End Overlay Layer Configuration 33 Register

Name: LCDC_HEOCFG33

Address: 0xF0030410

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI0COEFF2							
15	14	13	12	11	10	9	8
YPHI0COEFF1							
7	6	5	4	3	2	1	0
YPHI0COEFF0							

- **YPHI0COEFF0: Vertical Coefficient for phase 0 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI0COEFF1: Vertical Coefficient for phase 0 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI0COEFF2: Vertical Coefficient for phase 0 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.129High End Overlay Layer Configuration 34 Register

Name: LCDC_HEOCFG34

Address: 0xF0030414

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI1COEFF2							
15	14	13	12	11	10	9	8
YPHI1COEFF1							
7	6	5	4	3	2	1	0
YPHI1COEFF0							

- **YPHI1COEFF0: Vertical Coefficient for phase 1 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI1COEFF1: Vertical Coefficient for phase 1 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI1COEFF2: Vertical Coefficient for phase 1 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.130High End Overlay Layer Configuration 35 Register

Name: LCDC_HEOCFG35

Address: 0xF0030418

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI2COEFF2							
15	14	13	12	11	10	9	8
YPHI2COEFF1							
7	6	5	4	3	2	1	0
YPHI2COEFF0							

- **YPHI2COEFF0: Vertical Coefficient for phase 2 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI2COEFF1: Vertical Coefficient for phase 2 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI2COEFF2: Vertical Coefficient for phase 2 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.131High End Overlay Layer Configuration 36 Register

Name: LCDC_HEOCFG36

Address: 0xF003041C

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI3COEFF2							
15	14	13	12	11	10	9	8
YPHI3COEFF1							
7	6	5	4	3	2	1	0
YPHI3COEFF0							

- **YPHI3COEFF0: Vertical Coefficient for phase 3 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI3COEFF1: Vertical Coefficient for phase 3 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI3COEFF2: Vertical Coefficient for phase 3 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.132High End Overlay Layer Configuration 37 Register

Name: LCDC_HEOCFG37

Address: 0xF0030420

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI4COEFF2							
15	14	13	12	11	10	9	8
YPHI4COEFF1							
7	6	5	4	3	2	1	0
YPHI4COEFF0							

- **YPHI4COEFF0: Vertical Coefficient for phase 4 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI4COEFF1: Vertical Coefficient for phase 4 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI4COEFF2: Vertical Coefficient for phase 4 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.133High End Overlay Layer Configuration 38 Register

Name: LCDC_HEOCFG38

Address: 0xF0030424

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI5COEFF2							
15	14	13	12	11	10	9	8
YPHI5COEFF1							
7	6	5	4	3	2	1	0
YPHI5COEFF0							

- **YPHI5COEFF0: Vertical Coefficient for phase 5 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI5COEFF1: Vertical Coefficient for phase 5 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI5COEFF2: Vertical Coefficient for phase 5 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.134High End Overlay Layer Configuration 39 Register

Name: LCDC_HEOCFG39

Address: 0xF0030428

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI6COEFF2							
15	14	13	12	11	10	9	8
YPHI6COEFF1							
7	6	5	4	3	2	1	0
YPHI6COEFF0							

- **YPHI6COEFF0: Vertical Coefficient for phase 6 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI6COEFF1: Vertical Coefficient for phase 6 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI6COEFF2: Vertical Coefficient for phase 6 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.135High End Overlay Layer Configuration 40 Register

Name: LCDC_HEOCFG40

Address: 0xF003042C

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
YPHI7COEFF2							
15	14	13	12	11	10	9	8
YPHI7COEFF1							
7	6	5	4	3	2	1	0
YPHI7COEFF0							

- **YPHI7COEFF0: Vertical Coefficient for phase 7 tap 0**

Coefficient format is 1 sign bit and 7 fractional bits.

- **YPHI7COEFF1: Vertical Coefficient for phase 7 tap 1**

Coefficient format is 1 magnitude bit and 7 fractional bits.

- **YPHI7COEFF2: Vertical Coefficient for phase 7 tap 2**

Coefficient format is 1 sign bit and 7 fractional bits.

32.7.136High End Overlay Layer Configuration 41 Register

Name: LCDC_HEOCFG41

Address: 0xF0030430

Access: Read-write

Reset: 0x00000000

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

- **XPHIDEF: Horizontal Filter Phase Offset**

XPHIDEF defines the index of the first coefficient set used when the horizontal resampling operation is started.

- **YPHIDEF: Vertical Filter Phase Offset**

XPHIDEF defines the index of the first coefficient set used when the vertical resampling operation is started.

32.7.137Hardware Cursor Layer Channel Enable Register

Name: LCDC_HCRCHER

Address: 0xF0030440

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
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

- **CHEN: Channel Enable Register**

0: No effect.

1: Enables the DMA channel.

- **UPDATEEN: Update Overlay Attributes Enable Register**

0: No effect.

1: Updates windows attributes on the next start of frame.

- **A2QEN: Add to Queue Enable Register**

When set to one, it indicates that a valid descriptor has been written to memory, its memory location should be written to the DMA head pointer. The A2QSR status bit is set to one, and it is reset by hardware as soon as the descriptor pointed out by the DMA head pointer is added to the list.

32.7.138Hardware Cursor Layer Channel Disable Register

Name: LCDC_HCRCHDR

Address: 0xF0030444

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
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

- **CHDIS: Channel Disable Register**

When set to one this field disables the layer at the end of the current frame. The frame is completed.

- **CHRST: Channel Reset Register**

When set to one this field resets the layer immediately. The frame is aborted.

32.7.139Hardware Cursor Layer Channel Status Register

Name: LCDC_HCRCHSR

Address: 0xF0030448

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
–	–	–	–	–	A2QSR	UPDATESR	CHSR

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will be updated on the next frame.

- **A2QSR: Add To Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

32.7.140Hardware Cursor Layer Interrupt Enable Register

Name: LCDC_HCRIER

Address: 0xF003044C

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

32.7.141Hardware Cursor Layer Interrupt Disable Register

Name: LCDC_HCRIDR

Address: 0xF0030450

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DSCR: Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DONE: End of List Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **OVR: Overflow Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

32.7.142Hardware Cursor Layer Interrupt Mask Register

Name: LCDC_HCRIMR

Address: 0xF0030454

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR: Overflow Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.143Hardware Cursor Layer Interrupt Status Register

Name: LCDC_HCRISR

Address: 0xF0030458

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
–	OVR	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **DSCR: DMA Descriptor Loaded**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **ADD: Head Descriptor Loaded**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **DONE: End of List Detected**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **OVR: Overflow Detected**

When set to one this flag indicates that an Overflow has occurred. This flag is reset after a read operation.

32.7.144Hardware Cursor Layer Head Register

Name: LCDC_HCRHEAD

Address: 0xF003045C

Access: Read-write

Reset: 0x00000000

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

- **HEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.145Hardware Cursor Layer Address Register

Name: LCDC_HCRADDR

Address: 0xF0030460

Access: Read-write

Reset: 0x00000000

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: DMA Transfer start address**

Frame buffer start address.

32.7.146Hardware Cursor Layer Control Register

Name: LCDC_HCRCTRL

Address: 0xF0030464

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
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	LFETCH	DFETCH

- **DFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **LFETCH: Lookup Table Fetch Enable**

0: Lookup Table DMA fetch is disabled.

1: Lookup Table DMA fetch is enabled.

- **DMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **DSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **ADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **DONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.147Hardware Cursor Layer Next Register

Name: LCDC_HCRNEXT

Address: 0xF0030468

Access: Read-write

Reset: 0x00000000

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

- **NEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.148Hardware Cursor Layer Configuration 0 Register

Name: LCDC_HCRCFG0

Address: 0xF003046C

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
–	–	–	–	–	–	–	DLBO
7	6	5	4	3	2	1	0
–	–	BLEN		–	–	–	SIF

- **SIF: Source Interface**

0: Base Layer data is retrieved through AHB interface 0.

1: Base Layer data is retrieved through AHB interface 1.

- **BLEN: AHB Burst Length**

Value	Name	Description
0	AHB_BLEN_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_BLEN_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_BLEN_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_BLEN_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **DLBO: Defined Length Burst Only for Channel Bus Transaction.**

0: Undefined length INCR burst is used for a burst of 2 and 3 beats.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

32.7.149Hardware Cursor Layer Configuration 1 Register

Name: LCDC_HCRCFG1

Address: 0xF0030470

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
–				–	–	CLUTMODE	
7	6	5	4	3	2	1	0
RGBMODE				–	–	–	CLUTEN

- **CLUTEN: Color Lookup Table Enable**

0: RGB mode is selected.

1: Color Lookup table is selected.

- **RGBMODE: RGB input mode selection**

Value	Name	Description
0	12BPP_RGB_444	12 bpp RGB 444
1	16BPP_ARGB_4444	16 bpp ARGB 4444
2	16BPP_RGBA_4444	16 bpp RGBA 4444
3	16BPP_RGB_565	16 bpp RGB 565
4	16BPP_TRGB_1555	16 bpp TRGB 1555
5	18BPP_RGB_666	18 bpp RGB 666
6	18BPP_RGB_666PACKED	18 bpp RGB 666 PACKED
7	19BPP_TRGB_1666	19 bpp TRGB 1666
8	19BPP_TRGB_PACKED	19 bpp TRGB 1666 PACKED
9	24BPP_RGB_888	24 bpp RGB 888
10	24BPP_RGB_888_PACKED	24 bpp RGB 888 PACKED
11	25BPP_TRGB_1888	25 bpp TRGB 1888
12	32BPP_ARGB_8888	32 bpp ARGB 8888
13	32BPP_RGBA_8888	32 bpp RGBA 8888

- **CLUTMODE: Color Lookup table input mode selection**

Value	Name	Description
0	CLUT_1BPP	color lookup table mode set to 1 bit per pixel
1	CLUT_2BPP	color lookup table mode set to 2 bits per pixel
2	CLUT_4BPP	color lookup table mode set to 4 bits per pixel
3	CLUT_8BPP	color lookup table mode set to 8 bits per pixel

32.7.150Hardware Cursor Layer Configuration 2 Register

Name: LCDC_HCRCFG2

Address: 0xF0030474

Access: Read-write

Reset: 0x00000000

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

- **XPOS: Horizontal Window Position**

Hardware Cursor Horizontal window position.

- **YPOS: Vertical Window Position**

Hardware Cursor Vertical window position.

32.7.151Hardware Cursor Layer Configuration 3 Register

Name: LCDC_HCRCFG3

Address: 0xF0030478

Access: Read-write

Reset: 0x00000000

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

- **XSIZE: Horizontal Window Size**

Hardware cursor width is limited to 128 pixels

Hardware Cursor window width in pixels. The window width is set to (XSIZE+1).

The following constraint must be met:

$$XPOS + XSIZE \leq PPL$$

- **YSIZE: Vertical Window Size**

Hardware cursor height is limited to 128 pixels

Hardware Cursor window height in pixels. The window height is set to (YSIZE+1).

The following constrain must be met:

$$YPOS + YSIZE \leq RPF$$

32.7.152Hardware Cursor Layer Configuration 4 Register

Name: LCDC_HCRCFG4

Address: 0xF003047C

Access: Read-write

Reset: 0x00000000

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

- **XSTRIDE: Horizontal Stride**

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.153Hardware Cursor Layer Configuration 6 Register

Name: LCDC_HCRCFG6

Address: 0xF0030484

Access: Read-write

Reset: 0x00000000

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

- **RDEF: Red Default**

Default Red color when the Hardware Cursor DMA channel is disabled.

- **GDEF: Green Default**

Default Green color when the Hardware Cursor DMA channel is disabled.

- **BDEF: Blue Default**

- **Default Blue color when the Hardware Cursor DMA channel is disabled.**

32.7.154Hardware Cursor Layer Configuration 7 Register

Name: LCDC_HCRCFG7

Address: 0xF0030488

Access: Read-write

Reset: 0x00000000

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

- **RKEY: Red Color Component Chroma Key**

Reference Red chroma key used to match the Red color of the current overlay.

- **GKEY: Green Color Component Chroma Key**

Reference Green chroma key used to match the Green color of the current overlay.

- **BKEY: Blue Color Component Chroma Key**

Reference Blue chroma key used to match the Blue color of the current overlay.

32.7.155Hardware Cursor Layer Configuration 8 Register

Name: LCDC_HCRCFG8

Address: 0xF003048C

Access: Read-write

Reset: 0x00000000

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

- **RMASK: Red Color Component Chroma Key Mask**

Red Mask used when the compare function is used. If a bit is set then this bit is compared.

- **GMASK: Green Color Component Chroma Key Mask**

Green Mask used when the compare function is used. If a bit is set then this bit is compared.

- **BMASK: Blue Color Component Chroma Key Mask**

Blue Mask used when the compare function is used. If a bit is set then this bit is compared.

32.7.156Hardware Cursor Layer Configuration 9 Register

Name: LCDC_HCRCFG9

Address: 0xF0030490

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
GA							
15	14	13	12	11	10	9	8
–	–	–	–	–	DSTKEY	REP	DMA
7	6	5	4	3	2	1	0
OVR	LAEN	GAEN	REVALPHA	ITER	ITER2BL	INV	CRKEY

- **CRKEY: Blender Chroma Key Enable**

0: Chroma key matching is disabled.

1: Chroma key matching is enabled.

- **INV: Blender Inverted Blender Output Enable**

0: Iterated pixel is the blended pixel.

1: Iterated pixel is the inverted pixel.

- **ITER2BL: Blender Iterated Color Enable**

0: Final adder stage operand is set to 0.

1: Final adder stage operand is set to the iterated pixel value.

- **ITER: Blender Use Iterated Color**

0: Pixel difference is set to 0.

1: Pixel difference is set to the iterated pixel value.

- **REVALPHA: Blender Reverse Alpha**

0: Pixel difference is multiplied by alpha.

1: Pixel difference is multiplied by 1 - alpha.

- **GAEN: Blender Global Alpha Enable**

0: Global alpha blending coefficient is disabled.

1: Global alpha blending coefficient is enabled.

- **LAEN: Blender Local Alpha Enable**

0: Local alpha blending coefficient is disabled.

1: Local alpha blending coefficient is enabled.

- **OVR: Blender Overlay Layer Enable**

0: Overlay pixel color is set to the default overlay pixel color.

1: Overlay pixel color is set to the DMA channel pixel color.

- **DMA: Blender DMA Layer Enable**

0: The default color is used on the Overlay 1 Layer.

1: The DMA channel retrieves the pixels stream from the memory.

- **REP: Use Replication logic to expand RGB color to 24 bits**

0: When the selected pixel depth is less than 24 bpp the pixel is shifted and least significant bits are set to 0.

1: When the selected pixel depth is less than 24 bpp the pixel is shifted and the least significant bit replicates the msb.

- **DSTKEY: Destination Chroma Keying**

0: Source Chroma keying is enabled.

1: Destination Chroma keying is used.

- **GA: Blender Global Alpha**

Global alpha blender for the current layer.

32.7.157Post Processing Layer Channel Enable Register

Name: LCDC_PPCHER

Address: 0xF0030540

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
–	–	–	–	–	A2QEN	UPDATEEN	CHEN

- **CHEN: Channel Enable Register**

0: No effect.

1: Enables the DMA channel.

- **UPDATEEN: Update Overlay Attributes Enable Register**

0: No effect.

1: Updates windows attributes on the next start of frame.

- **A2QEN: Add to Queue Enable Register**

When set to one, it indicates that a valid descriptor has been written to memory, its memory location should be written to the DMA head pointer. The A2QSR status bit is set to one, and it is reset by hardware as soon as the descriptor pointed out by the DMA head pointer is added to the list.

32.7.158Post Processing Layer Channel Disable Register

Name: LCDC_PPCHDR

Address: 0xF0030544

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
–	–	–	–	–	–	–	CHRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CHDIS

- **CHDIS: Channel Disable Register**

When set to one this field disables the layer at the end of the current frame.

- **CHRST: Channel Reset Register**

When set to one this field disables the layer at the end of the current frame.

32.7.159Post Processing Layer Channel Status Register

Name: LCDC_PPCHSR

Address: 0xF0030548

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
–	–	–	–	–	A2QSR	UPDATESR	CHSR

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will update on the next Frame.

- **A2QSR: Add To Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

32.7.160Post Processing Layer Interrupt Enable Register

Name: LCDC_PPIER

Address: 0xF003054C

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
–	–	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Enable Register**

0: No effect.

1: Interrupt source is enabled.

32.7.161Post Processing Layer Interrupt Disable Register

Name: LCDC_PPIDR

Address: 0xF0030550

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
–	–	DONE	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DSCR: Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **ADD: Head Descriptor Loaded Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

- **DONE: End of List Interrupt Disable Register**

0: No effect.

1: Interrupt source is disabled.

32.7.162Post Processing Layer Interrupt Mask Register

Name: LCDC_PPIMR

Address: 0xF0030554

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	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DSCR: Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **ADD: Head Descriptor Loaded Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DONE: End of List Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

32.7.163Post Processing Layer Interrupt Status Register

Name: LCDC_PPISR

Address: 0xF0030558

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	ADD	DSCR	DMA	–	–

- **DMA: End of DMA Transfer**

When set to one this flag indicates that an End of Transfer has been detected. This flag is reset after a read operation.

- **DSCR: DMA Descriptor Loaded**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **ADD: Head Descriptor Loaded**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **DONE: End Of List Detected**

When set to one this flag indicates that a End of List condition has occurred. This flag is reset after a read operation.

32.7.164Post Processing Layer Head Register

Name: LCDC_PPHEAD

Address: 0xF003055C

Access: Read-Write

Reset: 0x00000000

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

- **HEAD: DMA Head Pointer**

The Head Pointer points to a new descriptor.

32.7.165Post Processing Layer Address Register

Name: LCDC_PPADDR

Address: 0xF0030560

Access: Read-Write

Reset: 0x00000000

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: DMA Transfer start address**

Post Processing Destination frame buffer address.

32.7.166Post Processing Layer Control Register

Name: LCDC_PPCTRL

Address: 0xF0030564

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
–	–	DONEIEN	ADDIEN	DSCRIEN	DMAIEN	–	DFETCH

- **DFETCH: Transfer Descriptor Fetch Enable**

0: Transfer Descriptor fetch is disabled.

1: Transfer Descriptor fetch is enabled.

- **DMAIEN: End of DMA Transfer Interrupt Enable**

0: DMA transfer completed interrupt is enabled.

1: DMA transfer completed interrupt is disabled.

- **DSCRIEN: Descriptor Loaded Interrupt Enable**

0: Transfer descriptor loaded interrupt is enabled.

1: Transfer descriptor loaded interrupt is disabled.

- **ADDIEN: Add Head Descriptor to Queue Interrupt Enable**

0: Transfer descriptor added to queue interrupt is enabled.

1: Transfer descriptor added to queue interrupt is enabled.

- **DONEIEN: End of List Interrupt Enable**

0: End of list interrupt is disabled.

1: End of list interrupt is enabled.

32.7.167Post Processing Layer Next Register

Name: LCDC_PPNEXT

Address: 0xF0030568

Access: Read-Write

Reset: 0x00000000

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

- **NEXT: DMA Descriptor Next Address**

The transfer descriptor address must be aligned on a 64-bit boundary.

32.7.168Post Processing Layer Configuration 0 Register

Name: LCDC_PPCFG0

Address: 0xF003056C

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
–	–	–	–	–	–	–	DLBO
7	6	5	4	3	2	1	0
–	–	BLEN		–	–	–	SIF

- **SIF: Source Interface**

0: Base Layer data is retrieved through AHB interface 0.

1: Base Layer data is retrieved through AHB interface 1.

- **BLEN: AHB Burst Length**

Value	Name	Description
0	AHB_BLEN_SINGLE	AHB Access is started as soon as there is enough space in the FIFO to store one data. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.
1	AHB_BLEN_INCR4	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 4 data. An AHB INCR4 Burst is used. SINGLE, INCR and INCR4 bursts are used. INCR is used for a burst of 2 and 3 beats.
2	AHB_BLEN_INCR8	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 8 data. An AHB INCR8 Burst is used. SINGLE, INCR, INCR4 and INCR8 bursts are used. INCR is used for a burst of 2 and 3 beats.
3	AHB_BLEN_INCR16	AHB Access is started as soon as there is enough space in the FIFO to store a total amount of 16 data. An AHB INCR16 Burst is used. SINGLE, INCR, INCR4, INCR8 and INCR16 bursts are used. INCR is used for a burst of 2 and 3 beats.

- **DLBO: Defined Length Burst Only For Channel Bus Transaction.**

0: Undefined length INCR burst is used for 2 and 3 beats burst.

1: Only Defined Length burst is used (SINGLE, INCR4, INCR8 and INCR16).

32.7.169Post Processing Layer Configuration 1 Register

Name: LCDC_PPCFG1

Address: 0xF0030570

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
–	–	–	ITUBT601	–	PPMODE		

- **PPMODE: Post Processing Output Format selection**

Table 32-55. PPMODE selection

Value	Name	Description
0	PPMODE_RGB_16BPP	RGB 16 bpp
1	PPMODE_RGB_24BPP_PACKED	RGB 24 bpp PACKED
2	PPMODE_RGB_24BPP_UNPACKED	RGB 24 bpp UNPACKED
3	PPMODE_YCBCR_422_MODE0	YCbCr 422 16 bpp (Mode 0)
4	PPMODE_YCBCR_422_MODE1	YCbCr 422 16 bpp (Mode 1)
5	PPMODE_YCBCR_422_MODE2	YCbCr 422 16 bpp (Mode 2)
6	PPMODE_YCBCR_422_MODE3	YCbCr 422 16 bpp (Mode 3)

- **ITUBT601: Color Space Conversion Luminance**

0: Luminance and chrominance range is [0;255]

1: Luminance values are clamped to [16;235] range. Chrominance values are clamped to [16;240] range.

32.7.170Post Processing Layer Configuration 2 Register

Name: LCDC_PPCFG2

Address: 0xF0030574

Access: Read-Write

Reset: 0x00000000

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

- **XSTRIDE: Horizontal Stride**

XSTRIDE represents the memory offset, in bytes, between two rows of the image memory.

32.7.171Post Processing Layer Configuration 3 Register

Name: LCDC_PPCFG3

Address: 0xF0030578

Access: Read-Write

Reset: 0x00000000

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

- **CSCYR: Color Space Conversion R coefficient for Luminance component, signed format, step set to 1/1024**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCYG: Color Space Conversion G coefficient for Luminance component, signed format, step set to 1/512**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCYB: Color Space Conversion B coefficient for Luminance component, signed format, step set to 1/1024**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCYOFF: Color Space Conversion Luminance Offset**

0: The *Yoff* parameter value is set to 0.

1: The *Yoff* parameter value is set to 16.

32.7.172Post Processing Layer Configuration 4 Register

Name: LCDC_PPCFG4

Address: 0xF003057C

Access: Read-Write

Reset: 0x00000000

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

- **CSCUR: Color Space Conversion R coefficient for Chrominance B component, signed format. (step 1/1024)**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCUG: Color Space Conversion G coefficient for Chrominance B component, signed format. (step 1/512)**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCUB: Color Space Conversion B coefficient for Chrominance B component, signed format. (step 1/512)**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCUOFF: Color Space Conversion Chrominance B Offset**

0: The *Cboff* parameter value is set to 0.

1: The *Cboff* parameter value is set to 128.

32.7.173Post Processing Layer Configuration 5 Register

Name: LCDC_PPCFG5

Address: 0xF0030580

Access: Read-Write

Reset: 0x00000000

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

- **CSCVR: Color Space Conversion R coefficient for Chrominance R component, signed format. (step 1/1024)**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCVG: Color Space Conversion G coefficient for Chrominance R component, signed format. (step 1/512)**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCVB: Color Space Conversion B coefficient for Chrominance R component, signed format. (step 1/1024)**

Color Space Conversion coefficient format is 1 sign bit, 9 fractional bits.

- **CSCVOFF: Color Space Conversion Chrominance R Offset**

0: The *Croff* parameter value is set to 0.

1: The *Croff* parameter value is set to 128.

32.7.174Base CLUT Register x Register

Name: LCDC_BASECLUTx [x=0..255]

Address: 0xF0030600

Access: Read-write

Reset: 0x00000000

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

- **BCLUT: Blue Color entry**

This field indicates the 8-bit width Blue color of the color lookup table.

- **GCLUT: Green Color entry**

This field indicates the 8-bit width Green color of the color lookup table.

- **RCLUT: Red Color entry**

This field indicates the 8-bit width Red color of the color lookup table.

32.7.175Overlay 1 CLUT Register x Register

Name: LCDC_OVR1CLUTx [x=0..255]

Address: 0xF0030A00

Access: Read-write

Reset: 0x00000000

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

- **BCLUT: Blue Color entry**

This field indicates the 8-bit width Blue color of the color lookup table.

- **GCLUT: Green Color entry**

This field indicates the 8-bit width Green color of the color lookup table.

- **RCLUT: Red Color entry**

This field indicates the 8-bit width Red color of the color lookup table.

- **ACLUT: Alpha Color entry**

This field indicates the 8-bit width Alpha channel of the color lookup table.

32.7.176Overlay 2 CLUT Register x Register

Name: LCDC_OVR2CLUTx [x=0..255]

Address: 0xF0030E00

Access: Read-write

Reset: 0x00000000

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

- **BCLUT: Blue Color entry**

This field indicates the 8-bit width Blue color of the color lookup table.

- **GCLUT: Green Color entry**

This field indicates the 8-bit width Green color of the color lookup table.

- **RCLUT: Red Color entry**

This field indicates the 8-bit width Red color of the color lookup table.

- **ACLUT: Alpha Color entry**

This field indicates the 8-bit width Alpha channel of the color lookup table.

32.7.177High End Overlay CLUT Register x Register

Name: LCDC_HEOCLUTx [x=0..255]

Address: 0xF0031200

Access: Read-write

Reset: 0x00000000

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

- **BCLUT: Blue Color entry**

This field indicates the 8-bit width Blue color of the color lookup table.

- **GCLUT: Green Color entry**

This field indicates the 8-bit width Green color of the color lookup table.

- **RCLUT: Red Color entry**

This field indicates the 8-bit width Red color of the color lookup table.

- **ACLUT: Alpha Color entry**

This field indicates the 8-bit width Alpha channel of the color lookup table.

32.7.178Hardware Cursor CLUT Register x Register

Name: LCDC_HCRCLUTx [x=0..255]

Address: 0xF0031600

Access: Read-write

Reset: 0x00000000

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

- **BCLUT: Blue Color entry**

This field indicates the 8-bit width Blue color of the color lookup table.

- **GCLUT: Green Color entry**

This field indicates the 8-bit width Green color of the color lookup table.

- **RCLUT: Red Color entry**

This field indicates the 8-bit width Red color of the color lookup table.

- **ACLUT: Alpha Color entry**

This field indicates the 8-bit width Alpha channel of the color lookup table.

33. Image Sensor Interface (ISI)

33.1 Description

The Image Sensor Interface (ISI) connects a CMOS-type image sensor to the processor and provides image capture in various formats. It does data conversion, if necessary, before the storage in memory through DMA.

The ISI supports color CMOS image sensor and grayscale image sensors with a reduced set of functionalities.

In grayscale mode, the data stream is stored in memory without any processing and so is not compatible with the LCD controller.

Internal FIFOs on the preview and codec paths are used to store the incoming data. The RGB output on the preview path is compatible with the LCD controller. This module outputs the data in RGB format (LCD compatible) and has scaling capabilities to make it compliant to the LCD display resolution (See Table 33-3 on page 810).

Several input formats such as preprocessed RGB or YCbCr are supported through the data bus interface.

It supports two modes of synchronization:

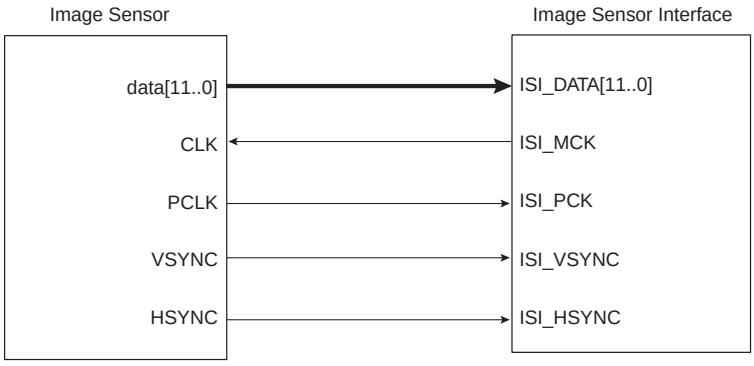
- The hardware with ISI_VSYNC and ISI_HSYNC signals
- The International Telecommunication Union Recommendation *ITU-R BT.656-4* Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) synchronization sequence.

Using EAV/SAV for synchronization reduces the pin count (ISI_VSYNC, ISI_HSYNC not used). The polarity of the synchronization pulse is programmable to comply with the sensor signals.

Table 33-1. I/O Description

Signal	Direction	Description
ISI_VSYNC	IN	Vertical Synchronization
ISI_HSYNC	IN	Horizontal Synchronization
ISI_DATA[11..0]	IN	Sensor Pixel Data
ISI_MCK	OUT	Master Clock Provided to the Image Sensor
ISI_PCK	IN	Pixel Clock Provided by the Image Sensor

Figure 33-1. ISI Connection Example



33.2 Embedded Characteristics

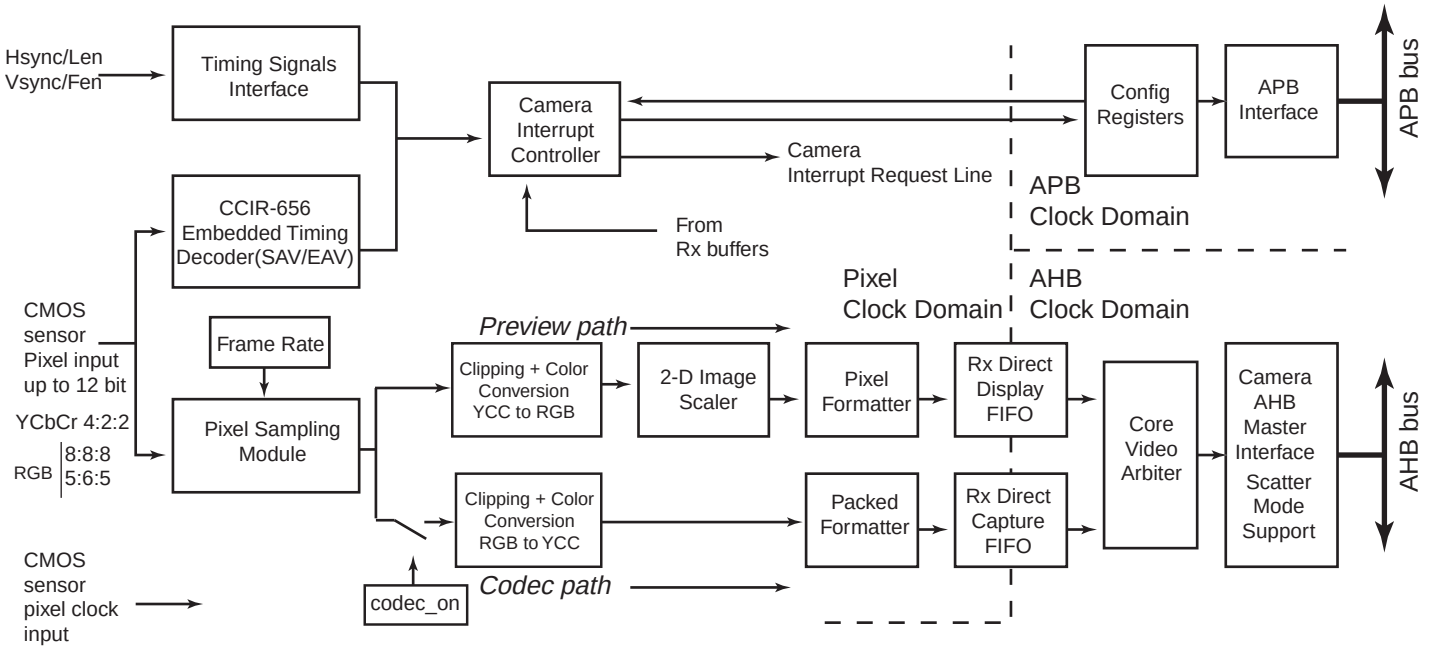
- ITU-R BT. 601/656 8-bit Mode External Interface Support
- Supports up to 12-bit Grayscale CMOS Sensors
- Support for ITU-R BT.656-4 SAV and EAV Synchronization
- Vertical and Horizontal Resolutions up to 2048*2048
- Preview Path

- Up to 2048*2048 in Grayscale Mode
- Up to 640*480 in RGB Mode

- 32 Bytes FIFO on Codec Path
- 32 Bytes FIFO on Preview Path
- Support for Packed Data Formatting for YCbCr 4:2:2 Formats
- Preview Scaler to Generate Smaller Size image
- Programmable Frame Capture Rate
- VGA, QVGA, CIF, QCIF Formats Supported for LCD Preview
- Custom Formats with Horizontal and Vertical Preview Size as Multiples of 16 Also Supported for LCD Preview

33.3 Block Diagram

Figure 33-2. Image Sensor Interface Block Diagram



33.4 Functional Description

The Image Sensor Interface (ISI) supports direct connection to the ITU-R BT. 601/656 8-bit mode compliant sensors and up to 12-bit grayscale sensors. It receives the image data stream from the image sensor on the 12-bit data bus.

This module receives up to 12 bits for data, the horizontal and vertical synchronizations and the pixel clock. The reduced pin count alternative for synchronization is supported for sensors that embed SAV (start of active video) and EAV (end of active video) delimiters in the data stream.

The Image Sensor Interface interrupt line is connected to the Advanced Interrupt Controller and can trigger an interrupt at the beginning of each frame and at the end of a DMA frame transfer. If the SAV/EAV synchronization is used, an interrupt can be triggered on each delimiter event.

For 8-bit color sensors, the data stream received can be in several possible formats: YCbCr 4:2:2, RGB 8:8:8, RGB 5:6:5 and may be processed before the storage in memory. When the preview DMA channel is configured and enabled, the preview path is activated and an ‘RGB frame’ is moved to memory. The preview path frame rate is configured with the FRATE field of the ISI_CFG1 register. When the codec DMA channel is configured and enabled, the codec path is activated and a ‘YCbCr 4:2:2 frame’ is captured as soon as the ISI_CDC bit of the ISI Control Register (ISI_CR) is set.

When the FULL bit of the ISI_CFG1 register is set, both preview DMA channel and codec DMA channel can operate simultaneously. When a zero is written to the FULL bit of the ISI_CFG1 register, a hardware scheduler checks the FRATE field. If its value is zero, a preview frame is skipped and a codec frame is moved to memory instead. If its value is other than zero, at least one free frame slot is available. The scheduler postpones the codec frame to that free available frame slot.

The data stream may be sent on both preview path and codec path if the value of bit ISI_CDC in the ISI_CR is one. To optimize the bandwidth, the codec path should be enabled only when a capture is required.

In grayscale mode, the input data stream is stored in memory without any processing. The 12-bit data, which represent the grayscale level for the pixel, is stored in memory one or two pixels per word, depending on the GS_MODE bit in the ISI_CFG2 register. The codec datapath is not available when grayscale image is selected.

A frame rate counter allows users to capture all frames or 1 out of every 2 to 8 frames.

33.4.1 Data Timing

The two data timings using horizontal and vertical synchronization and EAV/SAV sequence synchronization are shown in Figure 33-3 and Figure 33-4.

In the VSYNC/HSYNC synchronization, the valid data is captured with the active edge of the pixel clock (ISI_PCK), after SFD lines of vertical blanking and SLD pixel clock periods delay programmed in the ISI_CR .

The ITU-RBT.656-4 defines the functional timing for an 8-bit wide interface.

There are two timing reference signals, one at the beginning of each video data block SAV (0xFF000080) and one at the end of each video data block EAV(0xFF00009D). Only data sent between EAV and SAV is captured. Horizontal blanking and vertical blanking are ignored. Use of the SAV and EAV synchronization eliminates the ISI_VSYNC and ISI_HSYNC signals from the interface, thereby reducing the pin count. In order to retrieve both frame and line synchronization properly, at least one line of vertical blanking is mandatory.

Figure 33-3. HSYNC and VSYNC Synchronization

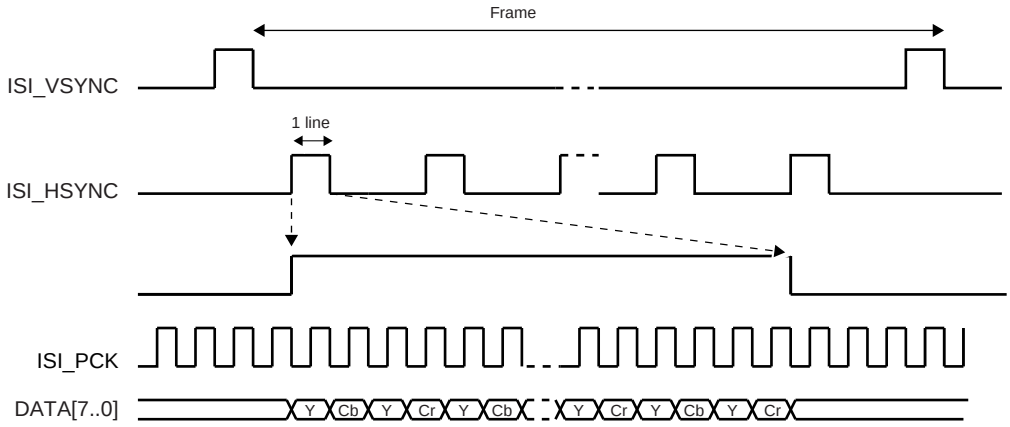
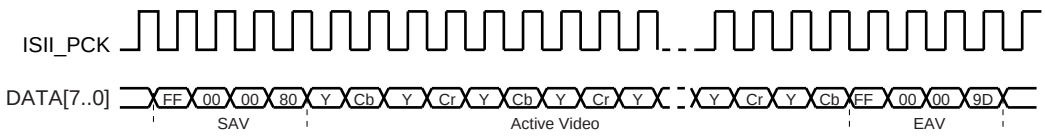


Figure 33-4. SAV and EAV Sequence Synchronization



33.4.2 Data Ordering

The RGB color space format is required for viewing images on a display screen preview, and the YCbCr color space format is required for encoding.

All the sensors do not output the YCbCr or RGB components in the same order. The ISI allows the user to program the same component order as the sensor, reducing software treatments to restore the right format.

Table 33-2. Data Ordering in YCbCr Mode

Mode	Byte 0	Byte 1	Byte 2	Byte 3
Default	Cb(i)	Y(i)	Cr(i)	Y(i+1)
Mode 1	Cr(i)	Y(i)	Cb(i)	Y(i+1)
Mode 2	Y(i)	Cb(i)	Y(i+1)	Cr(i)
Mode 3	Y(i)	Cr(i)	Y(i+1)	Cb(i)

Table 33-3. RGB Format in Default Mode, RGB_CFG = 00, No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R7(i)	R6(i)	R5(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	G7(i)	G6(i)	G5(i)	G4(i)	G3(i)	G2(i)	G1(i)	G0(i)
	Byte 2	B7(i)	B6(i)	B5(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 3	R7(i+1)	R6(i+1)	R5(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
RGB 5:6:5	Byte 0	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)	G5(i)	G4(i)	G3(i)
	Byte 1	G2(i)	G1(i)	G0(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 2	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)
	Byte 3	G2(i+1)	G1(i+1)	G0(i+1)	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)

Table 33-4. RGB Format, RGB_CFG = 10 (Mode 2), No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 5:6:5	Byte 0	G2(i)	G1(i)	G0(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)	G5(i)	G4(i)	G3(i)
	Byte 2	G2(i+1)	G1(i+1)	G0(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
	Byte 3	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)

Table 33-5. RGB Format in Default Mode, RGB_CFG = 00, Swap Activated

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)	R5(i)	R6(i)	R7(i)
	Byte 1	G0(i)	G1(i)	G2(i)	G3(i)	G4(i)	G5(i)	G6(i)	G7(i)
	Byte 2	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	B5(i)	B6(i)	B7(i)
	Byte 3	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)	R5(i+1)	R6(i+1)	R7(i+1)
RGB 5:6:5	Byte 0	G3(i)	G4(i)	G5(i)	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)
	Byte 1	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	G0(i)	G1(i)	G2(i)
	Byte 2	G3(i+1)	G4(i+1)	G5(i+1)	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)
	Byte 3	B0(i+1)	B1(i+1)	B2(i+1)	B3(i+1)	B4(i+1)	G0(i+1)	G1(i+1)	G2(i+1)

The RGB 5:6:5 input format is processed to be displayed as RGB 5:6:5 format, compliant with the 16-bit mode of the LCD controller.

33.4.3 Clocks

The sensor master clock (ISI_MCK) can be generated either by the Advanced Power Management Controller (APMC) through a Programmable Clock output or by an external oscillator connected to the sensor.

None of the sensors embed a power management controller, so providing the clock by the APMC is a simple and efficient way to control power consumption of the system.

Care must be taken when programming the system clock. The ISI has two clock domains, the sensor master clock and the pixel clock provided by sensor. The two clock domains are not synchronized, but the sensor master clock must be faster than the pixel clock.

33.4.4 Preview Path

33.4.4.1 Scaling, Decimation (Subsampling)

This module resizes captured 8-bit color sensor images to fit the LCD display format. The resize module performs only downscaling. The same ratio is applied for both horizontal and vertical resize, then a fractional decimation algorithm is applied.

The decimation factor is a multiple of 1/16; values 0 to 15 are forbidden.

Table 33-6. Decimation Factor

Decimation Value	0–15	16	17	18	19	...	124	125	126	127
Decimation Factor	—	1	1.063	1.125	1.188	...	7.750	7.813	7.875	7.938

Table 33-7. Decimation and Scaler Offset Values

OUTPUT	INPUT	352*288	640*480	800*600	1280*1024	1600*1200	2048*1536
VGA 640*480	F	—	16	20	32	40	51
QVGA 320*240	F	16	32	40	64	80	102
CIF 352*288	F	16	26	33	56	66	85
QCIF 176*144	F	32	53	66	113	133	170

Example:

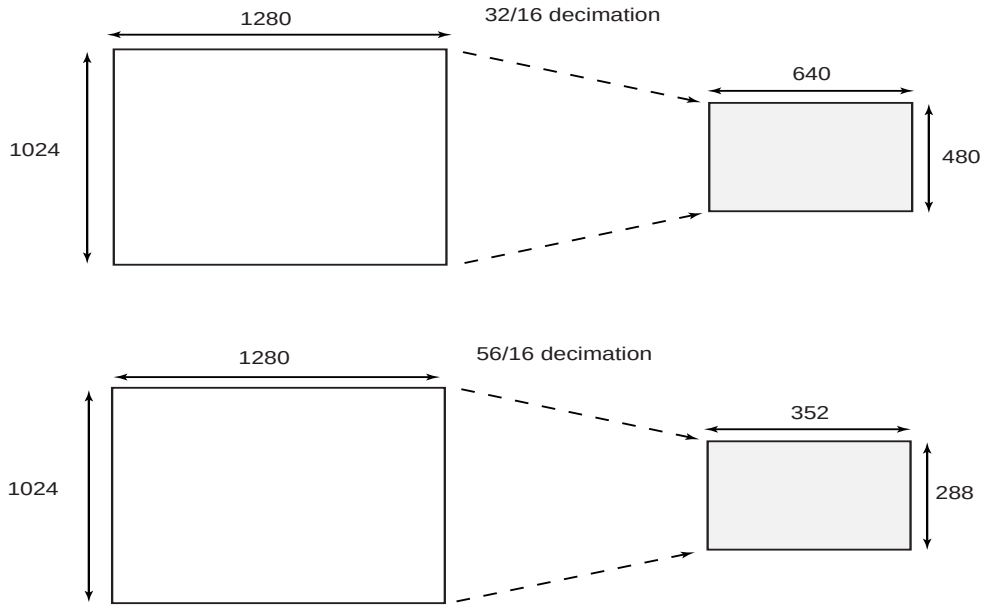
Input 1280*1024 Output = 640*480

Hratio = 1280/640 = 2

Vratio = 1024/480 = 2.1333

The decimation factor is 2 so 32/16.

Figure 33-5. Resize Examples



33.4.4.2 Color Space Conversion

This module converts YCrCb or YUV pixels to RGB color space. Clipping is performed to ensure that the samples value do not exceed the allowable range. The conversion matrix is defined below and is fully programmable:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} C_0 & 0 & C_1 \\ C_0 & -C_2 & -C_3 \\ C_0 & C_4 & 0 \end{bmatrix} \times \begin{bmatrix} Y - Y_{off} \\ C_b - C_{boff} \\ C_r - C_{roff} \end{bmatrix}$$

Example of programmable value to convert YCrCb to RGB:

$$\begin{cases} R = 1{,}164 \cdot (Y - 16) + 1{,}596 \cdot (C_r - 128) \\ G = 1{,}164 \cdot (Y - 16) - 0{,}813 \cdot (C_r - 128) - 0{,}392 \cdot (C_b - 128) \\ B = 1{,}164 \cdot (Y - 16) + 2{,}107 \cdot (C_b - 128) \end{cases}$$

An example of programmable value to convert from YUV to RGB:

$$\begin{cases} R = Y + 1{,}596 \cdot V \\ G = Y - 0{,}394 \cdot U - 0{,}436 \cdot V \\ B = Y + 2{,}032 \cdot U \end{cases}$$

33.4.4.3 Memory Interface

Preview datapath contains a data formatter that converts 8:8:8 pixel to RGB 5:6:5 format compliant with 16-bit format of the LCD controller. In general, when converting from a color channel with more bits to one with fewer bits, formatter module discards the lower-order bits. Example: Converting from RGB 8:8:8 to RGB 5:6:5, it discards the three LSBs from the red and blue channels, and two LSBs from the green channel. When grayscale mode is enabled, two memory formats are supported. One mode supports two pixels per word, and the other mode supports one pixel per word.

Table 33-8. Grayscale Memory Mapping Configuration for 12-bit Data

GS_MODE	DATA[31:24]	DATA[23:16]	DATA[15:8]	DATA[7:0]
0	P_0[11:4]	P_0[3:0], 0000	P_1[11:4]	P_1[3:0], 0000
1	P_0[11:4]	P_0[3:0], 0000	0	0

33.4.4.4 FIFO and DMA Features

Both preview and Codec datapaths contain FIFOs. These asynchronous buffers are used to safely transfer formatted pixels from the pixel clock domain to the AHB clock domain. A video arbiter is used to manage FIFO thresholds and triggers a relevant DMA request through the AHB master interface. Thus, depending on the FIFO state, a specified length burst is asserted. Regarding AHB master interface, it supports Scatter DMA mode through linked list operation. This mode of operation improves flexibility of image buffer location and allows the user to allocate two or more frame buffers. The destination frame buffers are defined by a series of Frame Buffer Descriptors (FBD). Each FBD controls the transfer of one entire frame and then optionally loads a further FBD to switch the DMA operation at another frame buffer address. The FBD is defined by a series of three words. The first one defines the current frame buffer address (named DMA_X_ADDR register), the second defines control information (named DMA_X_CTRL register) and the third defines the next descriptor address (named DMA_X_DSCR). DMA transfer mode with linked list support is available for both codec and preview datapath. The data to be transferred described by an FBD requires several burst accesses. In the following example, the use of two ping-pong frame buffers is described.

Example

The first FBD, stored at address 0x00030000, defines the location of the first frame buffer. This address is programmed in the ISI user interface DMA_P_DSCR. To enable the descriptor fetch operation, the value 0x00000001 must be written to the DMA_P_CTRL register. LLI_0 and LLI_1 are the two descriptors of the linked list.

Destination address: frame buffer ID0 0x02A000 (LLI_0.DMA_P_ADDR)

Transfer 0 Control Information, fetch and writeback: 0x00000003 (LLI_0.DMA_P_CTRL)

Next FBD address: 0x00030010 (LLI_0.DMA_P_DSCR)

Second FBD, stored at address 0x00030010, defines the location of the second frame buffer.

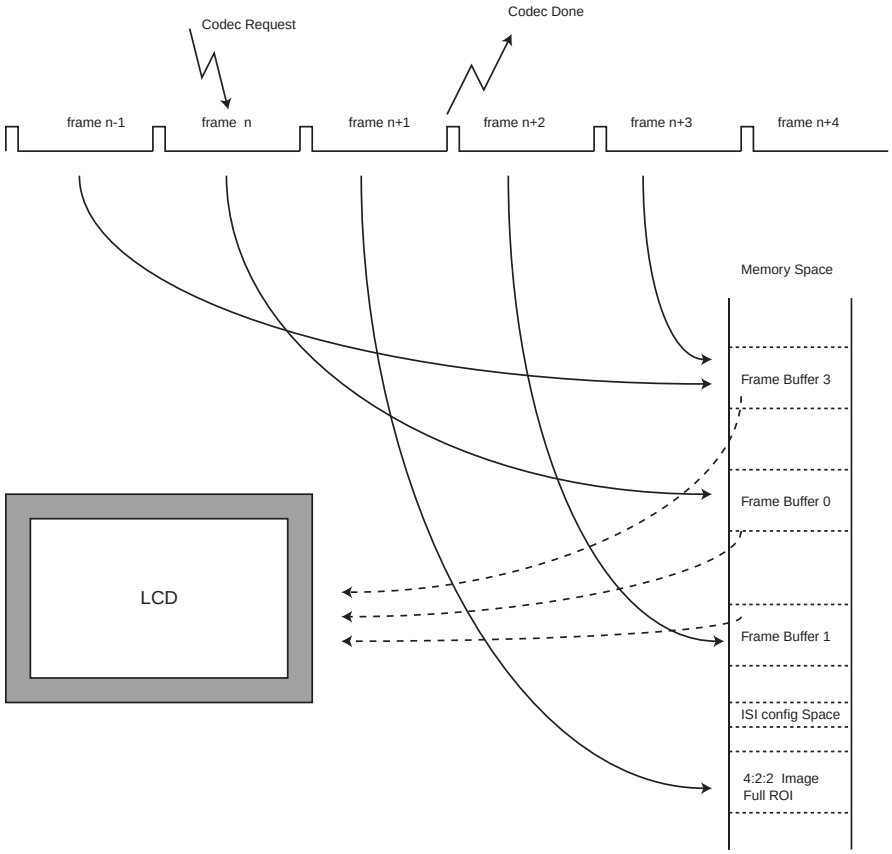
Destination address: frame buffer ID1 0x0003A000 (LLI_1.DMA_P_ADDR

Transfer 1 Control information fetch and writeback: 0x00000003 (LLI_1.DMA_P_CTRL)

Next FBD address: 0x00030000, wrapping to first FBD (LLI_1.DMA_P_DSCR)

Using this technique, several frame buffers can be configured through the linked list. Figure 33-6 illustrates a typical three frame buffer application. Frame n is mapped to frame buffer 0, frame n+1 is mapped to frame buffer 1, frame n+2 is mapped to frame buffer 2, further frames wrap. A codec request occurs, and the full-size 4:2:2 encoded frame is stored in a dedicated memory space.

Figure 33-6. Three Frame Buffers Application and Memory Mapping



33.4.5 Codec Path

33.4.5.1 Color Space Conversion

Depending on user selection, this module can be bypassed so that input YCrCb stream is directly connected to the format converter module. If the RGB input stream is selected, this module converts RGB to YCrCb color space with the formulas given below:

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} C_0 & C_1 & C_2 \\ C_3 & -C_4 & -C_5 \\ -C_6 & -C_7 & C_8 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y_{off} \\ Cr_{off} \\ Cb_{off} \end{bmatrix}$$

An example of coefficients is given below:

$$\begin{cases} Y = 0{,}257 \cdot R + 0{,}504 \cdot G + 0{,}098 \cdot B + 16 \\ C_r = 0{,}439 \cdot R - 0{,}368 \cdot G - 0{,}071 \cdot B + 128 \\ C_b = -0{,}148 \cdot R - 0{,}291 \cdot G + 0{,}439 \cdot B + 128 \end{cases}$$

33.4.5.2 Memory Interface

Dedicated FIFOs are used to support packed memory mapping. YCrCb pixel components are sent in a single 32-bit word in a contiguous space (packed). Data is stored in the order of natural scan lines. Planar mode is not supported.

33.4.5.3 DMA Features

Like preview datapath, codec datapath DMA mode uses linked list operation.

33.5 Image Sensor Interface (ISI) User Interface

Table 33-9. Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	ISI Configuration 1 Register	ISI_CFG1	Read-write	0x00000000
0x04	ISI Configuration 2 Register	ISI_CFG2	Read-write	0x00000000
0x08	ISI Preview Size Register	ISI_PSIZE	Read-write	0x00000000
0x0C	ISI Preview Decimation Factor Register	ISI_PDECF	Read-write	0x00000010
0x10	ISI Color Space Conversion YCrCb To RGB Set 0 Register	ISI_Y2R_SET0	Read-write	0x6832CC95
0x14	ISI Color Space Conversion YCrCb To RGB Set 1 Register	ISI_Y2R_SET1	Read-write	0x00007102
0x18	ISI Color Space Conversion RGB To YCrCb Set 0 Register	ISI_R2Y_SET0	Read-write	0x01324145
0x1C	ISI Color Space Conversion RGB To YCrCb Set 1 Register	ISI_R2Y_SET1	Read-write	0x01245E38
0x20	ISI Color Space Conversion RGB To YCrCb Set 2 Register	ISI_R2Y_SET2	Read-write	0x01384A4B
0x24	ISI Control Register	ISI_CR	Write-only	0x00000000
0x28	ISI Status Register	ISI_SR	Read-only	0x00000000
0x2C	ISI Interrupt Enable Register	ISI_IER	Write-only	0x00000000
0x30	ISI Interrupt Disable Register	ISI_IDR	Write-only	0x00000000
0x34	ISI Interrupt Mask Register	ISI_IMR	Read-only	0x00000000
0x38	DMA Channel Enable Register	ISI_DMA_CHER	Write-only	0x00000000
0x3C	DMA Channel Disable Register	ISI_DMA_CHDR	Write-only	0x00000000
0x40	DMA Channel Status Register	ISI_DMA_CHSR	Read-only	0x00000000
0x44	DMA Preview Base Address Register	ISI_DMA_P_ADDR	Read-write	0x00000000
0x48	DMA Preview Control Register	ISI_DMA_P_CTRL	Read-write	0x00000000
0x4C	DMA Preview Descriptor Address Register	ISI_DMA_P_DSCR	Read-write	0x00000000
0x50	DMA Codec Base Address Register	ISI_DMA_C_ADDR	Read-write	0x00000000
0x54	DMA Codec Control Register	ISI_DMA_C_CTRL	Read-write	0x00000000
0x58	DMA Codec Descriptor Address Register	ISI_DMA_C_DSCR	Read-write	0x00000000
0x5C–0xE0	Reserved	–	–	–
0xE4	Write Protection Control Register	ISI_WPCR	Read-write	0x00000000
0xE8	Write Protection Status Register	ISI_WPSR	Read-only	0x00000000
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Note: Several parts of the ISI controller use the pixel clock provided by the image sensor (ISI_PCK). Thus the user must first program the image sensor to provide this clock (ISI_PCK) before programming the Image Sensor Controller.

33.5.1 ISI Configuration 1 Register

Name: ISI_CFG1

Address: 0xF0034000

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
SFD							

23	22	21	20	19	18	17	16
SLD							

15	14	13	12	11	10	9	8
–	THMASK		FULL	DISCR	FRATE		

7	6	5	4	3	2	1	0
CRC_SYNC	EMB_SYNC	–	PIXCLK_POL	VSYNC_POL	HSYNC_POL	–	–

- **HSYNC_POL: Horizontal Synchronization Polarity**

0: HSYNC active high

1: HSYNC active low

- **VSYNC_POL: Vertical Synchronization Polarity**

0: VSYNC active high

1: VSYNC active low

- **PIXCLK_POL: Pixel Clock Polarity**

0: Data is sampled on rising edge of pixel clock.

1: Data is sampled on falling edge of pixel clock.

- **EMB_SYNC: Embedded Synchronization**

0: Synchronization by HSYNC, VSYNC

1: Synchronization by embedded synchronization sequence SAV/EAV

- **CRC_SYNC: Embedded Synchronization Correction**

0: No CRC correction is performed on embedded synchronization.

1: CRC correction is performed. If the correction is not possible, the current frame is discarded and the CRC_ERR bit is set in the status register.

- **FRATE: Frame Rate [0..7]**

0: All the frames are captured, else one frame every FRATE + 1 is captured.

- **DISCR: Disable Codec Request**

0: Codec datapath DMA interface requires a request to restart.

1: Codec datapath DMA automatically restarts.

- **FULL: Full Mode is Allowed**

0: The codec frame is transferred to memory when an available frame slot is detected.

1: Both preview and codec DMA channels are operating simultaneously.

- **THMASK: Threshold Mask**

Value	Name	Description
0	BEATS_4	Only 4 beats AHB burst allowed
1	BEATS_8	Only 4 and 8 beats AHB burst allowed
2	BEATS_16	4, 8 and 16 beats AHB burst allowed

- **SLD: Start of Line Delay**

SLD pixel clock periods to wait before the beginning of a line.

- **SFD: Start of Frame Delay**

SFD lines are skipped at the beginning of the frame.

33.5.2 ISI Configuration 2 Register

Name: ISI_CFG2

Address: 0xF0034004

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
RGB_CFG		YCC_SWAP		–	IM_HSIZE		
23	22	21	20	19	18	17	16
IM_HSIZE							
15	14	13	12	11	10	9	8
COL_SPACE	RGB_SWAP	GRAYSCALE	RGB_MODE	GS_MODE	IM_VSIZE		
7	6	5	4	3	2	1	0
IM_VSIZE							

- **IM_VSIZE: Vertical Size of the Image Sensor [0..2047]**

Vertical size = IM_VSIZE + 1

- **GS_MODE: Grayscale Pixel Format Mode**

0: 2 pixels per word

1: 1 pixel per word

- **RGB_MODE: RGB Input Mode**

0: RGB 8:8:8 24 bits

1: RGB 5:6:5 16 bits

- **GRAYSCALE: Grayscale Mode Format Enable**

0: Grayscale mode is disabled.

1: Input image is assumed to be grayscale coded.

- **RGB_SWAP: RGB Format Swap Mode**

0: D7 -> R7

1: D0 -> R7

The RGB_SWAP has no effect when the grayscale mode is enabled.

- **COL_SPACE: Color Space for the Image Data**

0: YCbCr

1: RGB

- **IM_HSIZE: Horizontal Size of the Image Sensor [0..2047]**

Horizontal size = IM_HSIZE + 1

- **YCC_SWAP: YCrCb Format Swap Mode**

Defines the YCC image data

Value	Name	Description
0	DEFAULT	Byte 0 Cb(i) Byte 1 Y(i) Byte 2 Cr(i) Byte 3 Y(i+1)
1	MODE1	Byte 0 Cr(i) Byte 1 Y(i) Byte 2 Cb(i) Byte 3 Y(i+1)
2	MODE2	Byte 0 Y(i) Byte 1 Cb(i) Byte 2 Y(i+1) Byte 3 Cr(i)
3	MODE3	Byte 0 Y(i) Byte 1 Cr(i) Byte 2 Y(i+1) Byte 3 Cb(i)

- **RGB_CFG: RGB Pixel Mapping Configuration**

Defines RGB pattern when RGB_MODE is set to 1

Value	Name	Description
0	DEFAULT	Byte 0 R/G(MSB) Byte 1 G(LSB)/B Byte 2 R/G(MSB) Byte 3 G(LSB)/B
1	MODE1	Byte 0 B/G(MSB) Byte 1 G(LSB)/R Byte 2 B/G(MSB) Byte 3 G(LSB)/R
2	MODE2	Byte 0 G(LSB)/R Byte 1 B/G(MSB) Byte 2 G(LSB)/R Byte 3 B/G(MSB)
3	MODE3	Byte 0 G(LSB)/B Byte 1 R/G(MSB) Byte 2 G(LSB)/B Byte 3 R/G(MSB)

If RGB_MODE is set to RGB 8:8:8, then RGB_CFG = 0 implies RGB color sequence, else it implies BGR color sequence.

33.5.3 ISI Preview Size Register

Name: ISI_PSIZE

Address: 0xF0034008

Access: Read-write

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
PREV_HSIZE							

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

7	6	5	4	3	2	1	0
PREV_VSIZE							

- **PREV_VSIZE: Vertical Size for the Preview Path**

Vertical Preview size = PREV_VSIZE + 1 (480 max only in RGB mode)

- **PREV_HSIZE: Horizontal Size for the Preview Path**

Horizontal Preview size = PREV_HSIZE + 1 (640 max only in RGB mode)

33.5.4 ISI Preview Decimation Factor Register

Name: ISI_PDECF

Address: 0xF003400C

Access: Read-write

Reset: 0x00000010

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

- **DEC_FACTOR: Decimation Factor**

DEC_FACTOR is 8-bit width, range is from 16 to 255. Values from 0 to 16 do not perform any decimation.

33.5.5 ISI Color Space Conversion YCrCb to RGB Set 0 Register

Name: ISI_Y2R_SET0

Address: 0xF0034010

Access: Read-write

Reset: 0x6832CC95

31	30	29	28	27	26	25	24
C3							
23	22	21	20	19	18	17	16
C2							
15	14	13	12	11	10	9	8
C1							
7	6	5	4	3	2	1	0
C0							

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element default step is 1/128, ranges from 0 to 1.9921875.

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element default step is 1/128, ranges from 0 to 1.9921875.

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element default step is 1/128, ranges from 0 to 1.9921875.

- **C3: Color Space Conversion Matrix Coefficient C3**

C3 element default step is 1/128, ranges from 0 to 1.9921875.

33.5.6 ISI Color Space Conversion YCrCb to RGB Set 1 Register

Name: ISI_Y2R_SET1

Address: 0xF0034014

Access: Read-write

Reset: 0x00007102

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	Cboff	Croff	Yoff	–	–	–	C4
7	6	5	4	3	2	1	0
C4							

- **C4: Color Space Conversion Matrix Coefficient C4**

C4 element default step is 1/128, ranges from 0 to 3.9921875.

- **Yoff: Color Space Conversion Luminance Default Offset**

0: No offset

1: Offset = 128

- **Croff: Color Space Conversion Red Chrominance Default Offset**

0: No offset

1: Offset = 16

- **Cboff: Color Space Conversion Blue Chrominance Default Offset**

0: No offset

1: Offset = 16

33.5.7 ISI Color Space Conversion RGB to YCrCb Set 0 Register

Name: ISI_R2Y_SET0

Address: 0xF0034018

Access: Read-write

Reset: 0x01324145

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

23	22	21	20	19	18	17	16
–	C2						

15	14	13	12	11	10	9	8
–	C1						

7	6	5	4	3	2	1	0
–	C0						

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element default step is 1/256, from 0 to 0.49609375.

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element default step is 1/128, from 0 to 0.9921875.

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element default step is 1/512, from 0 to 0.2480468875.

- **Roff: Color Space Conversion Red Component Offset**

0: No offset

1: Offset = 16

33.5.8 ISI Color Space Conversion RGB to YCrCb Set 1 Register

Name: ISI_R2Y_SET1

Address: 0xF003401C

Access: Read-write

Reset: 0x01245E38

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Goff
23	22	21	20	19	18	17	16
–	C5						
15	14	13	12	11	10	9	8
–	C4						
7	6	5	4	3	2	1	0
–	C3						

- **C3: Color Space Conversion Matrix Coefficient C3**

C0 element default step is 1/128, ranges from 0 to 0.9921875.

- **C4: Color Space Conversion Matrix Coefficient C4**

C1 element default step is 1/256, ranges from 0 to 0.49609375.

- **C5: Color Space Conversion Matrix Coefficient C5**

C1 element default step is 1/512, ranges from 0 to 0.2480468875.

- **Goff: Color Space Conversion Green Component Offset**

0: No offset

1: Offset = 128

33.5.9 ISI Color Space Conversion RGB to YCrCb Set 2 Register

Name: ISI_R2Y_SET2

Address: 0xF0034020

Access: Read-write

Reset: 0x01384A4B

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Boff
23	22	21	20	19	18	17	16
–	C8						
15	14	13	12	11	10	9	8
–	C7						
7	6	5	4	3	2	1	0
–	C6						

- **C6: Color Space Conversion Matrix Coefficient C6**

C6 element default step is 1/512, ranges from 0 to 0.2480468875.

- **C7: Color Space Conversion Matrix Coefficient C7**

C7 element default step is 1/256, ranges from 0 to 0.49609375.

- **C8: Color Space Conversion Matrix Coefficient C8**

C8 element default step is 1/128, ranges from 0 to 0.9921875.

- **Boff: Color Space Conversion Blue Component Offset**

0: No offset

1: Offset = 128

33.5.10 ISI Control Register

Name: ISI_CR

Address: 0xF0034024

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
–	–	–	–	–	–	–	ISI_CDC
7	6	5	4	3	2	1	0
–	–	–	–	–	ISI_SRST	ISI_DIS	ISI_EN

- **ISI_EN: ISI Module Enable Request**

Write a one to this bit to enable the module. Software must poll ENABLE bit in the ISI_SR to verify that the command has successfully completed.

- **ISI_DIS: ISI Module Disable Request**

Write a one to this bit to disable the module. If both ISI_EN and ISI_DIS are asserted at the same time, the disable request is not taken into account. Software must poll DIS_DONE bit in the ISI_SR to verify that the command has successfully completed.

- **ISI_SRST: ISI Software Reset Request**

Write a one to this bit to request a software reset of the module. Software must poll SRST bit in the ISI_SR to verify that the software request command has terminated.

- **ISI_CDC: ISI Codec Request**

Write a one to this bit to enable the codec datapath and capture a full resolution frame. A new request cannot be taken into account while CDC_PND bit is active in the ISI_SR.

33.5.11 ISI Status Register

Name: ISI_SR

Address: 0xF0034028

Access: Read-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	SIP	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	CDC_PND
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	ENABLE

- **ENABLE: Module Enable**

This bit is a status bit.

0: Module is disabled

1: Module is enabled

- **DIS_DONE: Module Disable Request has Terminated**

0: Indicates that the request is not completed (if a request was issued).

1: Disable request has completed. This flag is reset after a read operation.

- **SRST: Module Software Reset Request has Terminated**

0: Indicates that the request is not completed (if a request was issued).

1: Software reset request has completed. This flag is reset after a read operation.

- **CDC_PND: Pending Codec Request**

This bit is a status bit.

0: Indicates that no Codec request is pending

1: Indicates that the request has been taken into account but cannot be serviced within the current frame. The operation is postponed to the next frame.

- **VSYNC: Vertical Synchronization**

0: Indicates that the vertical synchronization has not been detected since the last read of the status register.

1: Indicates that a vertical synchronization has been detected since the last read of the status register.

- **PXFR_DONE: Preview DMA Transfer has Terminated**

When set, this bit indicates that the DATA transfer on the preview channel has completed. This flag is reset after a read operation.

0: Preview transfer done not detected.

1: Preview transfer done detected.

- **CXFR_DONE: Codec DMA Transfer has Terminated**

When set, this bit indicates that the DATA transfer on the codec channel has completed. This flag is reset after a read operation.

0: Codec transfer done not detected.

1: Codec transfer done detected.

- **SIP: Synchronization in Progress**

This is a status bit. When the status of the preview or codec DMA channel is modified, a minimum amount of time is required to perform the clock domain synchronization.

0: The clock domain synchronization process is terminated.

1: This bit is set when the clock domain synchronization operation occurs. No modification of the channel status is allowed when this bit is set, to guarantee data integrity.

- **P_OVR: Preview Datapath Overflow**

0: No overflow

1: An overrun condition has occurred in input FIFO on the preview path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO. This flag is reset after a read operation.

- **C_OVR: Codec Datapath Overflow**

0: No overflow

1: An overrun condition has occurred in input FIFO on the codec path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO. This flag is reset after a read operation.

- **CRC_ERR: CRC Synchronization Error**

0: No CRC error in the embedded synchronization frame (SAV/EAV)

1: Embedded Synchronization Correction is enabled (CRC_SYNC bit is set) in the control register and an error has been detected and not corrected. The frame is discarded and the ISI waits for a new one. This flag is reset after a read operation.

- **FR_OVR: Frame Rate Overrun**

0: No frame overrun

1: Frame overrun, the current frame is being skipped because a vsync signal has been detected while flushing FIFOs. This flag is reset after a read operation.

33.5.12 ISI Interrupt Enable Register

Name: ISI_IER

Address: 0xF003402C

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	–

- **DIS_DONE: Disable Done Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **SRST: Software Reset Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **VSYNC: Vertical Synchronization Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **PXFR_DONE: Preview DMA Transfer Done Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **CXFR_DONE: Codec DMA Transfer Done Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **P_OVR: Preview Datapath Overflow Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **C_OVR: Codec Datapath Overflow Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **CRC_ERR: Embedded Synchronization CRC Error Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

- **FR_OVR: Frame Rate Overflow Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt.

33.5.13 ISI Interrupt Disable Register

Name: ISI_IDR

Address: 0xF0034030

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	–

- **DIS_DONE: Disable Done Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **SRST: Software Reset Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **VSYNC: Vertical Synchronization Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **PXFR_DONE: Preview DMA Transfer Done Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **CXFR_DONE: Codec DMA Transfer Done Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **P_OVR: Preview Datapath Overflow Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **C_OVR: Codec Datapath Overflow Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **CRC_ERR: Embedded Synchronization CRC Error Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

- **FR_OVR: Frame Rate Overflow Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt.

33.5.14 ISI Interrupt Mask Register

Name: ISI_IMR

Address: 0xF0034034

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	–

- **DIS_DONE: Module Disable Operation Completed**

0: The disable completed interrupt is disabled.

1: The disable completed interrupt is enabled.

- **SRST: Software Reset Completed**

0: The software reset completed interrupt is disabled.

1: The software reset completed interrupt is enabled.

- **VSYNC: Vertical Synchronization**

0: The vertical synchronization interrupt is enabled.

1: The vertical synchronization interrupt is disabled.

- **PXFR_DONE: Preview DMA Transfer Interrupt**

0: The Preview DMA transfer completed interrupt is enabled

1: The Preview DMA transfer completed interrupt is disabled

- **CXFR_DONE: Codec DMA Transfer Interrupt**

0: The Codec DMA transfer completed interrupt is enabled

1: The Codec DMA transfer completed interrupt

- **P_OVR: FIFO Preview Overflow**

0: The preview FIFO overflow interrupt is disabled.

1: The preview FIFO overflow interrupt is enabled.

- **C_OVR: FIFO Codec Overflow**

0: The codec FIFO overflow interrupt is disabled.

1: The codec FIFO overflow interrupt is enabled.

- **CRC_ERR: CRC Synchronization Error**

0: The CRC error interrupt is disabled.

1: The CRC error interrupt is enabled.

- **FR_OVR: Frame Rate Overrun**

0: The frame overrun interrupt is disabled.

1: The frame overrun interrupt is enabled.

33.5.15 DMA Channel Enable Register

Name: ISI_DMA_CHER

Address: 0xF0034038

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
–	–	–	–	–	–	C_CH_EN	P_CH_EN

- **P_CH_EN: Preview Channel Enable**

Write a one to this bit to enable the preview DMA channel.

- **C_CH_EN: Codec Channel Enable**

Write a one to this bit to enable the codec DMA channel.

33.5.16 DMA Channel Disable Register

Name: ISI_DMA_CHDR

Address: 0xF003403C

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
–	–	–	–	–	–	C_CH_DIS	P_CH_DIS

- **P_CH_DIS: Preview Channel Disable Request**

0: No effect

1: Disables the channel. Poll P_CH_S in DMA_CHSR to verify that the preview channel status has been successfully modified.

- **C_CH_DIS: Codec Channel Disable Request**

0: No effect

1: Disables the channel. Poll C_CH_S in DMA_CHSR to verify that the codec channel status has been successfully modified.

33.5.17 DMA Channel Status Register

Name: ISI_DMA_CHSR

Address: 0xF0034040

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
–	–	–	–	–	–	C_CH_S	P_CH_S

- **P_CH_S: Preview DMA Channel Status**

0: Indicates that the Preview DMA channel is disabled

1: Indicates that the Preview DMA channel is enabled

- **C_CH_S: Code DMA Channel Status**

0: Indicates that the Codec DMA channel is disabled

1: Indicates that the Codec DMA channel is enabled

33.5.18 DMA Preview Base Address Register

Name: ISI_DMA_P_ADDR

Address: 0xF0034044

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
P_ADDR							

23	22	21	20	19	18	17	16
P_ADDR							

15	14	13	12	11	10	9	8
P_ADDR							

7	6	5	4	3	2	1	0
P_ADDR						–	–

- **P_ADDR: Preview Image Base Address**

This address is word aligned.

33.5.19 DMA Preview Control Register

Name: ISI_DMA_P_CTRL

Address: 0xF0034048

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
–	–	–	–	P_DONE	P_IEN	P_WB	P_FETCH

- **P_FETCH: Descriptor Fetch Control Bit**

0: Preview channel fetch operation is disabled.

1: Preview channel fetch operation is enabled.

- **P_WB: Descriptor Writeback Control Bit**

0: Preview channel writeback operation is disabled.

1: Preview channel writeback operation is enabled.

- **P_IEN: Transfer Done Flag Control**

0: Preview transfer done flag generation is enabled.

1: Preview transfer done flag generation is disabled.

- **P_DONE: Preview Transfer Done**

This bit is only updated in the memory.

0: The transfer related to this descriptor has not been performed.

1: The transfer related to this descriptor has completed. This bit is updated in memory at the end of the transfer, when writeback operation is enabled.

33.5.20 DMA Preview Descriptor Address Register

Name: ISI_DMA_P_DSCR

Address: 0xF003404C

Access: Read-write

Reset: 0x00000000

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

- **P_DSCR: Preview Descriptor Base Address**

This address is word aligned.

33.5.21 DMA Codec Base Address Register

Name: ISI_DMA_C_ADDR

Address: 0xF0034050

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
C_ADDR							

23	22	21	20	19	18	17	16
C_ADDR							

15	14	13	12	11	10	9	8
C_ADDR							

7	6	5	4	3	2	1	0
C_ADDR						–	–

- **C_ADDR: Codec Image Base Address**

This address is word aligned.

33.5.22 DMA Codec Control Register

Name: ISI_DMA_C_CTRL

Address: 0xF0034054

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
–	–	–	–	C_DONE	C_IEN	C_WB	C_FETCH

- **C_FETCH: Descriptor Fetch Control Bit**

0: Codec channel fetch operation is disabled.

1: Codec channel fetch operation is enabled.

- **C_WB: Descriptor Writeback Control Bit**

0: Codec channel writeback operation is disabled.

1: Codec channel writeback operation is enabled.

- **C_IEN: Transfer Done Flag Control**

0: Codec transfer done flag generation is enabled.

1: Codec transfer done flag generation is disabled.

- **C_DONE: Codec Transfer Done**

This bit is only updated in the memory.

0: The transfer related to this descriptor has not been performed.

1: The transfer related to this descriptor has completed. This bit is updated in memory at the end of the transfer, when. writeback operation is enabled.

33.5.23 DMA Codec Descriptor Address Register

Name: ISI_DMA_C_DSCR

Address: 0xF0034058

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
C_DSCR							

23	22	21	20	19	18	17	16
C_DSCR							

15	14	13	12	11	10	9	8
C_DSCR							

7	6	5	4	3	2	1	0
C_DSCR						–	–

- **C_DSCR: Codec Descriptor Base Address**

This address is word aligned.

33.5.24 ISI Write Protection Control Register

Name: ISI_WPCR

Address: 0xF00340E4

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 0x495349 (“ISI” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x495349 (“ISI” in ASCII).

- **WPKEY: Write Protection Key Password**

Value	Name	Description
0x495349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

33.5.25 ISI Write Protection Status Register

Name: ISI_WPSR

Address: 0xF00340E8

Access: Read-write

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**

Value	Description
0	No Write Protection Violation occurred since the last read of this register (WP_SR).
1	Write Protection detected unauthorized attempt to write a control register had occurred (since the last read).

- **WPVSRC: Write Protection Violation Source**

Value	Description
0	No Write Protection Violation occurred since the last read of this register (WP_SR).
1	Write access in ISI_CFG1 while Write Protection was enabled (since the last read).
2	Write access in ISI_CFG2 while Write Protection was enabled (since the last read).
3	Write access in ISI_PSIZE while Write Protection was enabled (since the last read).
4	Write access in ISI_PDECF while Write Protection was enabled (since the last read).
5	Write access in ISI_Y2R_SET0 while Write Protection was enabled (since the last read).
6	Write access in ISI_Y2R_SET1 while Write Protection was enabled (since the last read).
7	Write access in ISI_R2Y_SET0 while Write Protection was enabled (since the last read).
8	Write access in ISI_R2Y_SET1 while Write Protection was enabled (since the last read).
9	Write access in ISI_R2Y_SET2 while Write Protection was enabled (since the last read).

34. USB High Speed Device Port (UDPHS)

34.1 Description

The USB High Speed Device Port (UDPHS) is compliant with the Universal Serial Bus (USB), rev 2.0 High Speed device specification.

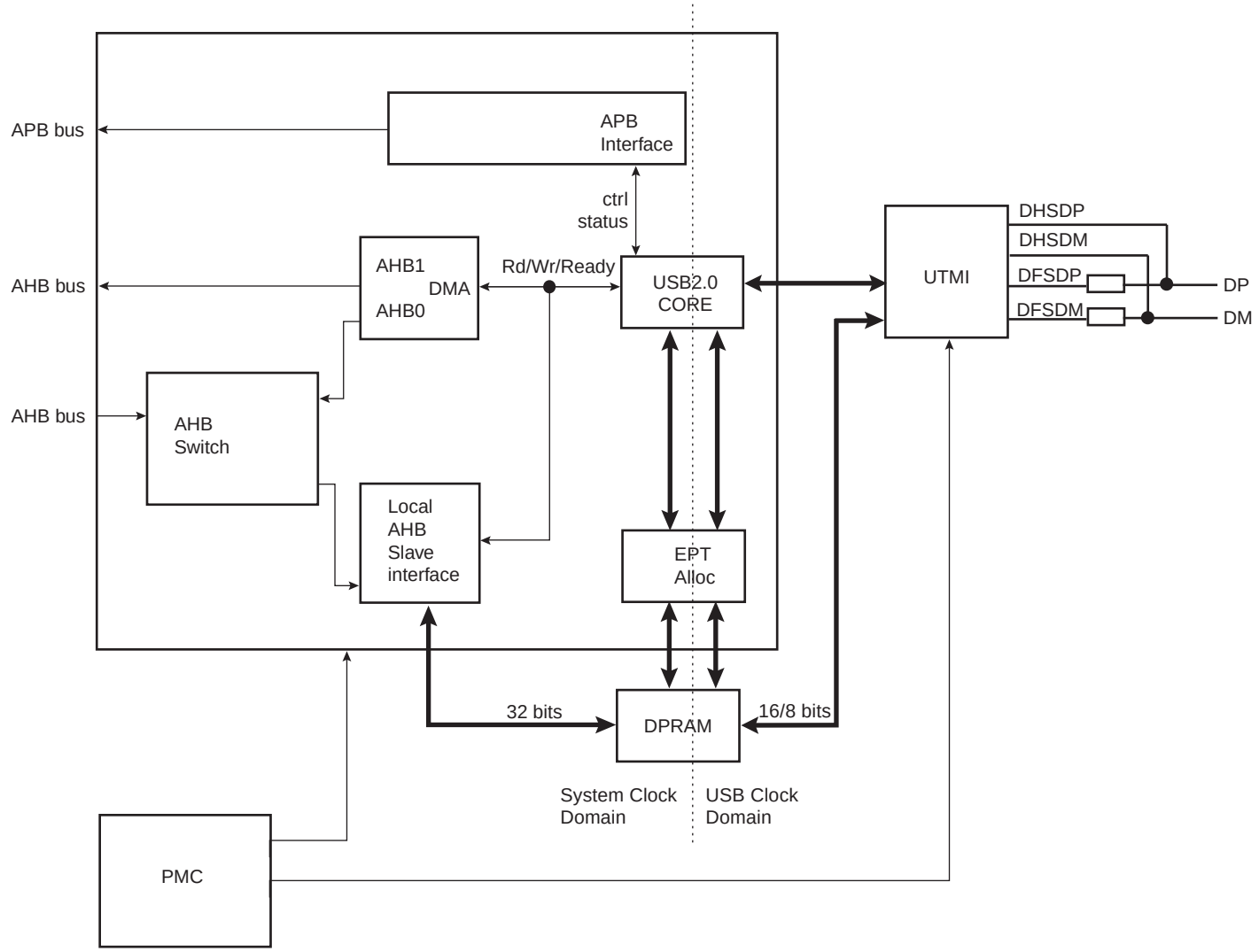
Each endpoint can be configured in one of several USB transfer types. It can be associated with one, two or three banks of a Dual-port RAM used to store the current data payload. If two or three banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints.

34.2 Embedded Characteristics

- 1 Device High Speed
- 1 UTMI transceiver shared between Host and Device
- USB v2.0 High Speed Compliant, 480 Mbit/s
- 16 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)
- 8 KBytes of DPRAM

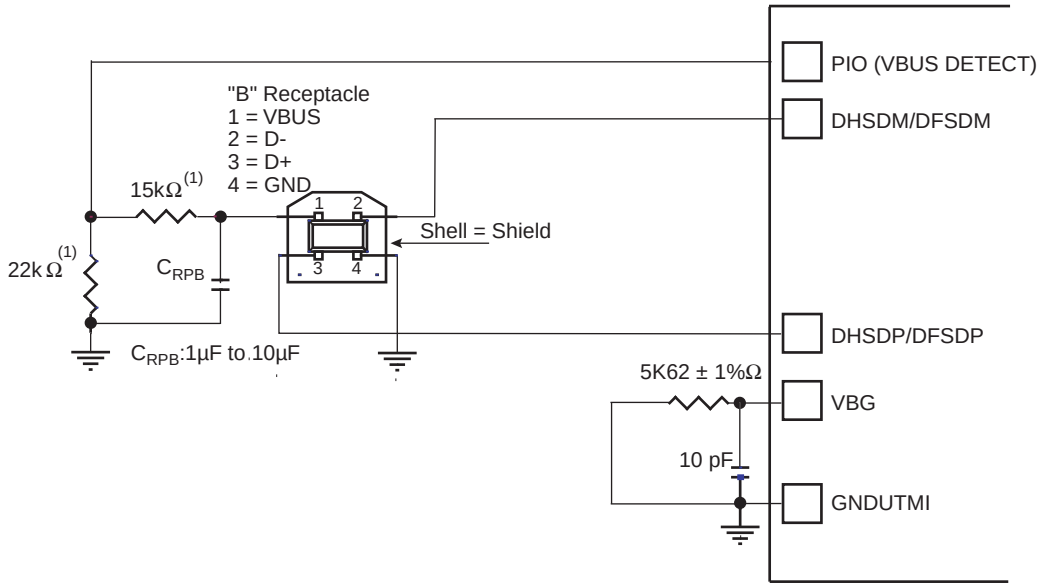
34.3 Block Diagram

Figure 34-1. Block Diagram



34.4 Typical Connection

Figure 34-2. Board Schematic



Note: The values shown on the 22 kΩ and 15 kΩ resistors are only valid with 3V3 supplied PIOs.

34.5 Product Dependencies

34.5.1 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 Peripheral Clock Enable Register (PMC_PCER). Then enable the PLL in the PMC UTMI Clock Configuration Register (CKGR_UCKR). Finally, enable BIAS in CKGR_UCKR.

However, if the application does not require UDPHS operations, the UDPHS clock can be stopped when not needed and restarted later.

34.5.2 Interrupt

The UDPHS interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the UDPHS interrupt requires the Interrupt Controller to be programmed first.

Table 34-1. Peripheral IDs

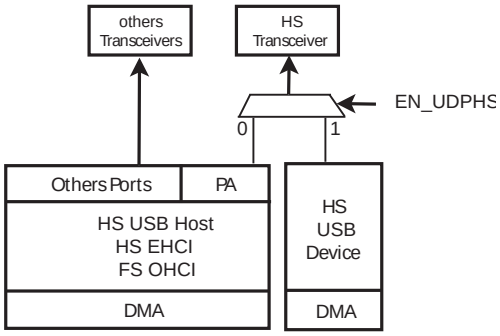
Instance	ID
UDPHS	33

34.6 Functional Description

34.6.1 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 register.

Figure 34-3. USB Selection



34.6.2 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.

34.6.3 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 34-2. 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

34.6.4 USB Transfer Event Definitions

A transfer is composed of one or several transactions as shown in the following table.

Table 34-3. USB Transfer Events

Transfer		Transaction
Direction	**Type**	
CONTROL (bidirectional)	Control Transfer (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 34-4. 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	3	Y	Y	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
2	EPT_2	3	Y	Y	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
3	EPT_3	2	Y	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
4	EPT_4	2	Y	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
5	EPT_5	2	Y	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
6	EPT_6	2	Y	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
7	EPT_7	2	Y	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
8	EPT_8	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
9	EPT_9	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
10	EPT_10	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
11	EPT_11	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
12	EPT_12	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
13	EPT_13	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
14	EPT_14	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt
15	EPT_15	2	N	N	1024	Ctrl/Bulk/Iso(34.3)/Interrupt

Note: 1. In Isochronous Mode (Iso), it is preferable that High Band Width capability is available.

The size of internal DPRAM is 8 KB.

Suspend and resume are automatically detected by the UDPHS device, which notifies the processor by raising an interrupt.

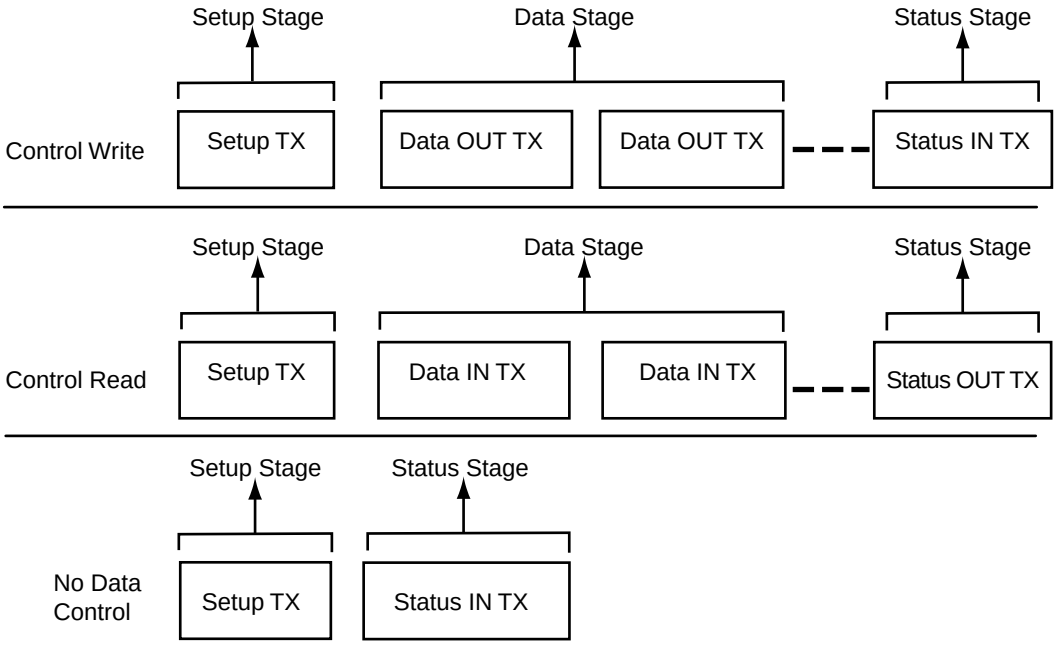
34.6.5 USB V2.0 High Speed BUS Transactions

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 34-4. Control Read and Write Sequences



A status IN or OUT transaction is identical to a data IN or OUT transaction.

34.6.6 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 897.

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 34-4. 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 8 KB. The DPR is shared by all active endpoints. The memory size required by the active endpoints must not exceed the size of the DPRAM.

SIZE_DPRAM = SIZE _EPT0

+ NB_BANK_EPT1 x SIZE_EPT1

+ NB_BANK_EPT2 x SIZE_EPT2

+ NB_BANK_EPT3 x SIZE_EPT3

+ NB_BANK_EPT4 x SIZE_EPT4

+ NB_BANK_EPT5 x SIZE_EPT5

+ NB_BANK_EPT6 x SIZE_EPT6

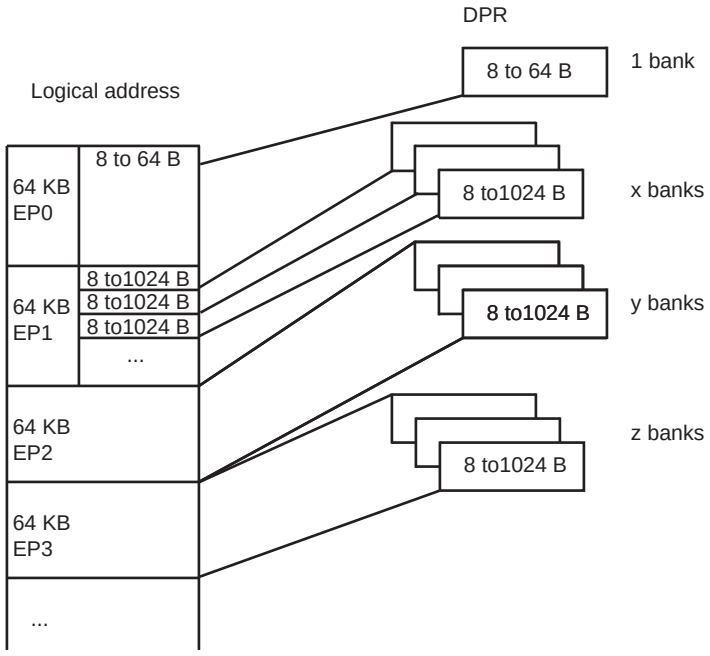
+... (refer to 34.7.8 UDPHS Endpoint Configuration Register)

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 KB logical address space.

The physical block of DPR allocated for the endpoint is remapped all along the 64 KB logical address space. The application can write a 64 KB buffer linearly.

Figure 34-5. 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.

34.6.7 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.

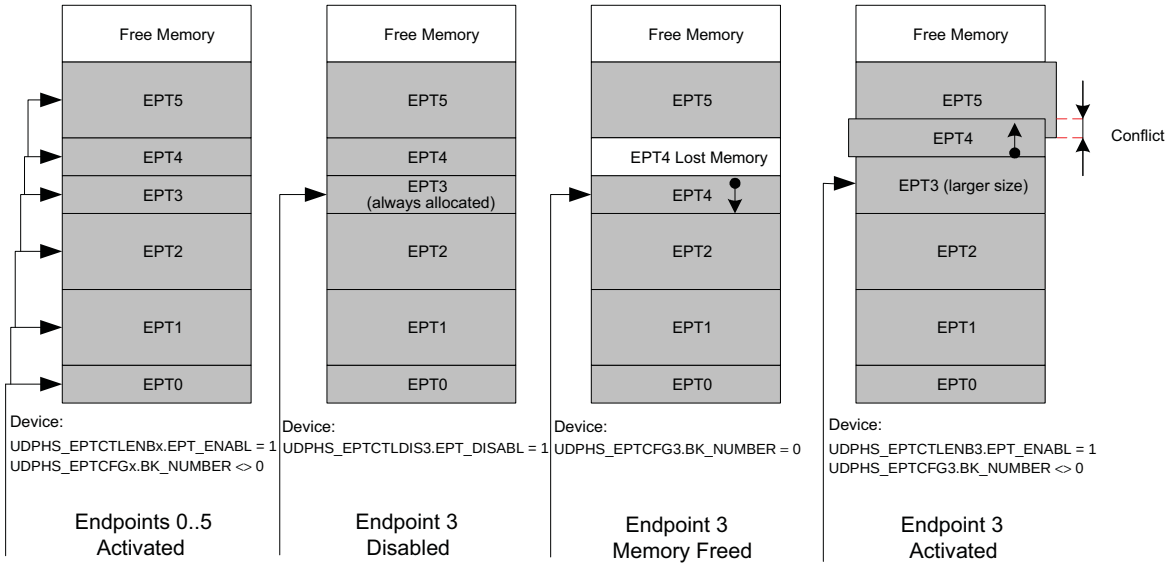
Disabling an endpoint, by writing a one to the Endpoint Disable bit in the UDPHS Endpoint Control Disable Register (UDPHS_EPTCTLDISx.EPT_DISABL), does not reset its configuration:

- 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 34-6 on page 856 illustrates the allocation and reorganization of the DPRAM in a typical example.

Figure 34-6. 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.

34.6.8 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 several times, during a single big USB packet DMA transfer in case another AHB master addresses the memory. The locked bursts result in up to 128-word single-cycle unbroken AHB bursts for bulk endpoints and 256-word single-cycle unbroken bursts for isochronous endpoints.

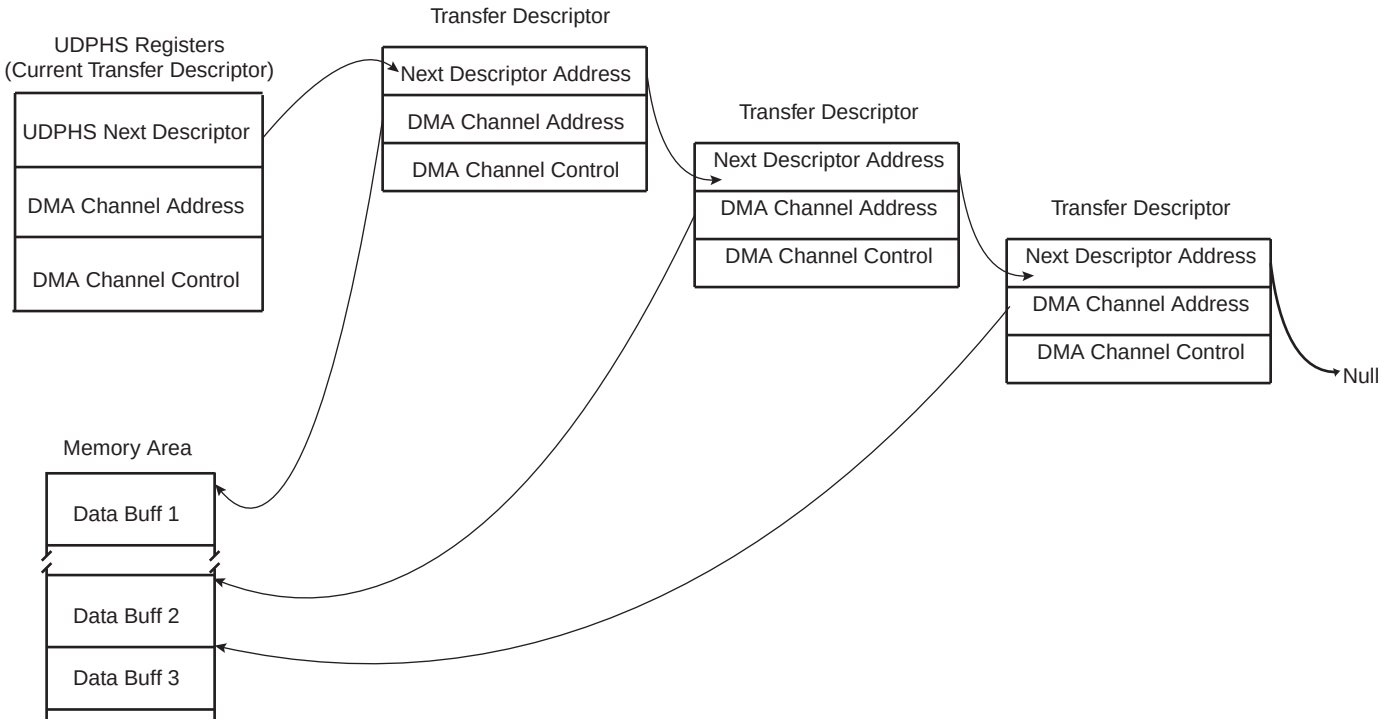
This maximum burst length is then controlled by the lowest programmed USB endpoint size (EPT_SIZE field in the UDPHS_EPTCFGx register) and DMA Size (BUFF_LENGTH field in the UDPHS_DMACONTROLx register).

The USB 2.0 device average throughput may be up to nearly 60 Mbyte/s. 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 918.

Note: In case of debug, be careful to address the DMA to an SRAM address even if a remap is done.

Figure 34-7. Example of DMA Chained List



34.6.9 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;
}
```

34.6.10 Handling Transactions with USB V2.0 Device Peripheral

34.6.10.1 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 34.6.10.15 “STALL” on page 869).

34.6.10.2 NYET

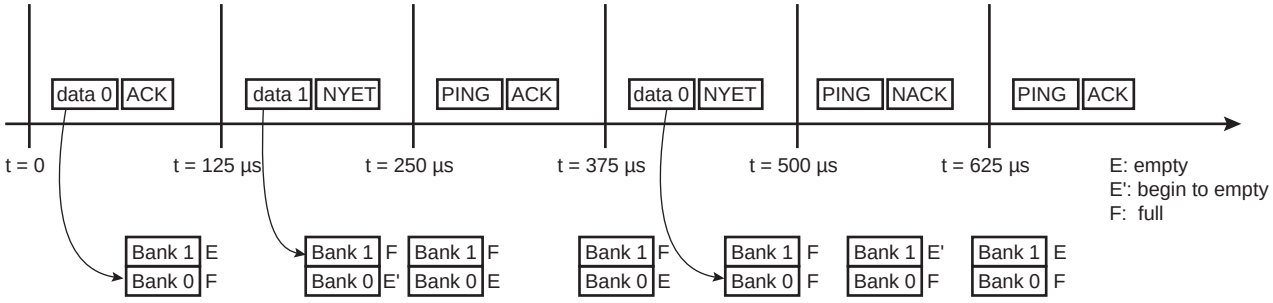
NYET is a High Speed only handshake. It is returned by a High Speed endpoint as part of the PING protocol.

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 34-8. NYET Example with Two Endpoint Banks



34.6.10.3 Data IN

34.6.10.4 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 34.6.10.5 below)
- 64 KB (see 34.6.10.5 below)

- DMA (see 34.6.10.6 below)

34.6.10.5 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 KB 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.

34.6.10.6 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 897 and Figure 34-13. 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 918). 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 34-9. Data IN Transfer for Endpoint with One Bank

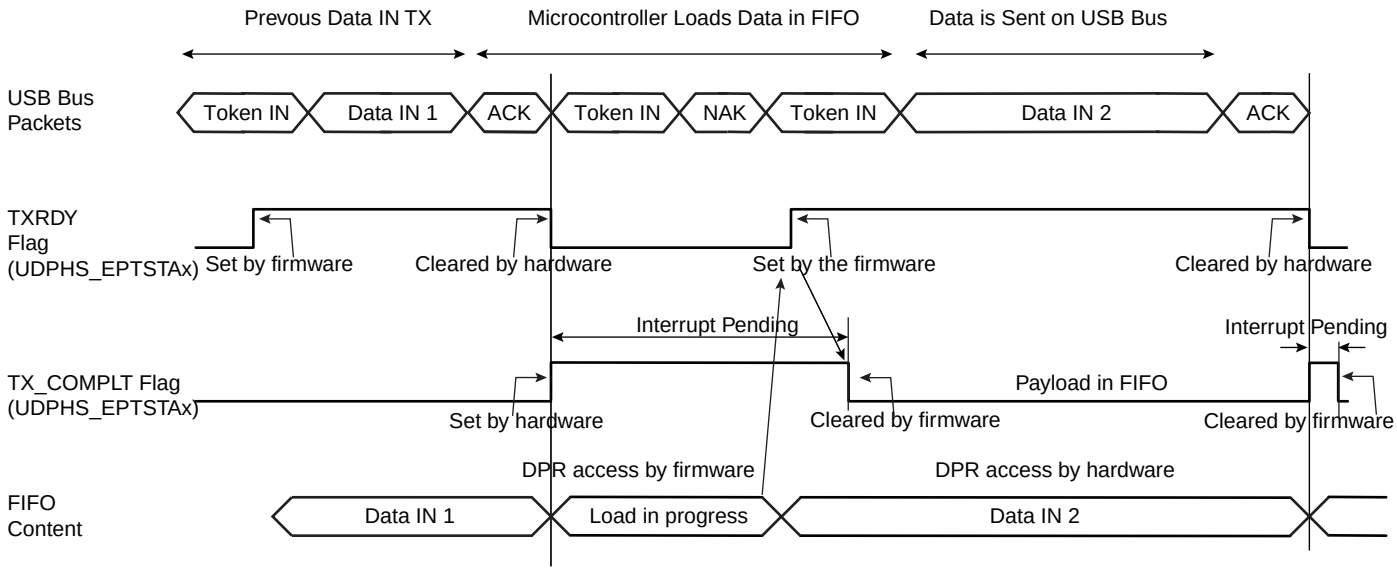


Figure 34-10.Data IN Transfer for Endpoint with Two Banks

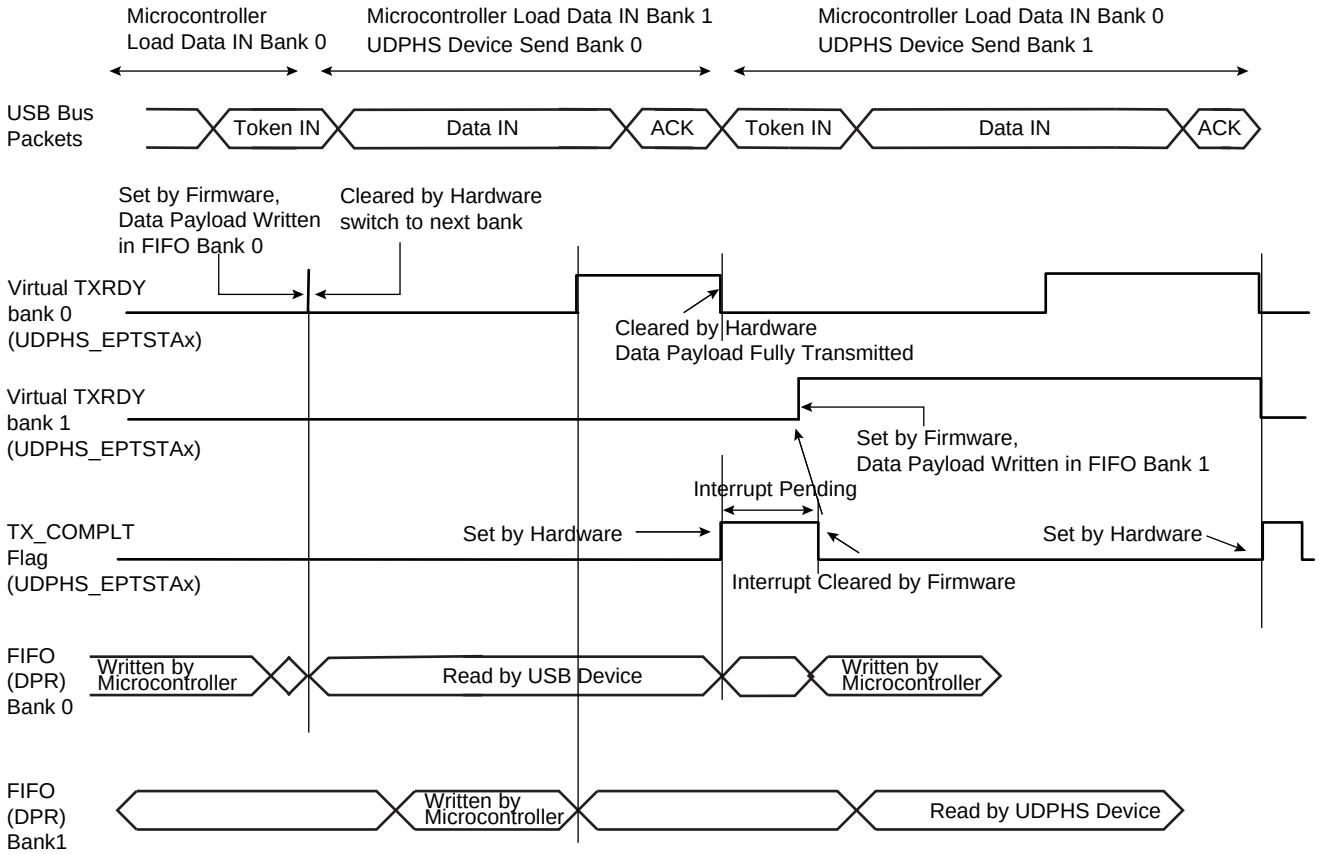
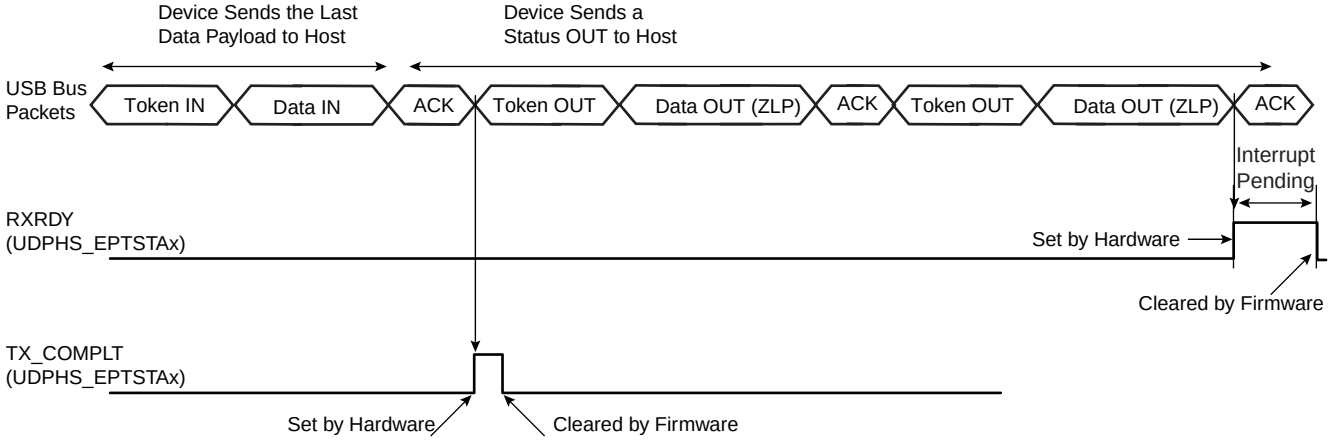
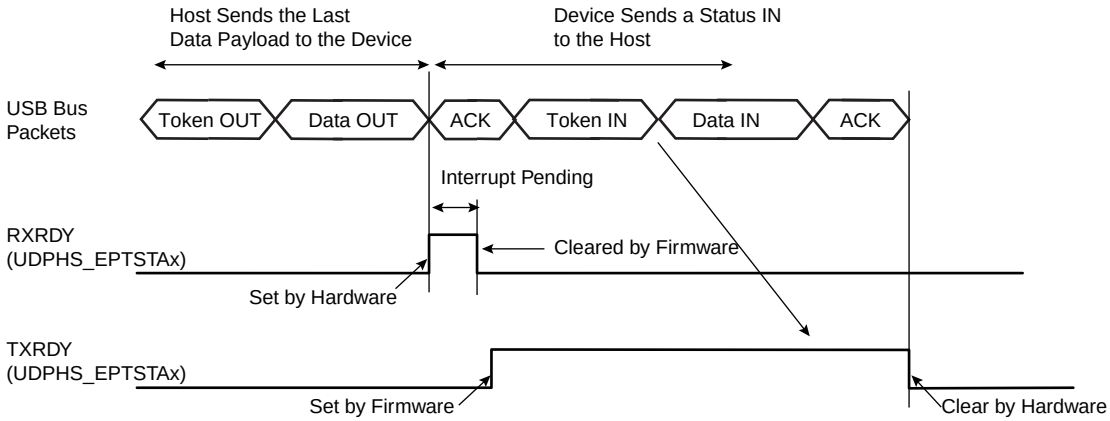


Figure 34-11.Data IN Followed By Status OUT Transfer at the End of a Control Transfer



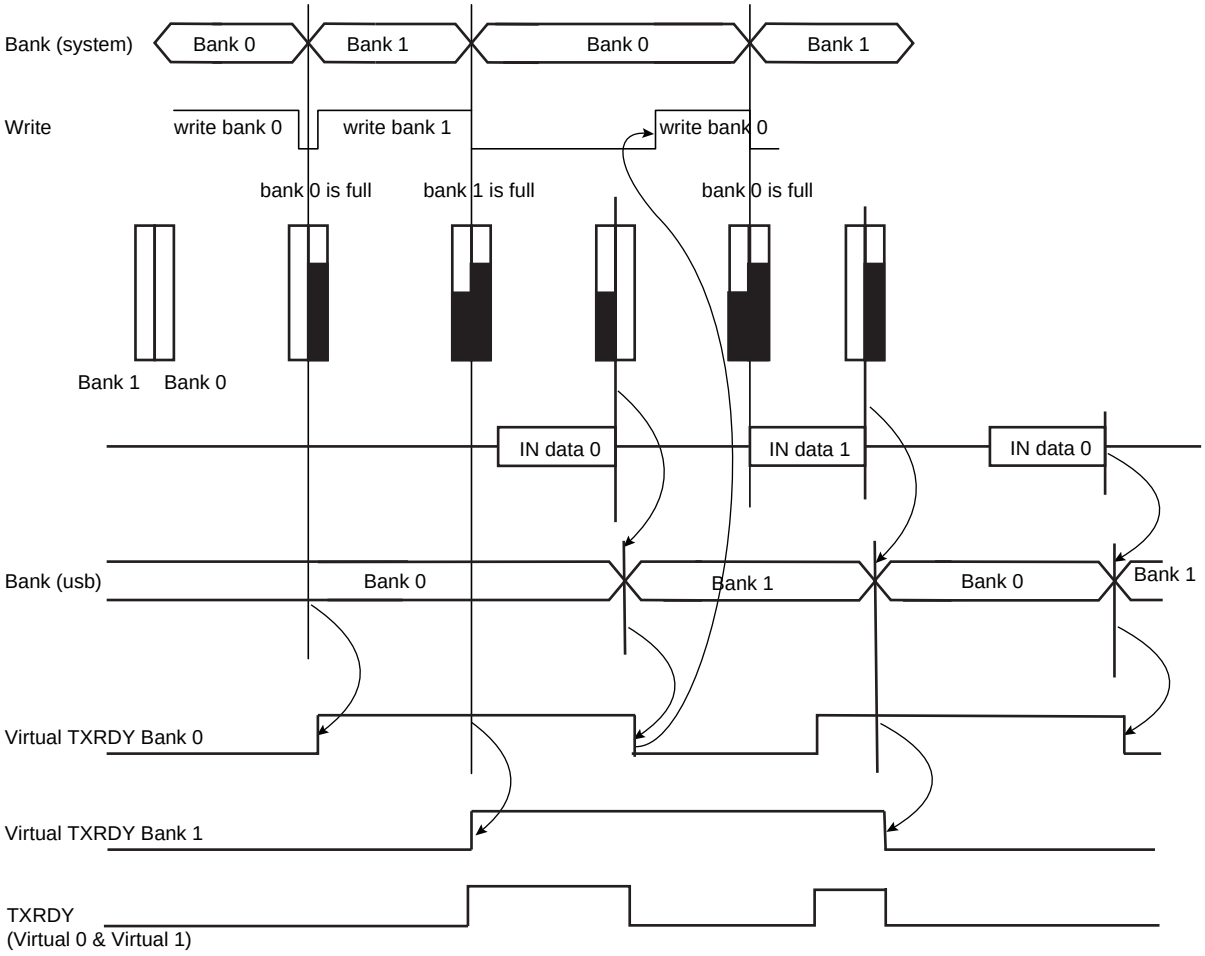
Note: A NAK handshake is always generated at the first status stage token.

Figure 34-12.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 34-13.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.

34.6.10.7 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.

34.6.10.8 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.

34.6.10.9 Data OUT

34.6.10.10 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.

34.6.10.11 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.

34.6.10.12 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 34.6.10.6 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 34-14.Data OUT Transfer for Endpoint with One Bank

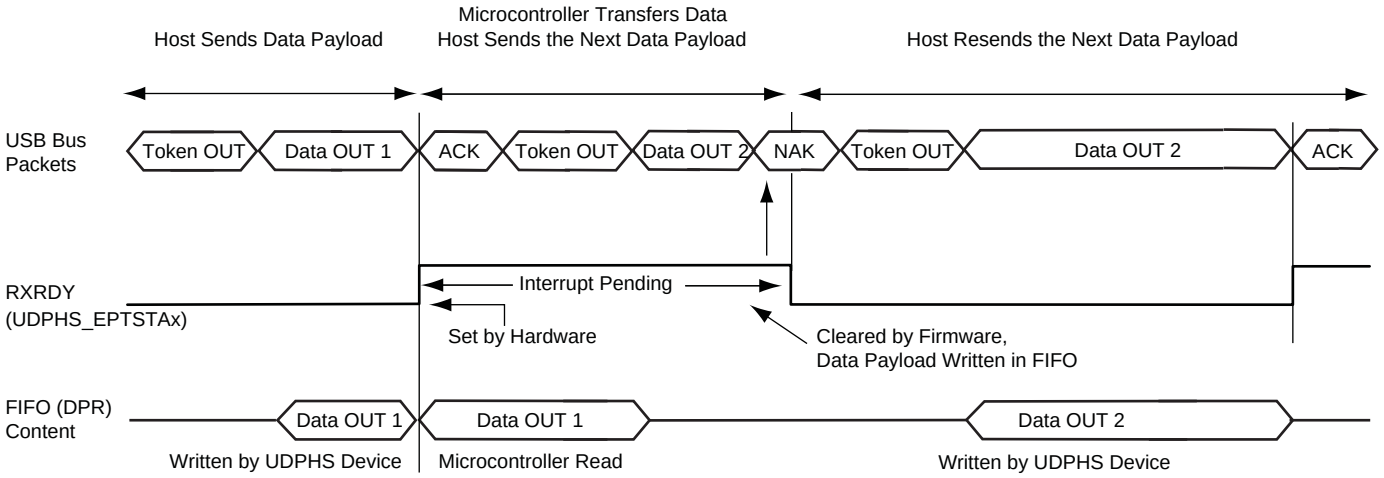
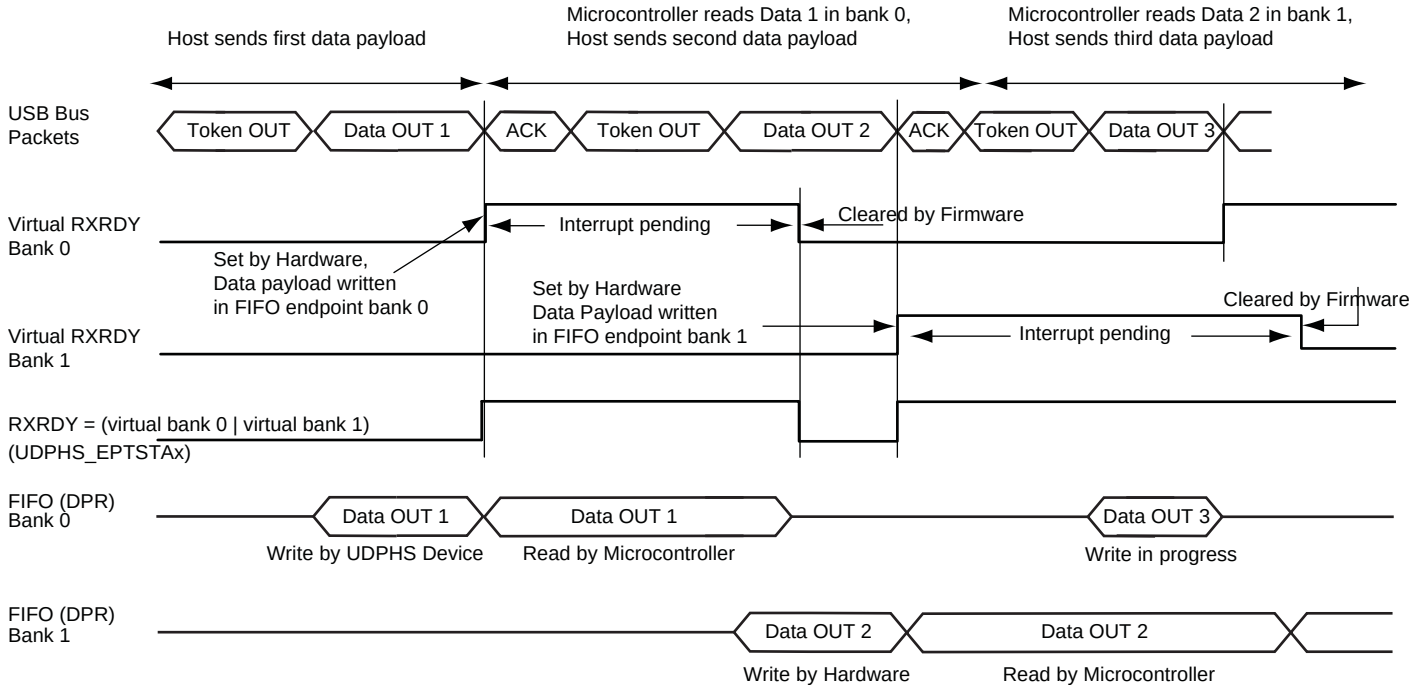
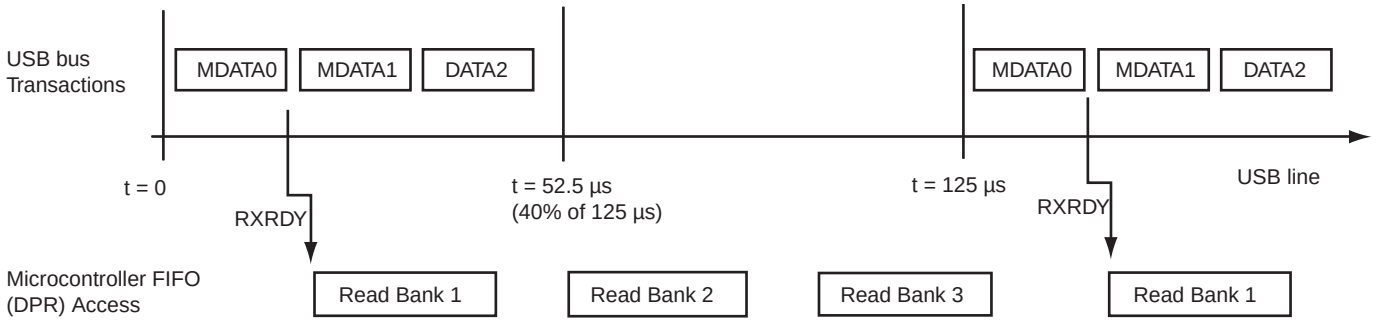


Figure 34-15.Data OUT Transfer for an Endpoint with Two Banks



34.6.10.13 High Bandwidth Isochronous Endpoint OUT

Figure 34-16.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

34.6.10.14 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 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.

34.6.10.15 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 34-17.Stall Handshake Data OUT Transfer

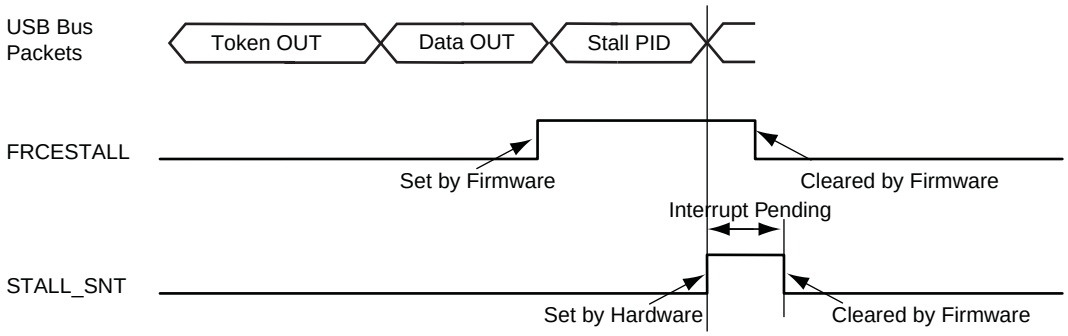
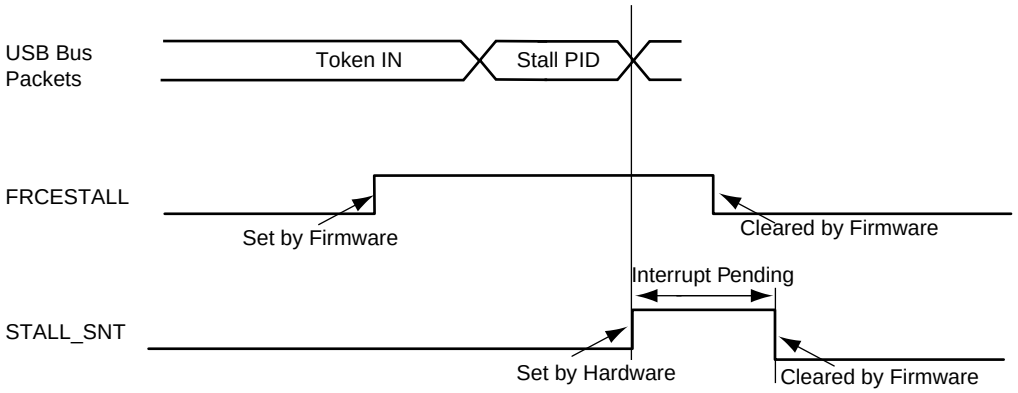


Figure 34-18.Stall Handshake Data IN Transfer



34.6.11 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.

34.6.12 USB V2.0 High Speed Global Interrupt

Interrupts are defined in Section 34.7.3 ”UDPHS Interrupt Enable Register” (UDPHS_IEN) and in Section 34.7.4 ”UDPHS Interrupt Status Register” (UDPHS_INTSTA).

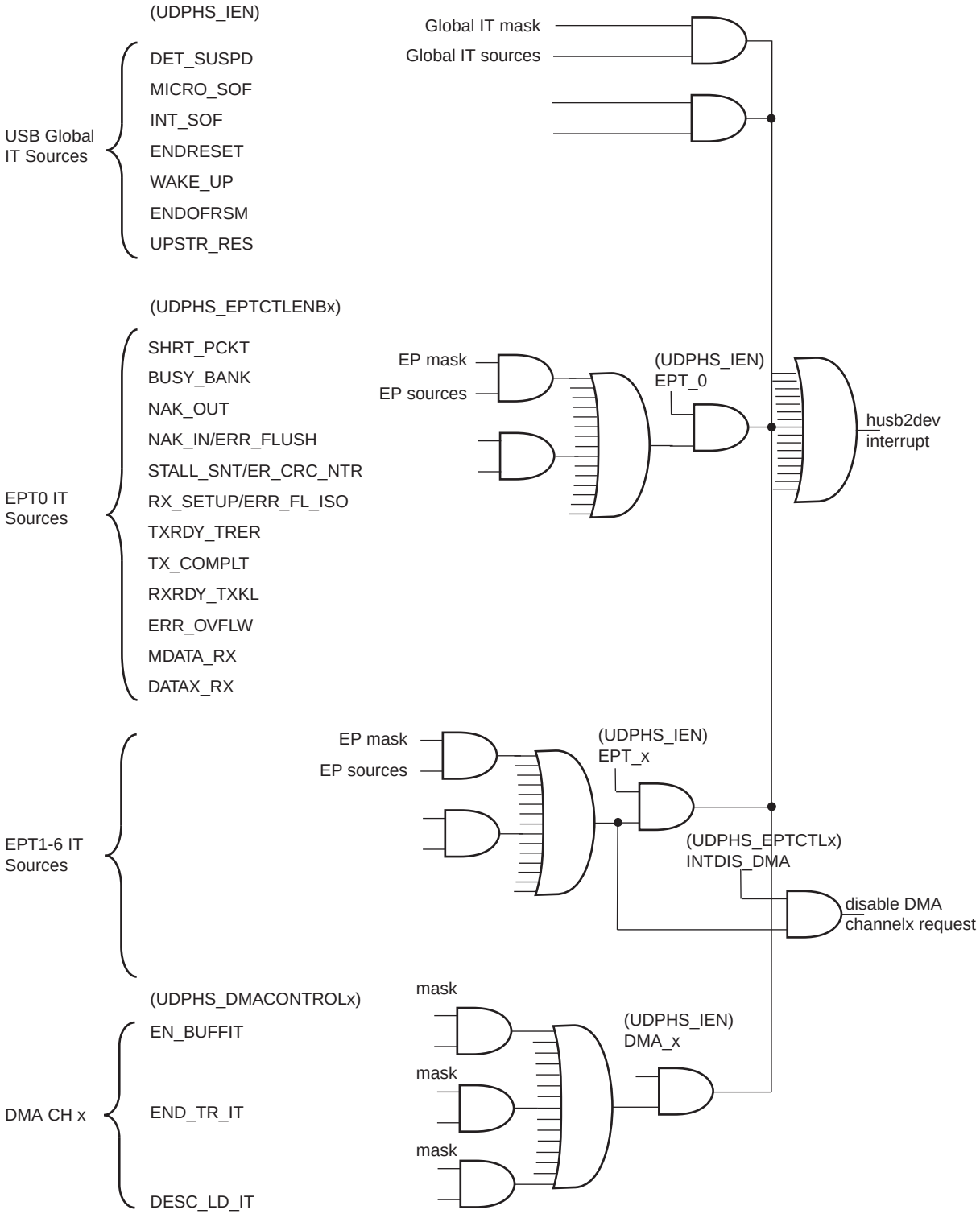
34.6.13 Endpoint Interrupts

Interrupts are enabled in UDPHS_IEN (see Section 34.7.3 ”UDPHS Interrupt Enable Register”) and individually masked in UDPHS_EPTCTLENBx (see Section 34.7.9 ”UDPHS Endpoint Control Enable Register (Control, Bulk, Interrupt Endpoints)”).

Table 34-5. 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 34-19.UDPHS Interrupt Control Interface

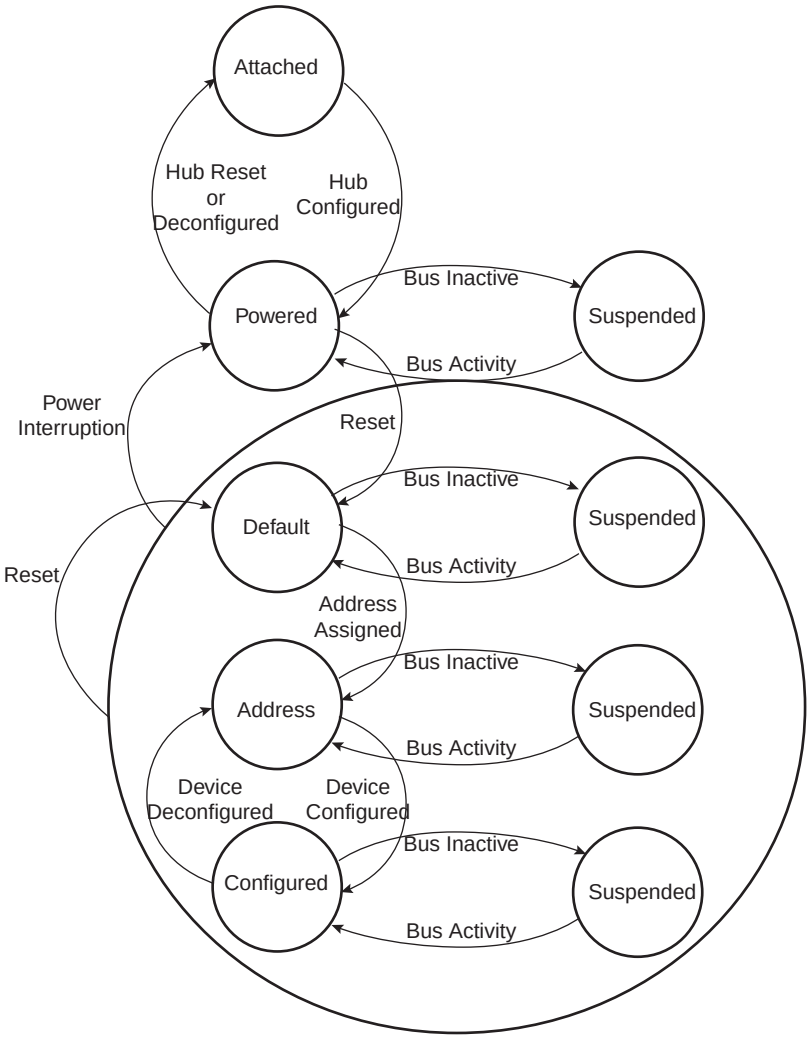


34.6.14 Power Modes

34.6.14.1 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 34-20.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.

34.6.14.2 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.

34.6.14.3 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.

34.6.14.4 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.

34.6.14.5 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.

34.6.14.6 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.

34.6.14.7 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.

34.6.14.8 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.

34.6.14.9 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.

34.6.15 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 34.7.7 “UDPHS Test Register” on page 886 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
};
```

34.7 USB High Speed Device Port (UDPHS) User Interface

Table 34-6. 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 - 0x2FC	UDPHS Endpoint1 to 15 (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 - 0x36C	DMA Channel1 to 6 (3) Registers			

Notes:
1. The reset value for UDPHS_EPTCTL0 is 0x0000_0001.
2. The addresses for the UDPHS Endpoint registers shown here are for UDPHS Endpoint0. The structure of this group of registers is repeated successively for each endpoint according to the consecution of endpoint registers located between 0x120 and 0x2FC.
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.

34.7.1 UDPHS Control Register

Name: UDPHS_CTRL

Address: 0xF8030000

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

34.7.2 UDPHS Frame Number Register

Name: UDPHS_FNUM

Address: 0xF8030004

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.

34.7.3 UDPHS Interrupt Enable Register

Name: UDPHS_IEN

Address: 0xF8030010

Access: Read-write

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
EPT_15	EPT_14	EPT_13	EPT_12	EPT_11	EPT_10	EPT_9	EPT_8
15	14	13	12	11	10	9	8
EPT_7	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0
7	6	5	4	3	2	1	0
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	–

- **DET_SUSPD: Suspend Interrupt Enable**

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.

34.7.4 UDPHS Interrupt Status Register

Name: UDPHS_INTSTA

Address: 0xF8030014

Access: Read-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
EPT_15	EPT_14	EPT_13	EPT_12	EPT_11	EPT_10	EPT_9	EPT_8
15	14	13	12	11	10	9	8
EPT_7	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0
7	6	5	4	3	2	1	0
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	SPEED

- **SPEED: Speed Status**

0: Reset by hardware when the hardware is in Full Speed mode.

1: Set by hardware when the hardware is in High Speed mode

- **DET_SUSPD: Suspend Interrupt**

0: Cleared by setting the DET_SUSPD bit in UDPHS_CLRINT register

1: Set by hardware when a UDPHS Suspend (Idle bus for three frame periods, a J state for 3 ms) is detected. This triggers a UDPHS interrupt when the DET_SUSPD bit is set in UDPHS_IEN register.

- **MICRO_SOF: Micro Start Of Frame Interrupt**

0: Cleared by setting the MICRO_SOF bit in UDPHS_CLRINT register.

1: Set by hardware when an UDPHS micro start of frame PID (SOF) has been detected (every 125 us) or synthesized by the macro. This triggers a UDPHS interrupt when the MICRO_SOF bit is set in UDPHS_IEN. In case of detected SOF, the MICRO_FRAME_NUM field in UDPHS_FNUM register is incremented and the FRAME_NUMBER field doesn't change.

Note: The Micro Start Of Frame Interrupt (MICRO_SOF), and the Start Of Frame Interrupt (INT_SOF) are not generated at the same time.

- **INT_SOF: Start Of Frame Interrupt**

0: Cleared by setting the INT_SOF bit in UDPHS_CLRINT.

1: Set by hardware when an UDPHS Start Of Frame PID (SOF) has been detected (every 1 ms) or synthesized by the macro. This triggers a UDPHS interrupt when the INT_SOF bit is set in UDPHS_IEN register. In case of detected SOF, in High Speed mode, the MICRO_FRAME_NUMBER field is cleared in UDPHS_FNUM register and the FRAME_NUMBER field is updated.

- **ENDRESET: End Of Reset Interrupt**

0: Cleared by setting the ENDRESET bit in UDPHS_CLRINT.

1: Set by hardware when an End Of Reset has been detected by the UDPHS controller. This triggers a UDPHS interrupt when the ENDRESET bit is set in UDPHS_IEN.

- **WAKE_UP: Wake Up CPU Interrupt**

0: Cleared by setting the WAKE_UP bit in UDPHS_CLRINT.

1: Set by hardware when the UDPHS controller is in SUSPEND state and is re-activated by a filtered non-idle signal from the UDPHS line (not by an upstream resume). This triggers a UDPHS interrupt when the WAKE_UP bit is set in UDPHS_IEN register. When receiving this interrupt, the user has to enable the device controller clock prior to operation.

Note: this interrupt is generated even if the device controller clock is disabled.

- **ENDOFRSM: End Of Resume Interrupt**

0: Cleared by setting the ENDOFRSM bit in UDPHS_CLRINT.

1: Set by hardware when the UDPHS controller detects a good end of resume signal initiated by the host. This triggers a UDPHS interrupt when the ENDOFRSM bit is set in UDPHS_IEN.

- **UPSTR_RES: Upstream Resume Interrupt**

0: Cleared by setting the UPSTR_RES bit in UDPHS_CLRINT.

1: Set by hardware when the UDPHS controller is sending a resume signal called “upstream resume”. This triggers a UDPHS interrupt when the UPSTR_RES bit is set in UDPHS_IEN.

- **EPT_x: Endpoint x Interrupt**

0: Reset when the UDPHS_EPTSTAx interrupt source is cleared.

1: Set by hardware when an interrupt is triggered by the UDPHS_EPTSTAx register and this endpoint interrupt is enabled by the EPT_x bit in UDPHS_IEN.

- **DMA_x: DMA Channel x Interrupt**

0: Reset when the UDPHS_DMASTATUSx interrupt source is cleared.

1: Set by hardware when an interrupt is triggered by the DMA Channelx and this endpoint interrupt is enabled by the DMA_x bit in UDPHS_IEN.

34.7.5 UDPHS Clear Interrupt Register

Name: UDPHS_CLRINT

Address: 0xF8030018

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.

34.7.6 UDPHS Endpoints Reset Register

Name: UDPHS_EPTRST

Address: 0xF803001C

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
EPT_15	EPT_14	EPT_13	EPT_12	EPT_11	EPT_10	EPT_9	EPT_8

7	6	5	4	3	2	1	0
EPT_7	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0

- **EPT_x: Endpoint x Reset**

0: No effect.

1: Reset the Endpointx state.

Setting this bit clears all bits in the Endpoint status UDPHS_EPTSTAx register except the TOGGLESQ_STA field.

34.7.7 UDPHS Test Register

Name: UDPHS_TST

Address: 0xF80300E0

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.

34.7.8 UDPHS Endpoint Configuration Register

Name: UDPHS_EPTCFGx [x=0..15]

Address: 0xF8030100 [0], 0xF8030120 [1], 0xF8030140 [2], 0xF8030160 [3], 0xF8030180 [4], 0xF80301A0 [5], 0xF80301C0 [6], 0xF80301E0 [7], 0xF8030200 [8], 0xF8030220 [9], 0xF8030240 [10], 0xF8030260 [11], 0xF8030280 [12], 0xF80302A0 [13], 0xF80302C0 [14], 0xF80302E0 [15]

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 34.6.6 ”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 34.6.6 ”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 34.6.6 ”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

34.7.9 UDPHS Endpoint Control Enable Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCTLENBx [x=0..15]

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 888.

For additional information, see “UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)” on page 899.

- **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.

34.7.10 UDPHS Endpoint Control Enable Register (Isochronous Endpoints)

Name: UDPHS_EPTCTLENBx [x=0..15] (ISOENDPT)

Address: 0xF8030104 [0], 0xF8030124 [1], 0xF8030144 [2], 0xF8030164 [3], 0xF8030184 [4], 0xF80301A4 [5], 0xF80301C4 [6], 0xF80301E4 [7], 0xF8030204 [8], 0xF8030224 [9], 0xF8030244 [10], 0xF8030264 [11], 0xF8030284 [12], 0xF80302A4 [13], 0xF80302C4 [14], 0xF80302E4 [15]

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 888.

For additional information, see “UDPHS Endpoint Control Register (Isochronous Endpoint)” on page 902.

- **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.

34.7.11 UDPHS Endpoint Control Disable Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCTLDISx [x=0..15]

Address: 0xF8030108 [0], 0xF8030128 [1], 0xF8030148 [2], 0xF8030168 [3], 0xF8030188 [4], 0xF80301A8 [5], 0xF80301C8 [6], 0xF80301E8 [7], 0xF8030208 [8], 0xF8030228 [9], 0xF8030248 [10], 0xF8030268 [11], 0xF8030288 [12], 0xF80302A8 [13], 0xF80302C8 [14], 0xF80302E8 [15]

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 888.

For additional information, see “UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)” on page 899.

- **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.

34.7.12 UDPHS Endpoint Control Disable Register (Isochronous Endpoint)

Name: UDPHS_EPTCTLDISx [x=0..15] (ISOENDPT)

Address: 0xF8030108 [0], 0xF8030128 [1], 0xF8030148 [2], 0xF8030168 [3], 0xF8030188 [4], 0xF80301A8 [5], 0xF80301C8 [6], 0xF80301E8 [7], 0xF8030208 [8], 0xF8030228 [9], 0xF8030248 [10], 0xF8030268 [11], 0xF8030288 [12], 0xF80302A8 [13], 0xF80302C8 [14], 0xF80302E8 [15]

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 888.

For additional information, see “UDPHS Endpoint Control Register (Isochronous Endpoint)” on page 902.

- **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.

34.7.13 UDPHS Endpoint Control Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCTLx [x=0..15]

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 888.

- **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.

34.7.14 UDPHS Endpoint Control Register (Isochronous Endpoint)

Name: UDPHS_EPTCTLx [x=0..15] (ISOENDPT)

Address: 0xF803010C [0], 0xF803012C [1], 0xF803014C [2], 0xF803016C [3], 0xF803018C [4], 0xF80301AC [5], 0xF80301CC [6], 0xF80301EC [7], 0xF803020C [8], 0xF803022C [9], 0xF803024C [10], 0xF803026C [11], 0xF803028C [12], 0xF80302AC [13], 0xF80302CC [14], 0xF80302EC [15]

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 888.

- **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.

34.7.15 UDPHS Endpoint Set Status Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTSETSTAx [x=0..15]

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 888.

For additional information, see “UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)” on page 911.

- **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.

34.7.16 UDPHS Endpoint Set Status Register (Isochronous Endpoint)

Name: UDPHS_EPTSETSTAx [x=0..15] (ISOENDPT)

Address: 0xF8030114 [0], 0xF8030134 [1], 0xF8030154 [2], 0xF8030174 [3], 0xF8030194 [4], 0xF80301B4 [5], 0xF80301D4 [6], 0xF80301F4 [7], 0xF8030214 [8], 0xF8030234 [9], 0xF8030254 [10], 0xF8030274 [11], 0xF8030294 [12], 0xF80302B4 [13], 0xF80302D4 [14], 0xF80302F4 [15]

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 888.

For additional information, see “UDPHS Endpoint Status Register (Isochronous Endpoint)” on page 914.

- **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.

34.7.17 UDPHS Endpoint Clear Status Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTCLRSTAx [x=0..15]

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 888.

For additional information, see “UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)” on page 911.

- **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.

34.7.18 UDPHS Endpoint Clear Status Register (Isochronous Endpoint)

Name: UDPHS_EPTCLRSTAx [x=0..15] (ISOENDPT)

Address: 0xF8030118 [0], 0xF8030138 [1], 0xF8030158 [2], 0xF8030178 [3], 0xF8030198 [4], 0xF80301B8 [5], 0xF80301D8 [6], 0xF80301F8 [7], 0xF8030218 [8], 0xF8030238 [9], 0xF8030258 [10], 0xF8030278 [11], 0xF8030298 [12], 0xF80302B8 [13], 0xF80302D8 [14], 0xF80302F8 [15]

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 888.

For additional information, see “UDPHS Endpoint Status Register (Isochronous Endpoint)” on page 914.

- **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.

34.7.19 UDPHS Endpoint Status Register (Control, Bulk, Interrupt Endpoints)

Name: UDPHS_EPTSTAx [x=0..15]

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 888.

- **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).

34.7.20 UDPHS Endpoint Status Register (Isochronous Endpoint)

Name: UDPHS_EPTSTAx [x=0..15] (ISOENDPT)

Address: 0xF803011C [0], 0xF803013C [1], 0xF803015C [2], 0xF803017C [3], 0xF803019C [4], 0xF80301BC [5], 0xF80301DC [6], 0xF80301FC [7], 0xF803021C [8], 0xF803023C [9], 0xF803025C [10], 0xF803027C [11], 0xF803029C [12], 0xF80302BC [13], 0xF80302DC [14], 0xF80302FC [15]

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 888.

- **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 916) 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).

34.7.21 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.

34.7.22 UDPHS DMA Next Descriptor Address Register

Name: UDPHS_DMANXTDSCx [x = 0..6]

Address: 0xF8030300 [0], 0xF8030310 [1], 0xF8030320 [2], 0xF8030330 [3], 0xF8030340 [4], 0xF8030350 [5], 0xF8030360 [6]

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.

34.7.23 UDPHS DMA Channel Address Register

Name: UDPHS_DMAADDRESSx [x = 0..6]

Address: 0xF8030304 [0], 0xF8030314 [1], 0xF8030324 [2], 0xF8030334 [3], 0xF8030344 [4], 0xF8030354 [5], 0xF8030364 [6]

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.

34.7.24 UDPHS DMA Channel Control Register

Name: UDPHS_DMACONTROLx [x = 0..6]

Address: 0xF8030308 [0], 0xF8030318 [1], 0xF8030328 [2], 0xF8030338 [3], 0xF8030348 [4], 0xF8030358 [5], 0xF8030368 [6]

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”.

34.7.25 UDPHS DMA Channel Status Register

Name: UDPHS_DMASTATUSx [x = 0..6]

Address: 0xF803030C [0], 0xF803031C [1], 0xF803032C [2], 0xF803033C [3], 0xF803034C [4], 0xF803035C [5], 0xF803036C [6]

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 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.

35. USB Host High Speed Port (UHPHS)

35.1 Description

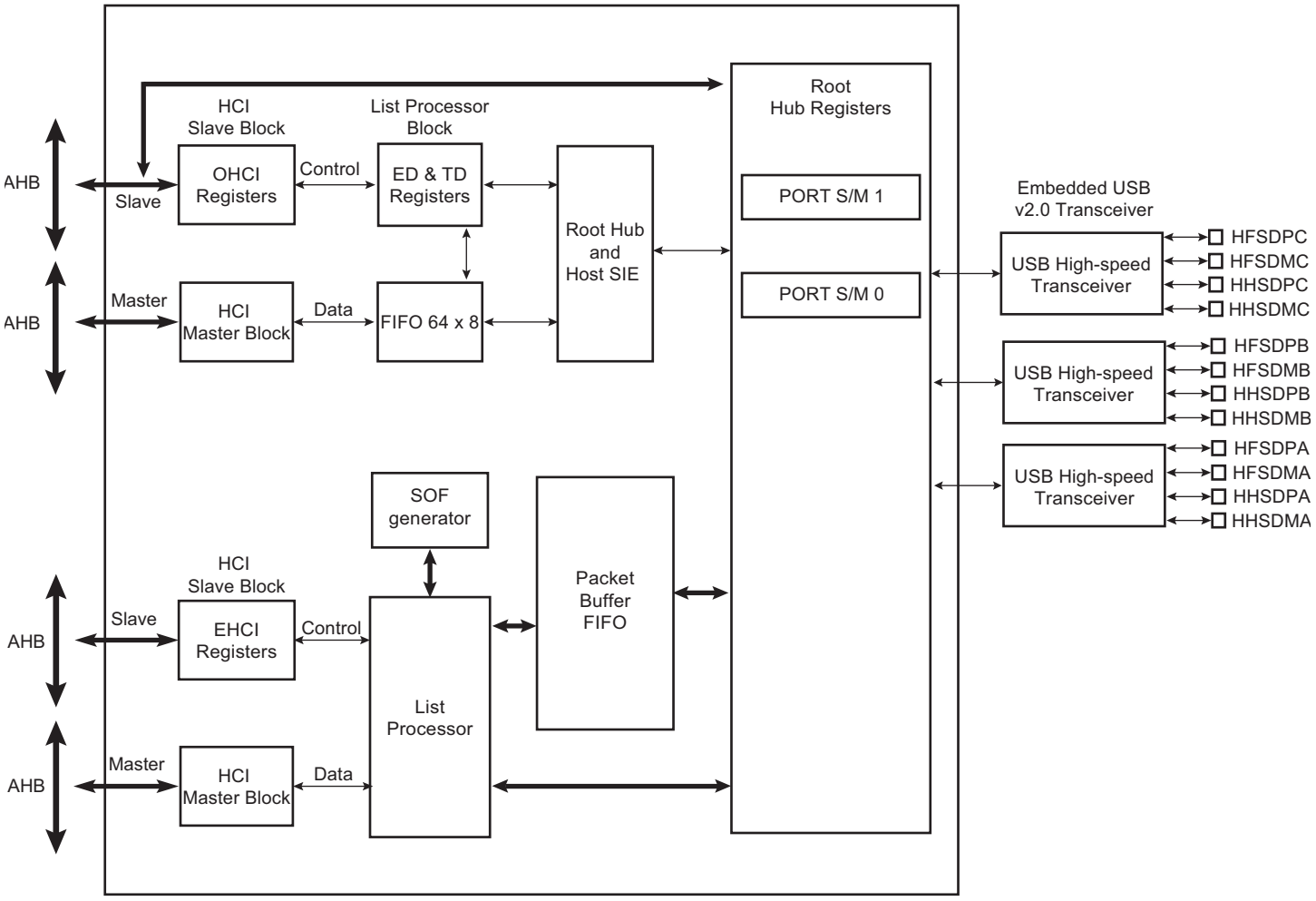
The USB Host High Speed Port (UHPHS) interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as Enhanced HCI protocol (Enhanced Host Controller Interface).

35.2 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 3 Downstream USB HS Ports
- Embedded USB Transceivers
- Supports Power Management
- 3 Hosts (A and B) High Speed (EHCI), Port A shared with UDPHS

35.3 Block Diagram

Figure 35-1. 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.

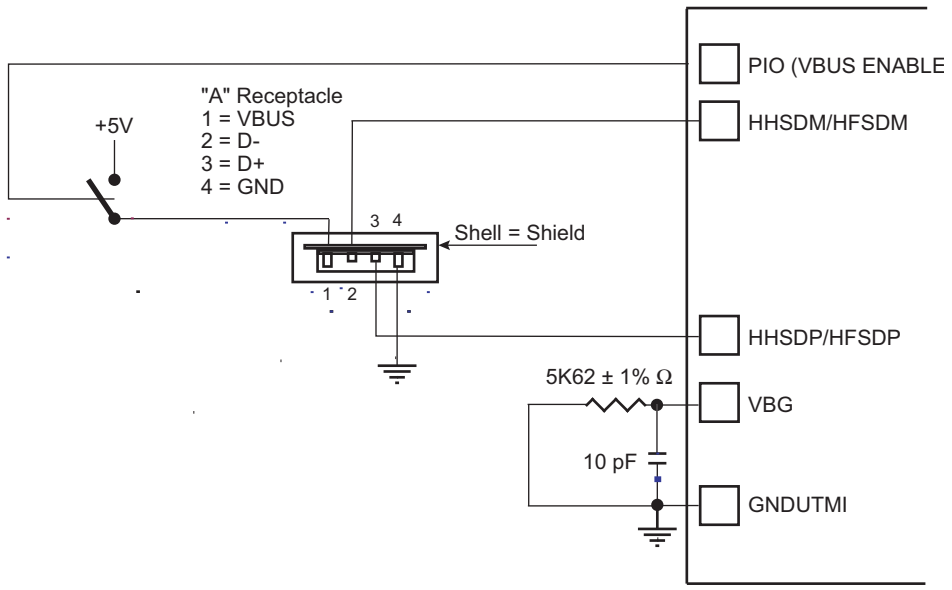
The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub’s operational registers. Device connection is automatically detected by the USB host port logic.

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.

35.4 Typical Connection

Figure 35-2. Board Schematic to Interface UHP High-speed Host Controller



35.5 Product Dependencies

35.5.1 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 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.

35.5.2 Power Management

The system embeds 3 transceivers.

The USB Host High Speed requires a 480 MHz clock for the embedded High-speed transceivers. This clock (UPLLCK) is provided by the UTMI PLL.

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.

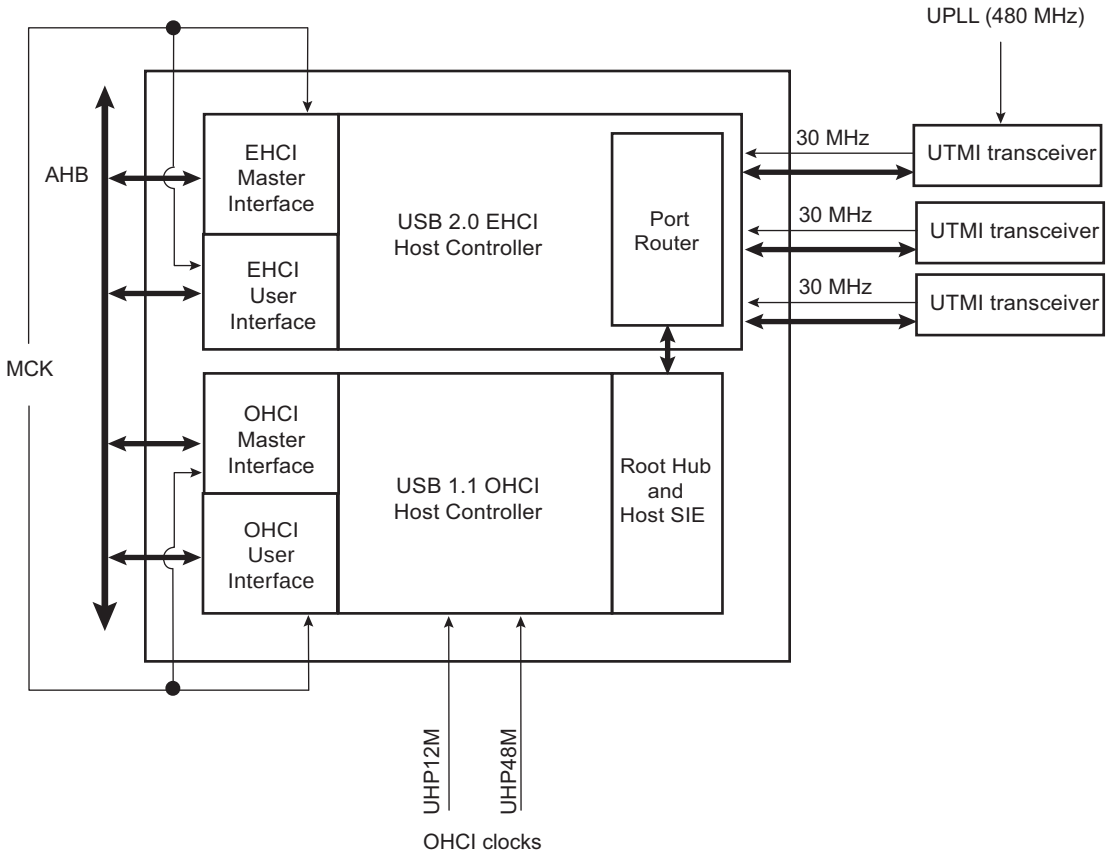
For High-speed operations, the user has to perform the following:

- Enable UHP peripheral clock, bit (1 << AT91C_ID_UHPHS) in PMC_PCER.
- Write UPLLCOUNT field in CKGR_UCKR.
- Enable UPLL, bit AT91C_CKGR_UPLLEN in CKGR_UCKR.
- Wait until UTMI PLL is locked. LOCKU bit set in PMC_SR.
- Enable BIAS, bit AT91C_CKGR_BIASEN in CKGR_UCKR.
- Select UPLLCK as Input clock of OHCI part (set 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 (set UHP bit in PMC_SCER).

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.
- Select PLLACK as Input clock of OHCI part (clear 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 (set UHP bit in PMC_SCER).

Figure 35-3. UHP Clock Trees



35.5.3 Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

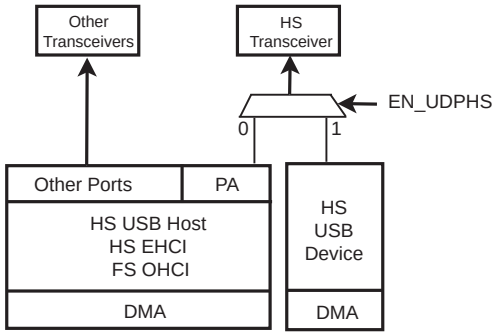
Handling USB host interrupts requires programming the AIC before configuring the UHPHS.

35.6 Functional Description

35.6.1 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 register.

Figure 35-4. USB Selection



35.6.2 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.

35.6.3 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.

36. Gigabit Ethernet MAC (GMAC)

36.1 Description

The Gigabit Ethernet MAC (GMAC) module implements a 10/100/1000 Mbps Ethernet MAC compatible with the IEEE 802.3 standard. The GMAC can operate in either half or full duplex mode at all supported speeds. The “Network Configuration Register” is used to select the speed, duplex mode and interface type (MII, GMII or RGMII).

The GMAC comprises two constituent components:

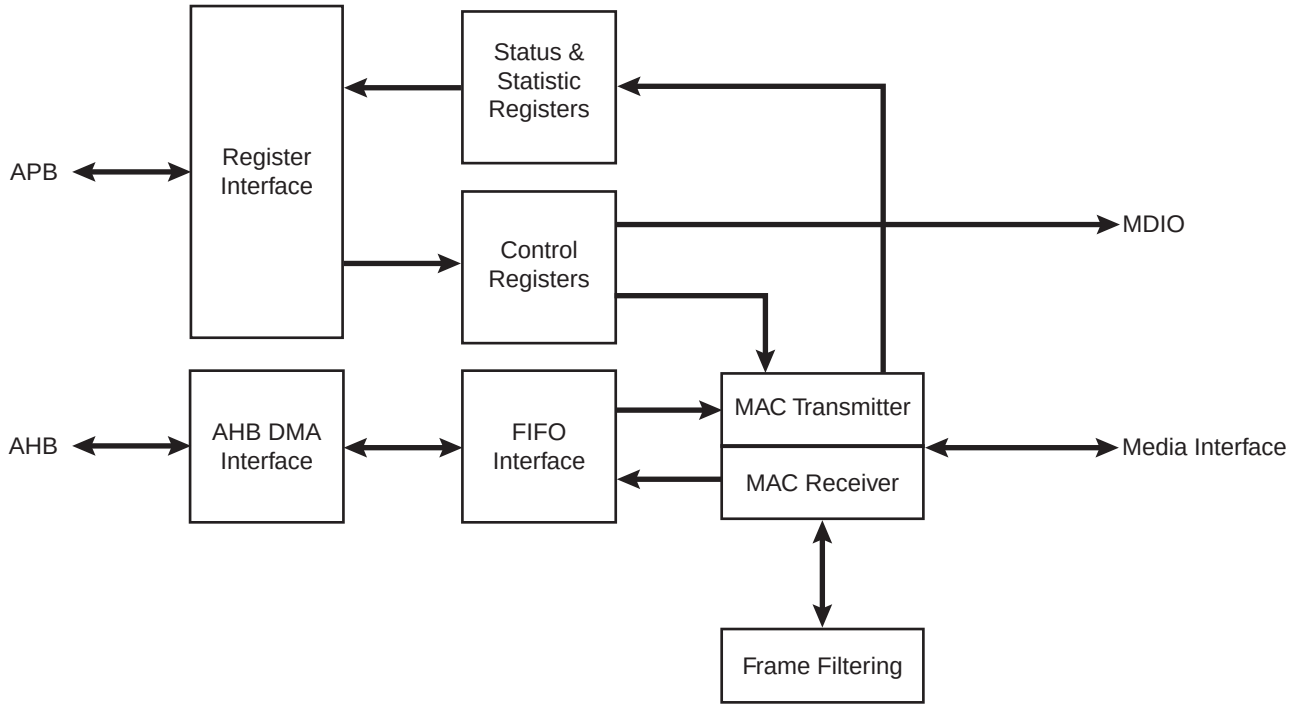
- GEM_MAC controlling transmit, receive, address checking and loopback
- GEM_TSU calculates the IEEE 1588 timer values

36.2 Embedded Characteristics

- Compatible with IEEE Standard 802.3
- 10, 100 and 1000 Mbps operation
- Full and half duplex operation at all supported speeds of operation
- Statistics Counter Registers for RMON/MIB
- MII/GMII/RGMII interface to the physical layer
- Integrated physical coding
- Direct memory access (DMA) interface to external memory
- Programmable burst length and endianism for DMA
- Interrupt generation to signal receive and transmit completion, or errors
- Automatic pad and cyclic redundancy check (CRC) generation on transmitted frames
- Frame extension and frame bursting at 1000 Mbps in half duplex mode
- Automatic discard of frames received with errors
- Receive and transmit IP, TCP and UDP checksum offload. Both IPv4 and IPv6 packet types supported
- Address checking logic for four specific 48-bit addresses, four type IDs, promiscuous mode, hash matching of unicast and multicast destination addresses and Wake-on-LAN
- Management Data Input/Output (MDIO) interface for physical layer management
- Support for jumbo frames up to 10240 bytes
- Full duplex flow control with recognition of incoming pause frames and hardware generation of transmitted pause frames
- Half duplex flow control by forcing collisions on incoming frames
- Support for 802.1Q VLAN tagging with recognition of incoming VLAN and priority tagged frames
- Support for 802.1Qbb priority-based flow control
- Programmable Inter Packet Gap (IPG) Stretch
- Recognition of IEEE 1588 PTP frames
- IEEE 1588 time stamp unit (TSU)
- Support for 802.1AS timing and synchronization

36.3 Block Diagram

Figure 36-1. Block Diagram



36.4 Signal Interface

The GMAC includes the following signal interfaces

- MII, GMII and RGMII to an external PHY
- MDIO interface for external PHY management
- Slave APB interface for accessing GMAC registers
- Master AHB interface for memory access

Table 36-1. GMAC connections in the different modes

Signal Name	Function	MII	GMII	RGMII
GTXCK	Transmit Clock or Reference Clock	TXCK	Not Used	TXCK
G125CK	125 MHz input Clock	Not Used	125 MHz Ref Clk	125 MHz Ref Clk
G125CKO	125 MHz output Clock	Not Used	TXCK	Not Used
GTXEN	Transmit Enable	TXEN	TXEN	TXCTL
GTX[7..0]	Transmit Data	TXD[3:0]	TXD[7:0]	TXD[3:0]
GTXER	Transmit Coding Error	TXER	TXER	Not Used
GRXCK	Receive Clock	RXCK	RXCK	RXCK
GRXDV	Receive Data Valid	RXDV	RXDV	Not Used
GRX[7..0]	Receive Data	RXD[3:0]	RXD[7:0]	RXD[3:0]
GRXER	Receive Error	RXER	RXER	RXCTL
GCRS	Carrier Sense and Data Valid	CRS	CRS	Not Used
GCOL	Collision Detect	COL	COL	Not Used
GMDC	Management Data Clock	MDC	MDC	MDC
GMDIO	Management Data Input/Output	MDIO	MDIO	MDIO

36.5 Functional Description

36.5.1 Media Access Controller

The Media Access Controller (MAC) transmit block takes data from FIFO, adds preamble and, if necessary, pad and frame check sequence (FCS). Both half duplex and full duplex Ethernet modes of operation are supported. When operating in half duplex mode, the MAC transmit block generates data according to the carrier sense multiple access with collision detect (CSMA/CD) protocol. The start of transmission is deferred if carrier sense (CRS) is active. If collision (COL) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. The CRS and COL signals have no effect in full duplex mode. When operating in gigabit mode half duplex, both carrier extension and frame bursting are performed in accordance with the IEEE 802.3 standard.

The MAC receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the MAC address checking block and FIFO. Software can configure the GMAC to receive jumbo frames up to 10240 bytes. It can optionally strip CRC from the received frame prior to transfer to FIFO.

The address checker recognizes four specific 48-bit addresses, can recognize four different type ID values, and contains a 64-bit Hash Register for matching multicast and unicast addresses as required. It can recognize the broadcast address of all ones and copy all frames. The MAC can also reject all frames that are not VLAN tagged and recognize Wake on LAN events.

The MAC receive block supports offloading of IP, TCP and UDP checksum calculations (both IPv4 and IPv6 packet types supported), and can automatically discard bad checksum frames.

36.5.2 1588 Time Stamp Unit

The 1588 time stamp unit (TSU) is a timer implemented as a 62-bit register. The 32 upper bits count seconds and the 30 lower bits count nanoseconds. The 30 lower bits roll over when they have counted to one second. The timer increments by a programmable number of nanoseconds with each MCK period and can be adjusted (incremented or decremented) through APB register accesses.

36.5.3 AHB Direct Memory Access Interface

GMAC is supplied with an AHB DMA interface. When the GMAC is configured to use the DMA, it is attached to the MAC module's FIFO to provide a scatter gather type capability for packet data storage in the embedded processor system or System on Chip.

The GMAC DMA controller performs six types of operation on the AHB bus. When the GMAC DMA is configured in internal FIFO mode, in order of priority these are as follows:

- Receive buffer manager write/read
- Transmit buffer manager write/read
- Receive data DMA write
- Transmit data DMA read

The DMA is configured in packet buffer mode, where dual-port memories are used to buffer multiple frames. The packet buffer DMA features follow.

36.5.3.1 Packet Buffer DMA

- Easier to guarantee maximum line rate due to the ability to store multiple frames in the packet buffer
- Full store and forward, or partial store and forward programmable options (partial store will cater for shorter latency requirements)
- Support for Transmit TCP/IP checksum offload
- Support for priority queueing
- When a collision on the line occurs during transmission, the packet will be automatically replayed directly from the packet buffer memory rather than having to re-fetch through the AHB (full store and forward ONLY)
- Received errored packets are automatically dropped before any of the packet is presented to the AHB (full store and forward ONLY), thus reducing AHB activity
- Supports manual RX packet flush capabilities
- Optional RX packet flush when there is lack of AHB resource

36.5.3.2 Partial Store and Forward Using Packet Buffer DMA

When the DMA is configured to use SRAM-based packet buffers, it can be programmed into a low latency mode, known as partial store and forward. This mode allows the integrator to attach small SRAMs to the GMAC DMA, smaller than the maximum sized packets to be transmitted/received. This allows for a reduced latency as the full packet is not buffered before forwarding. Note that this option is only available when the device is configured for full duplex operation. This feature is enabled via the TX and RX partial store and forward programmable registers. When the transmit partial store and forward mode is activated, the transmitter will only begin to forward the packet to the MAC when there is enough packet data stored in the packet buffer. Likewise, when the receive partial store and forward mode is activated, the receiver will only begin to forward the packet to the AHB when enough packet data is stored in the packet buffer. The amount of packet data required to activate the forwarding process is programmable via watermark registers which are located at the same address as the partial store and forward enable bits. Note that the minimum operational value for the TX partial store and forward watermark is 20. There is no operational limit for the RX partial store and forward watermark. Enabling partial store and forward is a useful means to reduce latency and reduce the size of the attached SRAMs, but there are performance implications. The GMAC DMA uses separate transmit and receive lists of buffer descriptors, with each descriptor describing a buffer area in memory. This allows Ethernet packets to be broken up and scattered around the AHB memory space.

36.5.3.3 Receive AHB Buffers

Received frames, optionally including FCS, are written to receive AHB buffers stored in memory. The receive buffer depth is programmable in the range of 64 bytes to 16 Kbytes through the DMA Configuration Register, with the default being 128 bytes.

The start location for each receive AHB buffer is stored in memory in a list of receive buffer descriptors at an address location pointed to by the receive buffer queue pointer. The base address for the receive buffer queue pointer is configured in software using the Receive Buffer Queue Base Address Register.

Each list entry consists of two words. The first is the address of the receive AHB buffer and the second the receive status. If the length of a receive frame exceeds the AHB buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit, which is always set for the first buffer in a frame. Bit zero of the address field is written to 1 to show the buffer has been used. The receive buffer manager then reads the location of the next receive AHB buffer and fills that with the next part of the received frame data. AHB buffers are filled until the frame is complete and the final buffer descriptor status word contains the complete frame status. Refer to Table 36-2, “Receive Buffer Descriptor Entry” for details of the receive buffer descriptor list.

Each receive AHB buffer start location is a word address. The start of the first AHB buffer in a frame 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 AHB buffer is offset, the available length of the first AHB buffer is reduced by the corresponding number of bytes.

Table 36-2. 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 GMAC to write data to the receive buffer. The GMAC 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	–
27	Specific Address Register match found, bit 25 and bit 26 indicate which Specific Address Register causes the match.
26:25	Specific Address Register match. Encoded as follows: 00: Specific Address Register 1 match 01: Specific Address Register 2 match 10: Specific Address Register 3 match 11: Specific Address Register 4 match If more than one specific address is matched only one is indicated with priority 4 down to 1.

Table 36-2. Receive Buffer Descriptor Entry (Continued)

Bit	Function
24	This bit has a different meaning depending on whether RX checksum offloading is enabled. **With RX checksum offloading disabled:** (bit 24 clear in Network Configuration Register) Type ID register match found, bit 22 and bit 23 indicate which type ID register causes the match. **With RX checksum offloading enabled:** (bit 24 set in Network Configuration Register) 0: The frame was not SNAP encoded and/or had a VLAN tag with the Canonical Format Indicator (CFI) bit set. 1: The frame was SNAP encoded and had either no VLAN tag or a VLAN tag with the CFI bit not set.
23:22	This bit has a different meaning depending on whether RX checksum offloading is enabled. **With RX checksum offloading disabled:** (bit 24 clear in Network Configuration) Type ID register match. Encoded as follows: 00: Type ID register 1 match 01: Type ID register 2 match 10: Type ID register 3 match 11: Type ID register 4 match If more than one Type ID is matched only one is indicated with priority 4 down to 1. **With RX checksum offloading enabled:** (bit 24 set in Network Configuration Register) 00: Neither the IP header checksum nor the TCP/UDP checksum was checked. 01: The IP header checksum was checked and was correct. Neither the TCP nor UDP checksum was checked. 10: Both the IP header and TCP checksum were checked and were correct. 11: Both the IP header and UDP checksum were checked and were correct.
21	VLAN tag detected—type ID of 0x8100. For packets incorporating the stacked VLAN processing feature, this bit will be set if the second VLAN tag has a type ID of 0x8100
20	Priority tag detected—type ID of 0x8100 and null VLAN identifier. For packets incorporating the stacked VLAN processing feature, this bit will be set if the second VLAN tag has a type ID of 0x8100 and a null VLAN identifier.
19:17	VLAN priority—only valid if bit 21 is set.
16	Canonical 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 valid status bit is start of frame (bit 14).
14	Start of frame—when set the buffer contains the start of a frame. If both bits 15 and 14 are set, the buffer contains a whole frame.
13	This bit has a different meaning depending on whether jumbo frames and ignore FCS modes are enabled. If neither mode is enabled this bit will be zero. **With jumbo frame mode enabled:** (bit 3 set in Network Configuration Register) Additional bit for length of frame (bit[13]), that is concatenated with bits[12:0] **With ignore FCS mode enabled and jumbo frames disabled:** (bit 26 set in Network Configuration Register and bit 3 clear in Network Configuration Register) This indicates per frame FCS status as follows: 0: Frame had good FCS 1: Frame had bad FCS, but was copied to memory as ignore FCS enabled.

Table 36-2. Receive Buffer Descriptor Entry (Continued)

Bit	Function
12:0	These bits represent the length of the received frame which may or may not include FCS depending on whether FCS discard mode is enabled. **With FCS discard mode disabled:** (bit 17 clear in Network Configuration Register) Least significant 12 bits for length of frame including FCS. If jumbo frames are enabled, these 12 bits are concatenated with bit[13] of the descriptor above. **With FCS discard mode enabled:** (bit 17 set in Network Configuration Register) Least significant 12 bits for length of frame excluding FCS. If jumbo frames are enabled, these 12 bits are concatenated with bit[13] of the descriptor above.

To receive frames, the AHB buffer descriptors must be initialized by writing an appropriate address to bits 31:2 in the first word of each list entry. Bit 0 must be written with zero. Bit 1 is the wrap bit and indicates the last entry in the buffer descriptor list.

The start location of the receive buffer descriptor list must be written with the receive buffer queue base address before reception is enabled (receive enable in the Network Control Register). Once reception is enabled, any writes to the Receive Buffer Queue Base Address Register are ignored. When read, it will return the current pointer position in the descriptor list, though this is only valid and stable when receive is disabled.

If the filter block indicates that a 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.

An internal counter within the GMAC represents the receive buffer queue pointer and it is not visible through the CPU interface. The receive buffer queue pointer increments by two words after each buffer has been used. It re-initializes to the receive buffer queue base address if any descriptor has its wrap bit set.

As receive AHB buffers are used, the receive AHB buffer manager sets bit zero of the first word of the descriptor to logic one indicating the AHB buffer has been used.

Software should search through the “used” bits in the AHB buffer descriptors to find out how many frames have been received, checking the start of frame and end of frame bits.

When the DMA is configured in the packet buffer partial store and forward mode, received frames are written out to the AHB buffers as soon as enough frame data exists in the packet buffer. For both cases, this may mean several full AHB buffers are used before some error conditions can be detected. If a receive error is detected the receive buffer currently being written will be recovered. Previous buffers will not be recovered. As an example, when receiving frames with cyclic redundancy check (CRC) errors or excessive length, it is possible that a frame fragment might be stored in a sequence of AHB 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 10/100/1000 Ethernet system there should be no excessive length frames or frames greater than 128 bytes with CRC errors. Collision fragments will be less than 128 bytes long, therefore it will be a rare occurrence to find a frame fragment in a receive AHB buffer, when using the default value of 128 bytes for the receive buffers size.

When in packet buffer full store and forward mode, only good received frames are written out of the DMA, so no fragments will exist in the AHB buffers due to MAC receiver errors. There is still the possibility of fragments due to DMA errors, for example used bit read on the second buffer of a multi-buffer frame.

If bit zero of the receive buffer descriptor is already set when the receive buffer manager reads the location of the receive AHB buffer, then the buffer has been already used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the “buffer not available” bit in the receive status register is set and an interrupt triggered. The receive resource error statistics register is also incremented.

When the DMA is configured in the packet buffer full store and forward mode, the user can optionally select whether received frames should be automatically discarded when no AHB buffer resource is available. This feature is selected via bit 24 of the DMA Configuration Register (by default, the received frames are not automatically discarded). If this feature

is off, then received packets will remain to be stored in the SRAM-based packet buffer until AHB buffer resource next becomes available. This may lead to an eventual packet buffer overflow if packets continue to be received when bit zero (used bit) of the receive buffer descriptor remains set. Note that after a used bit has been read, the receive buffer manager will reread the location of the receive buffer descriptor every time a new packet is received. When the DMA is not configured in the packet buffer full store and forward mode and a used bit is read, the frame currently being received will be automatically discarded.

When the DMA is configured in the packet buffer full store and forward mode, a receive overrun condition occurs when the receive SRAM-based packet buffer is full, or because HRESP was not OK. In all other modes, a receive overrun condition occurs when either the AHB bus was not granted quickly enough, or because HRESP was not OK, or because a new frame has been detected by the receive block, but the status update or write back for the previous frame has not yet finished. For a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame that is received whose address is recognized reuses the buffer.

When the DMA is configured for packet buffer mode, a write to bit 18 of the Network Control Register will force a packet from the external SRAM-based receive packet buffer to be flushed. This feature is only acted upon when the RX DMA is not currently writing packet data out to AHB, i.e., it is in an IDLE state. If the RX DMA is active, a write to this bit is ignored.

36.5.3.4 Transmit AHB Buffers

Frames to transmit are stored in one or more transmit AHB buffers. Transmit frames can be between 1 and 16384 bytes long, so it is possible to transmit frames longer than the maximum length specified in the IEEE 802.3 standard. It should be noted that zero length AHB buffers are allowed and that the maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit AHB buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the transmit buffer queue pointer. The base address for this queue pointer is set in software using the Transmit Buffer Queue Base Address Register. Each list entry consists of two words. The first is the byte address of the transmit buffer and the second containing the transmit control and status. For the packet buffer DMA, the start location for each AHB buffer is a byte address, the bottom bits of the address being used to offset the start of the data from the data-word boundary (i.e., bits 2,1 and 0 are used to offset the address for 64-bit datapaths).

Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad will also be automatically generated to take frames to a minimum length of 64 bytes. When CRC is not automatically generated (as defined in word 1 of the transmit buffer descriptor or through the control bus of FIFO), the frame is assumed to be at least 64 bytes long and pad is not generated.

An entry in the transmit buffer descriptor list is described in Table 36-3, “Transmit Buffer Descriptor Entry”.

To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits [31:0] in the first word of each descriptor list entry.

The second word of the transmit buffer descriptor is initialized with control information that indicates the length of the frame, whether or not the MAC is to append CRC and whether the buffer is the last buffer in the frame.

After transmission the status bits are written back to the second word of the first buffer along with the used bit. Bit 31 is the used bit which must be zero when the control word is read if transmission is to take place. It is written to one once the frame has been transmitted. Bits[29:20] 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 the queue pointer continues to increment.

The Transmit Buffer Queue Base Address Register can only be updated while transmission is disabled or halted; otherwise any attempted write will be ignored. When transmission is halted the transmit buffer queue pointer will maintain its value. Therefore when transmission is restarted the next descriptor read from the queue will be from immediately after the last successfully transmitted frame. while transmit is disabled (bit 3 of the network control set low), the transmit buffer queue pointer resets to point to the address indicated by the Transmit Buffer Queue Base Address Register. Note that disabling receive does not have the same effect on the receive buffer queue pointer.

Once the transmit queue is initialized, transmit is activated by writing to the transmit start bit (bit 9) of the Network Control Register. Transmit is halted when a buffer descriptor with its used bit set is read, 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. This is implemented with TXGO variable which is readable in the Transmit Status Register at bit location 3. The TXGO variable is reset when:

- Transmit is disabled.
- A buffer descriptor with its ownership bit set is read.
- Bit 10, THALT, of the Network Control Register is written.
- There is a transmit error such as too many retries, late collision (gigabit mode only) or a transmit under run.

To set TXGO, write TSTART to the bit 9 of the Network Control Register. Transmit halt does not take effect until any ongoing transmit finishes.

If the DMA is configured for internal FIFO mode, or for packet buffer partial store and forward mode and a collision occurs during transmission of a multi-buffer frame, transmission will automatically restart from the first buffer of the frame. For packet buffer mode, the entire contents of the frame are read into the transmit packet buffer memory, so the retry attempt will be replayed directly from the packet buffer memory rather than having to re-fetch through the AHB.

If a used bit is read mid way through transmission of a multi buffer frame this is treated as a transmit error. Transmission stops, GTXER is asserted and the FCS will be bad.

If transmission stops due to a transmit error or a used bit being read, transmission will restart from the first buffer descriptor of the frame being transmitted when the transmit start bit is rewritten.

Table 36-3. Transmit Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	Byte address of buffer
Word 1	
31	Used—must be zero for the GMAC to read data to the transmit buffer. The GMAC sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap—marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29	Retry limit exceeded, transmit error detected
28	Transmit under run—occurs when the start of packet data has been written into the FIFO and either HRESP is not OK, or the transmit data could not be fetched in time, or when buffers are exhausted. This is not set when the DMA is configured for packet buffer mode.
27	Transmit frame corruption due to AHB error—set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and GTXER asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size.
26	Late collision, transmit error detected. Late collisions only force this status bit to be set in gigabit mode.
25:23	Reserved

Table 36-3. Transmit Buffer Descriptor Entry (Continued)

Bit	Function
22:20	Transmit IP/TCP/UDP checksum generation offload errors: 000: No Error. 001: The Packet was identified as a VLAN type, but the header was not fully complete, or had an error in it. 010: The Packet was identified as a SNAP type, but the header was not fully complete, or had an error in it. 011: The Packet was not of an IP type, or the IP packet was invalidly short, or the IP was not of type IPv4/IPv6. 100: The Packet was not identified as VLAN, SNAP or IP. 101: Non supported packet fragmentation occurred. For IPv4 packets, the IP checksum was generated and inserted. 110: Packet type detected was not TCP or UDP. TCP/UDP checksum was therefore not generated. For IPv4 packets, the IP checksum was generated and inserted. 111: A premature end of packet was detected and the TCP/UDP checksum could not be generated.
19:17	Reserved
16	No CRC to be appended by MAC. When set, this implies that the data in the buffers already contains a valid CRC, hence no CRC or padding is to be appended to the current frame by the MAC. This control bit must be set for the first buffer in a frame and will be ignored for the subsequent buffers of a frame. Note that this bit must be clear when using the transmit IP/TCP/UDP checksum generation offload, otherwise checksum generation and substitution will not occur.
15	Last buffer, when set this bit will indicate the last buffer in the current frame has been reached.
14	Reserved
13:0	Length of buffer

36.5.3.5 DMA Bursting on the AHB

The DMA will always use SINGLE, or INCR type AHB accesses for buffer management operations. When performing data transfers, the AHB burst length used can be programmed using bits [4:0] of the DMA Configuration Register so that either SINGLE, INCR or fixed length incrementing bursts (INCR4, INCR8 or INCR16) are used where possible.

When there is enough space and enough data to be transferred, the programmed fixed length bursts will be used. If there is not enough data or space available, for example when at the beginning or the end of a buffer, SINGLE type accesses are used. Also SINGLE type accesses are used at 1024 byte boundaries, so that the 1 Kbyte boundaries are not burst over as per AHB requirements.

The DMA will not terminate a fixed length burst early, unless an error condition occurs on the AHB or if receive or transmit are disabled in the Network Control Register.

36.5.3.6 DMA Packet Buffer

The DMA can be configured to use packet buffers for both transmit and receive paths. This mode allows multiple packets to be buffered in both transmit and receive directions. This allows the DMA to withstand far greater access latencies on the AHB and make more efficient use of the AHB bandwidth.

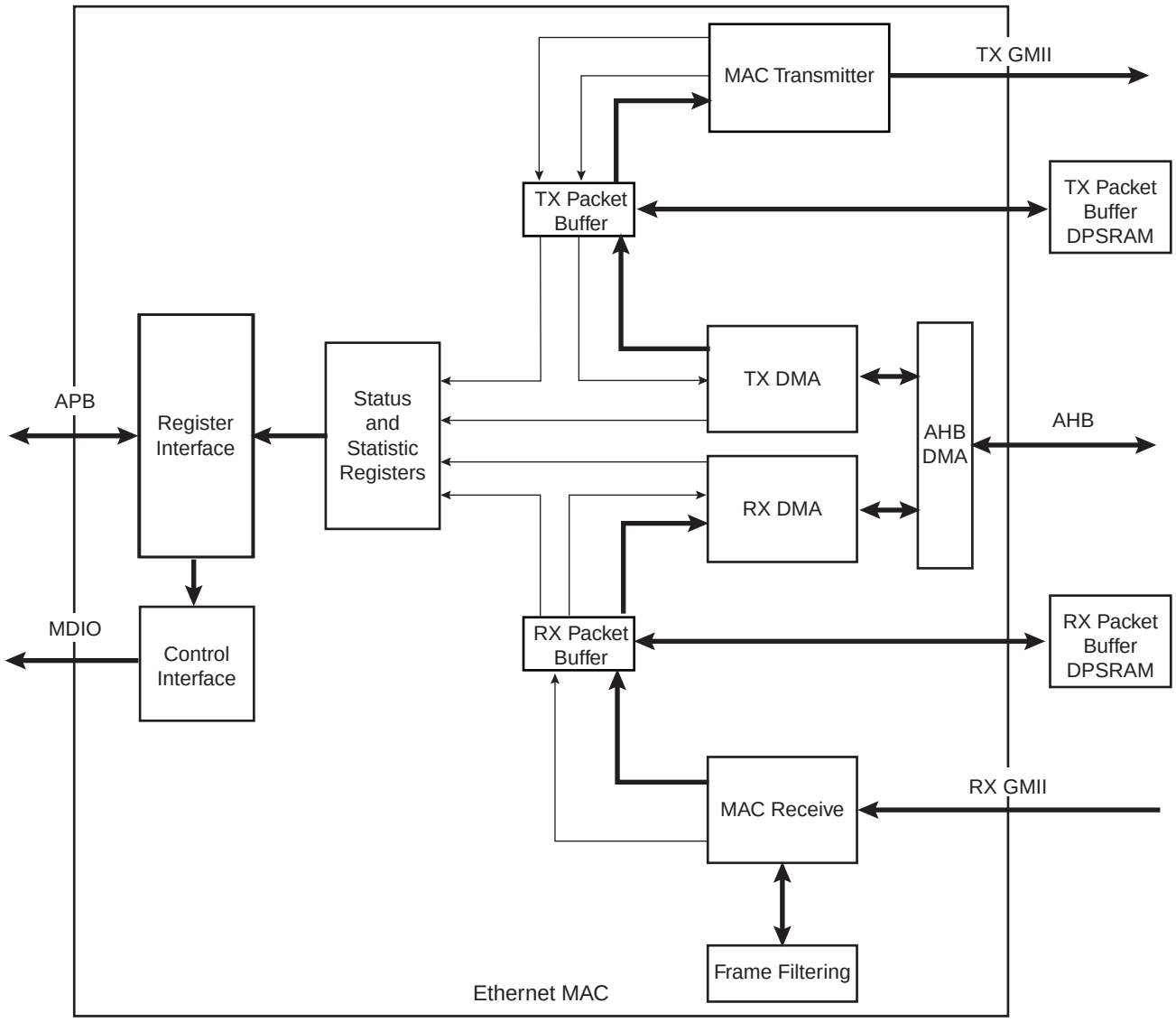
As described earlier, (see Section 36.5.3.2 “Partial Store and Forward Using Packet Buffer DMA”) when the DMA is configured to use packet buffers, it can be programmed into a low latency mode, known as partial store and forward. Further details of this mode have been given (see Section 36.5.3.2) and are not repeated here.

When it is programmed in full store and forward mode, full packets are buffered which provides the possibility to:

- Discard packets with error on the receive path before they are partially written out of the DMA, thus saving AHB bus bandwidth and driver processing overhead,
- Retry collided transmit frames from the buffer, thus saving AHB bus bandwidth,
- Implement transmit IP/TCP/UDP checksum generation offload.

With the packet buffers included, the structure of the GMAC data paths is shown in Figure 36-2.

Figure 36-2. Data Paths with Packet Buffers Included



36.5.3.7 Transmit Packet Buffer

The transmitter packet buffer will continue attempting to fetch frame data from the AHB system memory until the packet buffer itself is full, at which point it will attempt to maintain its full level.

To accommodate the status and statistics associated with each frame, 3 words per packet (or 2 if the GMAC is configured in 64-bit datapath mode) are reserved at the end of the packet data. If the packet is bad and requires to be dropped, the status and statistics are the only information held on that packet. Storing the status in the DPRAM is required in order to decouple the DMA interface of the buffer from the MAC interface, to update the MAC status/statistics and to generate interrupts in the order in which the packets that they represent were fetched from the AHB memory.

If any errors occur on the AHB while reading the transmit frame, the fetching of packet data from AHB memory is halted. The MAC transmitter will continue to fetch packet data, thereby emptying the packet buffer and allowing any good non-errored frames to be transmitted successfully. Once these have been fully transmitted, the status/statistics for the errored frame will be updated and software will be informed via an interrupt that an AHB error occurred. This way, the error is reported in the correct packet order.

The transmit packet buffer will only attempt to read more frame data from the AHB when space is available in the packet buffer memory. If space is not available it must wait until the a packet fetched by the MAC completes transmission and is subsequently removed from the packet buffer memory. Note that if full store and forward mode is active and if a single frame is fetched that is too large for the packet buffer memory, the frame is flushed and the DMA halted with an error status. This is because a complete frame must be written into the packet buffer before transmission can begin, and therefore the minimum packet buffer memory size should be chosen to satisfy the maximum frame to be transmitted in the application.

In full store and forward mode, once the complete transmit frame is written into the packet buffer memory, a trigger is sent across to the MAC transmitter, which will then begin reading the frame from the packet buffer memory. Since the whole frame is present and stable in the packet buffer memory an underflow of the transmitter is not possible. The frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In partial store and forward mode, a trigger is sent across to the MAC transmitter as soon as sufficient packet data is available, which will then begin fetching the frame from the packet buffer memory. If, after this point, the MAC transmitter is able to fetch data from the packet buffer faster than the AHB DMA can fill it, an underflow of the transmitter is possible. In this case, the transmission is terminated early, and the packet buffer is completely flushed. Transmission can only be restarted by writing to the transmit START bit.

In half duplex mode, the frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In full duplex mode, the frame is removed from the packet buffer on the fly.

Other than underflow, the only MAC related errors that can occur are due to collisions during half duplex transmissions. When a collision occurs the frame still exists in the packet buffer memory so can be retried directly from there. Only once the MAC transmitter has failed to transmit after sixteen attempts is the frame finally flushed from the packet buffer.

36.5.3.8 Receive Packet Buffer

The receive packet buffer stores frames from the MAC receiver along with their status and statistics. Frames with errors are flushed from the packet buffer memory, while good frames are pushed onto the DMA AHB interface.

The receiver packet buffer monitors FIFO from the MAC receiver and translates the FIFO pushes into packet buffer writes. At the end of the received frame the status and statistics are buffered so that the information can be used when the frame is read out. When programmed in full store and forward mode, if the frame has an error the frame data is immediately flushed from the packet buffer memory allowing subsequent frames to utilise the freed up space. The status and statistics for bad frames are still used to update the GMAC registers.

To accommodate the status and statistics associated with each frame, three words per packet (or two if configured in 64-bit datapath mode) are reserved at the end of the packet data. If the packet is bad and requires to be dropped, the status and statistics are the only information held on that packet.

The receiver packet buffer will also detect a full condition so that an overflow condition can be detected. If this occurs, subsequent packets are dropped and an RX overflow interrupt is raised.

For full store and forward, the DMA only begins packet fetches once the status and statistics for a frame are available. If the frame has a bad status due to a frame error, the status and statistics are passed onto the GMAC registers. If the frame has a good status, the information is used to read the frame from the packet buffer memory and burst onto the AHB using the DMA buffer management protocol. Once the last frame data has been transferred to the FIFO, the status and statistics are updated to the GMAC registers.

If partial store and forward mode is active, the DMA will begin fetching the packet data before the status is available. As soon as the status becomes available, the DMA will fetch this information as soon as possible before continuing to fetch the remainder of the frame. Once the last frame data has been transferred to the FIFO, the status and statistics are updated to the GMAC registers.

36.5.4 MAC Transmit Block

The MAC transmitter can operate in either half duplex or full duplex mode and transmits frames in accordance with the Ethernet IEEE 802.3 standard. In half duplex mode, the CSMA/CD protocol of the IEEE 802.3 specification is followed.

A small input buffer receives data through FIFO which, depending on the DMA bus width control bits in the Network Configuration Register, will extract data in 32-bit form. All subsequent processing prior to the final output is performed in bytes.

Transmit data can be output using the GMII/MII interface.

Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from FIFO a word at a time. When the GMAC is configured for gigabit operation, the data output to the PHY uses all 8 bits of the txd[7:0] output. In 10/100 mode, transmit data to the PHY is nibble wide and least significant nibble first using txd[3:0] with txd[7:4] tied to logic 0.

If necessary, padding is added to take the frame length to 60 bytes. CRC is calculated using an order 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. The no CRC bit can also be set through FIFO.

In full duplex mode (at all data rates), 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, the transmitter waits for the signal to become inactive, and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter will transmit a jam sequence of 32 bits taken from the data register and then retry transmission after the back off time has elapsed. If the collision occurs during either the preamble or Start Frame Delimiter (SFD), then these fields will be completed prior to generation of the jam sequence.

The back off time is based on an XOR of the 10 least significant bits of the data coming from 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, then the second 2 bits and so on up to the maximum of 10 bits. All 10 bits are used above ten collisions. An error will be indicated and no further attempts will be made if 16 consecutive attempts cause collision. This operation is compliant with the description in Clause 4.2.3.2.5 of the IEEE 802.3 standard which refers to the truncated binary exponential back off algorithm.

In 10/100 mode, both collisions and late collisions are treated identically, and back off and retry will be performed up to 16 times. When operating in gigabit mode, late collisions are treated as an exception and transmission is aborted, without retry. This condition is reported in the transmit buffer descriptor word 1 (late collision, bit 26) and also in the Transmit Status Register (late collision, bit 7). An interrupt can also be generated (if enabled) when this exception occurs, and bit 5 in the Interrupt Status Register will be set.

When operating in gigabit mode (half duplex) both carrier extension and frame bursting are performed in accordance with the IEEE 802.3 standard. For frames less than 512 bytes carrier extension is used to ensure the minimum slot time is not violated.

Frame bursting is used by the transmitter in gigabit mode (half duplex) when more than one frame is queued for transmission. The first frame of a burst must be carrier extended (if necessary) to ensure the minimum slot time of 512 bytes is achieved, after which all subsequent frames within the burst must only satisfy the minimum frame length of 64 bytes or greater. Each interframe gap within the burst is filled by the transmitter with carrier extensions, thus ensuring control of the medium is not given up. Several frames may be transmitted up to the burst limit of 65,536 bytes. The transmitter relinquishes control of the medium when there are no more frames queued for transmission or the burst limit is exceeded.

In gigabit mode any collisions occurring after the minimum slot time for the first frame within a burst are treated as a late collision. The burst is terminated upon this event.

In all modes of operation, if the transmit DMA under runs, a bad CRC is automatically appended using the same mechanism as jam insertion and the GTXER signal is asserted. For a properly configured system this should never happen and also it is impossible if configured to use the DMA with packet buffers, as the complete frame is buffered in local packet buffer memory.

By setting when bit 28 is set in the Network Configuration Register, the Inter Packet Gap (IPG) may be stretched beyond 96 bits depending on the length of the previously transmitted frame and the value written to the IPG Stretch Register (GMAC_IPGS). The least significant 8 bits of the IPG Stretch Register multiply the previous frame length (including preamble). The next significant 8 bits (+1 so as not to get a divide by zero) divide the frame length to generate the IPG. IPG stretch only works in full duplex mode and when bit 28 is set in the Network Configuration Register. The IPG Stretch Register cannot be used to shrink the IPG below 96 bits.

If the back pressure bit is set in the Network Control Register, or if the HDFC configuration bit is set in the GMAC_UR register (10M or 100M 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. Note that this feature is not available in gigabit half duplex mode.

36.5.5 MAC Receive Block

All processing within the MAC receive block is implemented using a 16-bit data path. The MAC receive block checks for valid preamble, FCS, alignment and length, presents received frames to FIFO and stores the frame destination address for use by the address checking block.

If, during the frame reception, the frame is found to be too long, a bad frame indication is sent to FIFO. The receiver logic ceases to send data to memory as soon as this condition occurs.

At end of frame reception the receive block indicates to the DMA block whether the frame is good or bad. The DMA block will recover the current receive buffer if the frame was bad.

Ethernet frames are normally stored in DMA memory or to FIFO complete with the FCS. Setting the FCS remove bit in the network configuration (bit 17) causes frames to be stored without their corresponding FCS. The reported frame length field is reduced by four bytes to reflect this operation.

The receive block signals to the register block to increment the alignment, CRC (FCS), short frame, long frame, jabber or receive symbol errors when any of these exception conditions occur.

If bit 26 is set in the network configuration, CRC errors will be ignored and CRC errored frames will not be discarded, though the Frame Check Sequence Errors statistic register will still be incremented. Additionally, if configured to use the DMA and not enabled for jumbo frames mode, then bit[13] of the receiver descriptor word 1 will be updated to indicate the FCS validity for the particular frame. This is useful for applications such as EtherCAT whereby individual frames with FCS errors must be identified.

Received frames can be checked for length field error by setting the length field error frame discard bit of the Network Configuration Register (bit-16). When this bit is set, the receiver compares a frame's measured length with the length field (bytes 13 and 14) extracted from the frame. The frame is discarded if the measured length is shorter. This checking procedure is for received frames between 64 bytes and 1518 bytes in length.

Each discarded frame is counted in the 10 bit length field error statistics register. Frames where the length field is greater than or equal to 0x0600 hex will not be checked.

When operating in gigabit mode (half duplex), the receiver will discard frames which do not meet the minimal slot time of 512 bytes. If a burst is detected, the first frame is checked to ensure it meets the slot time, but all subsequent frames of the burst are checked to ensure they meet the minimum frame size of 64 bytes.

In gigabit mode (half duplex), carrier extension errors are detected by the receiver during the minimum slot time, and the frame discarded. An error of this nature causes the receive symbol errors statistic register to be incremented. Carrier extension errors occurring during the inter packet gap period are ignored and have no effect on the statistics.

36.5.6 Checksum Offload for IP, TCP and UDP

The GMAC can be programmed to perform IP, TCP and UDP checksum offloading in both receive and transmit directions, which is enabled by setting bit 24 in the Network Configuration Register for receive and bit 11 in the DMA Configuration Register for transmit.

IPv4 packets contain a 16-bit checksum field, which is the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header. TCP and UDP packets contain a 16-bit checksum field, which is the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header, the data and a conceptual IP pseudo header.

To calculate these checksums in software requires each byte of the packet to be processed. For TCP and UDP this can use a large amount of processing power. Offloading the checksum calculation to hardware can result in significant performance improvements.

For IP, TCP or UDP checksum offload to be useful, the operating system containing the protocol stack must be aware that this offload is available so that it can make use of the fact that the hardware can either generate or verify the checksum.

36.5.6.1 Receiver Checksum Offload

When receive checksum offloading is enabled in the GMAC, the IPv4 header checksum is checked as per RFC 791, where the packet meets the following criteria:

- If present, the VLAN header must be four octets long and the CFI bit must not be set.
- Encapsulation must be RFC 894 Ethernet Type Encoding or RFC 1042 SNAP Encoding.
- IPv4 packet
- IP header is of a valid length

The GMAC also checks the TCP checksum as per RFC 793, or the UDP checksum as per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- Good IP header checksum (if IPv4)
- No IP fragmentation
- TCP or UDP packet

When an IP, TCP or UDP frame is received, the receive buffer descriptor gives an indication if the GMAC was able to verify the checksums. There is also an indication if the frame had SNAP encapsulation. These indication bits will replace the type ID match indication bits when the receive checksum offload is enabled. For details of these indication bits refer to Table 36-2, "Receive Buffer Descriptor Entry".

If any of the checksums are verified as incorrect by the GMAC, the packet is discarded and the appropriate statistics counter incremented.

36.5.6.2 Transmitter Checksum Offload

The transmitter checksum offload is only available if the GMAC is configured to use the DMA in packet buffer mode and full store and forward mode is enabled. This is because the complete frame to be transmitted must be read into the packet buffer memory before the checksum can be calculated and written back into the headers at the beginning of the frame.

Transmitter checksum offload is enabled by setting bit [11] in the DMA Configuration Register. When enabled, it will monitor the frame as it is written into the transmitter packet buffer memory to automatically detect the protocol of the frame. Protocol support is identical to the receiver checksum offload.

For transmit checksum generation and substitution to occur, the protocol of the frame must be recognized and the frame must be provided without the FCS field, by making sure that bit [16] of the transmit descriptor word 1 is clear. If the frame data already had the FCS field, this would be corrupted by the substitution of the new checksum fields.

If these conditions are met, the transmit checksum offload engine will calculate the IP, TCP and UDP checksums as appropriate. Once the full packet is completely written into packet buffer memory, the checksums will be valid and the

relevant DPRAM locations will be updated for the new checksum fields as per standard IP/TCP and UDP packet structures.

If the transmitter checksum engine is prevented from generating the relevant checksums, bits [22:20] of the transmitter DMA writeback status will be updated to identify the reason for the error. Note that the frame will still be transmitted but without the checksum substitution, as typically the reason that the substitution did not occur was that the protocol was not recognized.

36.5.7 MAC Filtering Block

The filter block determines which frames should be written to FIFO and on to the optional DMA.

Whether a frame is passed depends on what is enabled in the Network Configuration Register, the state of the external matching pins, the contents of the specific address, type and Hash Registers and the frame's destination address and type field.

If bit 25 of the Network Configuration Register is not set, a frame will not be copied to memory if the GMAC is transmitting in half duplex mode at the time a destination address is received.

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, which is the LSB of the first byte of the frame, is the group or 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 GMAC 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 written to FIFO and on to DMA memory if used.

Frames may be filtered using the type ID field for matching. Four type ID registers exist in the register address space and each can be enabled for matching by writing a one to the MSB (bit 31) of the respective register. When a frame is received, the matching is implemented as an OR function of the various types of match.

The contents of each type ID register (when enabled) are compared against the length/type ID of the frame being received (e.g., bytes 13 and 14 in non-VLAN and non-SNAP encapsulated frames) and copied to memory if a match is found. The encoded type ID match bits (Word 0, Bit 22 and Bit 23) in the receive buffer descriptor status are set indicating which type ID register generated the match, if the receive checksum offload is disabled.

The reset state of the type ID registers is zero, hence each is initially disabled.

The following example illustrates the use of the address and type ID match registers for a MAC address of 21:43:65:87:A9:CB:

Preamble	55
SFD	D5
DA (Octet 0 - LSB)	21
DA (Octet 1)	43
DA (Octet 2)	65
DA (Octet 3)	87
DA (Octet 4)	A9
DA (Octet 5 - MSB)	CB
SA (LSB)	00(1)
SA	00(1)
SA	00(1)
SA	00(1)
SA	00(1)
SA (MSB)	00(1)
Type ID (MSB)	43
Type ID (LSB)	21

Note: 1. Contains the address of the transmitting device

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:

Specific Address 1 Bottom [31:0] Register (GMAC_SAB1) (Address 0x088) 0x87654321

Specific Address 1 Top [47:32] Register (GMAC_SAT1) (Address 0x08C) 0x0000CBA9

And for a successful match to the type ID, the following type ID match 1 register must be set up:

Type ID Match 1 Register (GMAC_TIDM1) (Address 0x0A8) 0x80004321

36.5.8 Broadcast Address

Frames with the broadcast address of 0xFFFFFFFFFFFF are stored to memory only if the 'no broadcast' bit in the Network Configuration Register is set to zero.

36.5.9 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 XOR of every sixth bit of the destination address.

```
hash_index[05] = da[05] ^ da[11] ^ da[17] ^ da[23] ^ da[29] ^ da[35] ^ da[41] ^
da[47]
hash_index[04] = da[04] ^ da[10] ^ da[16] ^ da[22] ^ da[28] ^ da[34] ^ da[40] ^
da[46]
hash_index[03] = da[03] ^ da[09] ^ da[15] ^ da[21] ^ da[27] ^ da[33] ^ da[39] ^
da[45]
hash_index[02] = da[02] ^ da[08] ^ da[14] ^ da[20] ^ da[26] ^ da[32] ^ da[38] ^
da[44]
hash_index[01] = da[01] ^ da[07] ^ da[13] ^ da[19] ^ da[25] ^ da[31] ^ da[37] ^
da[43]
hash_index[00] = da[00] ^ 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 will be matched according to whether the frame is multicast or unicast.

A multicast match will be signalled if the multicast hash enable bit is set, `da[0]` is logic 1 and the hash index points to a bit set in the Hash Register.

A unicast match will be signalled if the unicast hash enable bit is set, `da[0]` is logic 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.

36.5.10 Copy all Frames (Promiscuous Mode)

If the Copy All Frames bit is set in the Network Configuration Register then all frames (except those that are too long, too short, have FCS errors or have GRXER asserted during reception) will be copied to memory. Frames with FCS errors will be copied if bit 26 is set in the Network Configuration Register.

36.5.11 Disable Copy of Pause Frames

Pause frames can be prevented from being written to memory by setting the disable copying of pause frames control bit 23 in the Network Configuration Register. When set, pause frames are not copied to memory regardless of the Copy All Frames bit, whether a hash match is found, a type ID match is identified or if a destination address match is found.

36.5.12 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 36-4. 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. To support these extra four bytes, the GMAC can accept frame lengths up to 1536 bytes by setting bit 8 in the Network Configuration Register.

If the VID (VLAN identifier) is null (0x000) this indicates a priority-tagged frame.

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 will be 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.

The GMAC can be configured to reject all frames except VLAN tagged frames by setting the discard non-VLAN frames bit in the Network Configuration Register.

36.5.13 Wake on LAN Support

The receive block supports Wake on LAN by detecting the following events on incoming receive frames:

- Magic packet
- Address Resolution Protocol (ARP) request to the device IP address
- Specific address 1 filter match
- Multicast hash filter match

These events can be individually enabled through bits [19:16] of the Wake on LAN Register. Also, for Wake on LAN detection to occur, receive enable must be set in the Network Control Register, however a receive buffer does not have to be available.

In case of an ARP request, specific address 1 or multicast filter events will occur even if the frame is errored. For magic packet events, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

- Magic packet events are enabled through bit 16 of the Wake on LAN Register
- The frame's destination address matches specific address 1
- The frame is correctly formed with no errors
- The frame contains at least 6 bytes of 0xFF for synchronization
- There are 16 repetitions of the contents of Specific Address 1 Register immediately following the synchronization

An ARP request event is detected if all of the following are true:

- ARP request events are enabled through bit 17 of the Wake on LAN Register
- Broadcasts are allowed by bit 5 in the Network Configuration Register
- The frame has a broadcast destination address (bytes 1 to 6)
- The frame has a type ID field of 0x0806 (bytes 13 and 14)
- The frame has an ARP operation field of 0x0001 (bytes 21 and 22)
- The least significant 16 bits of the frame's ARP target protocol address (bytes 41 and 42) match the value programmed in bits[15:0] of the Wake on LAN Register

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for the Wake on LAN target address value will not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event will occur if all of the following are true:

- Specific address 1 events are enabled through bit 18 of the Wake on LAN Register
- The frame's destination address matches the value programmed in the Specific Address 1 Registers

A multicast filter match event will occur if all of the following are true:

- Multicast hash events are enabled through bit 19 of the Wake on LAN Register
- Multicast hash filtering is enabled through bit 6 of the Network Configuration Register
- The frame destination address matches against the multicast hash filter
- The frame destination address is not a broadcast

36.5.14 IEEE 1588 Support

IEEE 1588 is a standard for precision time synchronization in local area networks. It works with the exchange of special Precision Time Protocol (PTP) frames. The PTP messages can be transported over IEEE 802.3/Ethernet, over Internet Protocol Version 4 or over Internet Protocol Version 6 as described in the annex of IEEE P1588.D2.1.

GMAC output pins indicate the message time-stamp point (asserted on the start packet delimiter and de-asserted at end of frame) for all frames and the passage of PTP event frames (asserted when a PTP event frame is detected and de-asserted at end of frame).

Synchronization between master and slave clocks is a two stage process.

First, the offset between the master and slave clocks is corrected by the master sending a sync frame to the slave with a follow up frame containing the exact time the sync frame was sent. Hardware assist modules at the master and slave side detect exactly when the sync frame was sent by the master and received by the slave. The slave then corrects its clock to match the master clock.

Second, the transmission delay between the master and slave is corrected. The slave sends a delay request frame to the master which sends a delay response frame in reply. Hardware assist modules at the master and slave side detect exactly when the delay request frame was sent by the slave and received by the master. The slave will now have enough information to adjust its clock to account for delay. For example, if the slave was assuming zero delay, the actual delay will be half the difference between the transmit and receive time of the delay request frame (assuming equal transmit and receive times) because the slave clock will be lagging the master clock by the delay time already.

The time-stamp is taken when the message time-stamp point passes the clock time-stamp point. For Ethernet the message time-stamp point is the SFD and the clock time-stamp point is the MII interface. (The IEEE 1588 specification refers to sync and delay_req messages as event messages as these require time-stamping. Follow up, delay response and management messages do not require time-stamping and are referred to as general messages.)

1588 version 2 defines two additional PTP event messages. These are the peer delay request (Pdelay_Req) and peer delay response (Pdelay_Resp) messages. These messages are used to calculate the delay on a link. Nodes at both ends of a link send both types of frames (regardless of whether they contain a master or slave clock). The Pdelay_Resp message contains the time at which a Pdelay_Req was received and is itself an event message. The time at which a Pdelay_Resp message is received is returned in a Pdelay_Resp_Follow_Up message.

1588 version 2 introduces transparent clocks of which there are two kinds, peer-to-peer (P2P) and end-to-end (E2E). Transparent clocks measure the transit time of event messages through a bridge and amend a correction field within the message to allow for the transit time. P2P transparent clocks additionally correct for the delay in the receive path of the link using the information gathered from the peer delay frames. With P2P transparent clocks delay_req messages are not used to measure link delay. This simplifies the protocol and makes larger systems more stable.

The GMAC recognizes four different encapsulations for PTP event messages:

1. 1588 version 1 (UDP/IPv4 multicast)
2. 1588 version 2 (UDP/IPv4 multicast)
3. 1588 version 2 (UDP/IPv6 multicast)
4. 1588 version 2 (Ethernet multicast)

Table 36-5. Example of Sync Frame in 1588 Version 1 Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–32)	E00001
IP DA (Octet 33)	81 or 82 or 83 or 84
source IP port (Octets 34–35)	—
dest IP port (Octets 36–37)	013F
other stuff (Octets 38–42)	—
version PTP (Octet 43)	01
other stuff (Octets 44–73)	—
control (Octet 74)	00
other stuff (Octets 75–168)	—

Table 36-6. Example of Delay Request Frame in 1588 Version 1 Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–32)	E00001
IP DA (Octet 33)	81 or 82 or 83 or 84
source IP port (Octets 34–35)	—
dest IP port (Octets 36–37)	013F
other stuff (Octets 38–42)	—
version PTP (Octet 43)	01
other stuff (Octets 44–73)	—
control (Octet 74)	01
other stuff (Octets 75–168)	—

For 1588 version 1 messages, sync and delay request frames are indicated by the GMAC if the frame type field indicates TCP/IP, UDP protocol is indicated, the destination IP address is 224.0.1.129/130/131 or 132, the destination UDP port is 319 and the control field is correct.

The control field is 0x00 for sync frames and 0x01 for delay request frames.

For 1588 version 2 messages, the type of frame is determined by looking at the message type field in the first byte of the PTP frame. Whether a frame is version 1 or version 2 can be determined by looking at the version PTP field in the second byte of both version 1 and version 2 PTP frames.

In version 2 messages sync frames have a message type value of 0x0, delay_req have 0x1, Pdelay_Req have 0x2 and Pdelay_Resp have 0x3.

Table 36-7. Example of Sync Frame in 1588 Version 2 (UDP/IPv4) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–33)	E0000181
source IP port (Octets 34–35)	—
dest IP port (Octets 36–37)	013F
other stuff (Octets 38–41)	—
message type (Octet 42)	00
version PTP (Octet 43)	02

Table 36-8. Example of Pdelay_Req Frame in 1588 Version 2 (UDP/IPv4) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–33)	E000006B
source IP port (Octets 34–35)	—
dest IP port (Octets 36–37)	013F
other stuff (Octets 38–41)	—
message type (Octet 42)	02
version PTP (Octet 43)	02

Table 36-9. Example of Sync Frame in 1588 Version 2 (UDP/IPv6) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	86dd
IP stuff (Octets 14–19)	—
UDP (Octet 20)	11
IP stuff (Octets 21–37)	—
IP DA (Octets 38–53)	FF0X00000000018
source IP port (Octets 54–55)	—
dest IP port (Octets 56–57)	013F
other stuff (Octets 58–61)	—
message type (Octet 62)	00
other stuff (Octets 63–93)	—
version PTP (Octet 94)	02

Table 36-10. Example of Pdelay_Resp Frame in 1588 Version 2 (UDP/IPv6) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	86dd
IP stuff (Octets 14–19)	—
UDP (Octet 20)	11
IP stuff (Octets 21–37)	—
IP DA (Octets 38–53)	FF0200000000006B
source IP port (Octets 54–55)	—
dest IP port (Octets 56–57)	013F
other stuff (Octets 58–61)	—
message type (Octet 62)	03
other stuff (Octets 63–93)	—
version PTP (Octet 94)	02

For the multicast address 011B19000000 sync and delay request frames are recognized depending on the message type field, 00 for sync and 01 for delay request.

Table 36-11. Example of Sync Frame in 1588 Version 2 (Ethernet Multicast) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	011B19000000
SA (Octets 6–11)	—
Type (Octets 12–13)	88F7
message type (Octet 14)	00
version PTP (Octet 15)	02

Pdelay request frames need a special multicast address so they can pass through ports blocked by the spanning tree protocol. For the multicast address 0180C200000E sync, Pdelay_Req and Pdelay_Resp frames are recognized depending on the message type field, 00 for sync, 02 for pdelay request and 03 for pdelay response.

Table 36-12. Example of Pdelay_Req Frame in 1588 Version 2 (Ethernet Multicast) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	0180C200000E
SA (Octets 6–11)	—
Type (Octets 12–13)	88F7
message type (Octet 14)	00
version PTP (Octet 15)	02

36.5.15 Time Stamp Unit

The TSU consists of a timer and registers to capture the time at which PTP event frames cross the message timestamp point. An interrupt is issued when a capture register is updated.

The timer is implemented as a 62-bit register with the upper 32 bits counting seconds and the lower 30 bits counting nanoseconds. The lower 30 bits roll over when they have counted to one second. An interrupt is generated when the seconds increment. The timer value can be read, written and adjusted through the APB interface.

The timer is clocked by MCK.

The amount by which the timer increments each clock cycle is controlled by the timer increment register. Bits [7:0] are the default increment value in nanoseconds. If the rest of the register is written with zero the timer increments by the value in [7:0] each clock cycle.

Bits [15:8] of the increment register are the alternative increment value in nanoseconds and bits [23:16] are the number of increments after which the alternative increment value is used. If [23:16] are zero then the alternative increment value will never be used.

Taking the example of 10.2 MHz, there are 102 cycles every ten microseconds or 51 every five microseconds. So a timer with a 10.2 MHz clock source is constructed by incrementing by 98 ns for fifty cycles and then incrementing by 100 ns (98*50 + 100 = 5000). This is programmed by setting the 1588 Timer Increment Register to 0x00326462.

For a 49.8 MHz clock source it would be 20 ns for 248 cycles followed by an increment of 40 ns (20*248 + 40 = 5000) programmed as 0x00F82814.

Having eight bits for the “number of increments” field allows frequencies up to 50 MHz to be supported with 200 kHz resolution

Without the alternative increment field the period of the clock would be limited to an integer number of nanoseconds, resulting in supported clock frequencies of 8, 10, 20, 25, 40, 50, 100, 125, 200 and 250 MHz.

GMAC is configured to operate as PTP slave. The timer register increments as normal but the timer value is copied to the sync strobe register.

There are six additional 62-bit registers that capture the time at which PTP event frames are transmitted and received. An interrupt is issued when these registers are updated.

36.5.16 MAC 802.3 Pause Frame Support

Note: See Clause 31, and Annex 31A and 31B of the IEEE standard 802.3 for a full description of MAC 802.3 pause operation.

The following table shows the start of a MAC 802.3 pause frame.

Table 36-13. Start of an 802.3 Pause Frame

Address		Type (MAC Control Frame)	Pause	
Destination	Source		Opcode	Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The GMAC supports both hardware controlled pause of the transmitter, upon reception of a pause frame, and hardware generated pause frame transmission.

36.5.16.1 802.3 Pause Frame Reception

Bit 13 of the Network Configuration Register is the pause enable control for reception. If this bit is set, transmission will pause if a non zero pause quantum frame is received.

If a valid pause frame is received then the Pause Time Register is updated with the new frame's pause time, regardless of whether a previous pause frame is active or not. An interrupt (either bit 12 or bit 13 of the Interrupt Status Register) is triggered when a pause frame is received, but only if the interrupt has been enabled (bit 12 and bit 13 of the Interrupt Mask Register). Pause frames received with non zero quantum are indicated through the interrupt bit 12 of the Interrupt Status Register. Pause frames received with zero quantum are indicated on bit 13 of the Interrupt Status Register.

Once the Pause Time Register is loaded and the frame currently being transmitted has been sent, no new frames are transmitted until the pause time reaches zero. The loading of a new pause time, and hence the pausing of transmission, only occurs when the GMAC is configured for full duplex operation. If the GMAC is configured for half duplex there will be no transmission pause, but the pause frame received interrupt will still be triggered. A valid pause frame is defined as having a destination address that matches either the address stored in Specific Address Register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0001.

Pause frames that have frame check sequence (FCS) or other errors will be treated as invalid and will be discarded. 802.3 Pause frames that are received after Priority-based Flow Control (PFC) has been negotiated will also be discarded. Valid pause frames received will increment the pause frames received statistic register.

The pause time register decrements every 512 bit times once transmission has stopped. For test purposes, the retry test bit can be set (bit 12 in the Network Configuration Register) which causes the Pause Time Register to decrement every GTXCK cycle once transmission has stopped.

The interrupt (bit 13 in the Interrupt Status Register) is asserted whenever the Pause Time Register decrements to zero (assuming it has been enabled by bit 13 in the Interrupt Mask Register). This interrupt is also set when a zero quantum pause frame is received.

36.5.16.2 802.3 Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit pause frame bits of the Network Control Register. If either bit 11 or bit 12 of the Network Control Register is written with logic 1, an 802.3 pause frame will be

transmitted, providing full duplex is selected in the Network Configuration Register and the transmit block is enabled in the Network Control Register.

Pause frame transmission will happen immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise the following:

- A destination address of 01-80-C2-00-00-01
- A source address taken from Specific Address Register 1
- A type ID of 88-08 (MAC control frame)
- A pause opcode of 00-01
- A pause quantum register
- Fill of 00 to take the frame to minimum frame length
- Valid FCS

The pause quantum used in the generated frame will depend on the trigger source for the frame as follows:

- If bit 11 is written with a one, the pause quantum will be taken from the Transmit Pause Quantum Register. The Transmit Pause Quantum Register resets to a value of 0xFFFF giving maximum pause quantum as default.
- If bit 12 is written with a one, the pause quantum will be zero.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the Interrupt Status Register) and the only statistics register that will be incremented will be the Pause Frames Transmitted Register.

Pause frames can also be transmitted by the MAC using normal frame transmission methods.

36.5.17 MAC PFC Priority-based Pause Frame Support

Note: Refer to the 802.1Qbb standard for a full description of priority-based pause operation.

The following table shows the start of a Priority-based Flow Control (PFC) pause frame.

Table 36-14. Start of a PFC Pause Frame

Address		Type (Mac Control Frame)	Pause Opcode	Priority Enable Vector	Pause Time
Destination	Source				
0x0180C2000001	6 bytes	0x8808	0x1001	2 bytes	8*2 bytes

The GMAC supports PFC priority-based pause transmission and reception. Before PFC pause frames can be received, bit 16 of the Network Control Register must be set.

36.5.17.1 PFC Pause Frame Reception

The ability to receive and decode priority-based pause frames is enabled by setting bit 16 of the Network Control Register. When this bit is set, the GMAC will match either classic 802.3 pause frames or PFC priority-based pause frames. Once a priority-based pause frame has been received and matched, then from that moment on the GMAC will only match on priority-based pause frames (this is an 802.1Qbb requirement, known as PFC negotiation). Once priority-based pause has been negotiated, any received 802.3x format pause frames will not be acted upon.

If a valid priority-based pause frame is received then the GMAC will decode the frame and determine which, if any, of the eight priorities require to be paused. Up to eight Pause Time Registers are then updated with the eight pause times extracted from the frame regardless of whether a previous pause operation is active or not. An interrupt (either bit 12 or bit 13 of the Interrupt Status Register) is triggered when a pause frame is received, but only if the interrupt has been enabled (bit 12 and bit 13 of the Interrupt Mask Register). Pause frames received with non zero quantum are indicated through the interrupt bit 12 of the Interrupt Status Register. Pause frames received with zero quantum are indicated on bit 13 of the Interrupt Status Register. The loading of a new pause time only occurs when the GMAC is configured for full duplex operation. If the GMAC is configured for half duplex, the pause time counters will not be loaded, but the pause frame received interrupt will still be triggered. A valid pause frame is defined as having a destination address that

matches either the address stored in Specific Address Register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0101.

Pause frames that have frame check sequence (FCS) or other errors will be treated as invalid and will be discarded. Valid pause frames received will increment the Pause Frames Received Statistic Register.

The Pause Time Registers decrement every 512 bit times immediately following the PFC frame reception. For test purposes, the retry test bit can be set (bit 12 in the Network Configuration Register) which causes the Pause Time Register to decrement every GRXCK cycle once transmission has stopped.

The interrupt (bit 13 in the Interrupt Status Register) is asserted whenever the Pause Time Register decrements to zero (assuming it has been enabled by bit 13 in the Interrupt Mask Register). This interrupt is also set when a zero quantum pause frame is received.

36.5.17.2 PFC Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit priority-based pause frame bit of the Network Control Register. If bit 17 of the Network Control Register is written with logic 1, a PFC pause frame will be transmitted providing full duplex is selected in the Network Configuration Register and the transmit block is enabled in the Network Control Register. When bit 17 of the Network Control Register is set, the fields of the priority-based pause frame will be built using the values stored in the Transmit PFC Pause Register.

Pause frame transmission will happen immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise the following:

- A destination address of 01-80-C2-00-00-01
- A source address taken from Specific Address Register 1
- A type ID of 88-08 (MAC control frame)
- A pause opcode of 01-01
- A priority enable vector taken from Transmit PFC Pause Register
- 8 pause quantum registers
- Fill of 00 to take the frame to minimum frame length
- Valid FCS

The pause quantum registers used in the generated frame will depend on the trigger source for the frame as follows:

- If bit 17 of the Network Control Register is written with a one, then the priority enable vector of the priority-based pause frame will be set equal to the value stored in the Transmit PFC Pause Register [7:0]. For each entry equal to zero in the Transmit PFC Pause Register[15:8], the pause quantum field of the pause frame associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in the Transmit PFC Pause Register[15:8], the pause quantum associated with that entry will be zero.
- The Transmit Pause Quantum Register resets to a value of 0xFFFF giving maximum pause quantum as default.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the Interrupt Status Register) and the only statistics register that will be incremented will be the Pause Frames Transmitted Register.

PFC Pause frames can also be transmitted by the MAC using normal frame transmission methods.

36.5.18 PHY Interface

Different PHY interfaces are supported by the Gigabit Ethernet MAC:

- GMII
- MII
- RGMII

The GMII and RGMII should only be used for 1000 Mbps operation. The MII interface is provided for 10/100 operation and uses txd[3:0] and rxd[3:0].

36.5.19 10/100/1000 Operation

The gigabit select bit in the Network Configuration Register selects between 10/100 Mbps Ethernet operation and 1000 Mbps mode. The 10/100 Mbps speed bit in the Network Configuration Register is used to select between 10 Mbps and 100 Mbps.

36.5.20 Jumbo Frames

The jumbo frames enable bit in the Network Configuration Register allows the GMAC, in its default configuration, to receive jumbo frames up to 10240 bytes in size. This operation does not form part of the IEEE 802.3 specification and is normally disabled. When jumbo frames are enabled, frames received with a frame size greater than 10240 bytes are discarded.

36.6 Programming Interface

36.6.1 Initialization

36.6.1.1 Configuration

Initialization of the GMAC 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.

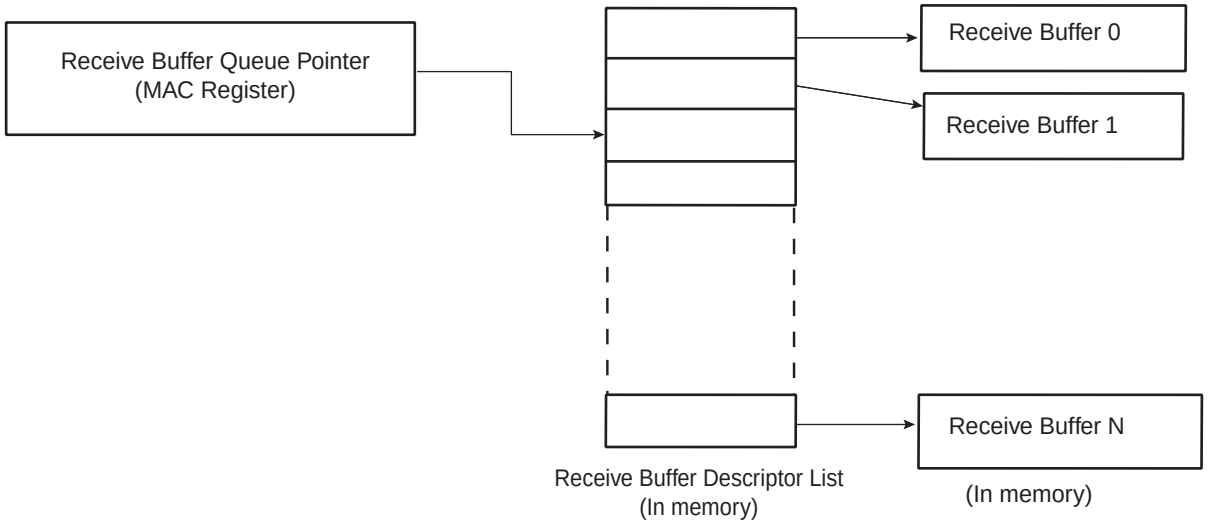
Note: These writes to the Network Control Register cannot be combined in any way.

36.6.1.2 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 Table 36-2, “Receive Buffer Descriptor Entry”

The Receive Buffer Queue Pointer Register points to this data structure.

Figure 36-3. Receive Buffer List



To create the list of buffers:

1. Allocate a number (N) of buffers of X bytes in system memory, where X is the DMA buffer length programmed in the DMA Configuration Register.
2. Allocate an area 8N bytes for the receive buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GMAC, i.e., bit 0 of word 0 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit 1 in word 0 set to 1).
4. Write address of receive buffer descriptor list and control information to GMAC register receive buffer queue pointer
5. The receive circuits can then be enabled by writing to the address recognition registers and the Network Control Register.

36.6.1.3 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 36-3, “Transmit Buffer Descriptor Entry”.

The Transmit Buffer Queue Pointer Register 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 8N bytes for the transmit buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GMAC, i.e., bit 31 of word 1 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit 30 in word 1 set to 1).
4. Write address of transmit buffer descriptor list and control information to GMAC register transmit buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the Network Control Register.

36.6.1.4 Address Matching

The GMAC register pair hash address and the four Specific Address Register-pairs must be written with the required values. Each register pair comprises of 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. Each register pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

As an example, to set Specific Address Register 1 to recognize destination address 21:43:65:87:A9:CB, the following values are written to Specific Address Register 1 bottom and Specific Address Register 1 top:

- Specific Address Register 1 bottom bits 31:0 (0x98): 8765_4321 hex.
- Specific Address Register 1 top bits 31:0 (0x9C): 0000_CBA9 hex.

36.6.1.5 PHY Maintenance

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 bits [18:16] are set to 010 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 Management Data Clock (MDC) cycle. This causes the transmission of a PHY management frame on MDIO. See section 22.2.4.5 of the IEEE 802.3 standard.

Reading during the shift operation will return the current contents of the shift register. At the end of the management operation the bits will 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 Management Data Clock (MDC) should not toggle faster than 2.5 MHz (minimum period of 400 ns), as defined by the IEEE 802.3 standard. MDC is generated by dividing down MCK. Three bits in the Network Configuration Register determine by how much MCK should be divided to produce MDC.

36.6.1.6 Interrupts

There are 18 interrupt conditions that are detected within the GMAC. 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 device interrupt controller documentation to identify that it is the GMAC that is generating the interrupt. To ascertain which interrupt, read the Interrupt Status Register. Note that in the default configuration this register will clear itself after being read, though this may be configured to be write-one-to-clear if desired.

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.

36.6.1.7 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 if they conclude on byte borders.
3. Set-up the transmit buffer list by writing buffer addresses to word zero of the transmit buffer descriptor entries and control and length to word one.
4. Write data for transmission into the buffers pointed to by the descriptors.
5. Write the address of the first buffer descriptor to transmit buffer descriptor queue pointer.
6. Enable appropriate interrupts.
7. Write to the transmit start bit (TSTART) in the Network Control Register.

36.6.1.8 Receiving Frames

When a frame is received and the receive circuits are enabled, the GMAC 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 one of the four type ID registers.
- If it matches the hash address function.
- If it is a broadcast address (0xFFFFFFFFFFFF) and broadcasts are allowed.
- If the GMAC is configured to “copy all frames”.

The register receive buffer queue pointer points to the next entry in the receive buffer descriptor list and the GMAC 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 GMAC then updates the receive buffer descriptor entry (see Table 36-2, “Receive Buffer Descriptor Entry”) with the reason for the address match and marks the area as being owned by software. Once this is complete, a receive complete interrupt is set. Software is then responsible for copying the data to the application area and releasing the buffer (by writing the ownership bit back to 0).

If the GMAC is unable to write the data at a rate to match the incoming frame, then a receive overrun interrupt is set. If there is no receive buffer available, i.e., the next buffer is still owned by software, a receive buffer not available interrupt is set. If the frame is not successfully received, a statistics register is incremented and the frame is discarded without informing software.

36.6.2 Statistics Registers

Statistics registers are described in the User Interface beginning with Section 36.7.39 on page 1015 and ending with Section 36.7.83 on page 1059.

The statistics register block begins at 0x100 and runs to 0x1B0, and comprises the registers listed below.

- Octets Transmitted [31:0] Register
- Octets Transmitted [47:32] Register
- Frames Transmitted Register
- Broadcast Frames Transmitted Register
- Multicast Frames Transmitted Register
- Pause Frames Transmitted Register
- 64 Byte Frames Transmitted Register
- 65 to 127 Byte Frames Transmitted Register
- 128 to 255 Byte Frames Transmitted Register
- 256 to 511 Byte Frames Transmitted Register
- 512 to 1023 Byte Frames Transmitted Register
- 1024 to 1518 Byte Frames Transmitted Register
- Greater Than 1518 Byte Frames Transmitted Register
- Transmit Under Runs Register
- Single Collision Frames Register
- Multiple Collision Frames Register
- Excessive Collisions Register
- Late Collisions Register
- Deferred Transmission Frames Register
- Carrier Sense Errors Register
- Octets Received [31:0] Received
- Octets Received [47:32] Received
- Frames Received Register
- Broadcast Frames Received Register
- Multicast Frames Received Register
- Pause Frames Received Register
- 64 Byte Frames Received Register
- 65 to 127 Byte Frames Received Register
- 128 to 255 Byte Frames Received Register
- 256 to 511 Byte Frames Received Register
- 512 to 1023 Byte Frames Received Register
- 1024 to 1518 Byte Frames Received Register
- 1519 to Maximum Byte Frames Received Register
- Undersize Frames Received Register
- Oversize Frames Received Register
- Jabbers Received Register
- Frame Check Sequence Errors Register
- Length Field Frame Errors Register
- Receive Symbol Errors Register
- Alignment Errors Register
- Receive Resource Errors Register
- Receive Overrun Register
- IP Header Checksum Errors Register
- TCP Checksum Errors Register
- UDP Checksum Errors Register

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 (RXEN) is set in the Network Control Register.

Once a statistics register has been read, it is automatically cleared. When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

36.7 Gigabit Ethernet MAC (GMAC) User Interface

Table 36-15. Register Mapping

Offset	Register	Name	Access	Reset
0x000	Network Control Register	GMAC_NCR	Read-write	0x0000_0000
0x004	Network Configuration Register	GMAC_NCFGR	Read-write	0x0008_0000
0x008	Network Status Register	GMAC_NSR	Read-only	0b01x0
0x00C	User Register	GMAC_UR	Read-write	0x0000_0000
0x010	DMA Configuration Register	GMAC_DCFGR	Read-write	0x0002_0004
0x014	Transmit Status Register	GMAC_TSR	Read-write	0x0000_0000
0x018	Receive Buffer Queue Base Address	GMAC_RBQB	Read-write	0x0000_0000
0x01C	Transmit Buffer Queue Base Address	GMAC_TBQB	Read-write	0x0000_0000
0x020	Receive Status Register	GMAC_RSR	Read-write	0x0000_0000
0x024	Interrupt Status Register	GMAC_ISR	Read-only	0x0000_0000
0x028	Interrupt Enable Register	GMAC_IER	Write-only	–
0x02C	Interrupt Disable Register	GMAC_IDR	Write-only	–
0x030	Interrupt Mask Register	GMAC_IMR	Read-only	0x07FF_FFFF
0x034	PHY Maintenance Register	GMAC_MAN	Read-write	0x0000_0000
0x038	Received Pause Quantum Register	GMAC_RPQ	Read-only	0x0000_0000
0x03C	Transmit Pause Quantum Register	GMAC_TPQ	Read-write	0x0000_FFFF
0x040	TX Partial Store and Forward Register	GMAC_TPSF	Read-write	0x0000_0FFF
0x044	RX Partial Store and Forward Register	GMAC_RPSF	Read-write	0x0000_0FFF
0x048-0x07C	Reserved	–	–	–
0x080	Hash Register Bottom [31:0]	GMAC_HRB	Read-write	0x0000_0000
0x084	Hash Register Top [63:32]	GMAC_HRT	Read-write	0x0000_0000
0x088	Specific Address 1 Bottom [31:0] Register	GMAC_SAB1	Read-write	0x0000_0000
0x08C	Specific Address 1 Top [47:32] Register	GMAC_SAT1	Read-write	0x0000_0000
0x090	Specific Address 2 Bottom [31:0] Register	GMAC_SAB2	Read-write	0x0000_0000
0x094	Specific Address 2 Top [47:32] Register	GMAC_SAT2	Read-write	0x0000_0000
0x098	Specific Address 3 Bottom [31:0] Register	GMAC_SAB3	Read-write	0x0000_0000
0x09C	Specific Address 3 Top [47:32] Register	GMAC_SAT3	Read-write	0x0000_0000
0x0A0	Specific Address 4 Bottom [31:0] Register	GMAC_SAB4	Read-write	0x0000_0000
0x0A4	Specific Address 4 Top [47:32] Register	GMAC_SAT4	Read-write	0x0000_0000
0x0A8	Type ID Match 1 Register	GMAC_TIDM1	Read-write	0x0000_0000
0x0AC	Type ID Match 2 Register	GMAC_TIDM2	Read-write	0x0000_0000
0x0B0	Type ID Match 3 Register	GMAC_TIDM3	Read-write	0x0000_0000
0x0B4	Type ID Match 4 Register	GMAC_TIDM4	Read-write	0x0000_0000
0x0B8	Wake on LAN Register	GMAC_WOL	Read-write	0x0000_0000
0x0BC	IPG Stretch Register	GMAC_IPGS	Read-write	0x0000_0000

Table 36-15. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0C0	Stacked VLAN Register	GMAC_SVLAN	Read-write	0x0000_0000
0x0C4	Transmit PFC Pause Register	GMAC_TPFCP	Read-write	0x0000_0000
0x0C8	Specific Address 1 Mask Bottom [31:0] Register	GMAC_SAMB1	Read-write	0x0000_0000
0x0CC	Specific Address 1 Mask Top [47:32] Register	GMAC_SAMT1	Read-write	0x0000_0000
0x0FC	Reserved	–	–	–
0x100	Octets Transmitted [31:0] Register	GMAC_OTLO	Read-only	0x0000_0000
0x104	Octets Transmitted [47:32] Register	GMAC_OTHI	Read-only	0x0000_0000
0x108	Frames Transmitted Register	GMAC_FT	Read-only	0x0000_0000
0x10C	Broadcast Frames Transmitted Register	GMAC_BCFT	Read-only	0x0000_0000
0x110	Multicast Frames Transmitted Register	GMAC_MFT	Read-only	0x0000_0000
0x114	Pause Frames Transmitted Register	GMAC_PFT	Read-only	0x0000_0000
0x118	64 Byte Frames Transmitted Register	GMAC_BFT64	Read-only	0x0000_0000
0x11C	65 to 127 Byte Frames Transmitted Register	GMAC_TBFT127	Read-only	0x0000_0000
0x120	128 to 255 Byte Frames Transmitted Register	GMAC_TBFT255	Read-only	0x0000_0000
0x124	256 to 511 Byte Frames Transmitted Register	GMAC_TBFT511	Read-only	0x0000_0000
0x128	512 to 1023 Byte Frames Transmitted Register	GMAC_TBFT1023	Read-only	0x0000_0000
0x12C	1024 to 1518 Byte Frames Transmitted Register	GMAC_TBFT1518	Read-only	0x0000_0000
0x130	Greater Than 1518 Byte Frames Transmitted Register	GMAC_GTBFT1518	Read-only	0x0000_0000
0x134	Transmit Under Runs Register	GMAC_TUR	Read-only	0x0000_0000
0x138	Single Collision Frames Register	GMAC_SCF	Read-only	0x0000_0000
0x13C	Multiple Collision Frames Register	GMAC_MCF	Read-only	0x0000_0000
0x140	Excessive Collisions Register	GMAC_EC	Read-only	0x0000_0000
0x144	Late Collisions Register	GMAC_LC	Read-only	0x0000_0000
0x148	Deferred Transmission Frames Register	GMAC_DTF	Read-only	0x0000_0000
0x14C	Carrier Sense Errors Register	GMAC_CSE	Read-only	0x0000_0000
0x150	Octets Received [31:0] Received	GMAC_ORLO	Read-only	0x0000_0000
0x154	Octets Received [47:32] Received	GMAC_ORHI	Read-only	0x0000_0000
0x158	Frames Received Register	GMAC_FR	Read-only	0x0000_0000
0x15C	Broadcast Frames Received Register	GMAC_BCFR	Read-only	0x0000_0000
0x160	Multicast Frames Received Register	GMAC_MFR	Read-only	0x0000_0000
0x164	Pause Frames Received Register	GMAC_PFR	Read-only	0x0000_0000
0x168	64 Byte Frames Received Register	GMAC_BFR64	Read-only	0x0000_0000
0x16C	65 to 127 Byte Frames Received Register	GMAC_TBFR127	Read-only	0x0000_0000
0x170	128 to 255 Byte Frames Received Register	GMAC_TBFR255	Read-only	0x0000_0000
0x174	256 to 511Byte Frames Received Register	GMAC_TBFR511	Read-only	0x0000_0000
0x178	512 to 1023 Byte Frames Received Register	GMAC_TBFR1023	Read-only	0x0000_0000
0x17C	1024 to 1518 Byte Frames Received Register	GMAC_TBFR1518	Read-only	0x0000_0000

Table 36-15. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x180	1519 to Maximum Byte Frames Received Register	GMAC_TMXBFR	Read-only	0x0000_0000
0x184	Undersize Frames Received Register	GMAC_UFR	Read-only	0x0000_0000
0x188	Oversize Frames Received Register	GMAC_OFR	Read-only	0x0000_0000
0x18C	Jabbers Received Register	GMAC_JR	Read-only	0x0000_0000
0x190	Frame Check Sequence Errors Register	GMAC_FCSE	Read-only	0x0000_0000
0x194	Length Field Frame Errors Register	GMAC_LFFE	Read-only	0x0000_0000
0x198	Receive Symbol Errors Register	GMAC_RSE	Read-only	0x0000_0000
0x19C	Alignment Errors Register	GMAC_AE	Read-only	0x0000_0000
0x1A0	Receive Resource Errors Register	GMAC_RRE	Read-only	0x0000_0000
0x1A4	Receive Overrun Register	GMAC_ROE	Read-only	0x0000_0000
0x1A8	IP Header Checksum Errors Register	GMAC_IHCE	Read-only	0x0000_0000
0x1AC	TCP Checksum Errors Register	GMAC_TCE	Read-only	0x0000_0000
0x1B0	UDP Checksum Errors Register	GMAC_UCE	Read-only	0x0000_0000
0x1C8	1588 Timer Sync Strobe Seconds Register	GMAC_TSSS	Read-write	0x0000_0000
0x1CC	1588 Timer Sync Strobe Nanoseconds Register	GMAC_TSSN	Read-write	0x0000_0000
0x1D0	1588 Timer Seconds Register	GMAC_TS	Read-write	0x0000_0000
0x1D4	1588 Timer Nanoseconds Register	GMAC_TN	Read-write	0x0000_0000
0x1D8	1588 Timer Adjust Register	GMAC_TA	Write-only	–
0x1DC	1588 Timer Increment Register	GMAC_TI	Read-write	0x0000_0000
0x1E0	PTP Event Frame Transmitted Seconds	GMAC_EFTS	Read-only	0x0000_0000
0x1E4	PTP Event Frame Transmitted Nanoseconds	GMAC_EFTN	Read-only	0x0000_0000
0x1E8	PTP Event Frame Received Seconds	GMAC_EFRS	Read-only	0x0000_0000
0x1EC	PTP Event Frame Received Nanoseconds	GMAC_EFRN	Read-only	0x0000_0000
0x1F0	PTP Peer Event Frame Transmitted Seconds	GMAC_PEFTS	Read-only	0x0000_0000
0x1F4	PTP Peer Event Frame Transmitted Nanoseconds	GMAC_PEFTN	Read-only	0x0000_0000
0x1F8	PTP Peer Event Frame Received Seconds	GMAC_PEFRS	Read-only	0x0000_0000
0x1FC	PTP Peer Event Frame Received Nanoseconds	GMAC_PEFRN	Read-only	0x0000_0000
0x200–0x23C	Reserved	–	–	–
0x280–0x298	Reserved	–	–	–

36.7.1 Network Control Register

Name: GMAC_NCR

Address: 0xF0028000

Access: Read-write

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

23	22	21	20	19	18	17	16
–	–	–	–	–	FNP	TXPBPF	ENPBPR

15	14	13	12	11	10	9	8
SRTSM	–	–	TXZQPF	TXPF	THALT	TSTART	BP

7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TXEN	RXEN	LBL	–

- **LBL: Loop Back Local**

Connects GTX to GRX, GTXEN to GRXDV and forces full duplex mode. GRXCK and GTXCK may malfunction as the GMAC 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.

- **RXEN: Receive Enable**

When set, RXEN enables the GMAC to receive data. When reset frame reception stops immediately and the receive pipeline will be cleared. The Receive Queue Pointer Register is unaffected.

- **TXEN: Transmit Enable**

When set, TXEN enables the GMAC transmitter to send data. When reset transmission will stop immediately, the transmit pipeline and control registers will be cleared and the Transmit Queue Pointer Register will reset 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 10M or 100M half duplex mode, forces collisions on all received frames. Ignored in gigabit half duplex mode.

- **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.

- **TXPF: Transmit Pause Frame**

Writing one to this bit causes a pause frame to be transmitted.

- **TXZQPF: Transmit Zero Quantum Pause Frame**

Writing one to this bit causes a pause frame with zero quantum to be transmitted.

- **SRTSM: Store Receive Time Stamp to Memory**

0: Normal operation.

1: Causes the CRC of every received frame to be replaced with the value of the nanoseconds field of the 1588 timer that was captured as the receive frame passed the message time stamp point.

- **ENPBPR: Enable PFC Priority-based Pause Reception**

Enables PFC Priority Based Pause Reception capabilities. Setting this bit enables PFC negotiation and recognition of priority-based pause frames.

- **TXPBPF: Transmit PFC Priority-based Pause Frame**

Takes the values stored in the Transmit PFC Pause Register.

- **FNP: Flush Next Packet**

Flush the next packet from the external RX DPRAM. Writing one to this bit will only have an effect if the DMA is not currently writing a packet already stored in the DPRAM to memory.

36.7.2 Network Configuration Register

Name: GMAC_NCFGR

Address: 0xF0028004

Access: Read-write

31	30	29	28	27	26	25	24
–	IRXER	RXBP	IPGSEN	–	IRXFCS	EFRHD	RXCOEN

23	22	21	20	19	18	17	16
DCPF	DBW		CLK			RFCS	LFERD

15	14	13	12	11	10	9	8
RXBUFO		PEN	RTY	–	GBE	–	MAXFS

7	6	5	4	3	2	1	0
UNIHEN	MTI HEN	NBC	CAF	JFRAME	DNVLAN	FD	SPD

- **SPD: Speed**

Set to logic one to indicate 100 Mbps operation, logic zero for 10 Mbps.

- **FD: Full Duplex**

If set to logic one, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting.

- **DNVLAN: Discard Non-VLAN FRAMES**

When set only VLAN tagged frames will be passed to the address matching logic.

- **JFRAME: Jumbo Frame Size**

Set to one to enable jumbo frames up to 10240 bytes to be accepted. The default length is 10240 bytes.

- **CAF: Copy All Frames**

When set to logic one, all valid frames will be accepted.

- **NBC: No Broadcast**

When set to logic one, frames addressed to the broadcast address of all ones will not be accepted.

- **MTIHEN: Multicast Hash Enable**

When set, multicast frames will be accepted when the 6-bit hash function of the destination address points to a bit that is set in the Hash Register.

- **UNIHEN: Unicast Hash Enable**

When set, unicast frames will be accepted when the 6-bit hash function of the destination address points to a bit that is set in the Hash Register.

- **MAXFS: 1536 Maximum Frame Size**

Setting this bit means the GMAC will accept frames up to 1536 bytes in length. Normally the GMAC would reject any frame above 1518 bytes.

- **GBE: Gigabit Mode Enable**

Setting this bit configures the GMAC for 1000 Mbps operation.

0: 10/100 operation using MII interface.

1: Gigabit operation using GMII interface.

- **RTY: Retry Test**

Must be set to zero for normal operation. If set to one the backoff between collisions will always be one slot time. Setting this bit to one helps test the too many retries condition. Also used in the pause frame tests to reduce the pause counter's decrement time from 512 bit times, to every GRXCK cycle.

- **PEN: Pause Enable**

When set, transmission will pause if a non-zero 802.3 classic pause frame is received and PFC has not been negotiated.

- **RXBUFO: Receive Buffer Offset**

Indicates the number of bytes by which the received data is offset from the start of the receive buffer

- **LFERD: Length Field Error Frame Discard**

Setting this bit causes frames with a measured length shorter than the extracted length field (as indicated by bytes 13 and 14 in a non-VLAN tagged frame) to be discarded. This only applies to frames with a length field less than 0x0600.

- **RFCS: Remove FCS**

Setting this bit will cause received frames to be written to memory without their frame check sequence (last 4 bytes). The frame length indicated will be reduced by four bytes in this mode.

- **CLK: MDC CLock Division**

Set according to MCK speed. These three bits determine the number MCK will be divided by to generate Management Data Clock (MDC). For conformance with the 802.3 specification, 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_48	MCK divided by 48 (MCK up to 120MHz)
4	MCK_64	MCK divided by 64 (MCK up to 160 MHz)
5	MCK_96	MCK divided by 96 (MCK up to 240 MHz)

- **DBW: Data Bus Width**

Should be always written to 0.

- **DCPF: Disable Copy of Pause Frames**

Set to one to prevent valid pause frames being copied to memory. When set, pause frames are not copied to memory regardless of the state of the Copy All Frames bit, whether a hash match is found or whether a type ID match is identified. If a destination address match is found, the pause frame will be copied to memory. Note that valid pause frames received will still increment pause statistics and pause the transmission of frames as required.

- **RXCOEN: Receive Checksum Offload Enable**

When set, the receive checksum engine is enabled. Frames with bad IP, TCP or UDP checksums are discarded.

- **EFRHD: Enable Frames Received in Half Duplex**

Enable frames to be received in half-duplex mode while transmitting.

- **IRXFCS: Ignore RX FCS**

When set, frames with FCS/CRC errors will not be rejected. FCS error statistics will still be collected for frames with bad FCS and FCS status will be recorded in frame's DMA descriptor. For normal operation this bit must be set to zero.

- **IPGSEN: IP Stretch Enable**

When set, the transmit IPG can be increased above 96 bit times depending on the previous frame length using the IPG Stretch Register.

- **RXBP: Receive Bad Preamble**

When set, frames with non-standard preamble are not rejected.

- **IRXER: Ignore IPG GRXER**

When set, GRXER has no effect on the GMAC's operation when GRXDV is low. Set this bit when using the RGMII wrapper in half-duplex mode.

36.7.3 Network Status Register

Name: GMAC_NSR

Address: 0xF0028008

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: MDIO Input Status**

Returns status of the GMDIO pin.

- **IDLE: PHY Management Logic Idle**

The PHY management logic is idle (i.e., has completed).

36.7.4 User Register

Name: GMAC_UR

Address: 0xF002800C

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
–	–	–	–	–	–	–	RGMII

- **RGMII: Reduced GMII Mode**

0: GMII mode is selected (default).

1: RGMII mode is selected.

36.7.5 DMA Configuration Register

Name: GMAC_DCFGR

Address: 0xF0028010

Access: Read-write

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

23	22	21	20	19	18	17	16
DRBS							

15	14	13	12	11	10	9	8
–	–	–	–	TXCOEN	TXPBMS	RXBMS	

7	6	5	4	3	2	1	0
ESPA	ESMA	–	FBLDO				

- **FBLDO: Fixed Burst Length for DMA Data Operations:**

Selects the burst length to attempt to use on the AHB when transferring frame data. Not used for DMA management operations and only used where space and data size allow. Otherwise SINGLE type AHB transfers are used.

Upper bits become non-writable if the configured DMA TX and RX FIFO sizes are smaller than required to support the selected burst size.

One-hot priority encoding enforced automatically on register writes as follows, where ‘x’ represents don’t care:

Value	Name	Description
0	–	Reserved
1	SINGLE	00001: Always use SINGLE AHB bursts
2	–	Reserved
4	INCR4	001xx: Attempt to use INCR4 AHB bursts (Default)
8	INCR8	01xxx: Attempt to use INCR8 AHB bursts
16	INCR16	1xxxx: Attempt to use INCR16 AHB bursts

- **ESMA: Endian Swap Mode Enable for Management Descriptor Accesses**

When set, selects swapped endianism for AHB transfers. When clear, selects little endian mode.

- **ESPA: Endian Swap Mode Enable for Packet Data Accesses**

When set, selects swapped endianism for AHB transfers. When clear, selects little endian mode.

- **RXBMS: Receiver Packet Buffer Memory Size Select**

The default receive packet buffer size is 4 Kbytes. The table below shows how to configure this memory to FULL, HALF, QUARTER or EIGHTH of the default size.

Value	Name	Description
0	EIGHTH	4/8 Kbyte Memory Size
1	QUARTER	4/4 Kbytes Memory Size
2	HALF	4/2 Kbytes Memory Size
3	FULL	4 Kbytes Memory Size

- **TXPBMS: Transmitter Packet Buffer Memory Size Select**

Having this bit at zero halves the amount of memory used for the transmit packet buffer. This reduces the amount of memory used by the GMAC. It is important to set this bit to one if the full configured physical memory is available. The value in brackets below represents the size that would result for the default maximum configured memory size of 4 Kbytes.

0: Do not use top address bit (2 Kbytes).

1: Use full configured addressable space (4 Kbytes).

- **TXCOEN: Transmitter Checksum Generation Offload Enable**

Transmitter IP, TCP and UDP checksum generation offload enable. When set, the transmitter checksum generation engine is enabled to calculate and substitute checksums for transmit frames. When clear, frame data is unaffected.

- **DRBS: DMA Receive Buffer Size**

DMA receive buffer size in AHB system memory. The value defined by these bits determines the size of buffer to use in main AHB system memory when writing received data.

The value is defined in multiples of 64 bytes, thus a value of 0x01 corresponds to buffers of 64 bytes, 0x02 corresponds to 128 bytes etc.

For example:

– 0x02: 128 bytes

– 0x18: 1536 bytes (1*max length frame/buffer)

– 0xA0: 10240 bytes (1*10K jumbo frame/buffer)

Note that this value should never be written as zero.

- **DDRP: DMA Discard Receive Packets**

When set, the GMAC DMA will automatically discard receive packets from the receiver packet buffer memory when no AHB resource is available.

When low, the received packets will remain to be stored in the SRAM based packet buffer until AHB buffer resource next becomes available.

A write to this bit is ignored if the DMA is not configured in the packet buffer full store and forward mode.

36.7.6 Transmit Status Register

Name: GMAC_TSR

Address: 0xF0028014

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
–	–	–	–	–	–	–	HRESP
7	6	5	4	3	2	1	0
LCO	UND	TXCOMP	TFC	TXGO	RLE	COL	UBR

- **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. When operating in 10/100 mode, this status indicates either a collision or a late collision. In gigabit mode, this status is not set for a late collision.

- **RLE: Retry Limit Exceeded**

Cleared by writing a one to this bit.

- **TXGO: Transmit Go**

Transmit go, if high transmit is active. When using FIFO, this bit represents bit 3 of the Network Control Register. When using the DMA interface this bit represents the TXGO variable as specified in the transmit buffer description.

- **TFC: Transmit Frame Corruption due to AHB error**

Transmit frame corruption due to AHB error. Set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and GTXER asserted).

Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size.

Cleared by writing a one to this bit.

- **TXCOMP: Transmit Complete**

Set when a frame has been transmitted. Cleared by writing a one to this bit.

- **UND: Transmit Under Run**

This bit is set if the transmitter was forced to terminate a frame that it had already began transmitting due to further data being unavailable.

This bit is set if a transmitter status write back has not completed when another status write back is attempted.

When using the DMA interface configured for internal FIFO mode, this bit is also set when the transmit DMA has written the SOP data into the FIFO and either the AHB bus was not granted in time for further data, or because an AHB not OK response was returned, or because a used bit was read.

When using the DMA interface configured for packet buffer mode, this bit will never be set.

- **LCO: Late Collision Occurred**

Only set if the condition occurs in gigabit mode, as retry is not attempted. Cleared by writing a one to this bit.

- **HRESP: HRESP Not OK**

Set when the DMA block sees HRESP not OK. Cleared by writing a one to this bit.

36.7.7 Receive Buffer Queue Base Address Register

Name: GMAC_RBQB

Address: 0xF0028018

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 holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the Network Control Register. Once reception is enabled, any write to the Receive Buffer Queue Base Address Register is ignored. 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.

In terms of AMBA AHB operation, the descriptors are read from memory using a single 32-bit AHB access. The descriptors should be aligned at 32-bit boundaries and the descriptors are written to using two individual non sequential accesses.

- **ADDR: Receive buffer queue base address**

Written with the address of the start of the receive queue.

36.7.8 Transmit Buffer Queue Base Address Register

Name: GMAC_TBQB

Address: 0xF002801C

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 holds the start address of the transmit buffer queue (transmit buffers descriptor list). The Transmit Buffer Queue Base Address Register must be initialized before transmit is started through bit 9 of the Network Control Register. Once transmission has started, any write to the Transmit Buffer Queue Base Address Register is illegal and therefore ignored.

Note that due to clock boundary synchronization, it takes a maximum of four MCK cycles from the writing of the transmit start bit before the transmitter is active. Writing to the Transmit Buffer Queue Base Address Register during this time may produce unpredictable results.

Reading this register returns the location of the descriptor currently being accessed. Since the DMA handles two frames at once, this may not necessarily be pointing to the current frame being transmitted.

In terms of AMBA AHB operation, the descriptors are written to memory using a single 32-bit AHB access. The descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual non sequential accesses.

- **ADDR: Transmit Buffer Queue Base Address**

Written with the address of the start of the transmit queue.

36.7.9 Receive Status Register

Name: GMAC_RSR

Address: 0xF0028020

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
–	–	–	–	HNO	RXOVR	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 will reread the pointer each time an end of frame is received until a valid pointer is found. This bit is set following each descriptor read attempt that fails, even if consecutive pointers are unsuccessful and software has in the mean time cleared the status flag. 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.

- **RXOVR: Receive Overrun**

This bit is set if RX FIFO is not able to store the receive frame due to a FIFO overflow, or if the receive status was not taken at the end of the frame. This bit is also set in DMA packet buffer mode if the packet buffer overflows. For DMA operation, the buffer will be recovered if an overrun occurs. This bit is cleared when set to 1.

- **HNO: HRESP Not OK**

Set when the DMA block sees HRESP not OK. This bit is cleared when set to 1.

36.7.10 Interrupt Status Register

Name: GMAC_ISR

Address: 0xF0028024

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
–	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

This register indicates the source of the interrupt. In order the bits of this register are read to 1, the corresponding interrupt source must be enabled in the mask register. If any bit is set in this register, the GMAC interrupt signal will be asserted in the system.

- **MFS: Management Frame Sent**

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: RX Used Bit Read**

Set when a receive buffer descriptor is read with its used bit set. Cleared on read.

- **TXUBR: TX Used Bit Read**

Set when a transmit buffer descriptor is read with its used bit set. Cleared on read.

- **TUR: Transmit Under Run**

This interrupt is set if the transmitter was forced to terminate a frame that it has already began transmitting due to further data being unavailable.

This interrupt is set if a transmitter status write back has not completed when another status write back is attempted.

This interrupt is also set when the transmit DMA has written the SOP data into the FIFO and either the AHB bus was not granted in time for further data, or because an AHB not OK response was returned, or because the used bit was read.

- **RLEX: Retry Limit Exceeded or Late Collision**

Transmit error. Late collision will only cause this status bit to be set in gigabit mode, as a retry is not attempted. Cleared on read.

- **TFC: Transmit Frame Corruption due to AHB error**

Transmit frame corruption due to AHB error. Set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame.

- **TCOMP: Transmit Complete**

Set when a frame has been transmitted. Cleared on read.

- **ROVR: Receive Overrun**

Set when the receive overrun status bit is set. Cleared on read.

- **HRESP: HRESP Not OK**

Set when the DMA block sees HRESP not OK. Cleared on read.

- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**

Indicates a valid pause has been received that has a non-zero pause quantum field. Cleared on read.

- **PTZ: Pause Time Zero**

Set when either the Pause Time Register at address 0x38 decrements to zero, or when a valid pause frame is received with a zero pause quantum field. Cleared on read.

- **PFTR: Pause Frame Transmitted**

Indicates a pause frame has been successfully transmitted after being initiated from the Network Control Register. Cleared on read.

- **DRQFR: PTP Delay Request Frame Received**

Indicates a PTP delay_req frame has been received. Cleared on read.

- **SFR: PTP Sync Frame Received**

Indicates a PTP sync frame has been received. Cleared on read.

- **DRQFT: PTP Delay Request Frame Transmitted**

Indicates a PTP delay_req frame has been transmitted. Cleared on read.

- **SFT: PTP Sync Frame Transmitted**

Indicates a PTP sync frame has been transmitted. Cleared on read.

- **PDRQFR: PDelay Request Frame Received**

Indicates a PTP pdelay_req frame has been received. Cleared on read.

- **PDRSFR: PDelay Response Frame Received**

Indicates a PTP pdelay_resp frame has been received. Cleared on read.

- **PDRQFT: PDelay Request Frame Transmitted**

Indicates a PTP pdelay_req frame has been transmitted. Cleared on read.

- **PDRSFT: PDelay Response Frame Transmitted**

Indicates a PTP pdelay_resp frame has been transmitted. Cleared on read.

- **SRI: TSU Seconds Register Increment**

Indicates the register has incremented. Cleared on read.

- **WOL: Wake On LAN**

WOL interrupt. Indicates a WOL event has been received.

36.7.11 Interrupt Enable Register

Name: GMAC_IER

Address: 0xF0028028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero.

- **MFS: Management Frame Sent**

Enable management done interrupt.

- **RCOMP: Receive Complete**

Enable receive complete interrupt.

- **RXUBR: RX Used Bit Read**

Enable receive used bit read interrupt.

- **TXUBR: TX Used Bit Read**

Enable transmit used bit read interrupt.

- **TUR: Transmit Under Run**

Enable transmit buffer under run interrupt.

- **RLEX: Retry Limit Exceeded or Late Collision**

Enable retry limit exceeded or late collision interrupt.

- **TFC: Transmit Frame Corruption due to AHB error**

Enable transmit frame corruption due to AHB error interrupt.

- **TCOMP: Transmit Complete**

Enable transmit complete interrupt.

- **ROVR: Receive Overrun**

Enable receive overrun interrupt.

- **HRESP: HRESP Not OK**

Enable HRESP not OK interrupt.

- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**

Enable pause frame with non-zero pause quantum interrupt.

- **PTZ: Pause Time Zero**

Enable pause time zero interrupt.

- **PFTR: Pause Frame Transmitted**

Enable pause frame transmitted interrupt.

- **EXINT: External Interrupt**

Enable external interrupt.

- **DRQFR: PTP Delay Request Frame Received**

Enable PTP delay_req frame received.

- **SFR: PTP Sync Frame Received**

Enable PTP sync frame received.

- **DRQFT: PTP Delay Request Frame Transmitted**

Enable PTP delay_req frame transmitted.

- **SFT: PTP Sync Frame Transmitted**

Enable PTP sync frame transmitted.

- **PDRQFR: PDelay Request Frame Received**

Enable PTP pdelay_req frame received

- **PDRSFR: PDelay Response Frame Received**

Enable PTP pdelay_resp frame received.

- **PDRQFT: PDelay Request Frame Transmitted**

Enable PTP pdelay_req frame transmitted.

- **PDRSFT: PDelay Response Frame Transmitted**

Enable PTP pdelay_resp frame transmitted.

- **SRI: TSU Seconds Register Increment**

Enable TSU Seconds Register increment.

- **WOL: Wake On LAN**

Enable WOL event received interrupt.

36.7.12 Interrupt Disable Register

Name: GMAC_IDR

Address: 0xF002802C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero.

- **MFS: Management Frame Sent**

Disable management done interrupt.

- **RCOMP: Receive Complete**

Disable receive complete interrupt.

- **RXUBR: RX Used Bit Read**

Disable receive used bit read interrupt.

- **TXUBR: TX Used Bit Read**

Disable transmit used bit read interrupt.

- **TUR: Transmit Under Run**

Disable transmit buffer under run interrupt.

- **RLEX: Retry Limit Exceeded or Late Collision**

Disable retry limit exceeded or late collision interrupt.

- **TFC: Transmit Frame Corruption due to AHB error**

Disable transmit frame corruption due to AHB error interrupt.

- **TCOMP: Transmit Complete**

Disable transmit complete interrupt.

- **ROVR: Receive Overrun**

Disable receive overrun interrupt.

- **HRESP: HRESP Not OK**

Disable HRESP not OK interrupt.

- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**

Disable pause frame with non-zero pause quantum interrupt.

- **PTZ: Pause Time Zero**

Disable pause time zero interrupt.

- **PFTR: Pause Frame Transmitted**

Disable pause frame transmitted interrupt.

- **EXINT: External Interrupt**

Disable external interrupt.

- **DRQFR: PTP Delay Request Frame Received**

Disable PTP delay_req frame received.

- **SFR: PTP Sync Frame Received**

Disable PTP sync frame received.

- **DRQFT: PTP Delay Request Frame Transmitted**

Disable PTP delay_req frame transmitted.

- **SFT: PTP Sync Frame Transmitted**

Disable PTP sync frame transmitted.

- **PDRQFR: PDelay Request Frame Received**

Disable PTP pdelay_req frame received

- **PDRSFR: PDelay Response Frame Received**

Disable PTP pdelay_resp frame received.

- **PDRQFT: PDelay Request Frame Transmitted**

Disable PTP pdelay_req frame transmitted.

- **PDRSFT: PDelay Response Frame Transmitted**

Disable PTP pdelay_resp frame transmitted.

- **SRI: TSU Seconds Register Increment**

Disable TSU Seconds Register increment.

- **WOL: Wake On LAN**

Disable WOL event received interrupt.

36.7.13 Interrupt Mask Register

Name: GMAC_IMR

Address: 0xF0028030

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

The Interrupt Mask Register is a read-only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the Interrupt Enable Register or set individually by writing to the Interrupt Disable Register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the Interrupt Mask Register.

For test purposes there is a write-only function to this register that allows the bits in the Interrupt Status Register to be set or cleared, regardless of the state of the mask register.

- **MFS: Management Frame Sent**

A read of this register returns the value of the management done interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **RCOMP: Receive Complete**

A read of this register returns the value of the receive complete interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **RXUBR: RX Used Bit Read**

A read of this register returns the value of the receive used bit read interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **TXUBR: TX Used Bit Read**

A read of this register returns the value of the transmit used bit read interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **TUR: Transmit Under Run**

A read of this register returns the value of the transmit buffer under run interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **RLEX: Retry Limit Exceeded or Late Collision**

A read of this register returns the value of the retry limit exceeded or late collision (gigabit mode only) interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **TFC: Transmit Frame Corruption due to AHB error**

A read of this register returns the value of the transmit frame corruption due to AHB error interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **TCOMP: Transmit Complete**

A read of this register returns the value of the transmit complete interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **ROVR: Receive Overrun**

A read of this register returns the value of the receive overrun interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **HRESP: HRESP Not OK**

A read of this register returns the value of the HRESP not OK interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

- **A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.**

- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**

A read of this register returns the value of the pause frame with non-zero pause quantum interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **PTZ: Pause Time Zero**

A read of this register returns the value of the pause time zero interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **PFTR: Pause Frame Transmitted**

A read of this register returns the value of the pause frame transmitted interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **EXINT: External Interrupt**

A read of this register returns the value of the external interrupt mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **DRQFR: PTP Delay Request Frame Received**

A read of this register returns the value of the PTP delay_req frame received mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **SFR: PTP Sync Frame Received**

A read of this register returns the value of the PTP sync frame received mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **DRQFT: PTP Delay Request Frame Transmitted**

A read of this register returns the value of the PTP delay_req frame transmitted mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **SFT: PTP Sync Frame Transmitted**

A read of this register returns the value of the PTP sync frame transmitted mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **PDRQFR: PDelay Request Frame Received**

A read of this register returns the value of the PTP pdelay_req frame received mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **PDRSFR: PDelay Response Frame Received**

A read of this register returns the value of the PTP pdelay_resp frame received mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **PDRQFT: PDelay Request Frame Transmitted**

A read of this register returns the value of the PTP pdelay_req frame transmitted mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

- **PDRSFT: PDelay Response Frame Transmitted**

A read of this register returns the value of the PTP pdelay_resp frame transmitted mask.

0: Interrupt is enabled.

1: Interrupt is disabled.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

36.7.14 PHY Maintenance Register

Name: GMAC_MAN

Address: 0xF0028034

Access: Read-write

31	30	29	28	27	26	25	24
WZO	CLTTO	OP		PHYA			

23	22	21	20	19	18	17	16
PHYA	REGA					WTN	

15	14	13	12	11	10	9	8
DATA							

7	6	5	4	3	2	1	0
DATA							

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 2 is set in the Network Status Register. It takes about 2000 MCK cycles to complete, when MDC is set for MCK divide by 32 in the Network Configuration Register. An interrupt is generated upon completion.

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. *See Section 22.2.4.5 of the IEEE 802.3 standard*.

Reading during the shift operation will return the current contents of the shift register. At the end of management operation, the bits will have shifted back to their original locations. For a read operation, the data bits will be 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, bit 30 should be written with a 0 rather than a 1.

For a description of MDC generation, see Section 36.7.2 “Network Configuration Register”.

- **DATA: PHY Data**

For a write operation this field is written with the data to be written to the PHY. After a read operation this field contains the data read from the PHY.

- **WTN: Write Ten**

Must be written to 10.

- **REGA: Register Address**

Specifies the register in the PHY to access.

- **PHYA: PHY Address**

- **OP: Operation**

10 is read. 01 is write.

- **CLTTO: Clause 22 Operation**

Must be written to 1 for Clause 22 operation. To read clause 45 PHYs, bit 30 should be written with a 0 rather than a 1.

- **WZO: Write ZERO**

Must be written with 0.

36.7.15 Receive Pause Quantum Register

Name: GMAC_RPQ

Address: 0xF0028038

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
RPQ							

7	6	5	4	3	2	1	0
RPQ							

- **RPQ: Received Pause Quantum**

Stores the current value of the Receive Pause Quantum Register which is decremented every 512 bit times.

36.7.16 Transmit Pause Quantum Register

Name: GMAC_TPQ

Address: 0xF002803C

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
TPQ							

7	6	5	4	3	2	1	0
TPQ							

- **TPQ: Transmit Pause Quantum**

Written with the pause quantum value for pause frame transmission.

36.7.17 TX Partial Store and Forward Register

Name: GMAC_TPSF

Address: 0xF0028040

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	TPB1ADR			

7	6	5	4	3	2	1	0
TPB1ADR							

Partial store and forward is only applicable when using the AHB DMA configured in SRAM based packet buffer mode. It is also not available when the priority queueing feature is enabled.

- **TPB1ADR: Transmit Partial Store and Forward Address**

Watermark value. Reset = 1.

- **ENTXP: Enable TX Partial Store and Forward Operation**

36.7.18 RX Partial Store and Forward Register

Name: GMAC_RPSF

Address: 0xF0028044

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	RPB1ADR			

7	6	5	4	3	2	1	0
RPB1ADR							

Partial store and forward is only applicable when using the AHB DMA configured in SRAM based packet buffer mode. It is also not available when the priority queueing feature is enabled.

- **RPB1ADR: Receive Partial Store and Forward Address**

Watermark value. Reset = 1.

- **ENRXP: Enable RX Partial Store and Forward Operation**

36.7.19 Hash Register Bottom [31:0]

Name: GMAC_HRB

Address: 0xF0028080

Access: Read-only

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							

The unicast hash enable (UNIHEN) and the multicast hash enable (MITIHEN) bits in the Network Configuration Register (Section 36.7.2 “Network Configuration Register”) enable the reception of hash matched frames. See Section 36.5.9 “Hash Addressing”.

- **ADDR: Hash Address**

The first 32 bits of the Hash Address Register.

36.7.20 Hash Register Top [63:32]

Name: GMAC_HRT

Address: 0xF0028084

Access: Read-only

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							

The unicast hash enable (UNIHEN) and the multicast hash enable (MITIHEN) bits in the "Network Configuration Register" enable the reception of hash matched frames. See Section 36.5.9 "Hash Addressing".

- **ADDR: Hash Address**

Bits 63 to 32 of the Hash Address Register.

36.7.21 Specific Address 1 Bottom [31:0] Register

Name: GMAC_SAB1

Address: 0xF0028088

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 1**

Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

36.7.22 Specific Address 1 Top [47:32] Register

Name: GMAC_SAT1

Address: 0xF002808C

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 1**

The most significant bits of the destination address, that is bits 47:32.

36.7.23 Specific Address 2 Bottom [31:0] Register

Name: GMAC_SAB2

Address: 0xF0028090

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 2**

Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

36.7.24 Specific Address 2 Top [47:32] Register

Name: GMAC_SAT2

Address: 0xF0028094

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 2**

The most significant bits of the destination address, that is bits 47:32.

36.7.25 Specific Address 3 Bottom [31:0] Register

Name: GMAC_SAB3

Address: 0xF0028098

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 3**

Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

36.7.26 Specific Address 3 Top [47:32] Register

Name: GMAC_SAT3

Address: 0xF002809C

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 3**

The most significant bits of the destination address, that is bits 47:32.

36.7.27 Specific Address 4 Bottom Register[31:0]

Name: GMAC_SAB4

Address: 0xF00280A0

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 4**

Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

36.7.28 Specific Address 4 Top Register[47:32]

Name: GMAC_SAT4

Address: 0xF00280A4

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							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 4**

The most significant bits of the destination address, that is bits 47:32.

36.7.29 Type ID Match 1 Register

Name: GMAC_TIDM1

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 Match 1**

For use in comparisons with received frames type ID/length frames.

36.7.30 Type ID Match 2 Register

Name: GMAC_TIDM2

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 Match 2**

For use in comparisons with received frames type ID/length frames.

36.7.31 Type ID Match 3 Register

Name: GMAC_TIDM3

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 Match 3**

For use in comparisons with received frames type ID/length frames.

36.7.32 Type ID Match 4 Register

Name: GMAC_TIDM4

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 Match 4**

For use in comparisons with received frames type ID/length frames.

36.7.33 Wake on LAN Register

Name: GMAC_WOL

Address: 0xF00280B8

Access: Read-write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–				MTI	SA1	ARP	MAG

15	14	13	12	11	10	9	8
IP							

7	6	5	4	3	2	1	0
IP							

- **IP: ARP Request IP Address**

Wake on LAN ARP request IP address. Written to define the least significant 16 bits of the target IP address that is matched to generate a Wake on LAN event. A value of zero will not generate an event, even if this is matched by the received frame.

- **MAG: Magic Packet Event Enable**

Wake on LAN magic packet event enable.

- **ARP: ARP Request Event Enable**

Wake on LAN ARP request event enable.

- **SA1: Specific Address Register 1 Event Enable**

Wake on LAN Specific Address Register 1 event enable.

- **MTI: Multicast Hash Event Enable**

Wake on LAN multicast hash event enable.

36.7.34 IPG Stretch Register

Name: GMAC_IPGS

Address: 0xF00280BC

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
FL							

7	6	5	4	3	2	1	0
FL							

- **FL: Frame Length**

Bits 7:0 are multiplied with the previously transmitted frame length (including preamble). Bits 15:8 +1 divide the frame length. If the resulting number is greater than 96 and bit 28 is set in the Network Configuration Register then the resulting number is used for the transmit inter-packet-gap. 1 is added to bits 15:8 to prevent a divide by zero. See Section 36.5.4 “MAC Transmit Block”.

36.7.35 Stacked VLAN Register

Name: GMAC_SVLAN

Address: 0xF00280C0

Access: Read-write

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

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

15	14	13	12	11	10	9	8
VLAN_TYPE							

7	6	5	4	3	2	1	0
VLAN_TYPE							

- **VLAN_TYPE: User Defined VLAN_TYPE Field**

User defined VLAN_TYPE field. When Stacked VLAN is enabled, the first VLAN tag in a received frame will only be accepted if the VLAN type field is equal to this user defined VLAN_TYPE, OR equal to the standard VLAN type (0x8100). Note that the second VLAN tag of a Stacked VLAN packet will only be matched correctly if its VLAN_TYPE field equals 0x8100.

- **ESVLAN: Enable Stacked VLAN Processing Mode**

0: Disable the stacked VLAN processing mode

1: Enable the stacked VLAN processing mode

36.7.36 Transmit PFC Pause Register

Name: GMAC_TPFCP

Address: 0xF00280C4

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
PQ							

7	6	5	4	3	2	1	0
PEV							

- **PEV: Priority Enable Vector**

If bit 17 of the Network Control Register is written with a one then the priority enable vector of the PFC priority based pause frame will be set equal to the value stored in this register [7:0].

- **PQ: Pause Quantum**

If bit 17 of the Network Control Register is written with a one then for each entry equal to zero in the Transmit PFC Pause Register[15:8], the PFC pause frame's pause quantum field associated with that entry will be taken from the Transmit Pause Quantum Register. For each entry equal to one in the Transmit PFC Pause Register [15:8], the pause quantum associated with that entry will be zero.

36.7.37 Specific Address 1 Mask Bottom [31:0] Register

Name: GMAC_SAMB1

Address: 0xF00280C8

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: Specific Address 1 Mask**

Setting a bit to one masks the corresponding bit in the Specific Address 1 Register.

36.7.38 Specific Address Mask 1 Top [47:32] Register

Name: GMAC_SAMT1

Address: 0xF00280CC

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: Specific Address 1 Mask**

Setting a bit to one masks the corresponding bit in the Specific Address 1 Register.

36.7.39 Octets Transmitted [31:0] Register

Name: GMAC_OTLO

Address: 0xF0028100

Access: Read-only

31	30	29	28	27	26	25	24
TXO							

23	22	21	20	19	18	17	16
TXO							

15	14	13	12	11	10	9	8
TXO							

7	6	5	4	3	2	1	0
TXO							

When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

- **TXO: Transmitted Octets**

Transmitted octets in frame without errors [31:0]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames.

36.7.40 Octets Transmitted [47:32] Register

Name: GMAC_OTHI

Address: 0xF0028104

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
TXO							

7	6	5	4	3	2	1	0
TXO							

When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

- **TXO: Transmitted Octets**

Transmitted octets in frame without errors [47:32]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames.

36.7.41 Frames Transmitted Register

Name: GMAC_FT

Address: 0xF0028108

Access: Read-only

31	30	29	28	27	26	25	24
FTX							

23	22	21	20	19	18	17	16
FTX							

15	14	13	12	11	10	9	8
FTX							

7	6	5	4	3	2	1	0
FTX							

- **FTX: Frames Transmitted without Error**

Frames transmitted without error. This register counts the number of frames successfully transmitted, i.e., no under run and not too many retries. Excludes pause frames.

36.7.42 Broadcast Frames Transmitted Register

Name: GMAC_BCFT

Address: 0xF002810C

Access: Read-only

31	30	29	28	27	26	25	24
BFTX							

23	22	21	20	19	18	17	16
BFTX							

15	14	13	12	11	10	9	8
BFTX							

7	6	5	4	3	2	1	0
BFTX							

- **BFTX: Broadcast Frames Transmitted without Error**

Broadcast frames transmitted without error. This register counts the number of broadcast frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

36.7.43 Multicast Frames Transmitted Register

Name: GMAC_MFT

Address: 0xF0028110

Access: Read-only

31	30	29	28	27	26	25	24
MFTX							

23	22	21	20	19	18	17	16
MFTX							

15	14	13	12	11	10	9	8
MFTX							

7	6	5	4	3	2	1	0
MFTX							

- **MFTX: Multicast Frames Transmitted without Error**

This register counts the number of multicast frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

36.7.44 Pause Frames Transmitted Register

Name: GMAC_PFT

Address: 0xF0028114

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
PFTX							

7	6	5	4	3	2	1	0
PFTX							

- **PFTX: Pause Frames Transmitted Register**

This register counts the number of pause frames transmitted. Only pause frames triggered by the register interface or through the external pause pins are counted as pause frames. Pause frames received through FIFO are counted in the frames transmitted counter.

36.7.45 64 Byte Frames Transmitted Register

Name: GMAC_BFT64

Address: 0xF0028118

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 64 Byte Frames Transmitted without Error**

This register counts the number of 64 byte frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

36.7.46 65 to 127 Byte Frames Transmitted Register

Name: GMAC_TBFT127

Address: 0xF002811C

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 65 to 127 Byte Frames Transmitted without Error**

This register counts the number of 65 to 127 byte frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

36.7.47 128 to 255 Byte Frames Transmitted Register

Name: GMAC_TBFT255

Address: 0xF0028120

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 128 to 255 Byte Frames Transmitted without Error**

This register counts the number of 128 to 255 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

36.7.48 256 to 511 Byte Frames Transmitted Register

Name: GMAC_TBFT511

Address: 0xF0028124

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 256 to 511 Byte Frames Transmitted without Error**

This register counts the number of 256 to 511 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

36.7.49 512 to 1023 Byte Frames Transmitted Register

Name: GMAC_TBFT1023

Address: 0xF0028128

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 512 to 1023 Byte Frames Transmitted without Error**

This register counts the number of 512 to 1023 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

36.7.50 1024 to 1518 Byte Frames Transmitted Register

Name: GMAC_TBFT1518

Address: 0xF002812C

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 1024 to 1518 Byte Frames Transmitted without Error**

This register counts the number of 1024 to 1518 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

36.7.51 Greater Than 1518 Byte Frames Transmitted Register

Name: GMAC_GTBFT1518

Address: 0xF0028130

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: Greater than 1518 Byte Frames Transmitted without Error**

This register counts the number of 1518 or above byte frames successfully transmitted without error i.e., no under run and not too many retries.

36.7.52 Transmit Under Runs Register

Name: GMAC_TUR

Address: 0xF0028134

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
–	–	–	–	–	–	TXUNR	

7	6	5	4	3	2	1	0
TXUNR							

- **TXUNR: Transmit Under Runs**

This register counts the number of frames not transmitted due to a transmit under run. If this register is incremented then no other statistics register is incremented.

36.7.53 Single Collision Frames Register

Name: GMAC_SCF

Address: 0xF0028138

Access: Read-only

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

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

15	14	13	12	11	10	9	8
SCOL							

7	6	5	4	3	2	1	0
SCOL							

- **SCOL: Single Collision**

This register counts the number of frames experiencing a single collision before being successfully transmitted i.e., no under run.

36.7.54 Multiple Collision Frames Register

Name: GMAC_MCF

Address: 0xF002813C

Access: Read-only

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

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

15	14	13	12	11	10	9	8
MCOL							

7	6	5	4	3	2	1	0
MCOL							

- **MCOL: Multiple Collision**

This register counts the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no under run and not too many retries.

36.7.55 Excessive Collisions Register

Name: GMAC_EC

Address: 0xF0028140

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
–	–	–	–	–	–	XCOL	

7	6	5	4	3	2	1	0
XCOL							

- **XCOL: Excessive Collisions**

This register counts the number of frames that failed to be transmitted because they experienced 16 collisions.

36.7.56 Late Collisions Register

Name: GMAC_LC

Address: 0xF0028144

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
–	–	–	–	–	–	LCOL	

7	6	5	4	3	2	1	0
LCOL							

- **LCOL: Late Collisions**

This register counts the number of late collisions occurring after the slot time (512 bits) has expired. In 10/100 mode, late collisions are counted twice i.e., both as a collision and a late collision. In gigabit mode, a late collision causes the transmission to be aborted, thus the single and multi collision registers are not updated.

36.7.57 Deferred Transmission Frames Register

Name: GMAC_DTF

Address: 0xF0028148

Access: Read-only

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

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

15	14	13	12	11	10	9	8
DEFT							

7	6	5	4	3	2	1	0
DEFT							

- **DEFT: Deferred Transmission**

This register counts 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 under run.

36.7.58 Carrier Sense Errors Register

Name: GMAC_CSE

Address: 0xF002814C

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
–	–	–	–	–	–	CSR	

7	6	5	4	3	2	1	0
CSR							

- **CSR: Carrier Sense Error**

This register counts 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 under run). 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.

36.7.59 Octets Received [31:0] Register

Name: GMAC_ORLO

Address: 0xF0028150

Access: Read-only

31	30	29	28	27	26	25	24
RXO							

23	22	21	20	19	18	17	16
RXO							

15	14	13	12	11	10	9	8
RXO							

7	6	5	4	3	2	1	0
RXO							

When reading the Octets Transmitted and Octets Received Registers, bits [31:0] should be read prior to bits [47:32] to ensure reliable operation.

- **RXO: Received Octets**

Received octets in frame without errors [31:0]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.60 Octets Received [47:32] Register

Name: GMAC_ORHI

Address: 0xF0028154

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
RXO							

7	6	5	4	3	2	1	0
RXO							

When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

- **RXO: Received Octets**

Received octets in frame without errors [47:32]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.61 Frames Received Register

Name: GMAC_FR

Address: 0xF0028158

Access: Read-only

31	30	29	28	27	26	25	24
FRX							

23	22	21	20	19	18	17	16
FRX							

15	14	13	12	11	10	9	8
FRX							

7	6	5	4	3	2	1	0
FRX							

- **FRX: Frames Received without Error**

Frames received without error. This register counts the number of frames successfully received. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.62 Broadcast Frames Received Register

Name: GMAC_BCFR

Address: 0xF002815C

Access: Read-only

31	30	29	28	27	26	25	24
BFRX							

23	22	21	20	19	18	17	16
BFRX							

15	14	13	12	11	10	9	8
BFRX							

7	6	5	4	3	2	1	0
BFRX							

- **BFRX: Broadcast Frames Received without Error**

Broadcast frames received without error. This register counts the number of broadcast frames successfully received. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.63 Multicast Frames Received Register

Name: GMAC_MFR

Address: 0xF0028160

Access: Read-only

31	30	29	28	27	26	25	24
MFRX							

23	22	21	20	19	18	17	16
MFRX							

15	14	13	12	11	10	9	8
MFRX							

7	6	5	4	3	2	1	0
MFRX							

- **MFRX: Multicast Frames Received without Error**

This register counts the number of multicast frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.64 Pause Frames Received Register

Name: GMAC_PFR

Address: 0xF0028164

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
PFRX							

7	6	5	4	3	2	1	0
PFRX							

- **PFRX: Pause Frames Received Register**

This register counts the number of pause frames received without error.

36.7.65 64 Byte Frames Received Register

Name: GMAC_BFR64

Address: 0xF0028168

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 64 Byte Frames Received without Error**

This register counts the number of 64 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.66 65 to 127 Byte Frames Received Register

Name: GMAC_TBFR127

Address: 0xF002816C

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 65 to 127 Byte Frames Received without Error**

This register counts the number of 65 to 127 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.67 128 to 255 Byte Frames Received Register

Name: GMAC_TBFR255

Address: 0xF0028170

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 128 to 255 Byte Frames Received without Error**

This register counts the number of 128 to 255 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.68 256 to 511 Byte Frames Received Register

Name: GMAC_TBFR511

Address: 0xF0028174

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 256 to 511 Byte Frames Received without Error**

This register counts the number of 256 to 511 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.69 512 to 1023 Byte Frames Received Register

Name: GMAC_TBFR1023

Address: 0xF0028178

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 512 to 1023 Byte Frames Received without Error**

This register counts the number of 512 to 1023 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

36.7.70 1024 to 1518 Byte Frames Received Register

Name: GMAC_TBFR1518

Address: 0xF002817C

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 1024 to 1518 Byte Frames Received without Error**

This register counts the number of 1024 to 1518 byte frames successfully received without error, i.e., no under run and not too many retries.

36.7.71 1519 to Maximum Byte Frames Received Register

Name: GMAC_TMXBFR

Address: 0xF0028180

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 1519 to Maximum Byte Frames Received without Error**

This register counts the number of 1519 byte or above frames successfully received without error. Maximum frame size is determined by the Network Configuration Register bit 8 (1536 maximum frame size) or bit 3 (jumbo frame size). Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory. See: Section 36.7.2 “Network Configuration Register”.

36.7.72 Undersized Frames Received Register

Name: GMAC_UFR

Address: 0xF0028184

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
–	–	–	–	–	–	UFRX	

7	6	5	4	3	2	1	0
UFRX							

- **UFRX: Undersize Frames Received**

This register counts the number of frames received less than 64 bytes in length (10/100 mode or gigabit mode, full duplex) that do not have either a CRC error or an alignment error. In gigabit mode, half duplex, this register counts either frames not conforming to the minimum slot time of 512 bytes or frames not conforming to the minimum frame size once bursting is active.

36.7.73 Oversized Frames Received Register

Name: GMAC_OFR

Address: 0xF0028188

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
–	–	–	–	–	–	OFRX	

7	6	5	4	3	2	1	0
OFRX							

- **OFRX: Oversized Frames Received**

This register counts the number of frames received exceeding 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register, 10240 bytes if bit 3 is set in the Network Configuration Register) in length but do not have either a CRC error, an alignment error nor a receive symbol error. See Section 36.7.2 “Network Configuration Register”.

36.7.74 Jabbers Received Register

Name: GMAC_JR

Address: 0xF002818C

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
–	–	–	–	–	–	JRX	

7	6	5	4	3	2	1	0
JRX							

- **JRX: Jabbers Received**

The register counts the number of frames received exceeding 1518 bytes in length (1536 if bit 8 is set in Network Configuration Register, 10240 bytes if bit 3 is set in the Network Configuration Register) and have either a CRC error, an alignment error or a receive symbol error. See: Section 36.7.2 “Network Configuration Register”.

36.7.75 Frame Check Sequence Errors Register

Name: GMAC_FCSE

Address: 0xF0028190

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
–	–	–	–	–	–	FCKR	

7	6	5	4	3	2	1	0
FCKR							

- **FCKR: Frame Check Sequence Errors**

The register counts frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit 8 is set in Network Configuration Register, 10240 bytes if bit 3 is set in the 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.

This register is incremented for a frame with bad FCS, regardless of whether it is copied to memory due to ignore FCS mode being enabled in bit 26 of the Network Configuration Register. See: Section 36.7.2 “Network Configuration Register”.

36.7.76 Length Field Frame Errors Register

Name: GMAC_LFFE

Address: 0xF0028194

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
–	–	–	–	–	–	LFER	

7	6	5	4	3	2	1	0
LFER							

- **LFER: Length Field Frame Errors**

This register counts the number of frames received that have a measured length shorter than that extracted from the length field (bytes 13 and 14). This condition is only counted if the value of the length field is less than 0x0600, the frame is not of excessive length and checking is enabled through bit 16 of the Network Configuration Register. See: Section 36.7.2 “Network Configuration Register”.

36.7.77 Receive Symbol Errors Register

Name: GMAC_RSE

Address: 0xF0028198

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
–	–	–	–	–	–	RXSE	

7	6	5	4	3	2	1	0
RXSE							

- **RXSE: Receive Symbol Errors**

This register counts the number of frames that had GRXER asserted during reception. For 10/100 mode symbol errors are counted regardless of frame length checks. For gigabit mode the frame must satisfy slot time requirements in order to count a symbol error. Additionally, in gigabit half duplex mode, carrier extension errors are also recorded. Receive symbol errors will also be counted as an FCS or alignment error if the frame is between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register, 10240 bytes if bit 3 is set in the Network Configuration Register). If the frame is larger it will be recorded as a jabber error. See Section 36.7.2 “Network Configuration Register”.

36.7.78 Alignment Errors Register

Name: GMAC_AE

Address: 0xF002819C

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
–	–	–	–	–	–	AER	

7	6	5	4	3	2	1	0
AER							

- **AER: Alignment Errors**

This register counts the 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 is set in Network Configuration Register, 10240 bytes if bit 3 is set in the 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. See: Section 36.7.2 “Network Configuration Register”.

36.7.79 Receive Resource Errors Register

Name: GMAC_RRE

Address: 0xF00281A0

Access: Read-only

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

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

15	14	13	12	11	10	9	8
RXRER							

7	6	5	4	3	2	1	0
RXRER							

- **RXRER: Receive Resource Errors**

This register counts 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 is set in Network Configuration Register, 10240 bytes if bit 3 is set in the 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. See: Section 36.7.2 “Network Configuration Register”.

36.7.80 Receive Overruns Register

Name: GMAC_ROE

Address: 0xF00281A4

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
–	–	–	–	–	–	RXOVR	

7	6	5	4	3	2	1	0
RXOVR							

- **RXOVR: Receive Overruns**

This register counts the number of frames that are address recognized but were not copied to memory due to a receive overrun.

36.7.81 IP Header Checksum Errors Register

Name: GMAC_IHCE

Address: 0xF00281A8

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
HCKER							

- **HCKER: IP Header Checksum Errors**

This register counts the number of frames discarded due to an incorrect IP header checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register or 10240 bytes if bit 3 is set in the Network Configuration Register) and do not have a CRC error, an alignment error, nor a symbol error.

36.7.82 TCP Checksum Errors Register

Name: GMAC_TCE

Address: 0xF00281AC

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
TCKER							

- **TCKER: TCP Checksum Errors**

This register counts the number of frames discarded due to an incorrect TCP checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register or 10240 bytes if bit 3 is set in the Network Configuration Register) and do not have a CRC error, an alignment error, nor a symbol error.

36.7.83 UDP Checksum Errors Register

Name: GMAC_UCE

Address: 0xF00281B0

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
UCKER							

- **UCKER: UDP Checksum Errors**

This register counts the number of frames discarded due to an incorrect UDP checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register or 10240 bytes if bit 3 is set in the Network Configuration Register) and do not have a CRC error, an alignment error, nor a symbol error.

36.7.84 1588 Timer Sync Strobe Seconds Register

Name: GMAC_TSSS

Address: 0xF00281C8

Access: Read-write

31	30	29	28	27	26	25	24
VTS							

23	22	21	20	19	18	17	16
VTS							

15	14	13	12	11	10	9	8
VTS							

7	6	5	4	3	2	1	0
VTS							

- **VTS: Value of Timer Seconds Register Capture**

The value of the Timer Seconds Register is captured.

36.7.85 1588 Timer Sync Strobe Nanoseconds Register

Name: GMAC_TSSN

Address: 0xF00281CC

Access: Read-write

31	30	29	28	27	26	25	24
–	–	VTN					

23	22	21	20	19	18	17	16
VTN							

15	14	13	12	11	10	9	8
VTN							

7	6	5	4	3	2	1	0
VTN							

- **VTN: Value Timer Nanoseconds Register Capture**

The value of the Timer Nanoseconds Register is captured.

36.7.86 1588 Timer Seconds Register

Name: GMAC_TS

Address: 0xF00281D0

Access: Read-write

31	30	29	28	27	26	25	24
TCS							

23	22	21	20	19	18	17	16
TCS							

15	14	13	12	11	10	9	8
TCS							

7	6	5	4	3	2	1	0
TCS							

- **TCS: Timer Count in Seconds**

This register is writable. It increments by one when the 1588 nanoseconds counter counts to one second. It may also be incremented when the Timer Adjust Register is written.

36.7.87 1588 Timer Nanoseconds Register

Name: GMAC_TN

Address: 0xF00281D4

Access: Read-write

31	30	29	28	27	26	25	24
–	–	TNS					

23	22	21	20	19	18	17	16
TNS							

15	14	13	12	11	10	9	8
TNS							

7	6	5	4	3	2	1	0
TNS							

- **TNS: Timer Count in Nanoseconds**

This register is writable. It can also be adjusted by writes to the 1588 Timer Adjust Register. It increments by the value of the 1588 Timer Increment Register each clock cycle.

36.7.88 1588 Timer Adjust Register

Name: GMAC_TA

Address: 0xF00281D8

Access: Write-only

31	30	29	28	27	26	25	24
ADJ	–	ITDT					

23	22	21	20	19	18	17	16
ITDT							

15	14	13	12	11	10	9	8
ITDT							

7	6	5	4	3	2	1	0
ITDT							

- **ITDT: Increment/Decrement**

The number of nanoseconds to increment or decrement the 1588 Timer Nanoseconds Register. If necessary, the 1588 Seconds Register will be incremented or decremented.

- **ADJ: Adjust 1588 Timer**

Write as one to subtract from the 1588 timer. Write as zero to add to it.

36.7.89 1588 Timer Increment Register

Name: GMAC_TI

Address: 0xF00281DC

Access: Read-write

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

23	22	21	20	19	18	17	16
NIT							

15	14	13	12	11	10	9	8
AC NS							

7	6	5	4	3	2	1	0
CNS							

- **CNS: Count Nanoseconds**

A count of nanoseconds by which the 1588 Timer Nanoseconds Register will be incremented each clock cycle.

- **ACNS: Alternative Count Nanoseconds**

Alternative count of nanoseconds by which the 1588 Timer Nanoseconds Register will be incremented each clock cycle.

- **NIT: Number of Increments**

The number of increments after which the alternative increment is used.

36.7.90 PTP Event Frame Transmitted Seconds Register

Name: GMAC_EFTS

Address: 0xF00281E0

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.91 PTP Event Frame Transmitted Nanoseconds Register

Name: GMAC_EFTN

Address: 0xF00281E4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.92 PTP Event Frame Received Seconds Register

Name: GMAC_EFRS

Address: 0xF00281E8

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.93 PTP Event Frame Received Nanoseconds Register

Name: GMAC_EFRN

Address: 0xF00281EC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.94 PTP Peer Event Frame Transmitted Seconds Register

Name: GMAC_PEFTS

Address: 0xF00281F0

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.95 PTP Peer Event Frame Transmitted Nanoseconds Register

Name: GMAC_PEFTN

Address: 0xF00281F4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.96 PTP Peer Event Frame Received Seconds Register

Name: GMAC_PEFRS

Address: 0xF00281F8

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

36.7.97 PTP Peer Event Frame Received Nanoseconds Register

Name: GMAC_PEFRN

Address: 0xF00281FC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

37. Ethernet MAC 10/100 (EMAC)

37.1 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.

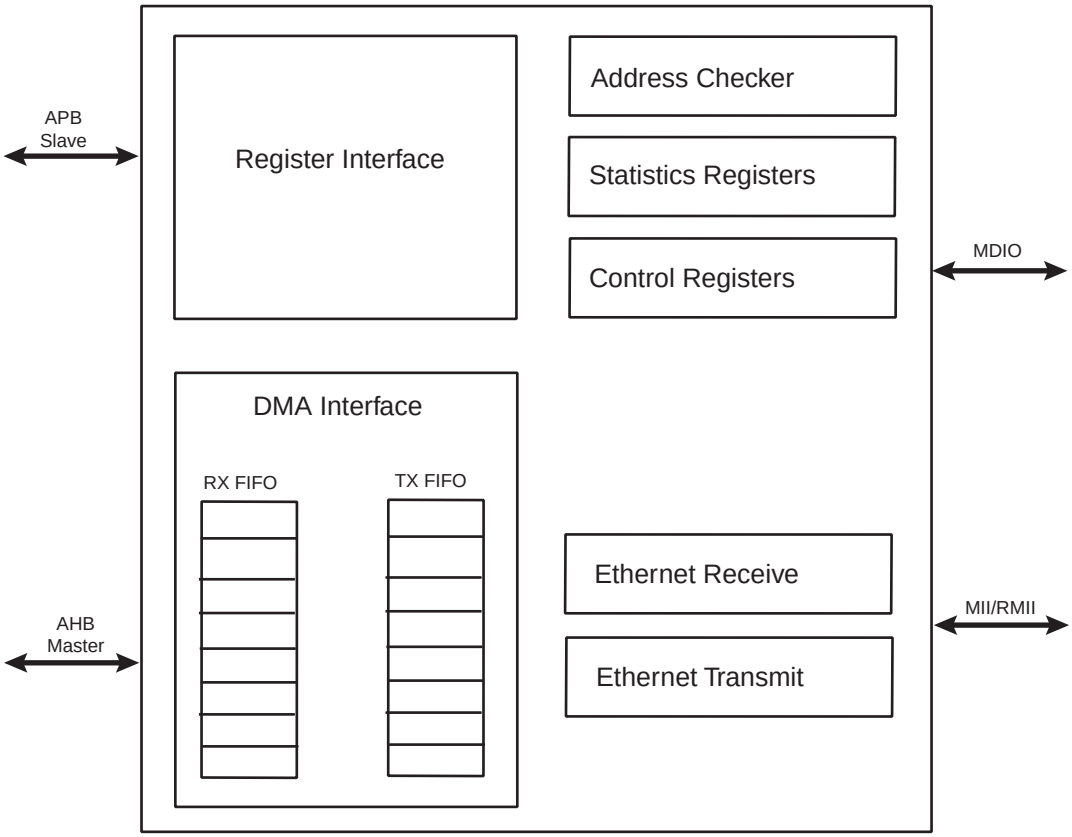
The statistics register block contains registers for counting various types of event associated with transmit and receive operations. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3.

37.2 Embedded Characteristics

- Supports RMII Interface to the physical layer
- 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
- Wake-on-LAN Support
- Jumbo Frames Up to 10240 bytes Supported

37.3 Block Diagram

Figure 37-1. EMAC Block Diagram



37.4 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 37-1 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.

37.4.1 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.

37.4.2 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

37.4.2.1 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.

37.4.2.2 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 37-1 for details of the receive buffer descriptor list.

Table 37-1. 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
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.

Table 37-1. Receive Buffer Descriptor Entry (Continued)

Bit	Function
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.

37.4.2.3 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 37-2 on page 1080 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 37-2. 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

37.4.3 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.

37.4.4 Pause Frame Support

The start of an 802.3 pause frame is as follows:

Table 37-3. 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).

37.4.5 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.

37.4.6 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

37.4.7 Broadcast Address

The broadcast address of 0xFFFFFFFFFFFF is recognized if the ‘no broadcast’ bit in the network configuration register is zero.

37.4.8 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.

37.4.9 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.

37.4.10 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.

37.4.11 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 37-4. 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

37.4.12 Wake-on-LAN Support

The receive block supports Wake-on-LAN by detecting the following events on incoming receive frames:

- Magic packet
- ARP request to the device IP address
- Specific address 1 filter match
- Multicast hash filter match

If one of these events occurs Wake-on-LAN detection is indicated by asserting the `wol` output pin for 64 `rx_clk` cycles. These events can be individually enabled through bits[19:16] of the Wake-on-LAN register. Also, for Wake-on-LAN detection to occur, receive enable must be set in the network control register, however a receive buffer does not have to be available. `wol` assertion due to ARP request, specific address 1 or multicast filter events occurs even if the frame is errored. For magic packet events, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

- Magic packet events are enabled through bit 16 of the Wake-on-LAN register
- The frame's destination address matches specific address 1
- The frame is correctly formed with no errors
- The frame contains at least 6 bytes of 0xFF for synchronization
- There are 16 repetitions of the contents of specific address 1 register immediately following the synchronization

An ARP request event is detected if all of the following are true:

- ARP request events are enabled through bit 17 of the Wake-on-LAN register
- Broadcasts are allowed by bit 5 in the network configuration register
- The frame has a broadcast destination address (bytes 1 to 6)
- The frame has a type ID field of 0x0806 (bytes 13 and 14)
- The frame has an ARP operation field of 0x0001 (bytes 21 and 22)
- The least significant 16 bits of the frame's ARP target protocol address (bytes 41 and 42) match the value programmed in bits[15:0] of the Wake-on-LAN register

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for the Wake-on-LAN target address value does not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event occurs if all of the following are true:

- Specific address 1 events are enabled through bit 18 of the Wake-on-LAN register
- The frame's destination address matches the value programmed in the specific address 1 registers

A multicast filter match event occurs if all of the following are true:

- Multicast hash events are enabled through bit 19 of the Wake-on-LAN register
- Multicast hash filtering is enabled through bit 6 of the network configuration register
- The frame's destination address matches against the multicast hash filter
- The frame's destination address is not a broadcast

37.4.13 PHY Maintenance

The register EMAC_MAN enables the EMAC to communicate with a PHY by means of the MDIO interface. It is used during auto-negotiation to ensure that the EMAC and the PHY are configured for the same speed and duplex configuration.

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the network status register (about 2000 MCK cycles later when bit ten is set to zero, and bit eleven is set to one in the network configuration register). An interrupt is generated as this bit is set. During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each MDC cycle. This causes transmission of a PHY management frame on MDIO.

Reading during the shift operation returns the current contents of the shift register. At the end of management operation, the bits have shifted back to their original locations. For a read operation, the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The MDIO interface can read IEEE 802.3 clause 45 PHYs as well as clause 22 PHYs. To read clause 45 PHYs, bits[31:28] should be written as 0x0011. For a description of MDC generation, see the network configuration register in the “Network Control Register” on page 1091.

37.4.14 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 RMII interface are described in Table 37-5.

Table 37-5. Pin Configuration

Pin Name	RMII
ETXCK_EREFCK	EREFCK: Reference Clock
ECRS	
ECOL	
ERXDV	ECRSDV: Carrier Sense/Data Valid
ERX0 - ERX3	ERX0 - ERX1: 2-bit Receive Data
ERXER	ERXER: Receive Error
ERXCK	
ETXEN	ETXEN: Transmit Enable
ETX0-ETX3	ETX0 - ETX1: 2-bit Transmit Data
ETXER	

The intent of the RMII is to provide a reduced pin count alternative to the IEEE 802.3u MII. It uses 2 bits for transmit (ETX0 and ETX1) and two bits for receive (ERX0 and ERX1). There is a Transmit Enable (ETXEN), a Receive Error (ERXER), a Carrier Sense (ECRS_DV), and a 50 MHz Reference Clock (ETXCK_EREFCK) for 100Mb/s data rate.

37.4.14.1 RMII Transmit and Receive Operation

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.

37.5 Programming Interface

37.5.1 Initialization

37.5.1.1 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:

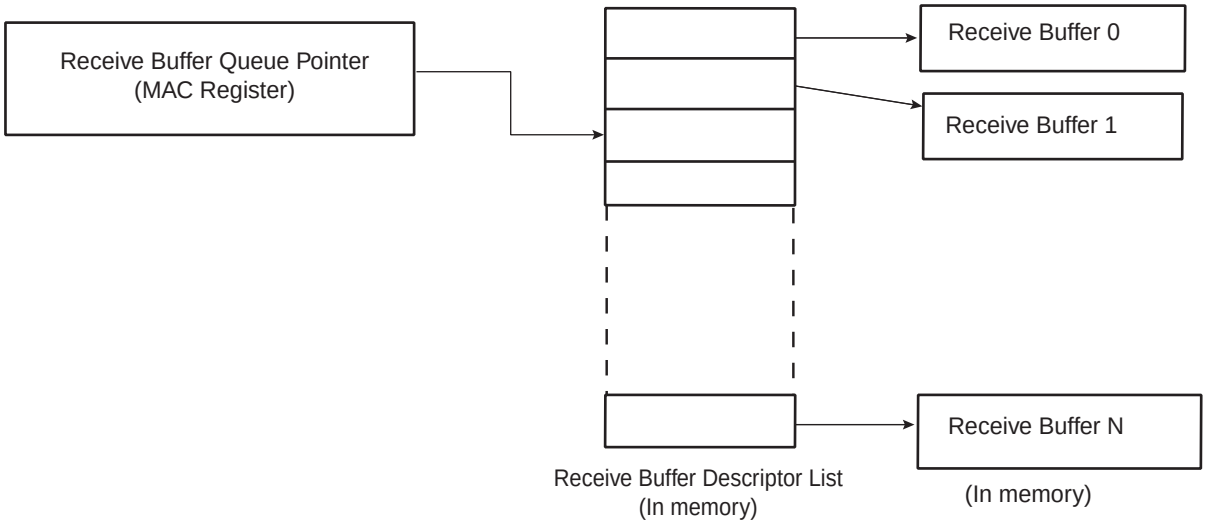
1. Write to network control register to disable transmit and receive circuits.
2. Write to network control register to change loop-back mode.
3. Write to network control register to re-enable transmit or receive circuits.

Note: These writes to network control register cannot be combined in any way.

37.5.1.2 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 1077. It points to this data structure.

Figure 37-2. 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.

37.5.1.3 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 37-2 on page 1080) 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.

37.5.1.4 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 1081. for details of address matching. Each register-pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

37.5.1.5 Interrupts

There are 15 interrupt conditions that are detected within the EMAC. These are ORed to make a single interrupt. Depending on the overall system design, this may be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU enters the interrupt handler (Refer to the 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.

37.5.1.6 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.

37.5.1.7 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 37-1 on page 1077) 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.

37.6 Ethernet MAC 10/100 (EMAC) User Interface

Table 37-6. 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_7FFF
0x34	Phy Maintenance Register	EMAC_MAN	Read-write	0x0000_0000
0x38	Pause Time Register	EMAC_PTR	Read-write	0x0000_0000
0x3C	Pause Frames Received Register	EMAC_PFR	Read-write	0x0000_0000
0x40	Frames Transmitted Ok Register	EMAC_FTO	Read-write	0x0000_0000
0x44	Single Collision Frames Register	EMAC_SCF	Read-write	0x0000_0000
0x48	Multiple Collision Frames Register	EMAC_MCF	Read-write	0x0000_0000
0x4C	Frames Received Ok Register	EMAC_FRO	Read-write	0x0000_0000
0x50	Frame Check Sequence Errors Register	EMAC_FCSE	Read-write	0x0000_0000
0x54	Alignment Errors Register	EMAC_ALE	Read-write	0x0000_0000
0x58	Deferred Transmission Frames Register	EMAC_DTF	Read-write	0x0000_0000
0x5C	Late Collisions Register	EMAC_LCOL	Read-write	0x0000_0000
0x60	Excessive Collisions Register	EMAC_ECOL	Read-write	0x0000_0000
0x64	Transmit Underrun Errors Register	EMAC_TUND	Read-write	0x0000_0000
0x68	Carrier Sense Errors Register	EMAC_CSE	Read-write	0x0000_0000
0x6C	Receive Resource Errors Register	EMAC_RRE	Read-write	0x0000_0000
0x70	Receive Overrun Errors Register	EMAC_ROV	Read-write	0x0000_0000
0x74	Receive Symbol Errors Register	EMAC_RSE	Read-write	0x0000_0000
0x78	Excessive Length Errors Register	EMAC_ELE	Read-write	0x0000_0000
0x7C	Receive Jabbers Register	EMAC_RJA	Read-write	0x0000_0000
0x80	Undersize Frames Register	EMAC_USF	Read-write	0x0000_0000
0x84	SQE Test Errors Register	EMAC_STE	Read-write	0x0000_0000
0x88	Received Length Field Mismatch Register	EMAC_RLE	Read-write	0x0000_0000

Table 37-6. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x90	Hash Register Bottom [31:0] Register	EMAC_HRB	Read-write	0x0000_0000
0x94	Hash Register Top [63:32] Register	EMAC_HRT	Read-write	0x0000_0000
0x98	Specific Address 1 Bottom Register	EMAC_SA1B	Read-write	0x0000_0000
0x9C	Specific Address 1 Top Register	EMAC_SA1T	Read-write	0x0000_0000
0xA0	Specific Address 2 Bottom Register	EMAC_SA2B	Read-write	0x0000_0000
0xA4	Specific Address 2 Top Register	EMAC_SA2T	Read-write	0x0000_0000
0xA8	Specific Address 3 Bottom Register	EMAC_SA3B	Read-write	0x0000_0000
0xAC	Specific Address 3 Top Register	EMAC_SA3T	Read-write	0x0000_0000
0xB0	Specific Address 4 Bottom Register	EMAC_SA4B	Read-write	0x0000_0000
0xB4	Specific Address 4 Top Register	EMAC_SA4T	Read-write	0x0000_0000
0xB8	Type ID Checking Register	EMAC_TID	Read-write	0x0000_0000
0xC0	User Input/Output Register	EMAC_USRIO	Read-write	0x0000_0000
0xC4	Wake on LAN Register	EMAC_WOL	Read-write	0x0000_0000
0xC8 - 0xFC	Reserved	–	–	–

37.6.1 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.

37.6.2 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.

37.6.3 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).

37.6.4 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.

37.6.5 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.

37.6.6 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.

37.6.7 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.

37.6.8 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
–	WOL	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.

- **WOL: Wake On LAN**

Set when a WOL event has been triggered (This flag can be set even if the EMAC is not clocked). Cleared on a read.

37.6.9 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
–	WOL	PTZ	PFR	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame sent**

Enable management done interrupt.

- **RCOMP: Receive Complete**

Enable receive complete interrupt.

- **RXUBR: Receive Used Bit Read**

Enable receive used bit read interrupt.

- **TXUBR: Transmit Used Bit Read**

Enable transmit used bit read interrupt.

- **TUND: Ethernet Transmit Buffer Underrun**

Enable transmit underrun interrupt.

- **RLE: Retry Limit Exceeded**

Enable retry limit exceeded interrupt.

- **TXERR**

Enable transmit buffers exhausted in mid-frame interrupt.

- **TCOMP: Transmit Complete**

Enable transmit complete interrupt.

- **ROVR: Receive Overrun**

Enable receive overrun interrupt.

- **HRESP: Hresp not OK**

Enable Hresp not OK interrupt.

- **PFR: Pause Frame Received**

Enable pause frame received interrupt.

- **PTZ: Pause Time Zero**

Enable pause time zero interrupt.

- **WOL: Wake On LAN**

Enable Wake On LAN interrupt.

37.6.10 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
–	WOL	PTZ	PFR	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame sent**

Disable management done interrupt.

- **RCOMP: Receive Complete**

Disable receive complete interrupt.

- **RXUBR: Receive Used Bit Read**

Disable receive used bit read interrupt.

- **TXUBR: Transmit Used Bit Read**

Disable transmit used bit read interrupt.

- **TUND: Ethernet Transmit Buffer Underrun**

Disable transmit underrun interrupt.

- **RLE: Retry Limit Exceeded**

Disable retry limit exceeded interrupt.

- **TXERR**

Disable transmit buffers exhausted in mid-frame interrupt.

- **TCOMP: Transmit Complete**

Disable transmit complete interrupt.

- **ROVR: Receive Overrun**

Disable receive overrun interrupt.

- **HRESP: Hresp not OK**

Disable Hresp not OK interrupt.

- **PFR: Pause Frame Received**

Disable pause frame received interrupt.

- **PTZ: Pause Time Zero**

Disable pause time zero interrupt.

- **WOL: Wake On LAN**

Disable Wake On LAN interrupt.

37.6.11 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
–	WOL	PTZ	PFR	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame sent**

Management done interrupt masked.

- **RCOMP: Receive Complete**

Receive complete interrupt masked.

- **RXUBR: Receive Used Bit Read**

Receive used bit read interrupt masked.

- **TXUBR: Transmit Used Bit Read**

Transmit used bit read interrupt masked.

- **TUND: Ethernet Transmit Buffer Underrun**

Transmit underrun interrupt masked.

- **RLE: Retry Limit Exceeded**

Retry limit exceeded interrupt masked.

- **TXERR**

Transmit buffers exhausted in mid-frame interrupt masked.

- **TCOMP: Transmit Complete**

Transmit complete interrupt masked.

- **ROVR: Receive Overrun**

Receive overrun interrupt masked.

- **HRESP: Hresp not OK**

Hresp not OK interrupt masked.

- **PFR: Pause Frame Received**

Pause frame received interrupt masked.

- **PTZ: Pause Time Zero**

Pause time zero interrupt masked.

- **WOL: Wake On LAN**

Wake On LAN interrupt masked.

37.6.12 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.

37.6.13 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.

37.6.14 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 1082.

37.6.15 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 1082.

37.6.16 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.

37.6.17 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.

37.6.18 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.

37.6.19 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.

37.6.20 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.

37.6.21 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.

37.6.22 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.

37.6.23 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.

37.6.24 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.

37.6.25 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: Reduce MII**

When set, this bit enables the RMII operation 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.

37.6.26 Wake-on-LAN Register

Name: EMAC_WOL

Access: Read-write

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

23	22	21	20	19	18	17	16
–	–	–	–	MTI	SA1	ARP	MAG

15	14	13	12	11	10	9	8
IP							

7	6	5	4	3	2	1	0
IP							

- **IP: ARP request IP address**

Written to define the least significant 16 bits of the target IP address that is matched to generate a Wake-on-LAN event. A value of zero does not generate an event, even if this is matched by the received frame.

- **MAG: Magic packet event enable**

When set, magic packet events causes the `wol` output to be asserted.

- **ARP: ARP request event enable**

When set, ARP request events causes the `wol` output to be asserted.

- **SA1: Specific address register 1 event enable**

When set, specific address 1 events causes the `wol` output to be asserted.

- **MTI: Multicast hash event enable**

When set, multicast hash events causes the `wol` output to be asserted.

37.6.27 EMAC Statistic Registers

These registers reset to zero on a read and stick at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data. The receive statistics registers are only incremented when the receive enable bit is set in the network control register. To write to these registers, bit 7 must be set in the network control register. The statistics register block contains the following registers.

37.6.27.1 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.

37.6.27.2 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.

37.6.27.3 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.

37.6.27.4 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.

37.6.27.5 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.

37.6.27.6 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.

37.6.27.7 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.

37.6.27.8 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.

37.6.27.9 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.

37.6.27.10 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.

37.6.27.11 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.

37.6.27.12 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.

37.6.27.13 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.

37.6.27.14 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.

37.6.27.15 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.

37.6.27.16 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.

37.6.27.17 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.

37.6.27.18 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.

37.6.27.19 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.

37.6.27.20 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.

38. High Speed MultiMedia Card Interface (HSMCI)

38.1 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.

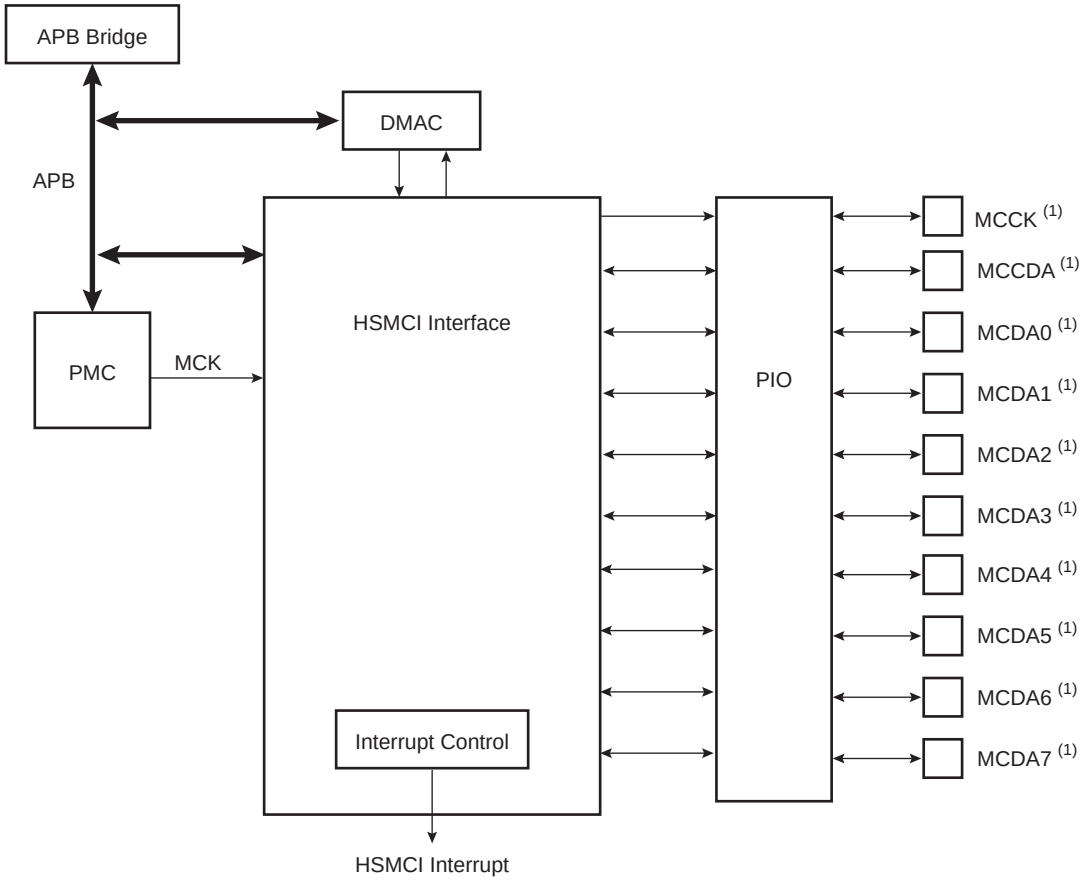
HSMCI fully supports CE-ATA Revision 1.1, built on the MMC System Specification v4.0. The module includes dedicated hardware to issue the command completion signal and capture the host command completion signal disable.

38.2 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

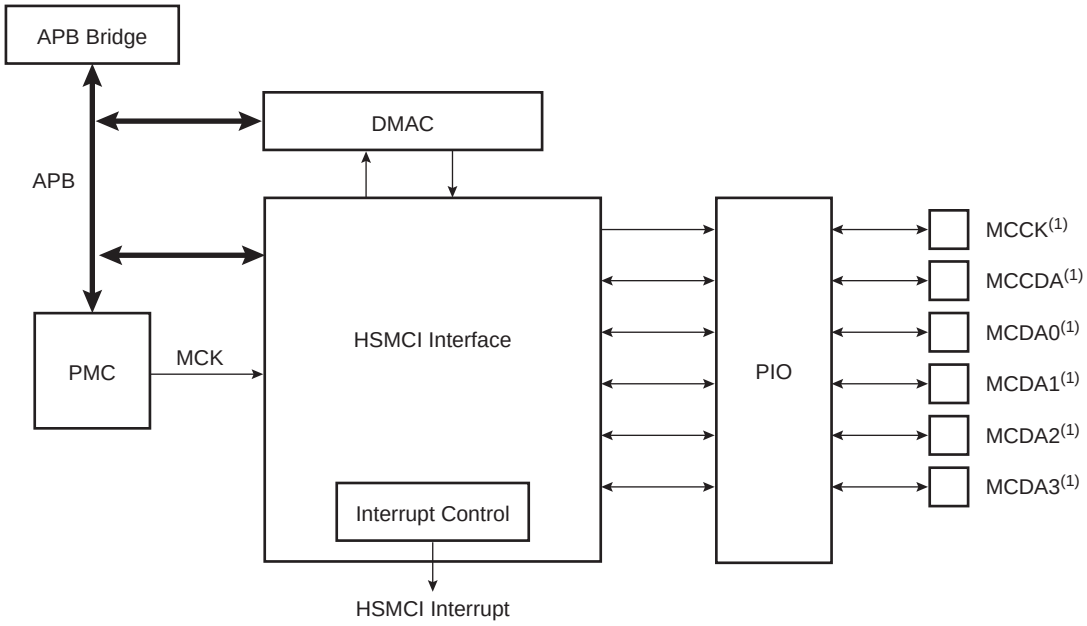
38.3 Block Diagram

Figure 38-1. Block Diagram (8-bit configuration)



Note: 1. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

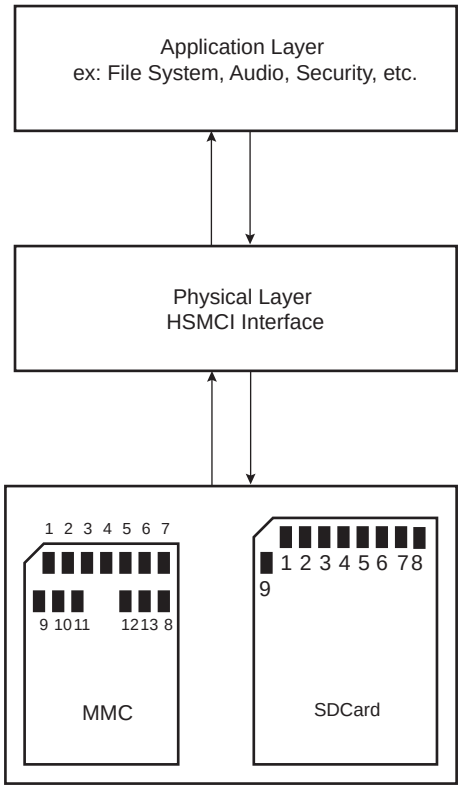
Figure 38-2. Block Diagram (4-bit configuration)



Note: 1. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

38.4 Application Block Diagram

Figure 38-3. Application Block Diagram



38.5 Pin Name List

Table 38-1. I/O Lines Description for 8-bit Configuration

Pin Name[1]	Pin Description	Type[2]	Comments
MCCDA/MCCDB	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0 - MCDA7	Data 0..7 of Slot A	I/O/PP	DAT[0..7] of an MMC DAT[0..3] of an SD Card/SDIO
MCDB0 - MCDB7	Data 0..7 of Slot B	I/O/PP	DAT[0..7] 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, MCCDB to HSMCIx_CDB, MCCDC to HSMCIx_CDC, MCCDD to HSMCIx_CDD, MCDAy to HSMCIx_DAy, MCDBy to HSMCIx_DBy, MCDCy to HSMCIx_DCy, MCDDy to HSMCIx_DDy.

2. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

Table 38-2. I/O Lines Description for 4-bit Configuration

Pin Name[1]	Pin Description	Type[2]	Comments
MCCDA/MCCDB	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0 - MCDA3	Data 0..3 of Slot A	I/O/PP	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.

38.6 Product Dependencies

38.6.1 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 38-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
HSMCI0	MCI0_CDA	PD0	A
HSMCI0	MCI0_CK	PD9	A
HSMCI0	MCI0_DA0	PD1	A
HSMCI0	MCI0_DA1	PD2	A
HSMCI0	MCI0_DA2	PD3	A
HSMCI0	MCI0_DA3	PD4	A
HSMCI0	MCI0_DA4	PD5	A
HSMCI0	MCI0_DA5	PD6	A
HSMCI0	MCI0_DA6	PD7	A
HSMCI0	MCI0_DA7	PD8	A
HSMCI1	MCI1_CDA	PB19	A
HSMCI1	MCI1_CK	PB24	A
HSMCI1	MCI1_DA0	PB20	A
HSMCI1	MCI1_DA1	PB21	A
HSMCI1	MCI1_DA2	PB22	A
HSMCI1	MCI1_DA3	PB23	A
HSMCI2	MCI2_CDA	PC10	A
HSMCI2	MCI2_CK	PC15	A
HSMCI2	MCI2_DA0	PC11	A
HSMCI2	MCI2_DA1	PC12	A
HSMCI2	MCI2_DA2	PC13	A
HSMCI2	MCI2_DA3	PC14	A

38.6.2 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.

38.6.3 Interrupt

The HSMCI interface has an interrupt line connected to the interrupt controller.

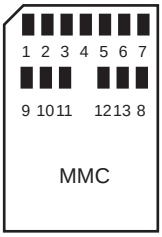
Handling the HSMCI interrupt requires programming the interrupt controller before configuring the HSMCI.

Table 38-4. Peripheral IDs

Instance	ID
HSMCI0	21
HSMCI1	22
HSMCI2	23

38.7 Bus Topology

Figure 38-4. 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.

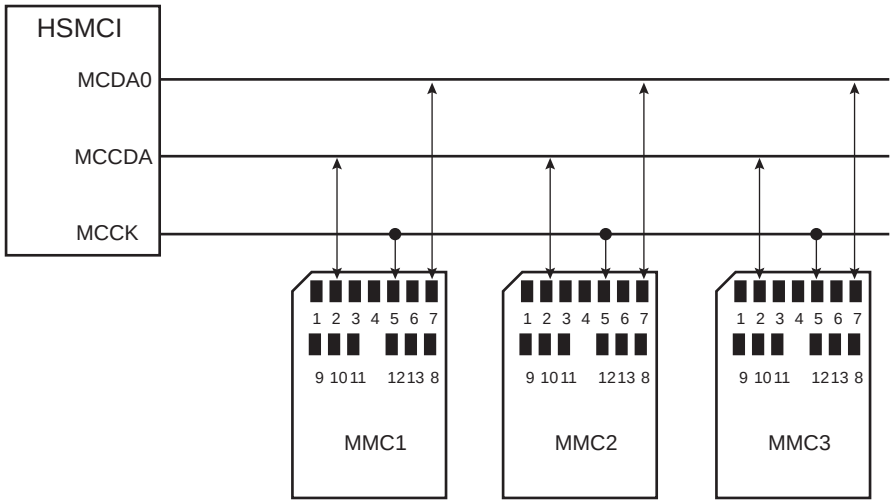
Table 38-5. Bus Topology

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.

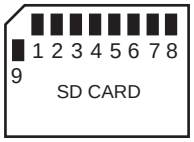
2. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCCDB to HSMCIx_CDB, MCDAy to HSMCIx_DAy, MCDBy to HSMCIx_DBy.

Figure 38-5. 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 38-6. SD Memory Card Bus Topology



The SD Memory Card bus includes the signals listed in Table 38-6.

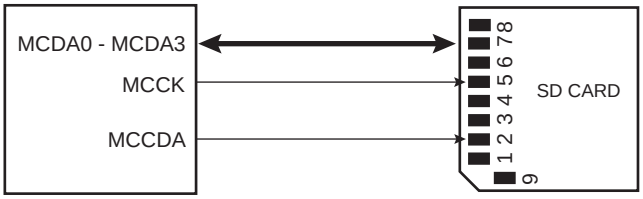
Table 38-6. SD Memory Card Bus Signals

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, MCCDB to HSMCIx_CDB, MCDAy to HSMCIx_DAy, MCDBy to HSMCIx_DBy.

Figure 38-7. 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. 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.

38.8 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 38-7 on page 1153.

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 1156.).

The HSMCI provides a set of registers to perform the entire range of High Speed MultiMedia Card operations.

38.8.1 Command - Response Operation

After reset, the HSMCI is disabled and becomes valid after setting the MCIEN bit in the HSMCI_CR.

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 (HSMCI_CMDR). The HSMCI_CMDR allows a command to be carried out.

For example, to perform an ALL_SEND_CID command:

	Host Command					N_{ID} Cycles			Response			High Impedance State		
CMD	S	T	Content	CRC	E	Z	******	Z	S	T	CID Content	Z	Z	Z

The command ALL_SEND_CID and the fields and values for the HSMCI_CMDR are described in Table 38-7 and Table 38-8.

Table 38-7. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Response	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 38-8. 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 38-8).

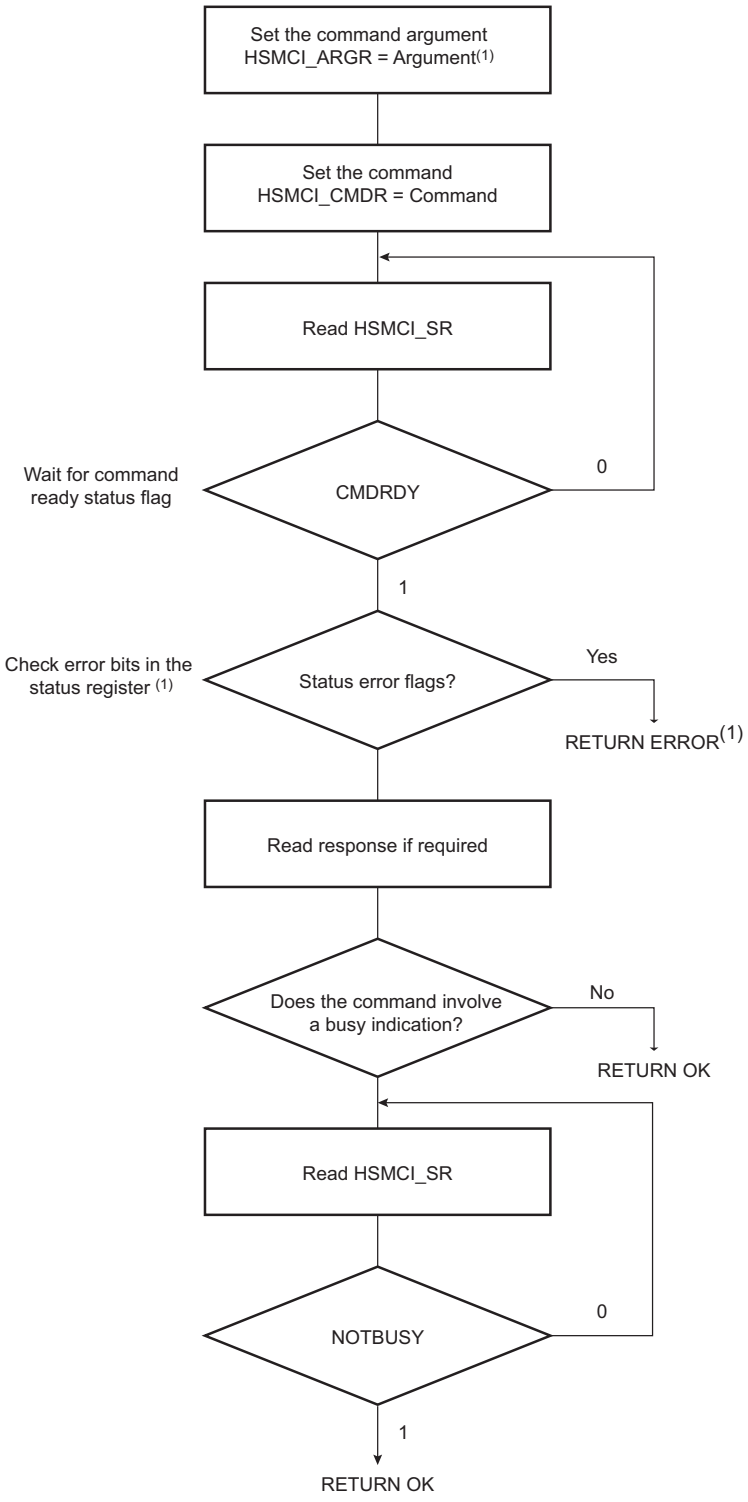
The command is sent immediately after writing the command register.

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 38-8. 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).

38.8.2 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):

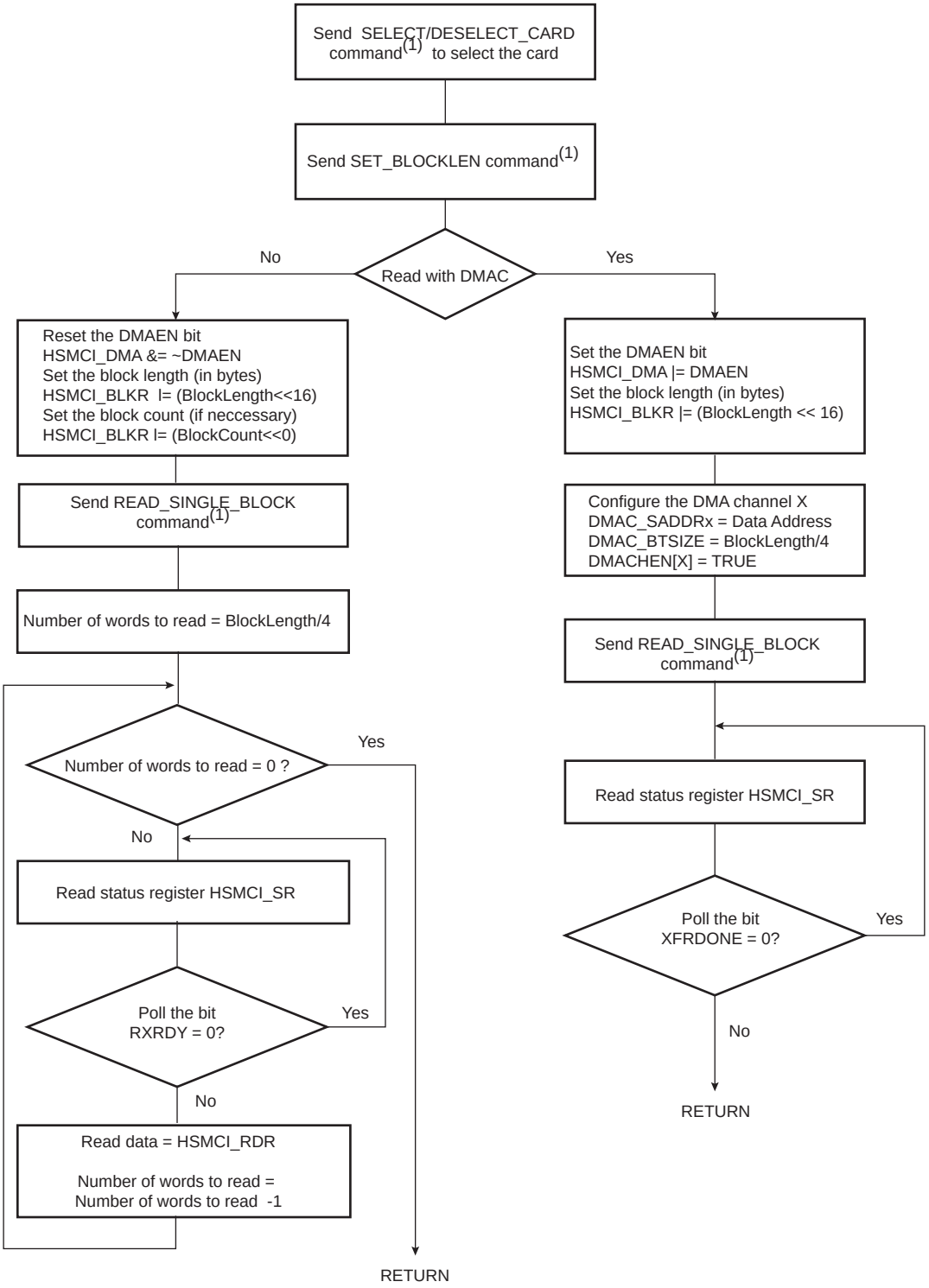
  The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

  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 HSMCI_BLKR defines the number of blocks to transfer (from 1 to 65535 blocks). Programming the value 0 in the BCNT field corresponds to an infinite block transfer.

38.8.3 Read Operation

The following flowchart (Figure 38-9) 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 38-9. Read Functional Flow Diagram



Notes: 1. It is assumed that this command has been correctly sent (see Figure 38-8).

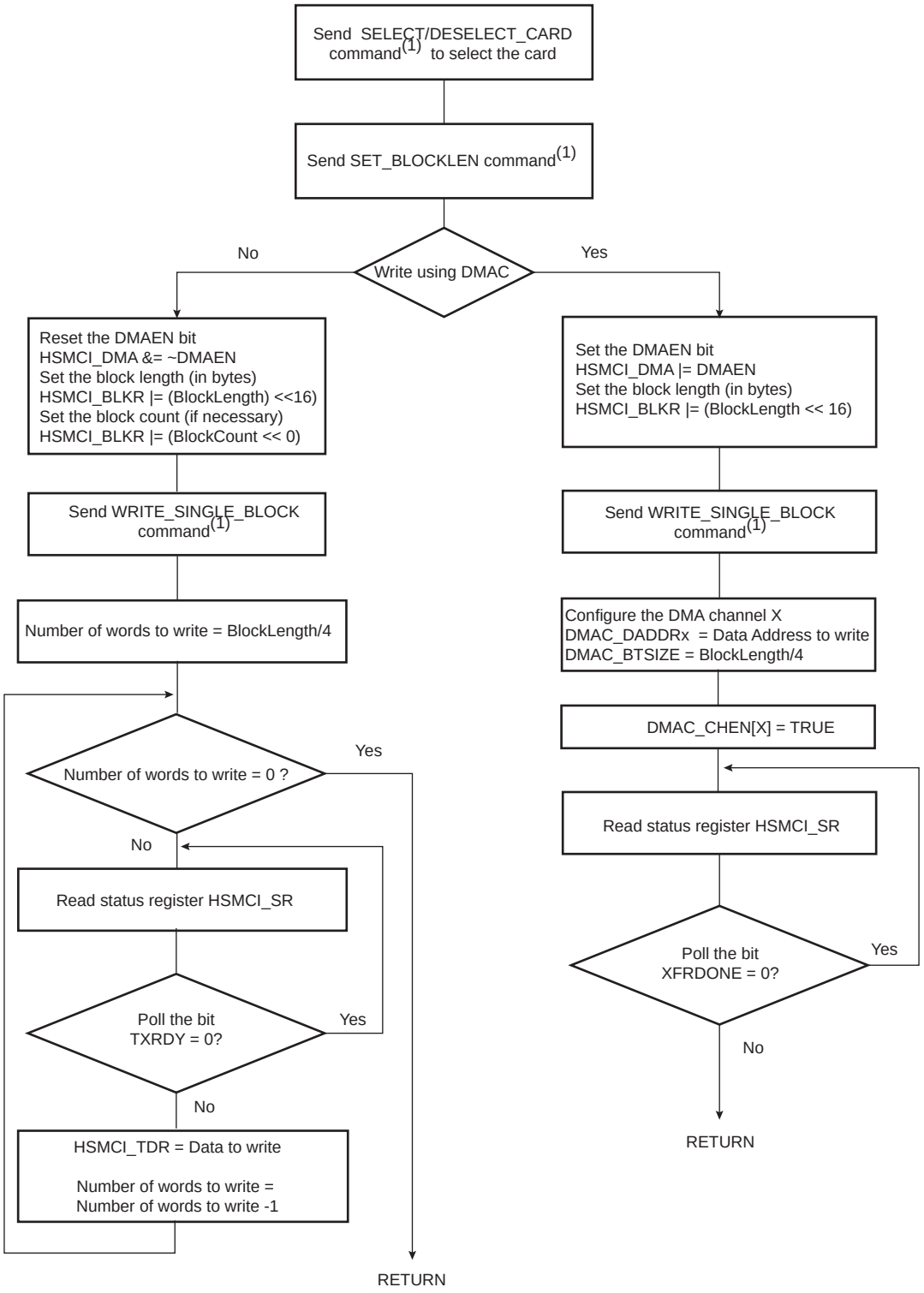
38.8.4 Write Operation

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 38-10) 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).

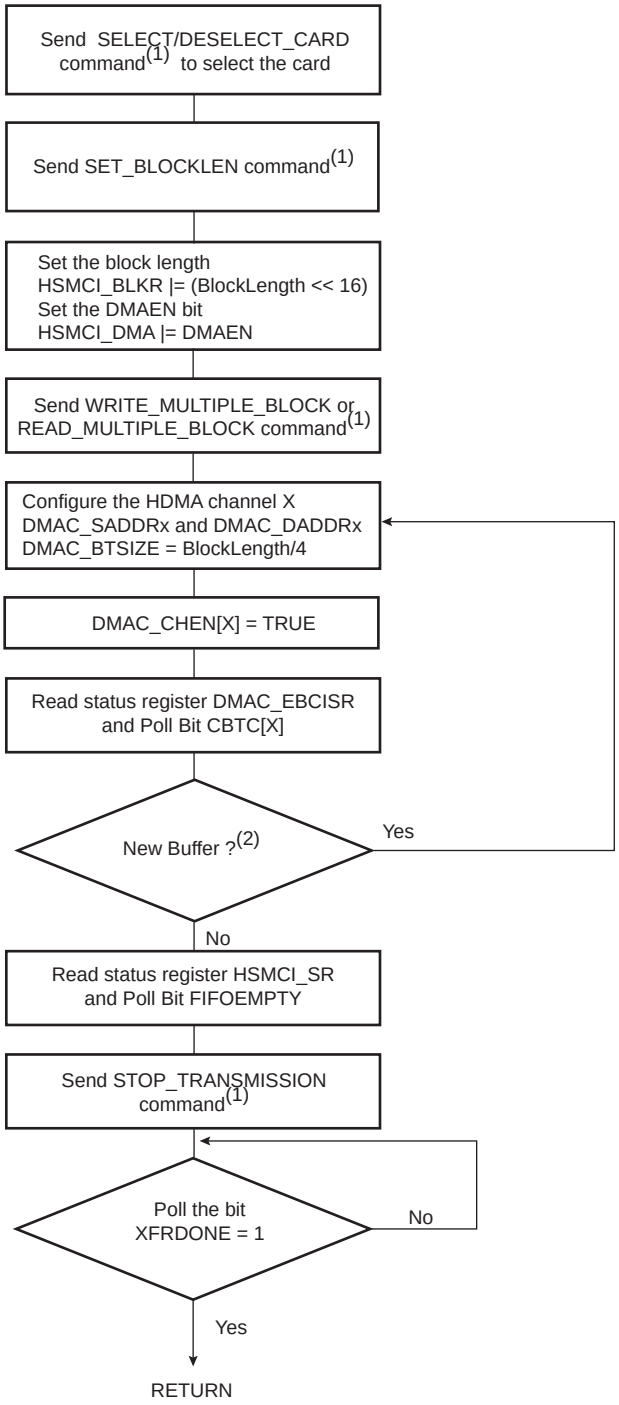
Figure 38-10.Write Functional Flow Diagram



Note: 1. It is assumed that this command has been correctly sent (see Figure 38-8).

The following flowchart (Figure 38-11) 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 38-11.Read Multiple Block and Write Multiple Block



Notes: 1. It is assumed that this command has been correctly sent (see Figure 38-8).
2. Handle errors reported in HSMCI_SR.

38.8.5 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. Configure the fields of the HSMCI_DMA register as follows:
    - 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.
    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. Configure the fields of DMAC_CTRLAx for Channel x as follows:

    –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. Configure the fields of DMAC_CTRLBx for Channel x as follows:

    –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. Configure the fields of DMAC_CFGx for Channel x as follows:

    –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.

38.8.6 READ_SINGLE_BLOCK Operation using DMA Controller

38.8.6.1 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. Configure the fields of the HSMCI_DMA register as follows:
    - 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.
    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. Configure the fields of the DMAC_CTRLAx register for Channel x as follows:

    –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. Configure the fields of the DMAC_CFGx register for Channel x as follows:

    –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. Configure the fields of the DMAC_CFGx register for Channel x as follows:

    –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.

38.8.6.2 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.
    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. Configure the fields of LLI_W.DMAC_CTRLAx as follows:

    –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. Configure the fields of LLI_W.DMAC_CTRLBx as follows:

    –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. Configure the fields of LLI_W.DMAC_CFGx for Channel x as follows:

    –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. Configure the fields of LLI_B.DMAC_CTRLAx as follows:

    –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. Configure the fields of LLI_B.DMAC_CTRLBx as follows:

–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. Configure the LLI_B.DMAC_CFGx memory location for Channel x as follows:

–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.

38.8.6.3 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.
   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. Configure the fields of DMAC_CTRLAx for Channel x as follows:

   –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. Configure the fields of DMAC_CTRLBx for Channel x as follows:

   –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. Configure the fields of DMAC_CFGx for Channel x as follows:

–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.

38.8.7 WRITE_MULTIPLE_BLOCK

38.8.7.1 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. Configure the fields of the HSMCI_DMA register as follows:
    - 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.
    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. Configure the fields of LLI(n).DMAC_CTRLAx for Channel x as follows:

    –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. Configure the fields of LLI(n).DMAC_CTRLBx for Channel x as follows:

    –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. Configure the fields of LLI(n).DMAC_CFGx for Channel x as follows:

–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 register 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.
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.

38.8.8 READ_MULTIPLE_BLOCK

38.8.8.1 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. Configure the fields of the HSMCI_DMA register as follows:
    - 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.
    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. Configure the fields of LLI_W(n).DMAC_CTRLAx as follows:

–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. Configure the fields of LLI_W(n).DMAC_CTRLBx as follows:

–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. Configure the fields of the LLI_W(n).DMAC_CFGx register for Channel x as follows:

–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.
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.

38.8.8.2 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.
    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. Configure the fields of LLI_W(n).DMAC_CTRLAx as follows:

–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. Configure the fields of LLI_W(n).DMAC_CTRLBx as follows:

–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. Configure the fields of LLI_W(n).DMAC_CFGx for Channel x as follows:

–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. Configure the fields of LLI_B(n).DMAC_CTRLAx as follows:

–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. Configure the fields of LLI_B(n).DMAC_CTRLBx as follows:

–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. Configure the LLI_B(n).DMAC_CFGx memory location for Channel x as follows:

–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.
5. Poll CBTC[x] bit in the DMAC_EBCISR.
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.

38.8.8.3 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.
    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. Configure the fields of LLI_W(n).DMAC_CTRLAx as follows:

    –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. Configure the fields of LLI_W(n).DMAC_CTRLBx as follows:

    –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. Configure the fields of LLI_W(n).DMAC_CFGx for Channel x as follows:

    –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.
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.

38.9 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).

38.9.1 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.

38.9.2 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.

38.10 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.

38.10.1 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.

38.10.2 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.

38.10.3 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.

38.10.4 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.

38.11 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.

38.11.1 Boot Procedure, Processor Mode

1. Configure the HSMCI data bus width programming SDCBUS Field in the HSMCI_SDCR. 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.
3. Issue the Boot Operation Request command by writing to the HSMCI_CMDR 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 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 with SPCMD field set to BOOTEND.

38.11.2 Boot Procedure DMA Mode

1. Configure the HSMCI data bus width by programming SDCBUS Field in the HSMCI_SDCR. 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.
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 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 with SPCMD field set to BOOTEND.

38.12 HSMCI Transfer Done Timings

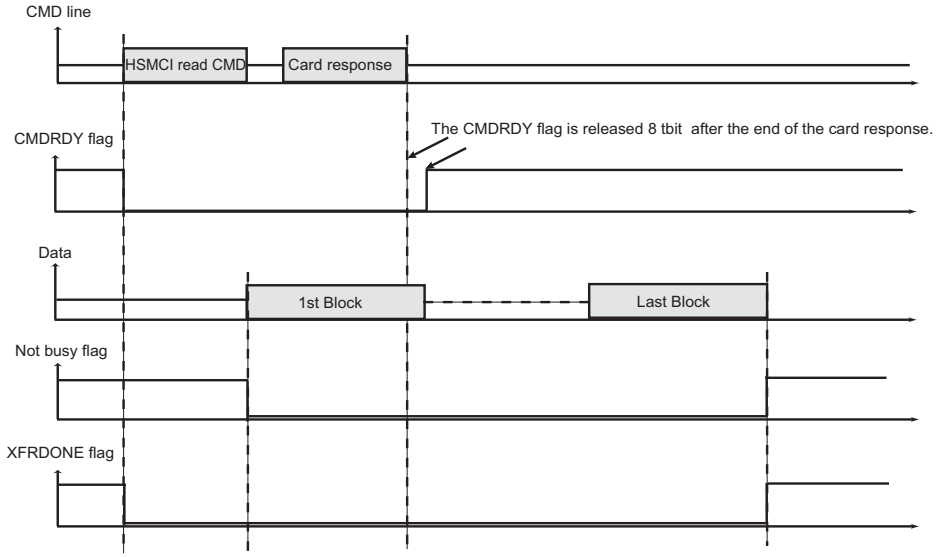
38.12.1 Definition

The XFRDONE flag in the HSMCI_SR indicates exactly when the read or write sequence is finished.

38.12.2 Read Access

During a read access, the XFRDONE flag behaves as shown in Figure 38-12.

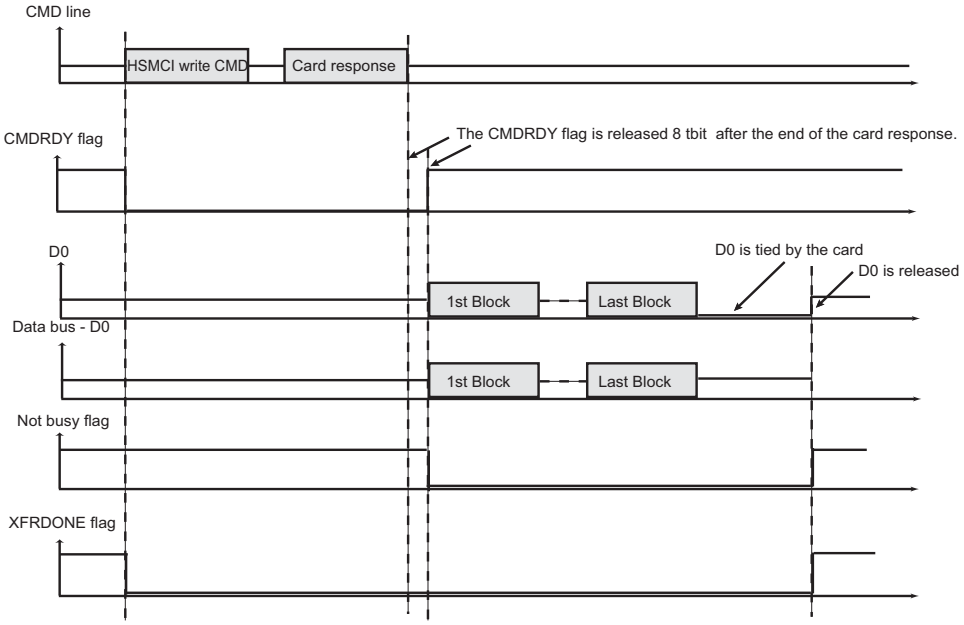
Figure 38-12.XFRDONE During a Read Access



38.12.3 Write Access

During a write access, the XFRDONE flag behaves as shown in Figure 38-13.

Figure 38-13.XFRDONE During a Write Access



38.13 Register Write Protection

To prevent any single software error from corrupting HSMCI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the "HSMCI Write Protection Mode Register" (HSMCI_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the "HSMCI Write Protection Status Register" (HSMCI_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 HSMCI_WPSR.

The following registers can be protected:

- "HSMCI Mode Register" on page 1176
- "HSMCI Data Timeout Register" on page 1178
- "HSMCI SDCard/SDIO Register" on page 1179
- "HSMCI Completion Signal Timeout Register" on page 1184
- "HSMCI DMA Configuration Register" on page 1197
- "HSMCI Configuration Register" on page 1198

38.14 High Speed MultiMedia Card Interface (HSMCI) User Interface

Table 38-9. 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-only	–
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-only	0x0
0x24	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x28	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x2C	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x30	Receive Data Register	HSMCI_RDR	Read-only	0x0
0x34	Transmit Data Register	HSMCI_TDR	Write-only	–
0x38–0x3C	Reserved	–	–	–
0x40	Status Register	HSMCI_SR	Read-only	0xC0E5
0x44	Interrupt Enable Register	HSMCI_IER	Write-only	–
0x48	Interrupt Disable Register	HSMCI_IDR	Write-only	–
0x4C	Interrupt Mask Register	HSMCI_IMR	Read-only	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.

38.14.1 HSMCI Control Register

Name: HSMCI_CR

Address: 0xF0000000 (0), 0xF8000000 (1), 0xF8004000 (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
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.

38.14.2 HSMCI Mode Register

Name: HSMCI_MR

Address: 0xF0000004 (0), 0xF8000004 (1), 0xF8004004 (2)

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 “HSMCI Write Protection Mode Register” on page 1199.

- **CLKDIV: Clock Divider**

High Speed MultiMedia Card Interface clock (MCCK or HSMCI_CK) is Master Clock (MCK) divided 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 bit is the least significant bit of the clock divider and indicates the clock divider parity.

38.14.3 HSMCI Data Timeout Register

Name: HSMCI_DTOR

Address: 0xF0000008 (0), 0xF8000008 (1), 0xF8004008 (2)

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in “HSMCI Write Protection Mode Register” on page 1199.

- **DTOCYC: Data Timeout Cycle Number**

This field determines 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.

38.14.4 HSMCI SDCard/SDIO Register

Name: HSMCI_SDCR

Address: 0xF000000C (0), 0xF800000C (1), 0xF800400C (2)

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in “HSMCI Write Protection Mode Register” on page 1199.

- **SDCSEL: SDCard/SDIO Slot**

Value	Name	Description
0	SLOTA	Slot A is selected.
1	SLOTB	Reserved
2	SLOTC	Reserved
3	SLOTD	Reserved

- **SDCBUS: SDCard/SDIO Bus Width**

Value	Name	Description
0	1	1 bit
1	–	Reserved
2	4	4 bits
3	8	8 bits

38.14.5 HSMCI Argument Register

Name: HSMCI_ARGR

Address: 0xF0000010 (0), 0xF8000010 (1), 0xF8004010 (2)

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**

38.14.6 HSMCI Command Register

Name: HSMCI_CMDR

Address: 0xF0000014 (0), 0xF8000014 (1), 0xF8004014 (2)

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. 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.

38.14.7 HSMCI Block Register

Name: HSMCI_BLKR

Address: 0xF0000018 (0), 0xF8000018 (1), 0xF8004018 (2)

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.

Bits 16 and 17 must be set to 0 if FBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

38.14.8 HSMCI Completion Signal Timeout Register

Name: HSMCI_CSTOR

Address: 0xF000001C (0), 0xF800001C (1), 0xF800401C (2)

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in “HSMCI Write Protection Mode Register” on page 1199.

- **CSTOCYC: Completion Signal Timeout Cycle Number**

This field determines 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**

This field determines 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.

38.14.9 HSMCI Response Register

Name: HSMCI_RSPR

Address: 0xF0000020 (0), 0xF8000020 (1), 0xF8004020 (2)

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.

38.14.10HSMCI Receive Data Register

Name: HSMCI_RDR

Address: 0xF0000030 (0), 0xF8000030 (1), 0xF8004030 (2)

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**

38.14.11HSMCI Transmit Data Register

Name: HSMCI_TDR

Address: 0xF0000034 (0), 0xF8000034 (1), 0xF8004034 (2)

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**

38.14.12HSMCI Status Register

Name: HSMCI_SR

Address: 0xF0000040 (0), 0xF8000040 (1), 0xF8004040 (2)

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.

- **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.

- **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.

- **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. Cleared by reading in the HSMCI_SR.

- **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.

- **DMADONE: DMA Transfer done**

0: DMA buffer transfer has not completed since the last read of the HSMCI_SR.

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.

- **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.

38.14.13HSMCI Interrupt Enable Register

Name: HSMCI_IER

Address: 0xF0000044 (0), 0xF8000044 (1), 0xF8004044 (2)

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

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **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**

38.14.14HSMCI Interrupt Disable Register

Name: HSMCI_IDR

Address: 0xF0000048 (0), 0xF8000048 (1), 0xF8004048 (2)

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

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **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**

38.14.15HSMCI Interrupt Mask Register

Name: HSMCI_IMR

Address: 0xF000004C (0), 0xF800004C (1), 0xF800404C (2)

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

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **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**

38.14.16HSMCI DMA Configuration Register

Name: HSMCI_DMA

Address: 0xF0000050 (0), 0xF8000050 (1), 0xF8004050 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	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 “HSMCI Write Protection Mode Register” on page 1199.

- **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
4	32	32 data available

- **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.

38.14.17HSMCI Configuration Register

Name: HSMCI_CFG

Address: 0xF0000054 (0), 0xF8000054 (1), 0xF8004054 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	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 “HSMCI Write Protection Mode Register” on page 1199.

- **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).

38.14.18HSMCI Write Protection Mode Register

Name: HSMCI_WPMR

Address: 0xF00000E4 (0), 0xF80000E4 (1), 0xF80040E4 (2)

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 Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

See Section 38.13 “Register Write Protection” for the list of registers which can be protected.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x4D4349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

38.14.19HSMCI Write Protection Status Register

Name: HSMCI_WPSR

Address: 0xF00000E8 (0), 0xF80000E8 (1), 0xF80040E8 (2)

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 Protect Violation has occurred since the last read of the HSMCI_WPSR.

1: A Write Protect Violation has occurred since the last read of the HSMCI_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.

38.14.20HSMCI FIFOx Memory Aperture

Name: HSMCI_FIFOx[x=0..255]

Address: 0xF0000200 (0), 0xF8000200 (1), 0xF8004200 (2)

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**

39. Serial Peripheral Interface (SPI)

39.1 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:

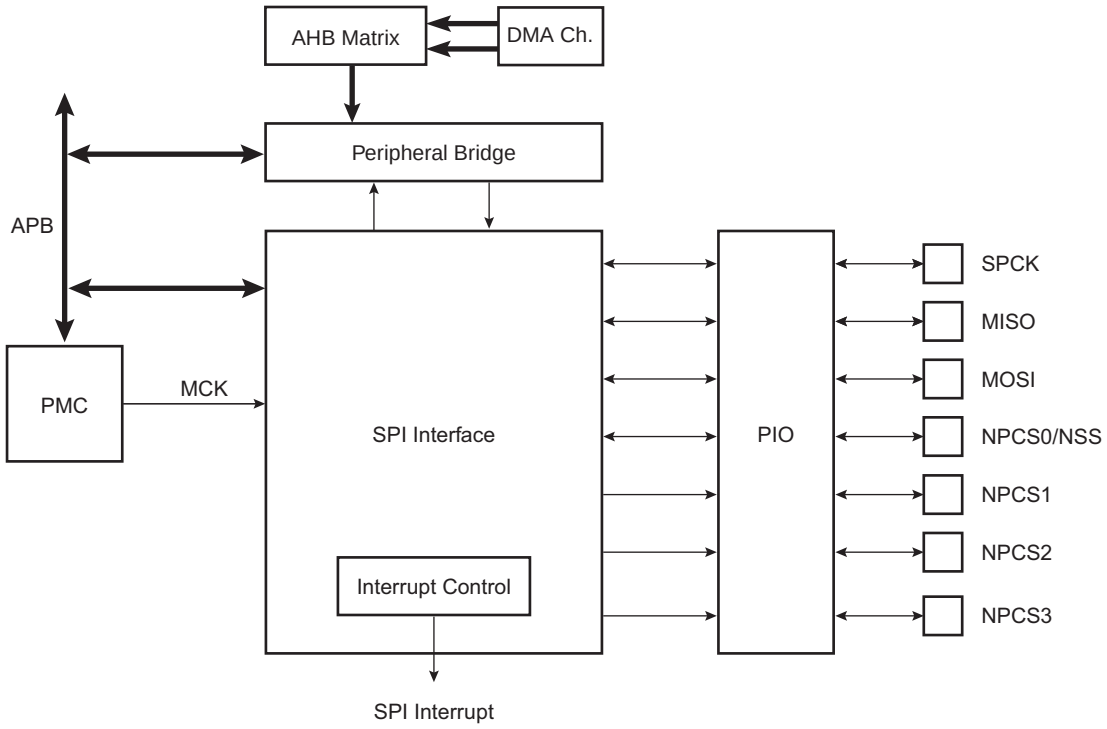
- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during 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.

39.2 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 Clock
  - 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

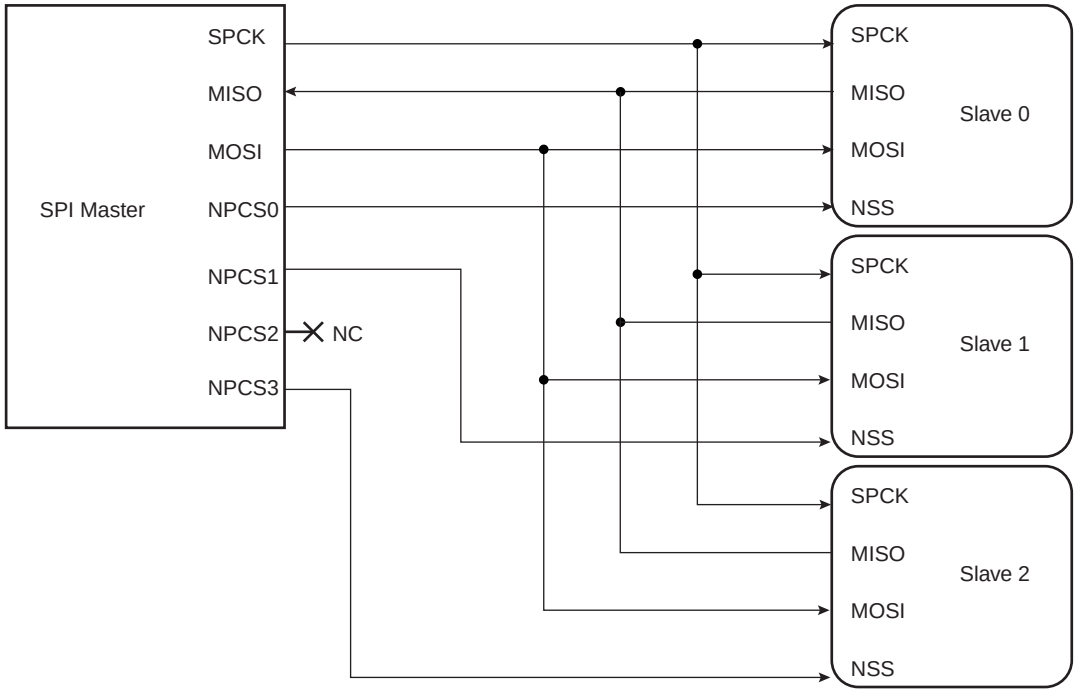
39.3 Block Diagram

Figure 39-1. Block Diagram



39.4 Application Block Diagram

Figure 39-2. Application Block Diagram: Single Master/Multiple Slave Implementation



39.5 Signal Description

Table 39-1. 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

39.6 Product Dependencies

39.6.1 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 39-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_MISO	PD10	A
SPI0	SPI0_MOSI	PD11	A
SPI0	SPI0_NPCS0	PD13	A

Table 39-2. I/O Lines (Continued)

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_NPCS1	PD14	B
SPI0	SPI0_NPCS2	PD15	B
SPI0	SPI0_NPCS3	PD16	B
SPI0	SPI0_SPCK	PD12	A
SPI1	SPI1_MISO	PC22	A
SPI1	SPI1_MOSI	PC23	A
SPI1	SPI1_NPCS0	PC25	A
SPI1	SPI1_NPCS1	PC26	A
SPI1	SPI1_NPCS2	PC27	A
SPI1	SPI1_NPCS3	PC28	A
SPI1	SPI1_SPCK	PC24	A

39.6.2 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.

39.6.3 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 39-3. Peripheral IDs

Instance	ID
SPI0	24
SPI1	25

39.6.4 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.

39.7 Functional Description

39.7.1 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.

39.7.2 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.

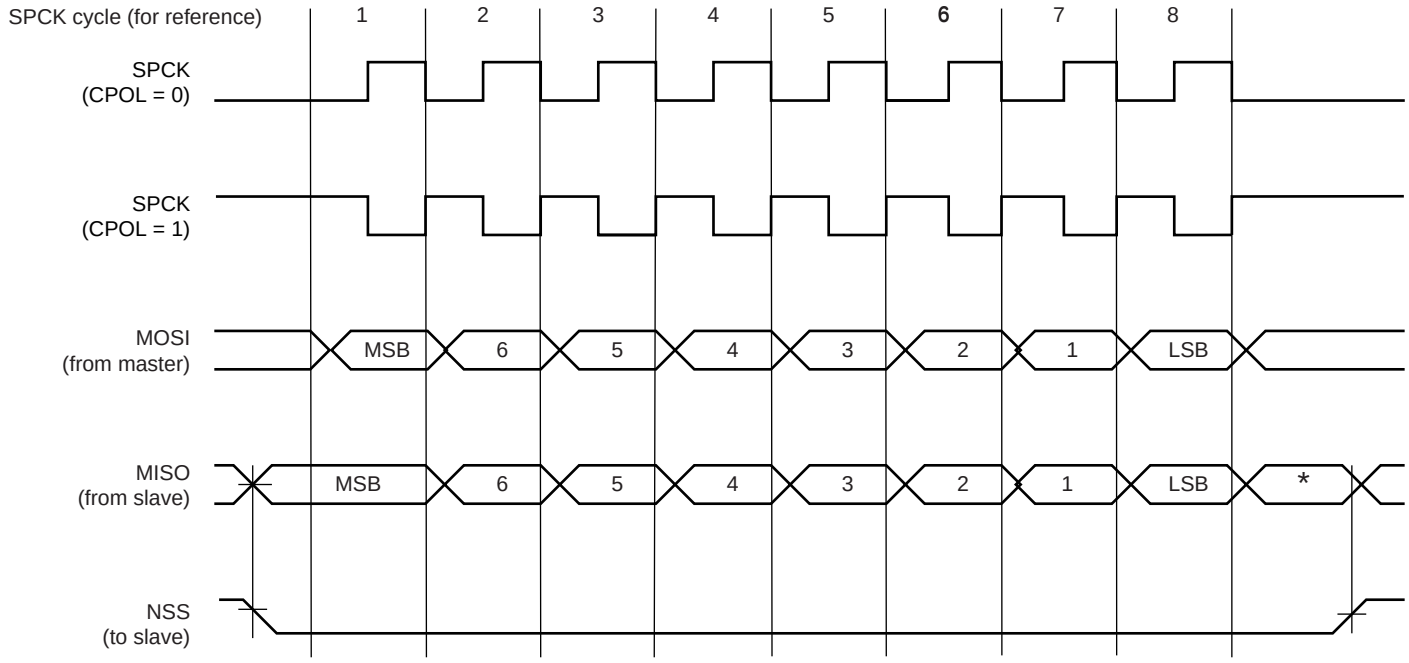
Table 39-4 shows the four modes and corresponding parameter settings.

Table 39-4. 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

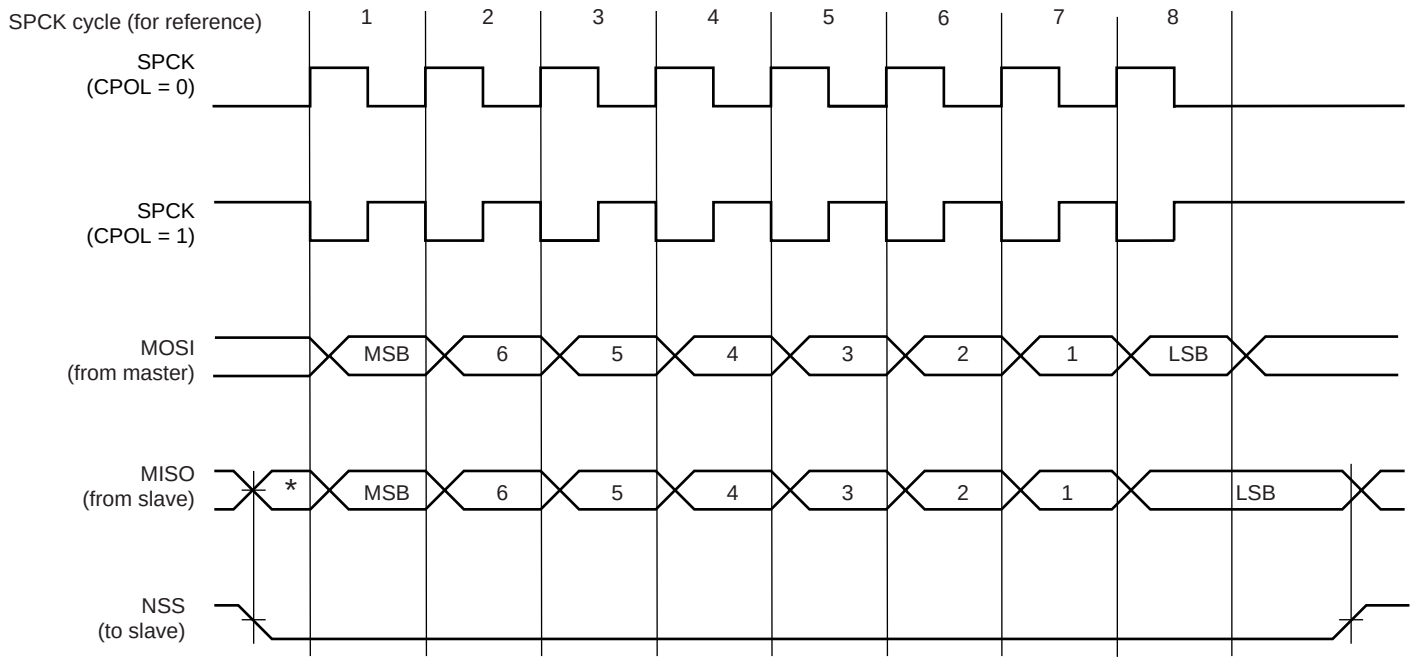
Figure 39-3 and Figure 39-4 show examples of data transfers.

Figure 39-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Figure 39-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

39.7.3 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.

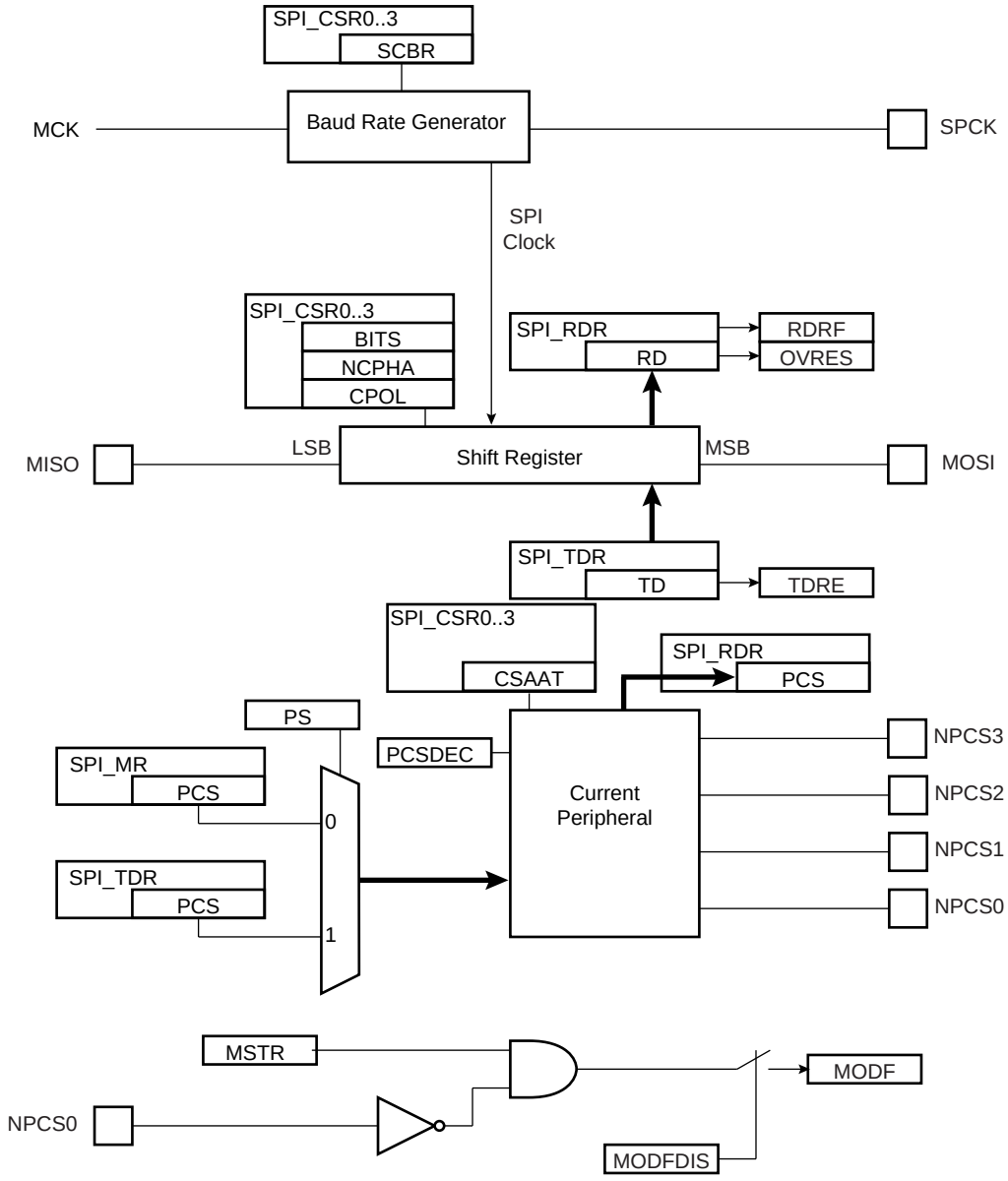
The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

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 39-5, shows a block diagram of the SPI when operating in Master Mode. Figure 39-6 on page 1209 shows a flow chart describing how transfers are handled.

39.7.3.1 Master Mode Block Diagram

Figure 39-5. Master Mode Block Diagram



39.7.3.2 Master Mode Flow Diagram

Figure 39-6. Master Mode Flow Diagram

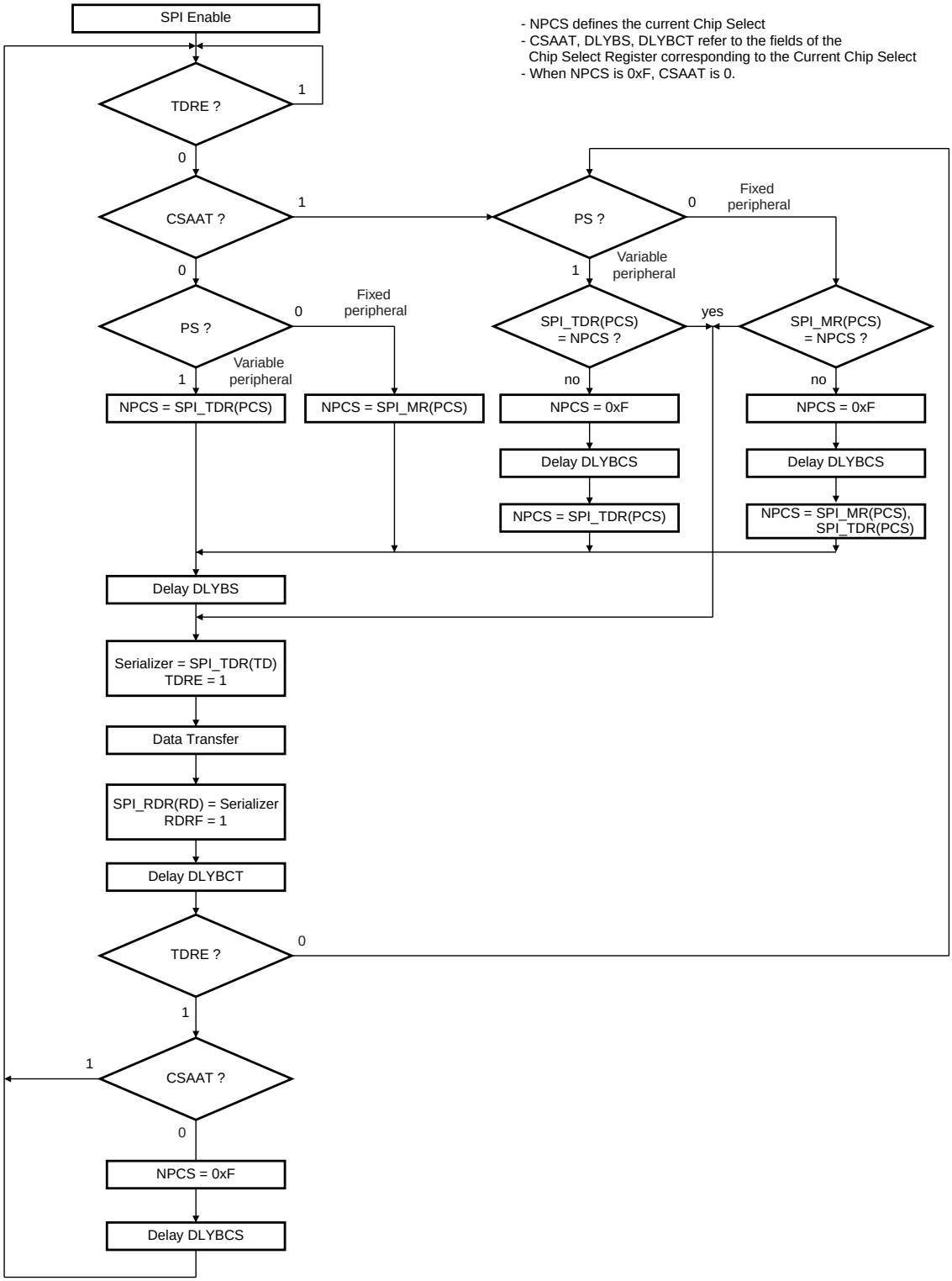
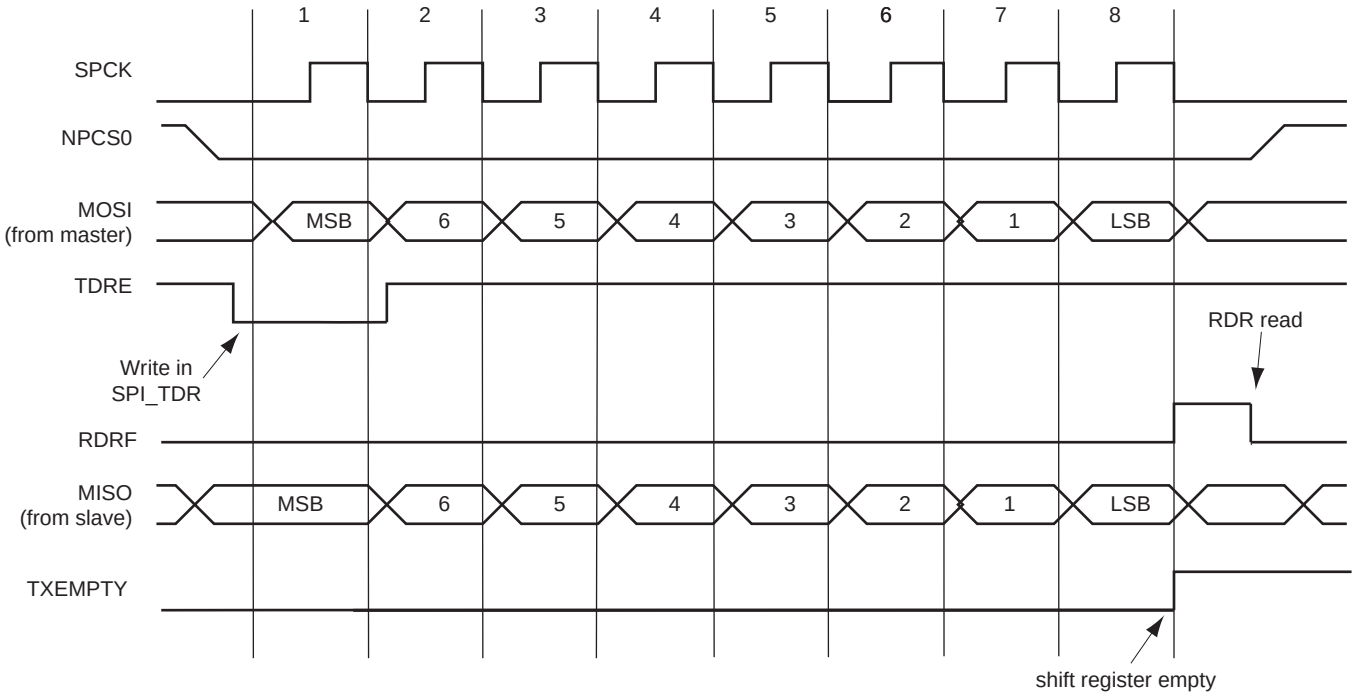


Figure 39-7 shows Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags behavior within the SPI_SR (Status Register) during an 8-bit data transfer in fixed mode and no Peripheral Data Controller involved.

Figure 39-7. Status Register Flags Behavior



39.7.3.3 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.

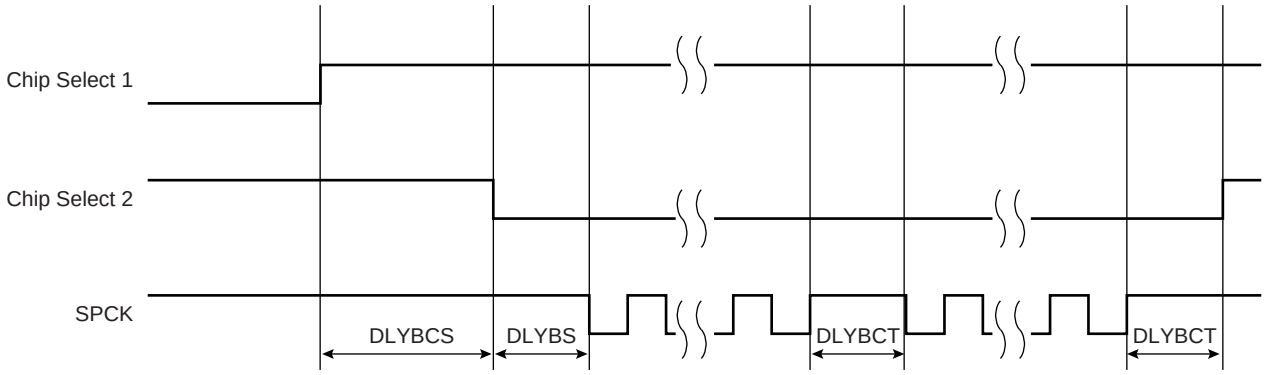
39.7.3.4 Transfer Delays

Figure 39-8 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 39-8. Programmable Delays



39.7.3.5 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 39.8.4 (SPI Transmit Data Register) and LASTXFER bit at 0 or 1 depending on CSAAT bit.

Note: 1. Optional.

CSAAT, LASTXFER and CSNAAT bits are discussed in Section 39.7.3.9 ”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.

39.7.3.6 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.

39.7.3.7 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.

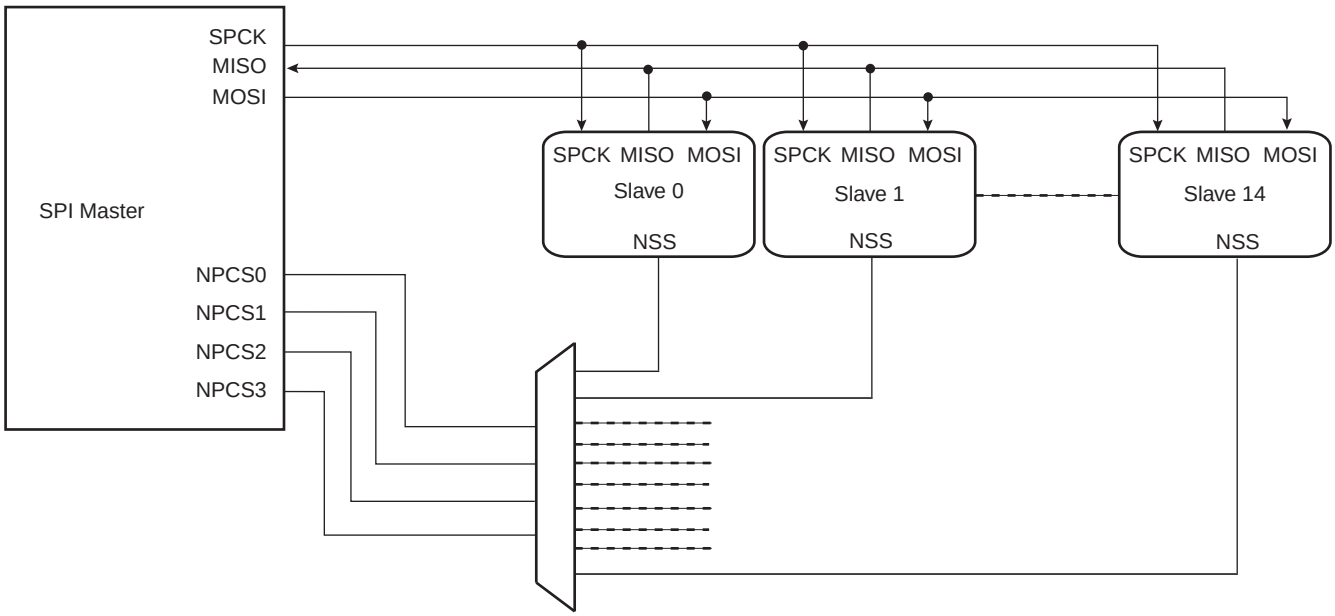
When operating with decoding, the SPI directly outputs the value defined by the PCS field on NPCS lines of either the Mode Register or the Transmit Data Register (depending on PS).

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 39-9 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 39-9. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation



39.7.3.8 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.

39.7.3.9 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.

If chip select must be deasserted between each data when communicating with a slave device such as an SPI ADC for example, one can workaround this by using another “ghost” chip select line. In this case, the variable mode must be used. In this mode, the assertion of the corresponding chip select (CS) depends on the PCS field within SPI_TDR.

- Thereafter, an example of such implementation:
  NPCS0[(1)]: Actual SPI Slave Device
- NPCS1[(1)]: Ghost SPI Slave Device

Transmit Buffer must be initialized in the following way according to the SPI_TDR register (See Section 39.8.4 “SPI Transmit Data Register” on page 1222) to send data equal to 0xAB with NPCS0 and to send data equal to 0xCD on NPCS1.

[BUFFER_ADDR[(2)]] = 0x0000 00AB <------ Transfer on NPCS0
[BUFFER_ADDR + 0x4] = 0x0001 000CD <------ Dummy transfer on NPCS1
[BUFFER_ADDR + 0x8] = 0x0000 000AB <------ Transfer on NPCS0
[BUFFER_ADDR + 0xC] = 0x0001 000CD <------ Dummy transfer on NPCS1

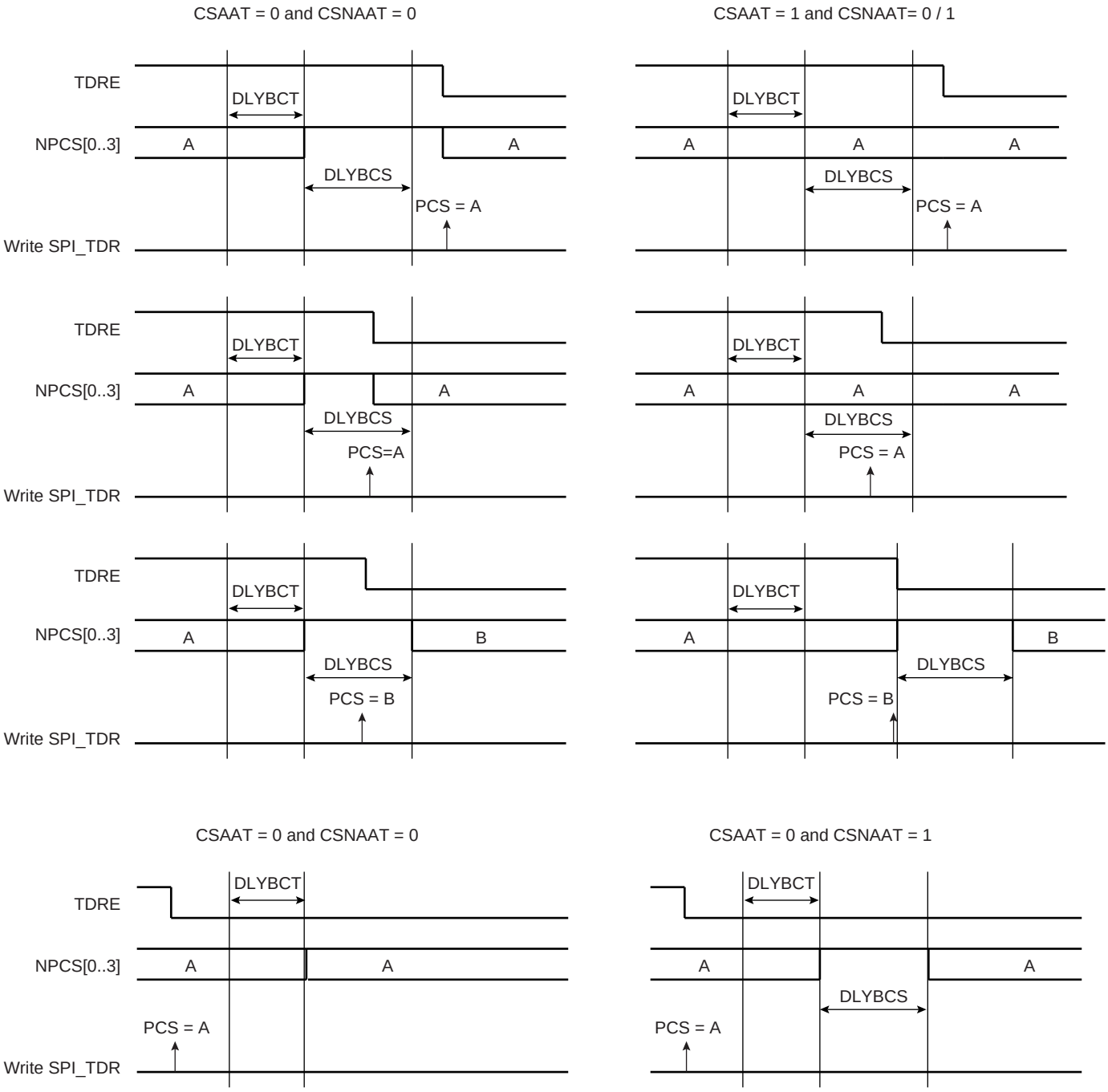
Please note that in this case the size of the buffer will be twice the actual size needed.

Notes: 1. Can be any other chip select line.
2. Transmit Buffer address.

When the CSAAT bit is set at 0, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a Chip Select, the flag TDRE rises as soon as the content of the SPI_TDR is transferred into the internal shifter. When this flag is detected the SPI_TDR can be reloaded. If this reload 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. This might lead to difficulties for interfacing with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSNAAT bit (Chip Select Not Active After Transfer) at 1. This allows to de-assert systematically the chip select lines during a time DLYBCS. (The value of the CSNAAT bit is taken into account only if the CSAAT bit is set at 0 for the same Chip Select).

Figure 39-10 shows different peripheral deselection cases and the effect of the CSAAT and CSNAAT bits.

Figure 39-10.Peripheral Deselection



39.7.3.10 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).

39.7.4 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 39.8.9 “SPI Chip Select Register” on page 1228.)

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.

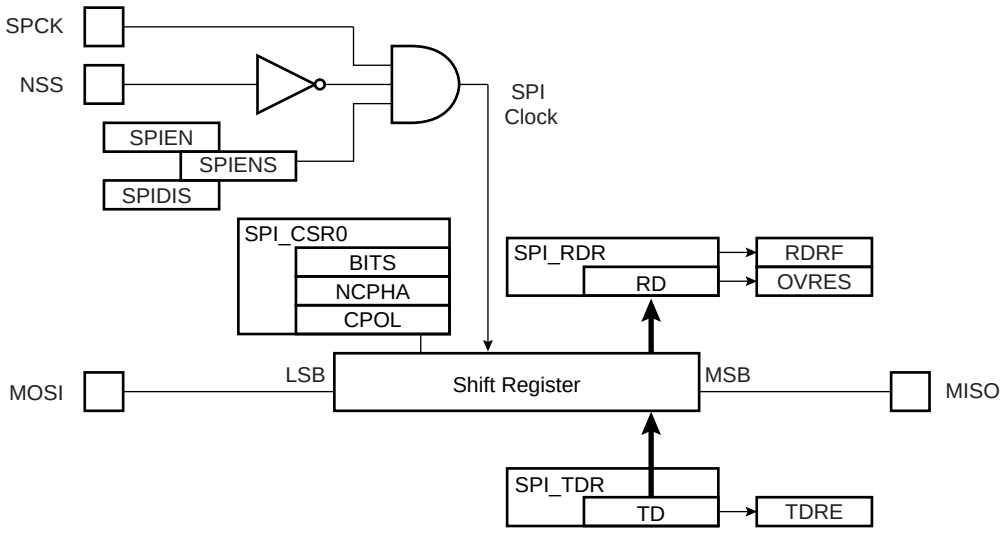
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets 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 SPI_TDR is retransmitted. In this case the Underrun Error Status Flag (UNDES) is set in the SPI_SR.

Figure 39-11 shows a block diagram of the SPI when operating in Slave Mode.

Figure 39-11.Slave Mode Functional Block Diagram



39.7.5 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 39.8.2 ”SPI Mode Register”

Section 39.8.9 ”SPI Chip Select Register”

39.8 Serial Peripheral Interface (SPI) User Interface

Table 39-5. 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
0x40 - 0xE0	Reserved	–	–	–
0xE4	Write Protection Control Register	SPI_WPMR	Read-write	0x0
0xE8	Write Protection Status Register	SPI_WPSR	Read-only	0x0
0x00EC - 0x00F8	Reserved	–	–	–
0x00FC	Reserved	–	–	–
0x100 - 0x124	Reserved for PDC Registers	–	–	–

39.8.1 SPI Control Register

Name: SPI_CR

Address: 0xF0004000 (0), 0xF8008000 (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 39.7.3.5 ”Peripheral Selection” for more details.

39.8.2 SPI Mode Register

Name: SPI_MR

Address: 0xF0004004 (0), 0xF8008004 (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:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

39.8.3 SPI Receive Data Register

Name: SPI_RDR

Address: 0xF0004008 (0), 0xF8008008 (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.

39.8.4 SPI Transmit Data Register

Name: SPI_TDR

Address: 0xF000400C (0), 0xF800800C (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).

39.8.5 SPI Status Register

Name: SPI_SR

Address: 0xF0004010 (0), 0xF8008010 (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
–	–	–	–	–	UNDES	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.

- **UNDES: Underrun Error Status (Slave Mode Only)**

0 = No underrun has been detected since the last read of SPI_SR.

1 = A transfer begins whereas no data has been loaded in the Transmit Data Register.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

39.8.6 SPI Interrupt Enable Register

Name: SPI_IER

Address: 0xF0004014 (0), 0xF8008014 (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
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	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**
- **UNDES: Underrun Error Interrupt Enable**

39.8.7 SPI Interrupt Disable Register

Name: SPI_IDR

Address: 0xF0004018 (0), 0xF8008018 (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
–	–	–	–	–	UNDES	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**
- **UNDES: Underrun Error Interrupt Disable**

39.8.8 SPI Interrupt Mask Register

Name: SPI_IMR

Address: 0xF000401C (0), 0xF800801C (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
–	–	–	–	–	UNDES	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**
- **UNDES: Underrun Error Interrupt Mask**

39.8.9 SPI Chip Select Register

Name: SPI_CSRx[x=0..3]

Address: 0xF0004030 (0), 0xF8008030 (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	CSNAAT	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.

- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**

0 = The Peripheral Chip Select does not rise between two transfers if the SPI_TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.

1 = The Peripheral Chip Select rises systematically after each transfer performed on the same slave. It remains active after the end of transfer for a minimal duration of:

  – $\frac{DLYBCT}{MCK}$ (if DLYBCT field is different from 0)

  – $\frac{DLYBCT + 1}{MCK}$ (if DLYBCT field equals 0)

- **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 39.8.9 “SPI Chip Select Register” on page 1228.)

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
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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

39.8.10 SPI Write Protection Mode Register

Name: SPI_WPMR

Address: 0xF00040E4 (0), 0xF80080E4 (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 Protect Enable**

0: Disables the Write Protect if WPKEY corresponds to 0x535049 (“SPI” in ASCII).

1: Enables the Write Protect if WPKEY corresponds to 0x535049 (“SPI” in ASCII).

Protects the registers:

- Section 39.8.2 ”SPI Mode Register”
- Section 39.8.9 ”SPI Chip Select Register”

- **WPKEY: Write Protect Key**

Value	Name	Description
0x535049	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

39.8.11 SPI Write Protection Status Register

Name: SPI_WPSR

Address: 0xF00040E8 (0), 0xF80080E8 (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)

40. Two-wire Interface (TWI)

40.1 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.

Table 40-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I^2C compatible device.

Table 40-1. 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

40.2 Embedded Characteristics

- 3 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 Kbit/s
- General Call Supported in Slave mode
- SMBUS Quick Command Supported in Master Mode
- Connection to DMA Controller (DMA) Channel Capabilities Optimizes Data Transfers

Note: 1. See Table 40-1 for details on compatibility with I²C Standard.

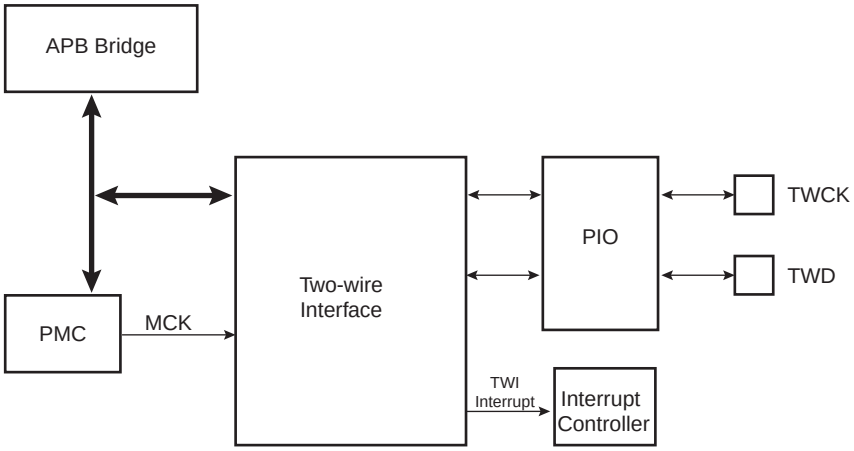
40.3 List of Abbreviations

Table 40-2. 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

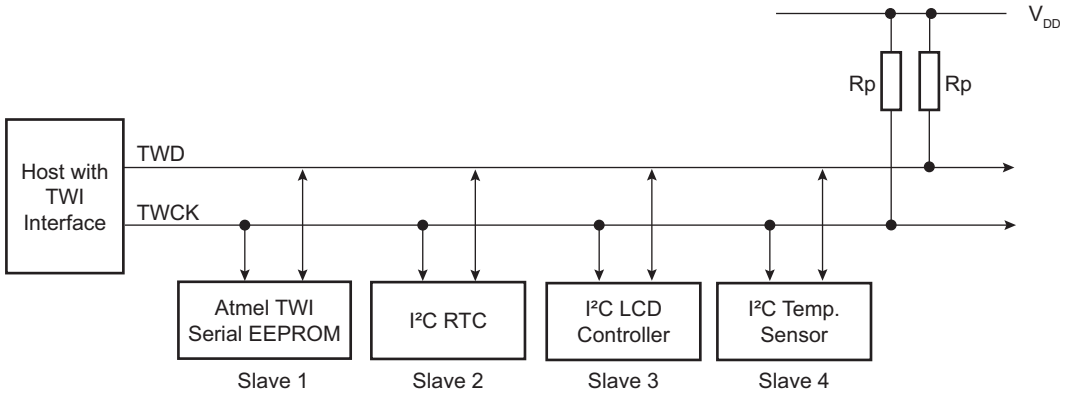
40.4 Block Diagram

Figure 40-1. Block Diagram



40.5 Application Block Diagram

Figure 40-2. Application Block Diagram



Rp: Pull-up value as given by the I²C Standard

40.5.1 I/O Lines Description

Table 40-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

40.6 Product Dependencies

40.6.1 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 40-2). 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 40-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TWI0	TWCK0	PA31	A
TWI0	TWD0	PA30	A
TWI1	TWCK1	PC27	B
TWI1	TWD1	PC26	B
TWI2	TWCK2	PA19	B
TWI2	TWD2	PA18	B

40.6.2 Power Management

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.

40.6.3 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 40-5. Peripheral IDs

Instance	ID
TWI0	18
TWI1	19
TWI2	20

40.7 Functional Description

40.7.1 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 40-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 40-3).

- 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 40-3. START and STOP Conditions

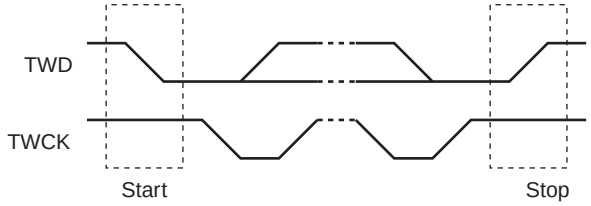
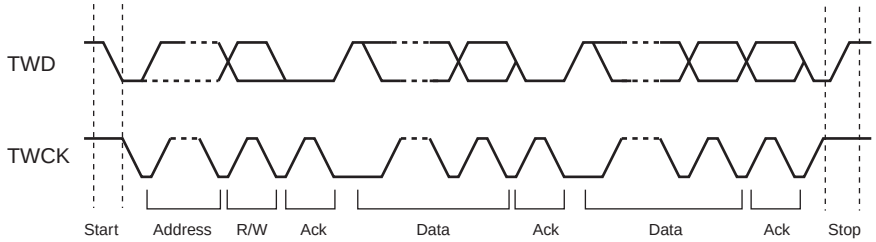


Figure 40-4. Transfer Format



40.7.2 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 sections.

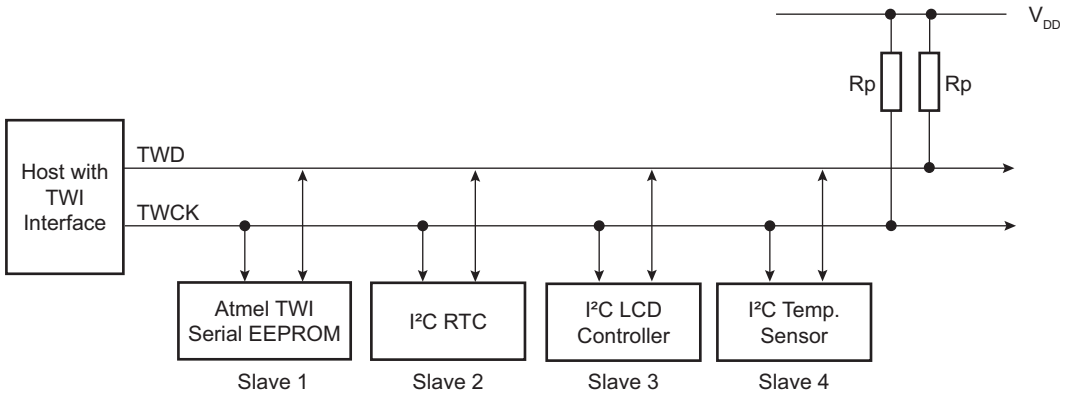
40.8 Master Mode

40.8.1 Definition

The Master is the device that starts a transfer, generates a clock and stops it.

40.8.2 Application Block Diagram

Figure 40-5. Master Mode Typical Application Block Diagram



Rp: Pull-up value as given by the I²C Standard

40.8.3 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.

40.8.4 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 40-6, Figure 40-7, and Figure 40-8.

Figure 40-6. Master Write with One Data Byte

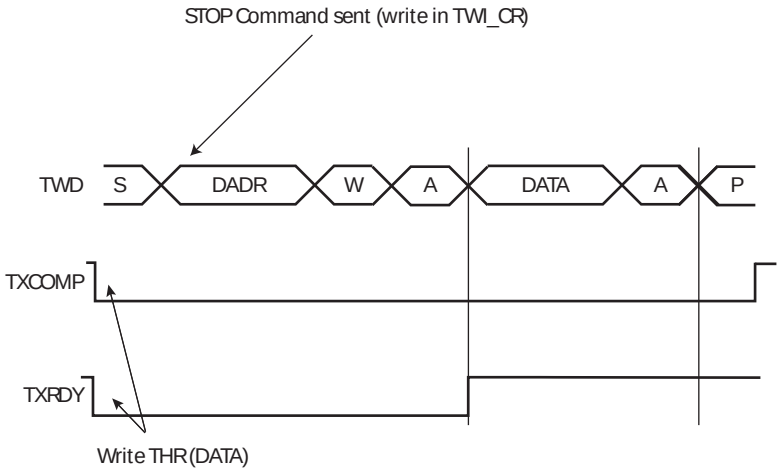


Figure 40-7. Master Write with Multiple Data Bytes

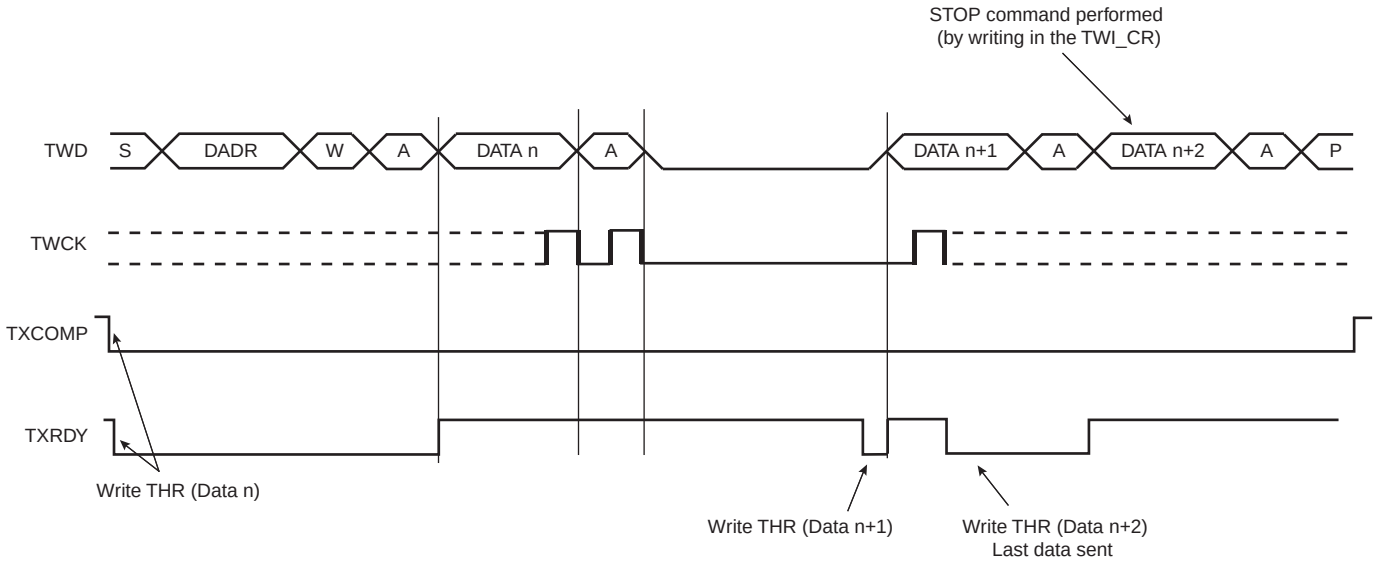
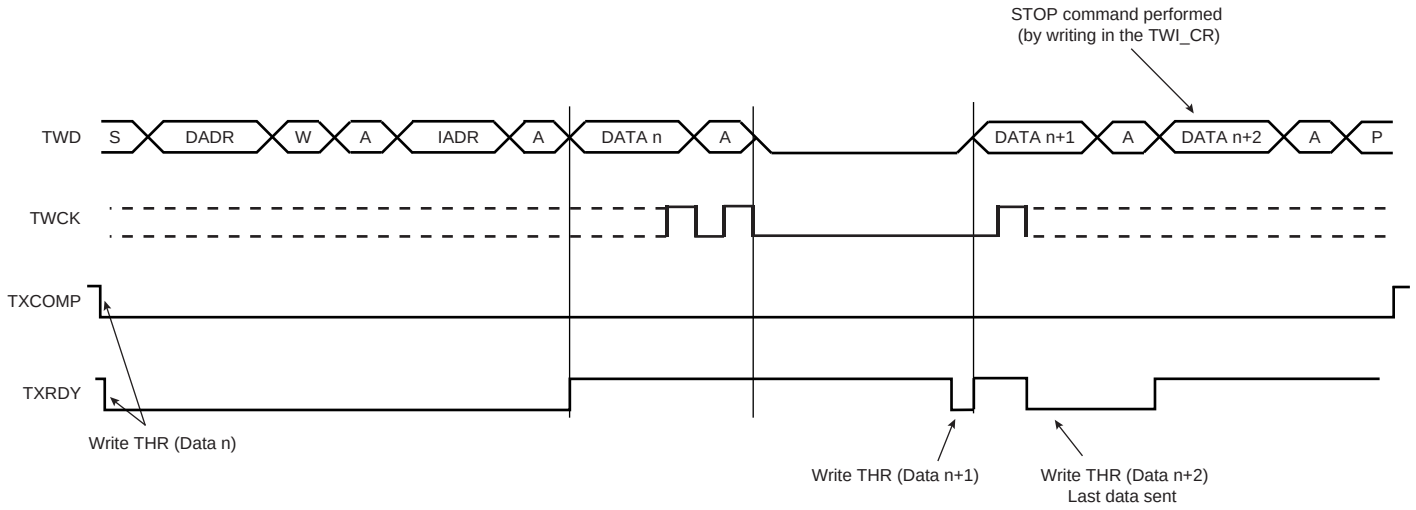


Figure 40-8. Master Write with One Byte Internal Address and Multiple Data Bytes



40.8.5 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 40-9. 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 40-9. 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 40-10. For Internal Address usage see Section 40.8.6.

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 is read. Once the TWI_RHR is read, the master will stop stretching the Serial Clock Line and end the data reception. See Figure 40-11.

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 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 40-9. Master Read with One Data Byte

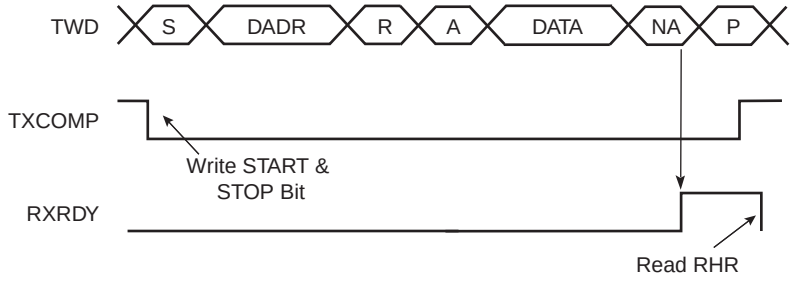


Figure 40-10.Master Read with Multiple Data Bytes

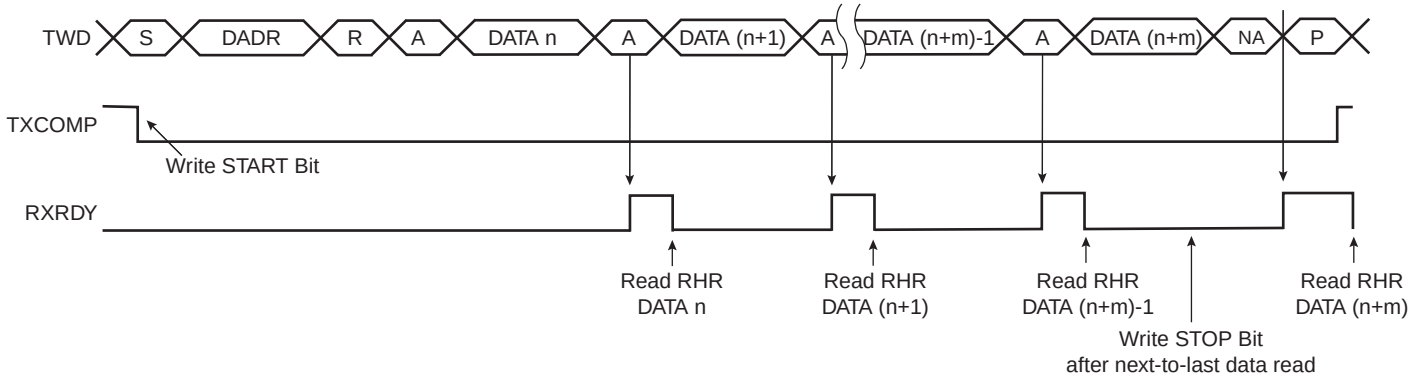
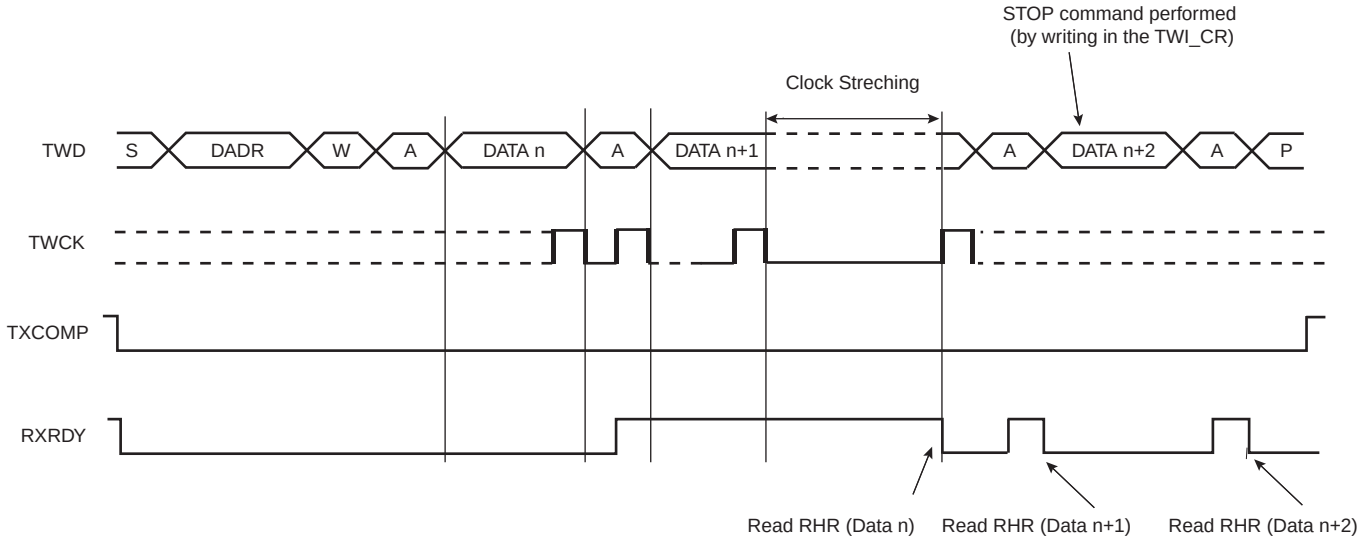


Figure 40-11.Master Read Clock Stretching with Multiple Data Bytes



40.8.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

40.8.6.1 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 40-13. See Figure 40-12 and Figure 40-14 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.

Table 40-6 shows the abbreviations used in Figure 40-12 and Figure 40-13.

Table 40-6. Abbreviations

Abbreviation	Definition
S	Start
Sr	Repeated Start
P	Stop
W	Write
R	Read
A	Acknowledge
NA	Not Acknowledge
DADR	Device Address
IADR	Internal Address

Figure 40-12.Master Write with One, Two or Three Bytes Internal Address and One Data Byte

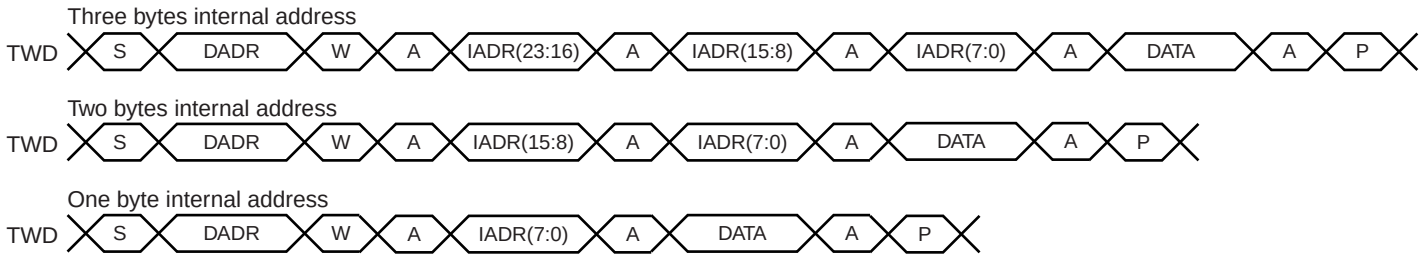
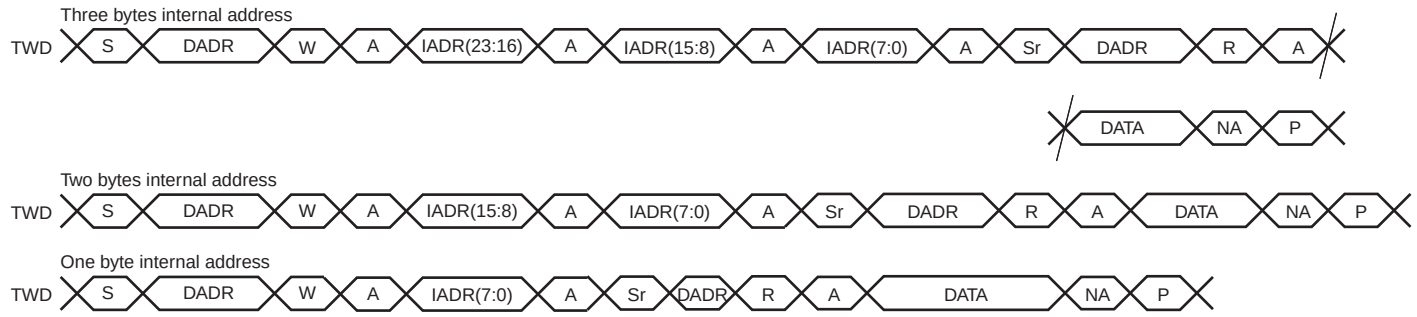


Figure 40-13.Master Read with One, Two or Three Bytes Internal Address and One Data Byte



40.8.6.2 10-bit Slave Addressing

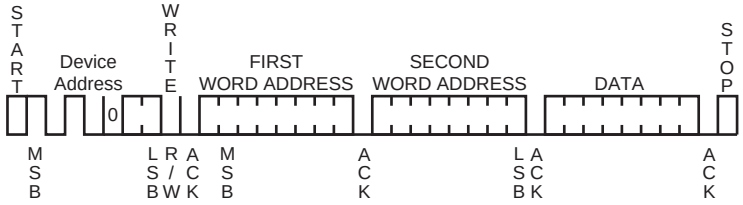
For a slave address higher than 7 bits, the user must configure the address size **(**IADRSZ) and set the other slave address bits in the internal address register (TWI_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

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 40-14 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 40-14. Internal Address Usage



40.8.7 Using the DMA Controller

The use of the DMA significantly reduces the CPU load.

To assure correct implementation, respect the following programming sequence.

40.8.7.1 Data Transmit with the DMA

1. Initialize the DMA (channels, memory pointers, size -1, etc.);
2. Configure the master mode (DADR, CKDIV, etc.) or slave mode.
3. Enable the DMA.
4. Wait for the DMA BTC flag.
5. Disable the DMA.
6. Wait for the TXRDY flag in TWI_SR.
7. Set the STOP command in TWI_CR.
8. Write the last character in TWI_THR.
9. (Optional) Wait for the TXCOMP flag in TWI_SR before disabling the peripheral clock if required.

40.8.7.2 Data Receive with the DMA

The DMA transfer size must be defined with the buffer size minus 2. The two remaining characters must be managed without DMA to ensure that the exact number of bytes are received whatever the system bus latency conditions encountered during the end of buffer transfer period.

In slave mode, the number of characters to receive must be known in order to configure the DMA.

1. Initialize the DMA (channels, memory pointers, size -2, etc.);
2. Configure the master mode (DADR, CKDIV, etc.) or slave mode.
3. Enable the DMA.
4. (Master Only) Write the START bit in the TWI_CR to start the transfer.
5. Wait for the DMA BTC flag.
6. Disable the DMA.
7. Wait for the RXRDY flag in the TWI_SR.
8. Set the STOP command in TWI_CR.

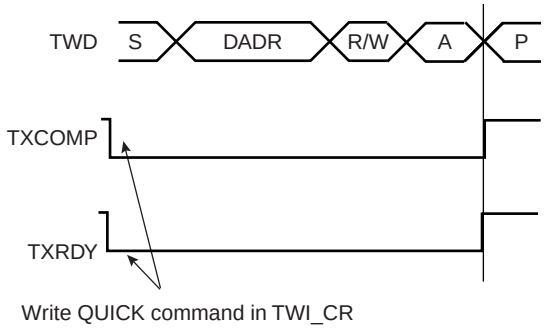
9. Read the penultimate character in TWI_RHR.
10. Wait for the RXRDY flag in the TWI_SR.
11. Read the last character in TWI_RHR.
12. (Optional) Wait for the TXCOMP flag in TWI_SR before disabling the peripheral clock if required.

40.8.8 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 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 40-15.SMBUS Quick Command



40.8.9 Read-write Flowcharts

The following flowcharts shown in Figure 40-17 on page 1246, Figure 40-18 on page 1247, Figure 40-19 on page 1248, Figure 40-20 on page 1249 and Figure 40-21 on page 1250 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 40-16.TWI Write Operation with Single Data Byte without Internal Address

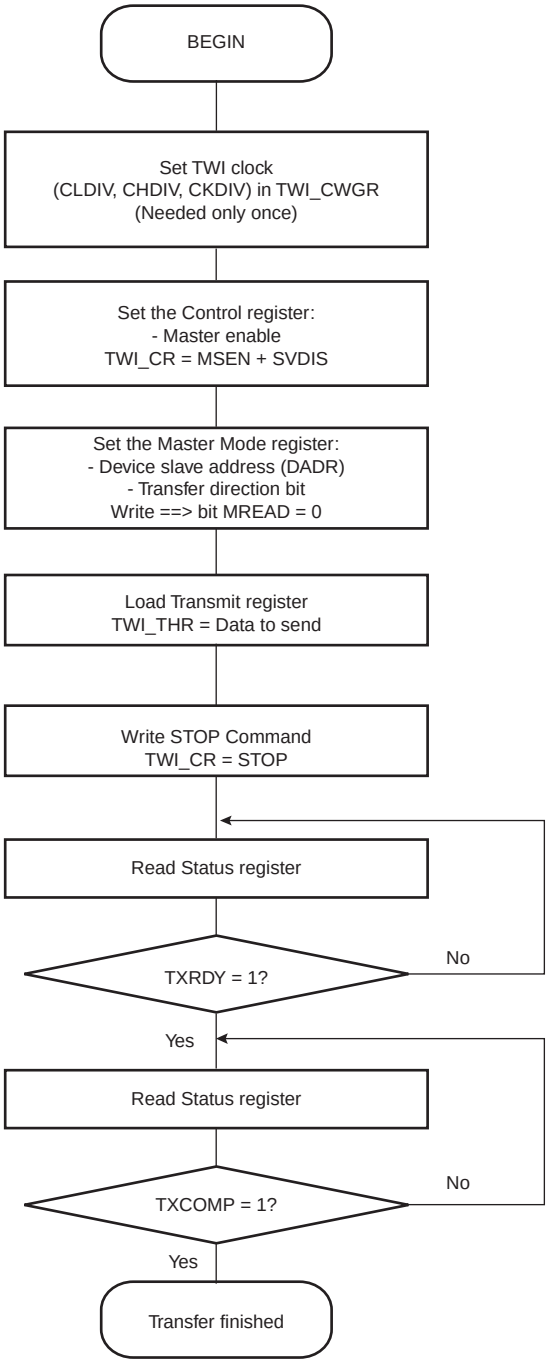


Figure 40-17.TWI Write Operation with Single Data Byte and Internal Address

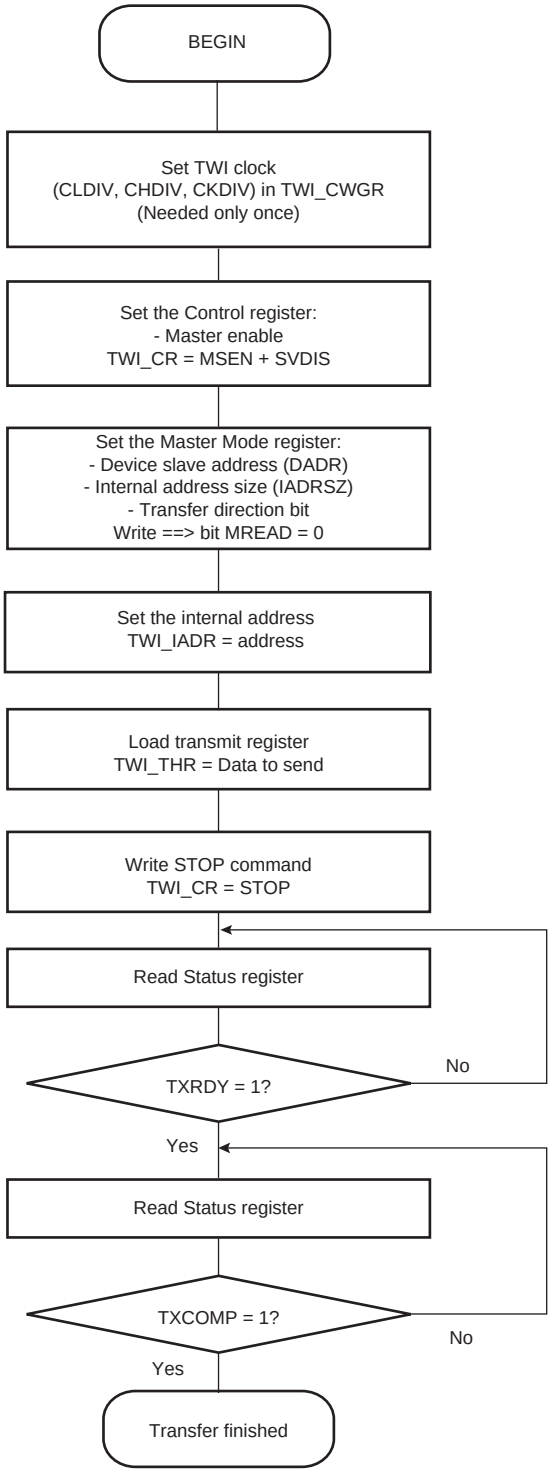


Figure 40-18.TWI Write Operation with Multiple Data Bytes with or without Internal Address

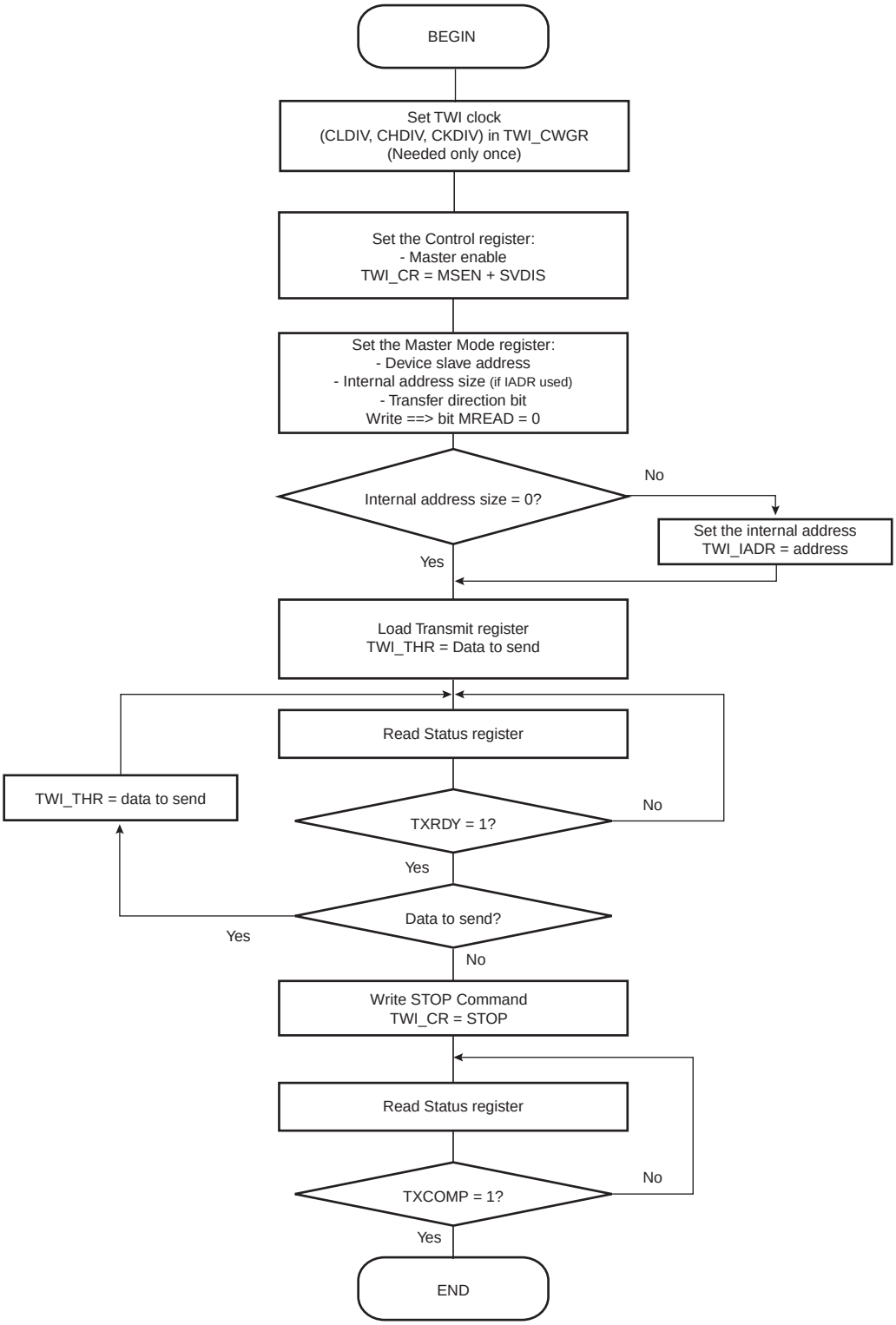


Figure 40-19.TWI Read Operation with Single Data Byte without Internal Address

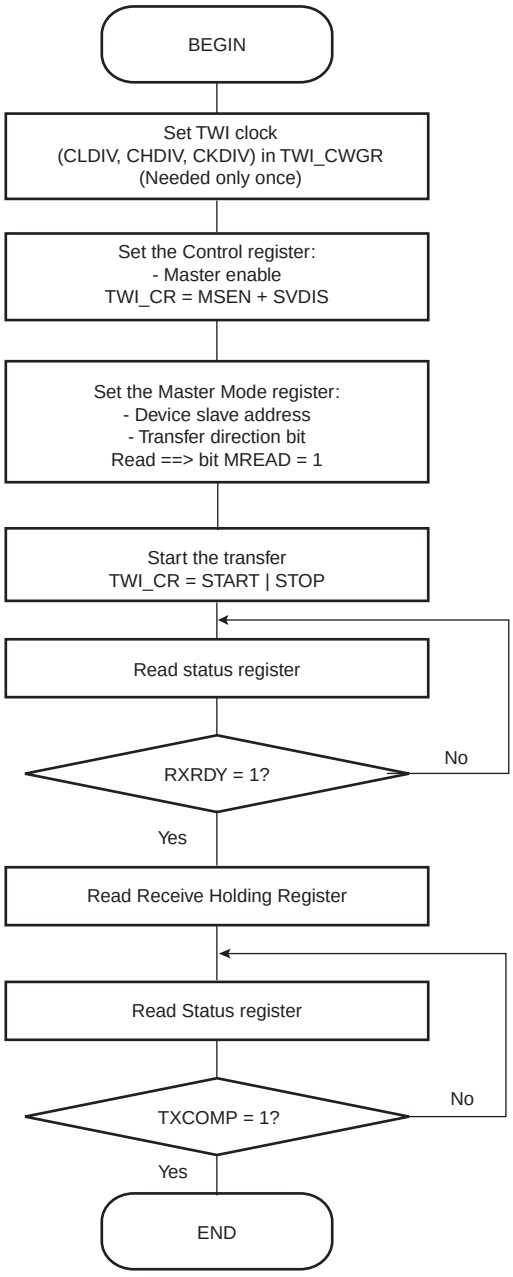


Figure 40-20.TWI Read Operation with Single Data Byte and Internal Address

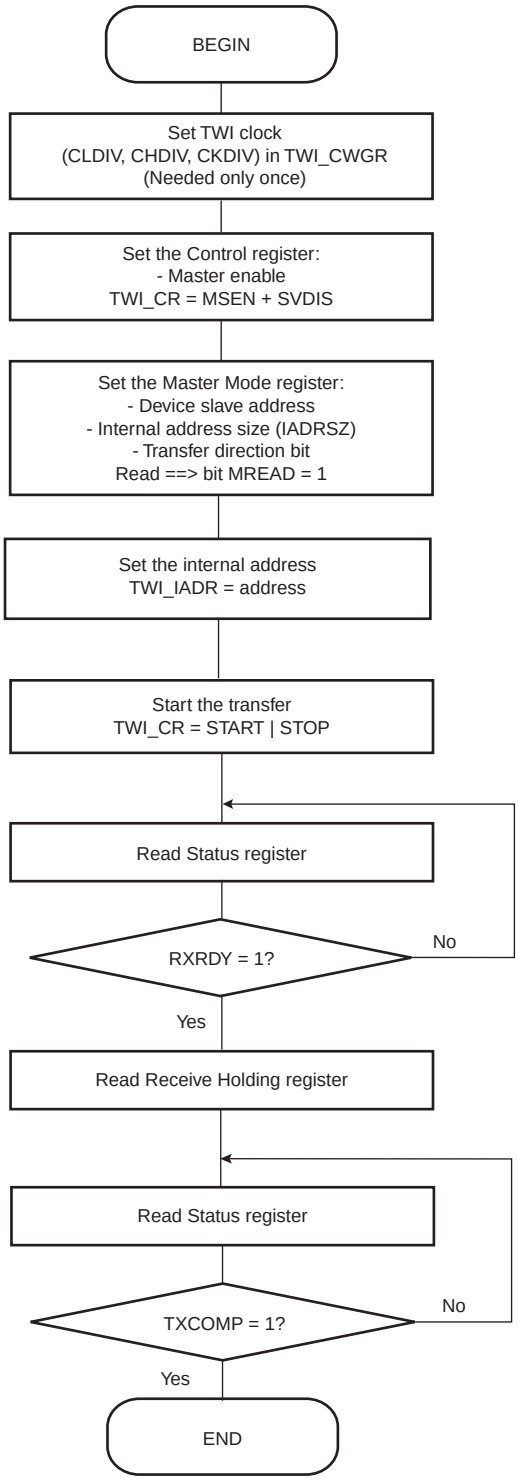
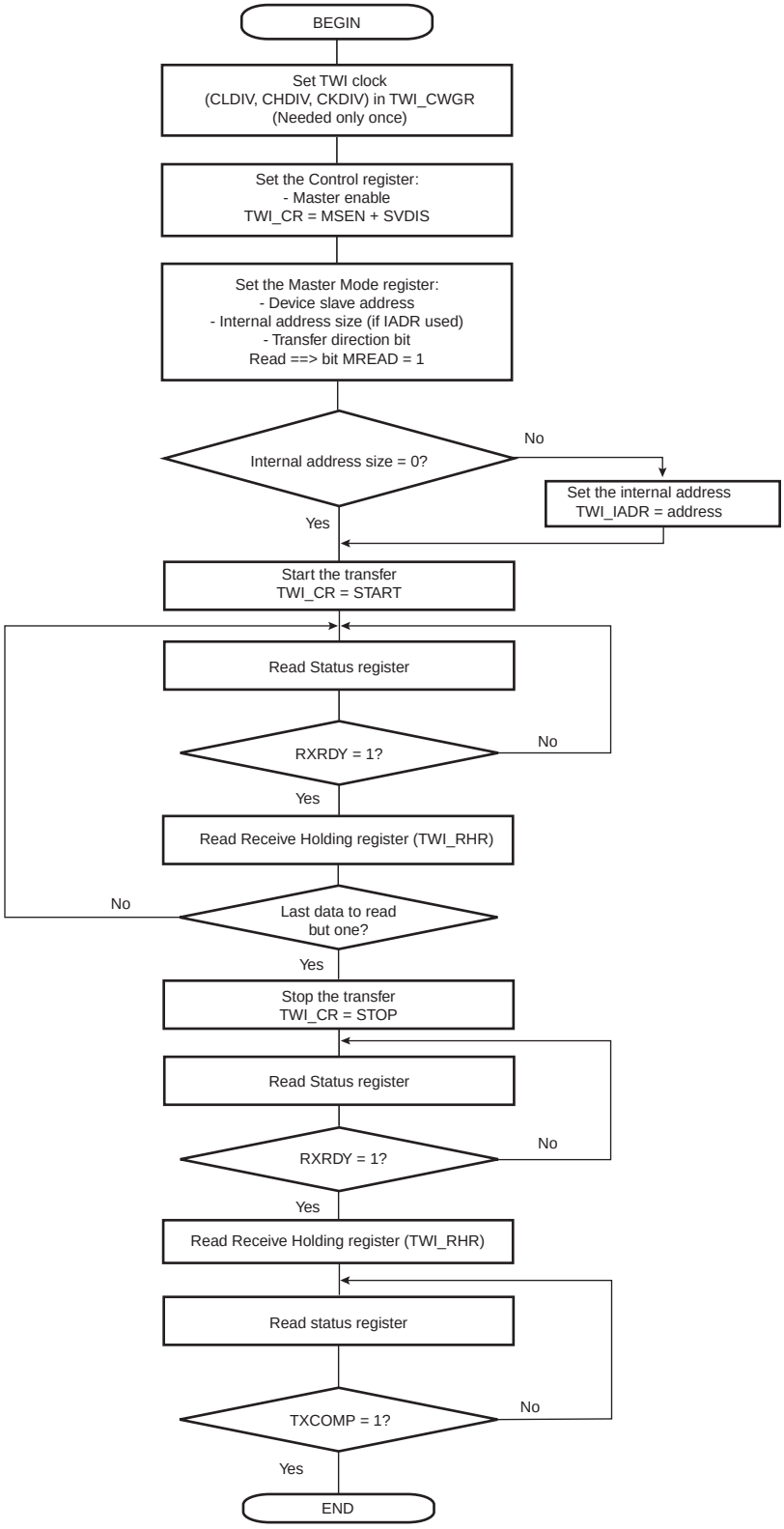


Figure 40-21.TWI Read Operation with Multiple Data Bytes with or without Internal Address



40.9 Multi-master Mode

40.9.1 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 40-23 on page 1252.

40.9.2 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.

40.9.2.1 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 40-22 on page 1252).

Note: The state of the bus (busy or free) is not indicated in the user interface.

40.9.2.2 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 40-22.Programmer Sends Data While the Bus is Busy

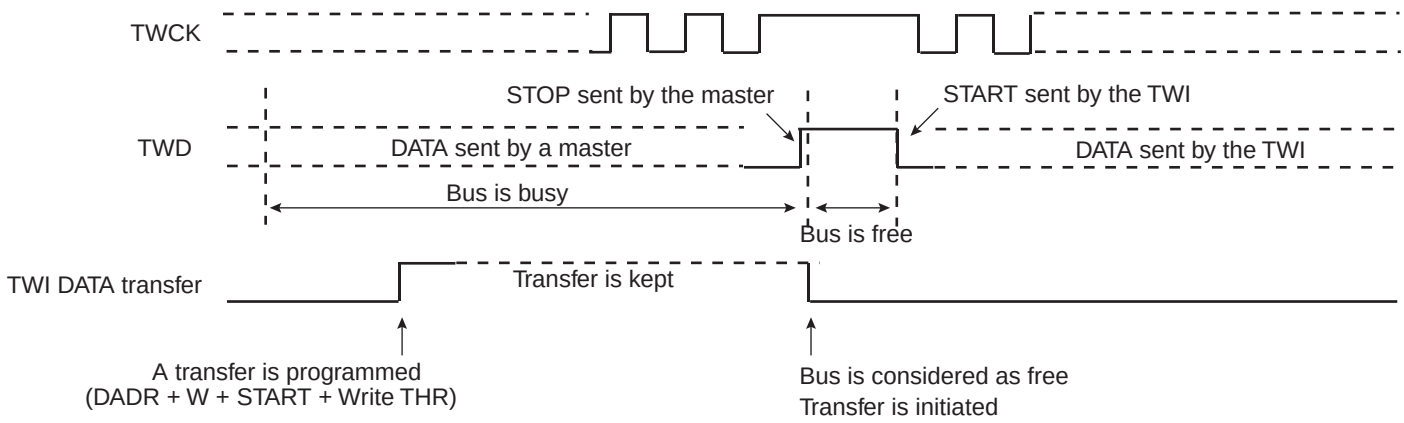
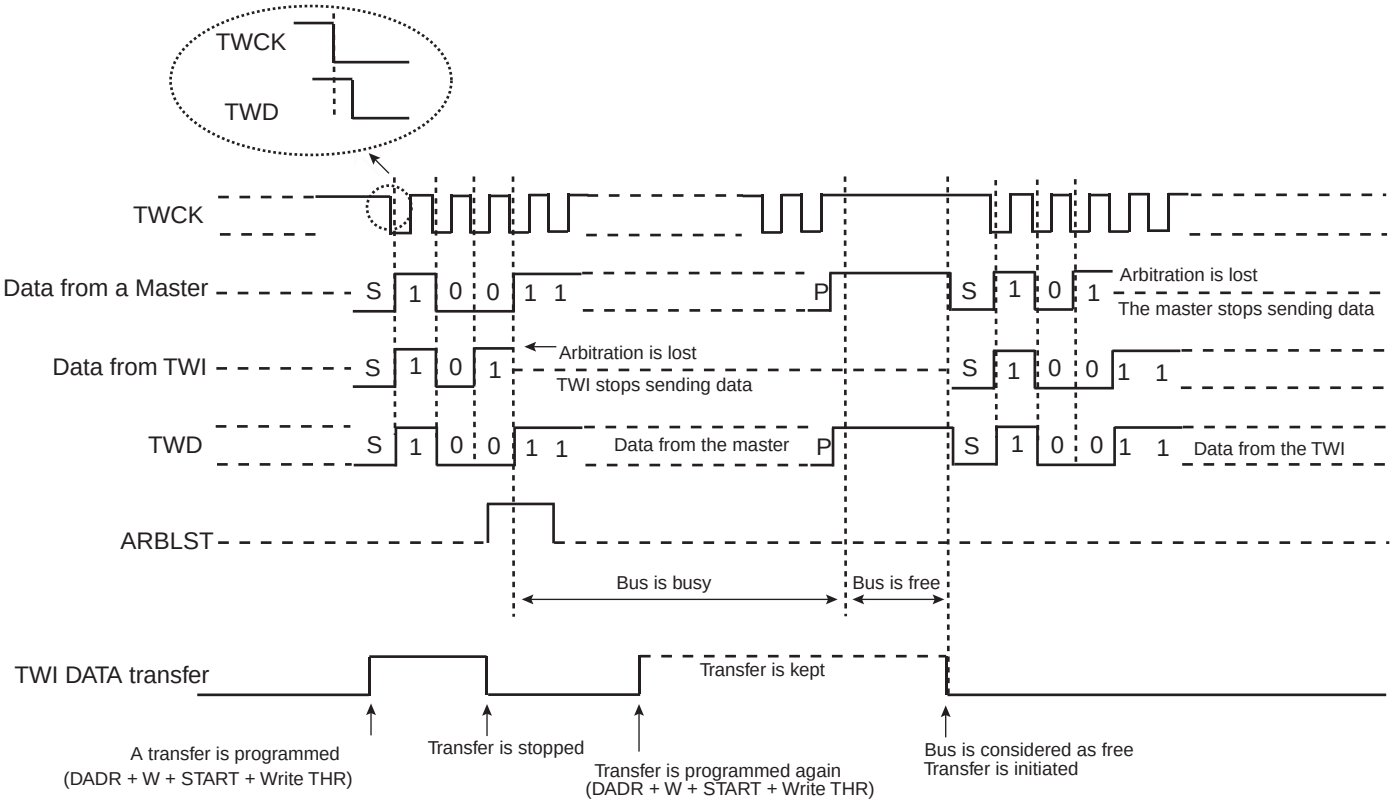
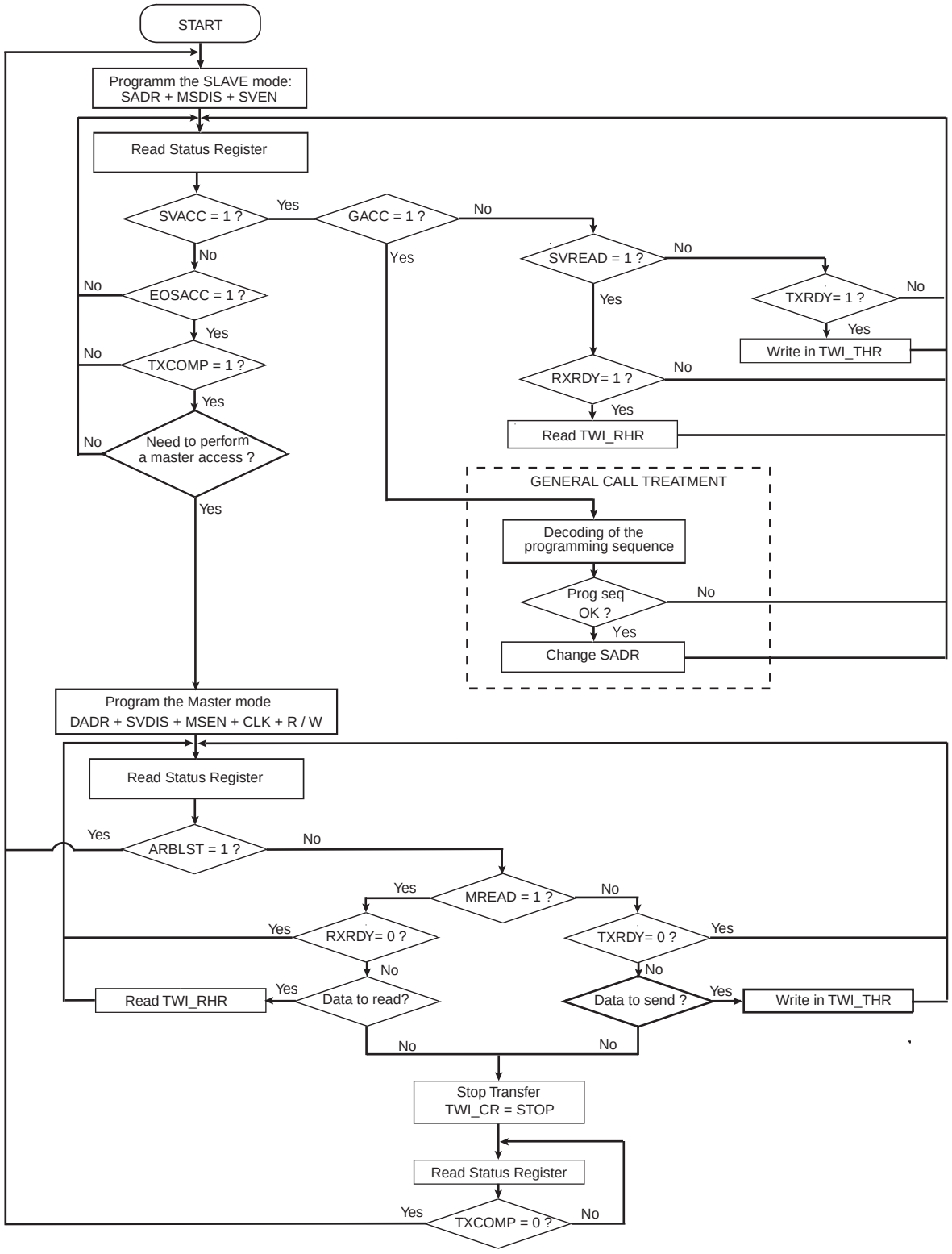


Figure 40-23.Arbitration Cases



The flowchart shown in Figure 40-24 on page 1253 gives an example of read and write operations in Multi-master mode.

Figure 40-24.Multi-master Flowchart



40.10 Slave Mode

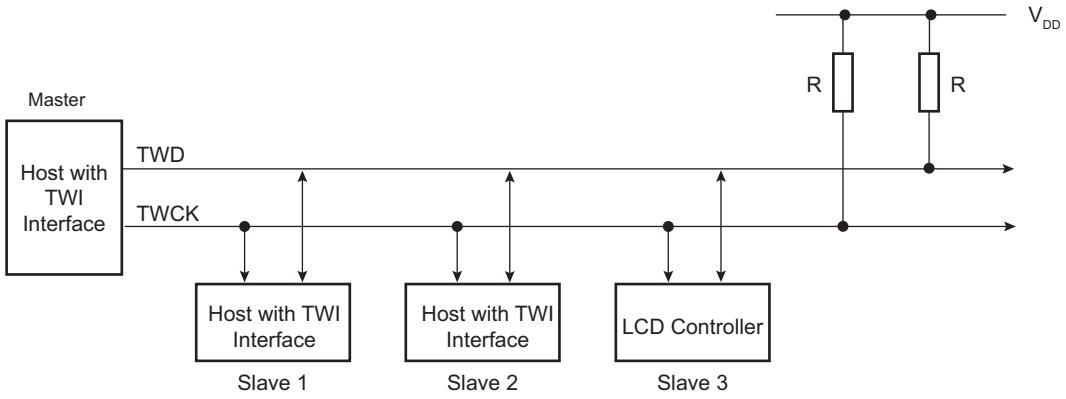
40.10.1 Definition

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).

40.10.2 Application Block Diagram

Figure 40-25.Slave Mode Typical Application Block Diagram



40.10.3 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.

40.10.4 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.

40.10.4.1 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 40-26 on page 1255.

40.10.4.2 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 40-27 on page 1256.

40.10.4.3 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 40-29 on page 1257 and Figure 40-30 on page 1258.

40.10.4.4 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 40-28 on page 1256.

40.10.5 Data Transfer

40.10.5.1 Read Operation

The read mode is defined as a data requirement from the master.

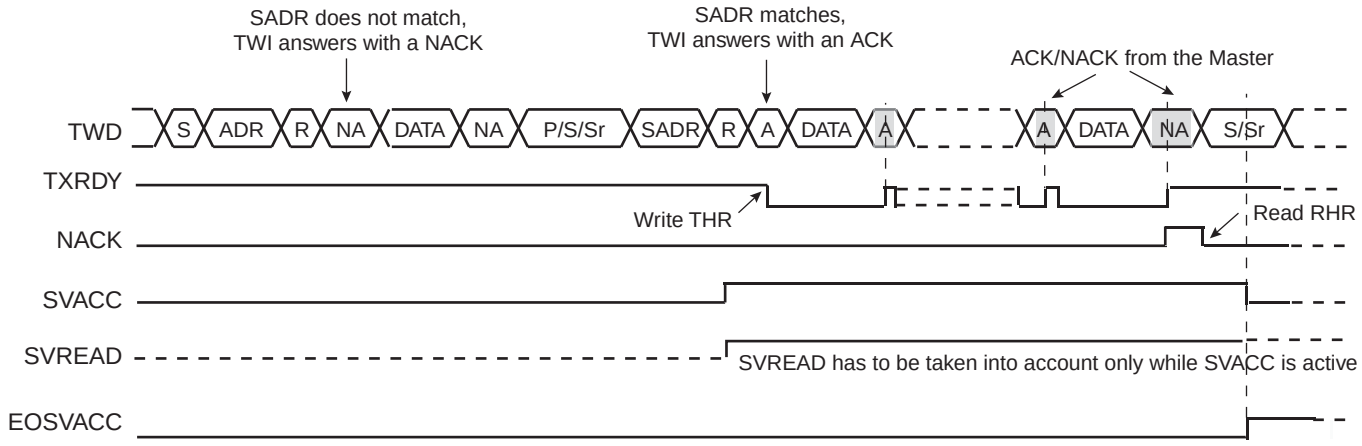
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 40-26 on page 1255 describes the write operation.

Figure 40-26.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.

40.10.5.2 Write Operation

The write mode is defined as a data transmission from the master.

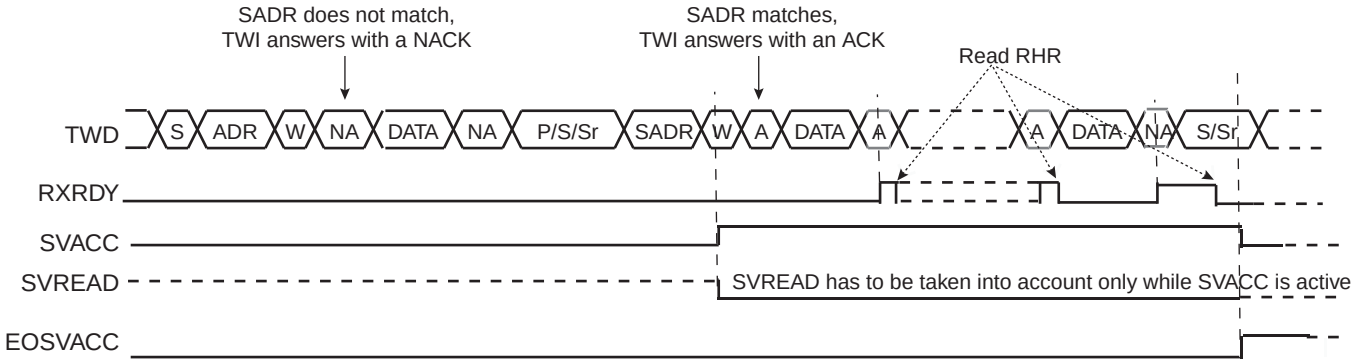
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 40-27 describes the Write operation.

Figure 40-27.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.

40.10.5.3 General Call

The general call is performed in order to change the address of the slave.

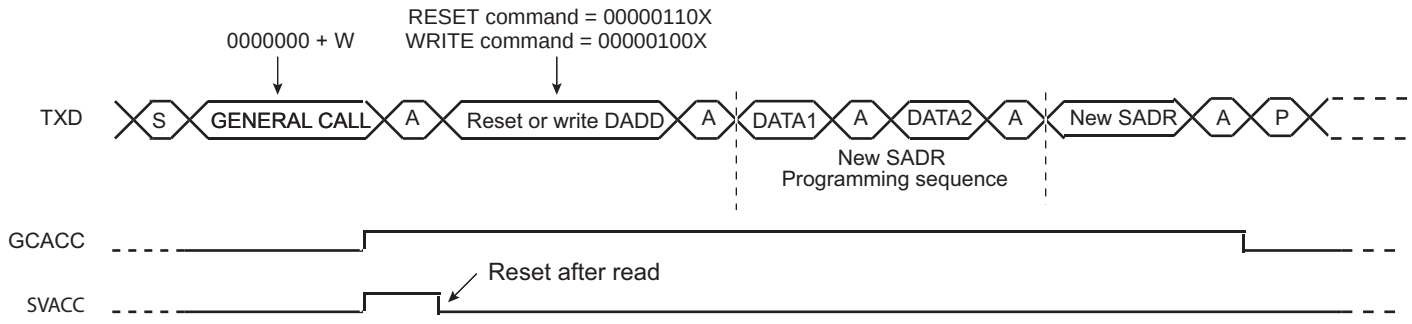
If a GENERAL CALL is detected, GACC is set.

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 40-28 describes the General Call access.

Figure 40-28.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.

40.10.5.4 Clock Synchronization

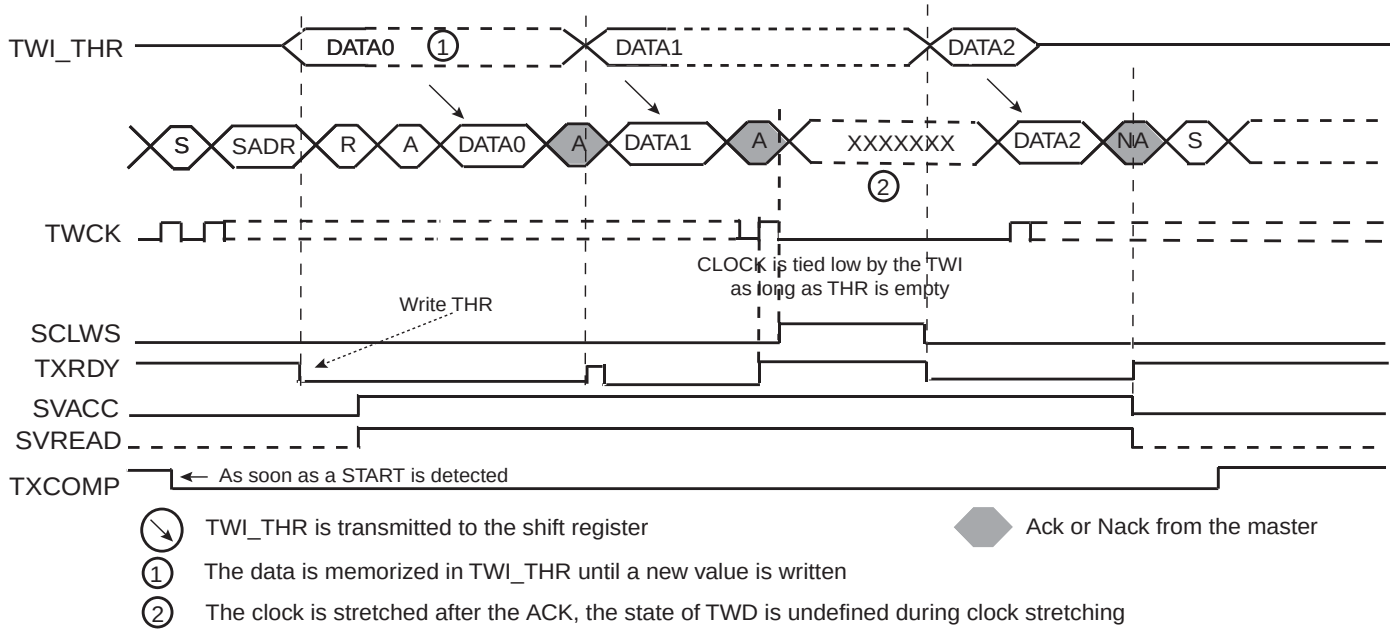
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

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 40-29 describes the clock synchronization in Read mode.

Figure 40-29.Clock Synchronization in Read Mode



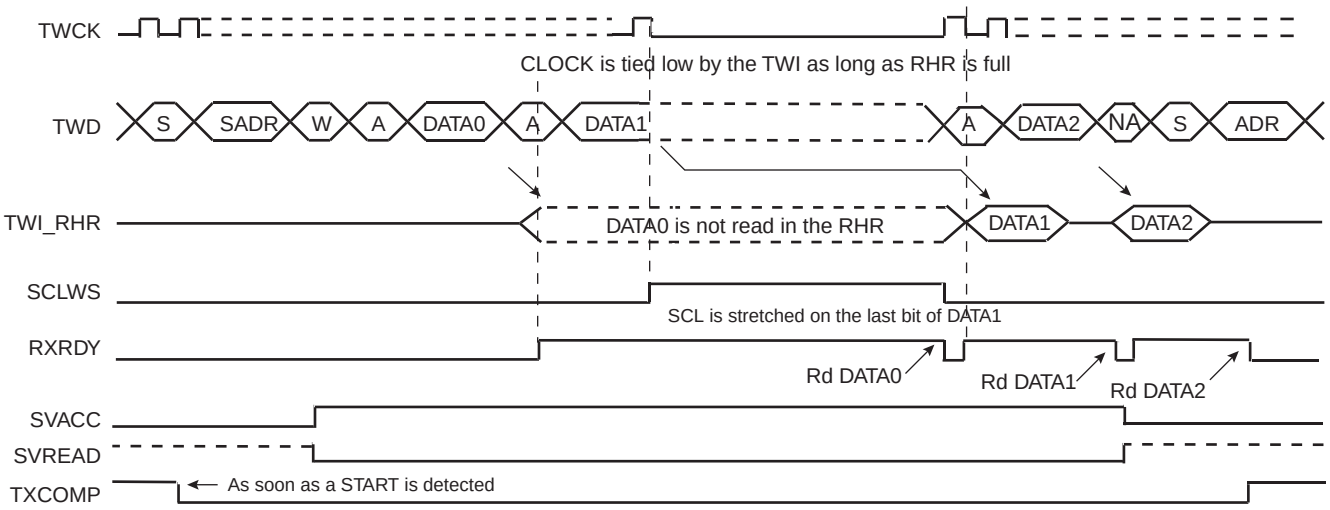
Notes:
1. TXRDY is reset when data has been written in the TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically 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 40-30 describes the clock synchronization in Read mode.

Figure 40-30.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.

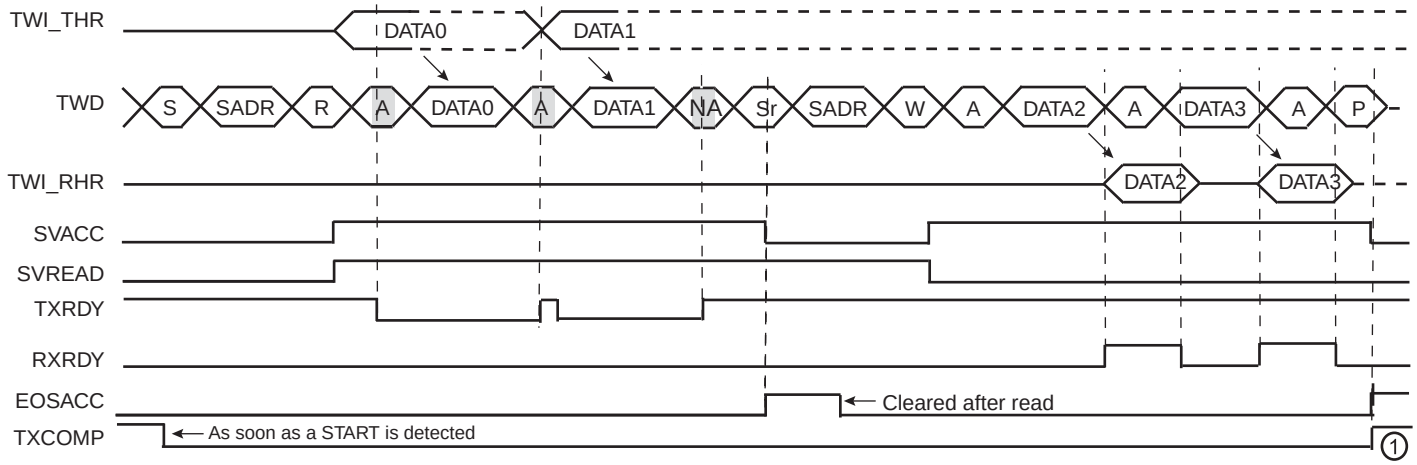
40.10.5.5 Reversal after a Repeated Start

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 40-31 describes the repeated start + reversal from Read to Write mode.

Figure 40-31.Repeated Start + Reversal from Read to Write Mode

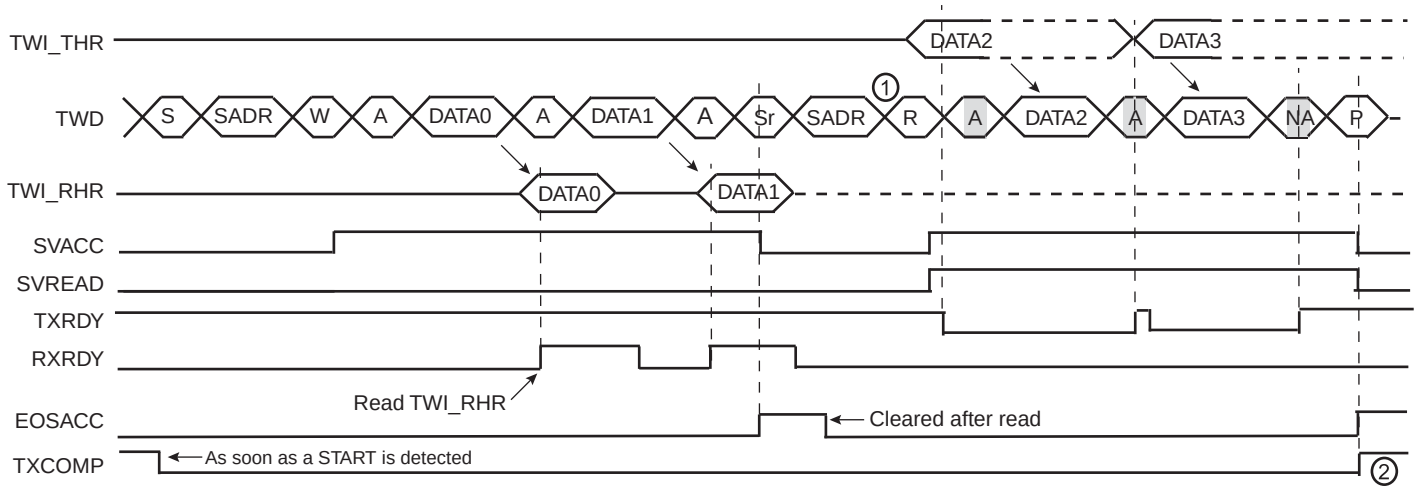


Note: 1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command. Figure 40-32 describes the repeated start + reversal from Write to Read mode.

Figure 40-32.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.

40.10.6 Using the DMA Controller

The use of the DMA significantly reduces the CPU load.

To assure correct implementation, respect the following programming sequence.

40.10.6.1 Data Transmit with the DMA

1. Initialize the DMA (channels, memory pointers, size, etc.).
2. Configure the slave mode.
3. Enable the DMA.
4. Wait for the DMA BTC flag.
5. Disable the DMA.
6. (Optional) Wait for the TXCOMP flag in TWI_SR before disabling the peripheral clock if required.

40.10.6.2 Data Receive with the DMA

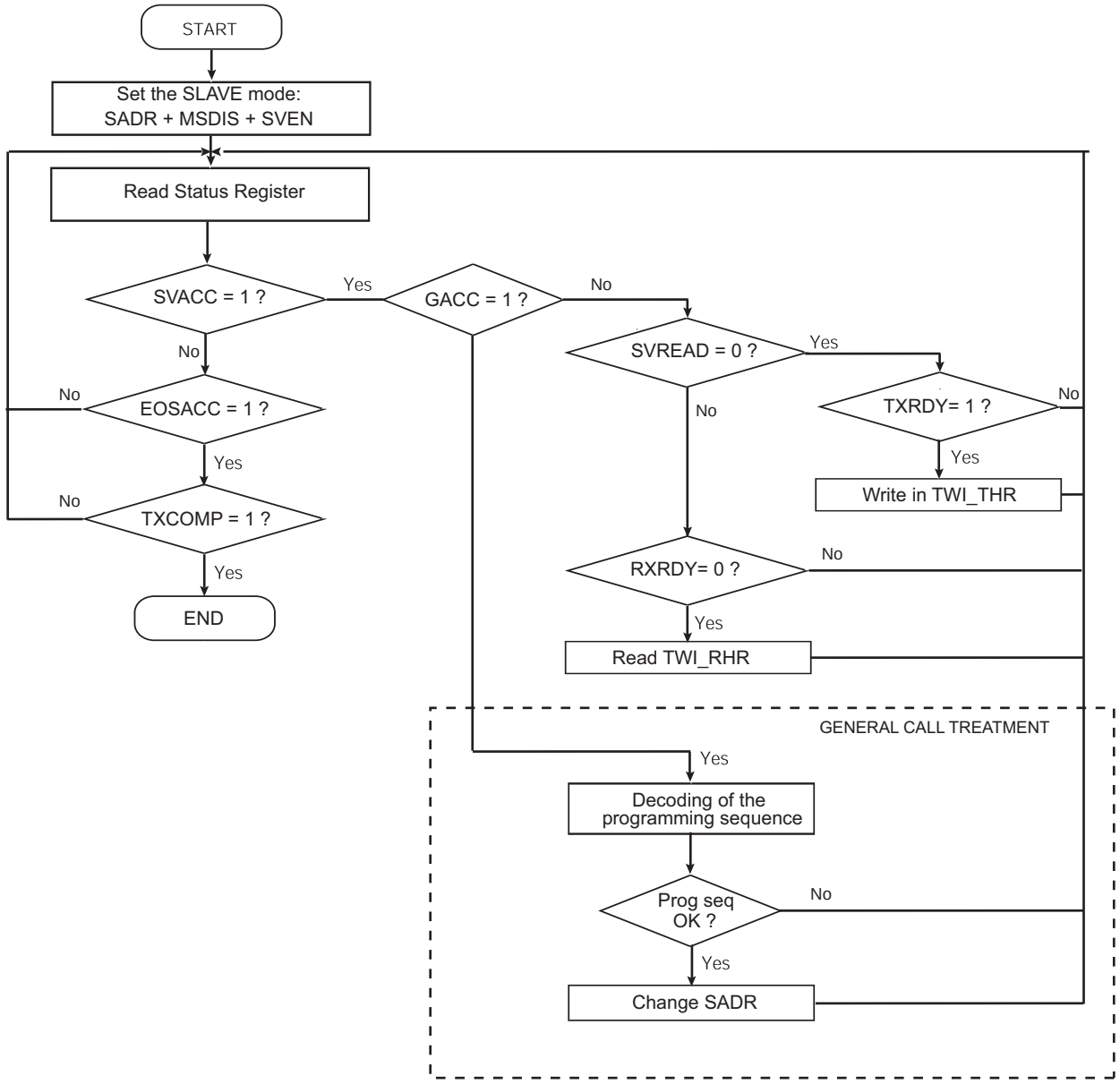
The DMA transfer size must be defined with the buffer size. In slave mode, the number of characters to receive must be known in order to configure the DMA.

1. Initialize the DMA (channels, memory pointers, size, etc.).
2. Configure the slave mode.
3. Enable the DMA.
4. Wait for the DMA BTC flag.
5. Disable the DMA.
6. (Optional) Wait for the TXCOMP flag in TWI_SR before disabling the peripheral clock if required.

40.10.7 Read Write Flowcharts

The flowchart shown in Figure 40-33 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 40-33.Read Write Flowchart in Slave Mode



40.11 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 the “TWI Clock Waveform Generator Register” and the “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 WPVS flag is set and the address of the corresponding canceled register write is available in the WPVSRC field of the “TWI Write Protection 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 WPKEY field of the “TWI Write Protection Mode Register” must be filled with the “TWI” ASCII code (0x545749) otherwise the register write will be canceled.

40.12 Two-wire Interface (TWI) User Interface

Table 40-7. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TWI_CR	Write-only	–
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	–
0x28	Interrupt Disable Register	TWI_IDR	Write-only	–
0x2C	Interrupt Mask Register	TWI_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWI_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWI_THR	Write-only	0x00000000
0x38–0xE0	Reserved	–	–	–
0xE4	Protection Mode Register	TWI_WPMR	Read-write	0x00000000
0xE8	Protection Status Register	TWI_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–

Note: All unlisted offset values are considered as “reserved”.

40.12.1 TWI Control Register

Name: TWI_CR

Address: 0xF0014000 (0), 0xF0018000 (1), 0xF801C000 (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.

40.12.2 TWI Master Mode Register

Name: TWI_MMR

Address: 0xF0014004 (0), 0xF0018004 (1), 0xF801C004 (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.

40.12.3 TWI Slave Mode Register

Name: TWI_SMR

Address: 0xF0014008 (0), 0xF0018008 (1), 0xF801C008 (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.

40.12.4 TWI Internal Address Register

Name: TWI_IADR

Address: 0xF001400C (0), 0xF001800C (1), 0xF801C00C (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.

40.12.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Address: 0xF0014010 (0), 0xF0018010 (1), 0xF801C010 (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.

40.12.6 TWI Status Register

Name: TWI_SR

Address: 0xF0014020 (0), 0xF0018020 (1), 0xF801C020 (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 40-8 on page 1240 and in Figure 40-10 on page 1241.

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 40-29 on page 1257, Figure 40-30 on page 1258, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **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 40-10 on page 1241.

RXRDY behavior in Slave mode can be seen in Figure 40-27 on page 1256, Figure 40-30 on page 1258, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **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.

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 40.8.4 on page 1238.

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 40-26 on page 1255, Figure 40-29 on page 1257, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **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 40-26 on page 1255, Figure 40-27 on page 1256, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **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 40-26 on page 1255, Figure 40-27 on page 1256, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **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 40-28 on page 1256.

- **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 or an address 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 40-29 on page 1257 and Figure 40-30 on page 1258.

- **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 40-31 on page 1258 and Figure 40-32 on page 1259

40.12.7 TWI Interrupt Enable Register

Name: TWI_IER

Address: 0xF0014024 (0), 0xF0018024 (1), 0xF801C024 (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

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **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**

40.12.8 TWI Interrupt Disable Register

Name: TWI_IDR

Address: 0xF0014028 (0), 0xF0018028 (1), 0xF801C028 (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

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **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**

40.12.9 TWI Interrupt Mask Register

Name: TWI_IMR

Address: 0xF001402C (0), 0xF001802C (1), 0xF801C02C (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

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **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**

40.12.10TWI Receive Holding Register

Name: TWI_RHR

Address: 0xF0014030 (0), 0xF0018030 (1), 0xF801C030 (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**

40.12.11TWI Transmit Holding Register

Name: TWI_THR

Address: 0xF0014034 (0), 0xF0018034 (1), 0xF801C034 (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
TXDATA							

- **TXDATA: Master or Slave Transmit Holding Data**

40.12.12TWI Write Protection Mode Register

Name: TWI_WPMR

Address: 0xF00140E4 (0), 0xF00180E4 (1), 0xF801C0E4 (2)

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 0x545749 (“TWI” in ASCII).

1: Enables the Write Protect if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

The write protected registers are:

- “TWI Clock Waveform Generator Register”
- “TWI Slave Mode Register”

- **WPKEY: Write Protect Key**

Value	Name	Description
0x545749	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

40.12.13TWI Write Protection Status Register

Name: TWI_WPSR

Address: 0xF00140E8 (0), 0xF00180E8 (1), 0xF801C0E8 (2)

Access: Read-only

31	30	29	28	27	26	25	24
WPVSRC							
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 TWI_WPSR.

1: A Write Protect Violation has occurred since the last read of the TWI_WPSR. 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 TWI_WPSR automatically clears all fields.

41. Synchronous Serial Controller (SSC)

41.1 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:

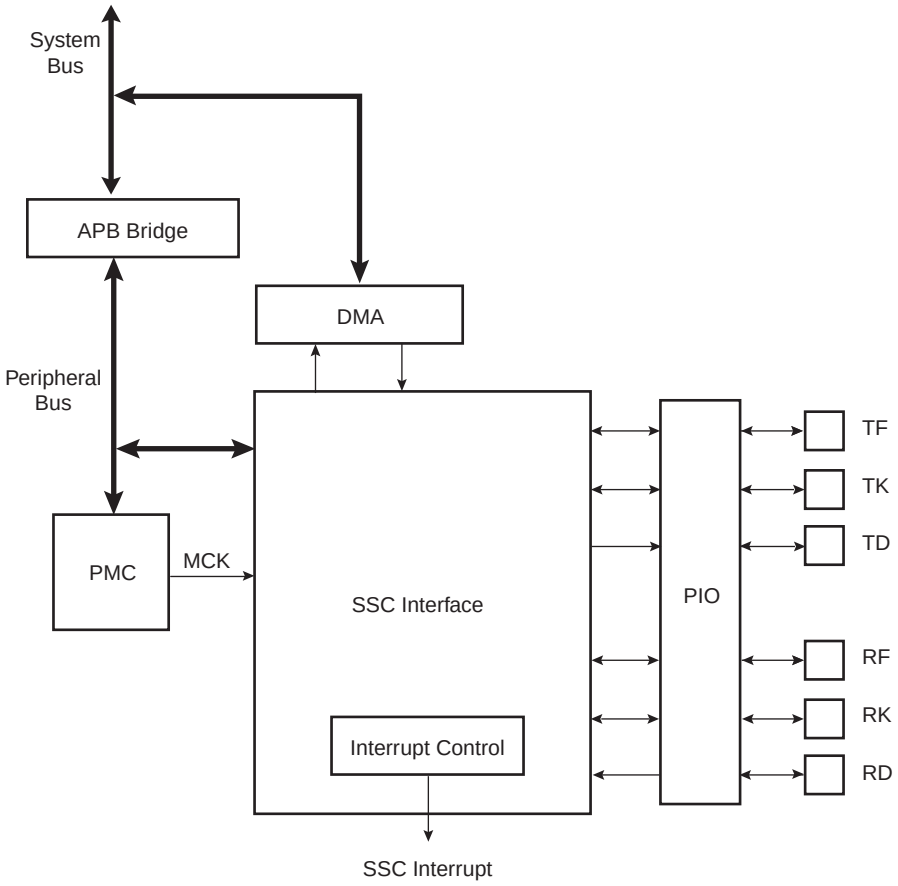
- CODEC’s in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

41.2 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

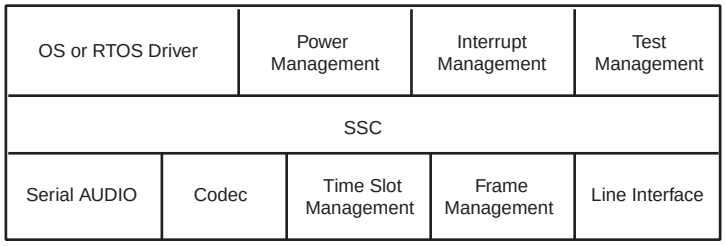
41.3 Block Diagram

Figure 41-1. Block Diagram



41.4 Application Block Diagram

Figure 41-2. Application Block Diagram



41.5 Pin Name List

Table 41-1. 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

41.6 Product Dependencies

41.6.1 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 41-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SSC0	RD0	PC21	A
SSC0	RF0	PC20	A
SSC0	RK0	PC19	A
SSC0	TD0	PC18	A
SSC0	TF0	PC17	A
SSC0	TK0	PC16	A
SSC1	RD1	PB11	B
SSC1	RF1	PB10	B
SSC1	RK1	PB7	B
SSC1	TD1	PB6	B
SSC1	TF1	PB3	B
SSC1	TK1	PB2	B

41.6.2 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.

41.6.3 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

Table 41-3. Peripheral IDs

Instance	ID
SSC0	38
SSC1	39

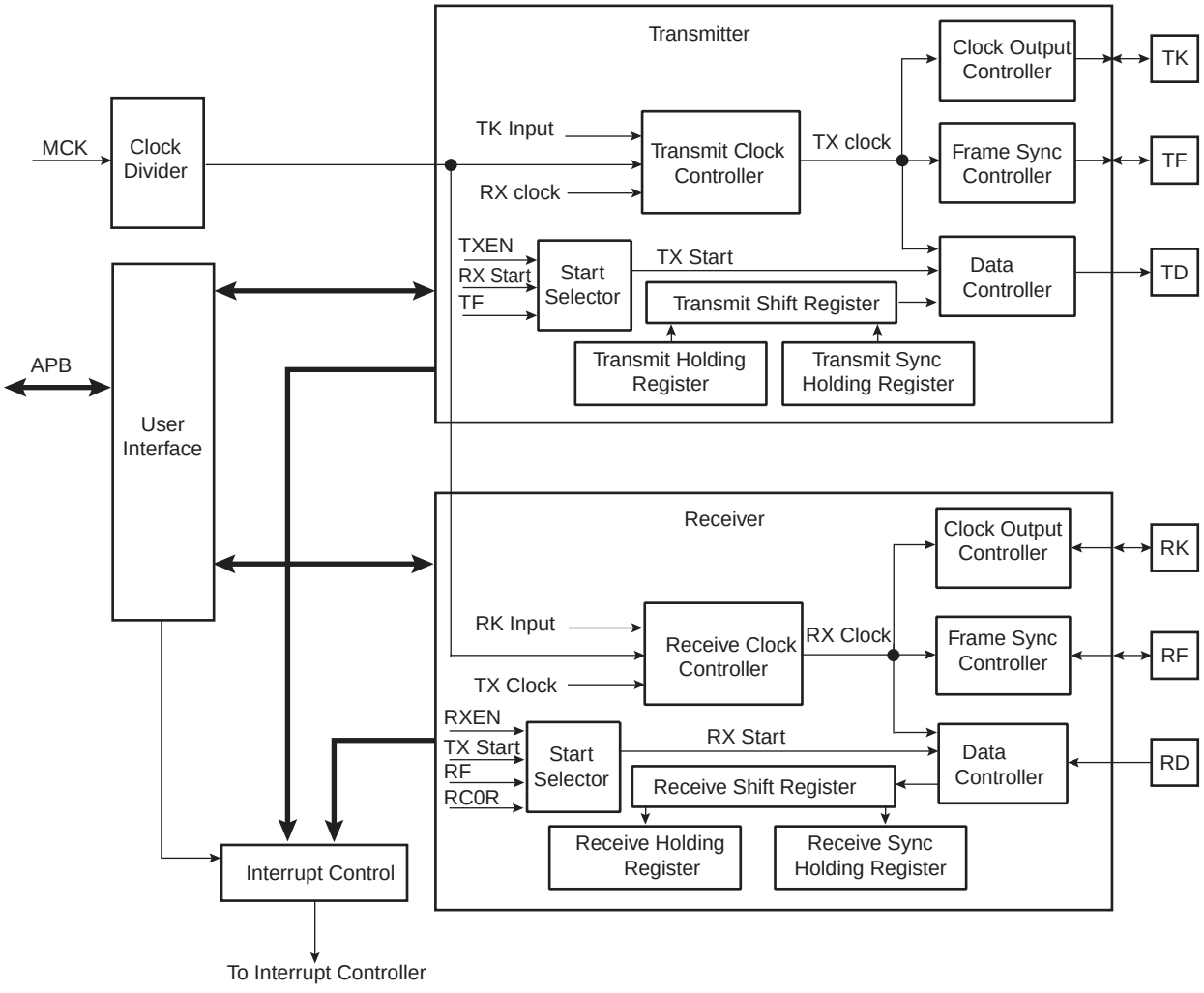
SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

41.7 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 41-3. SSC Functional Block Diagram



41.7.1 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

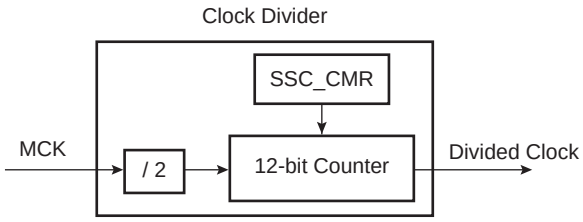
- the transmitter clock
- the internal clock divider

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.

41.7.1.1 Clock Divider

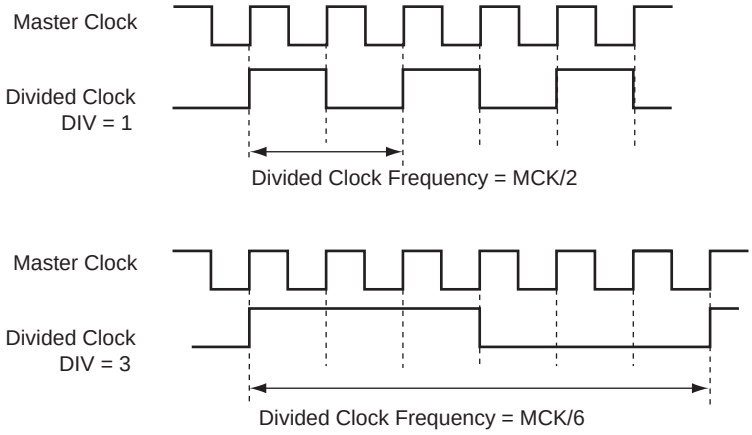
Figure 41-4. Divided Clock Block Diagram



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 41-5. Divided Clock Generation

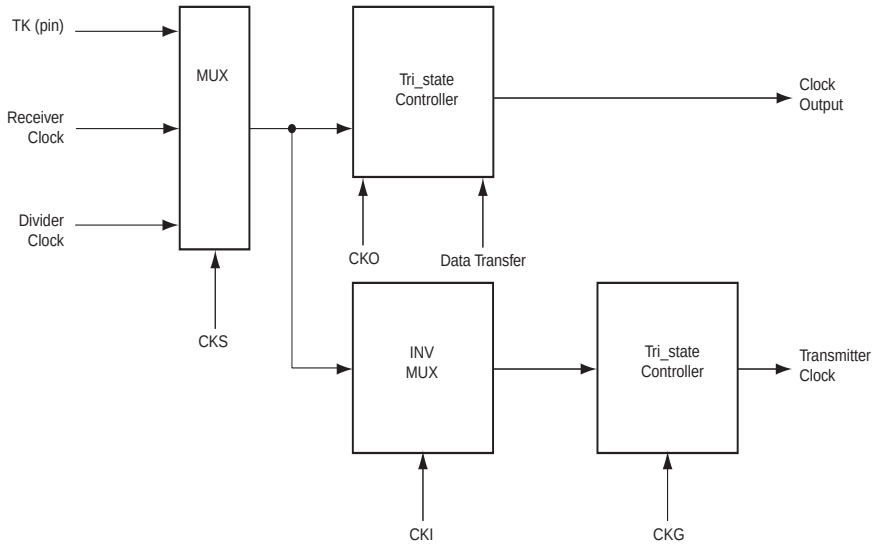


41.7.1.2 Transmitter Clock Management

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 41-6. Transmitter Clock Management

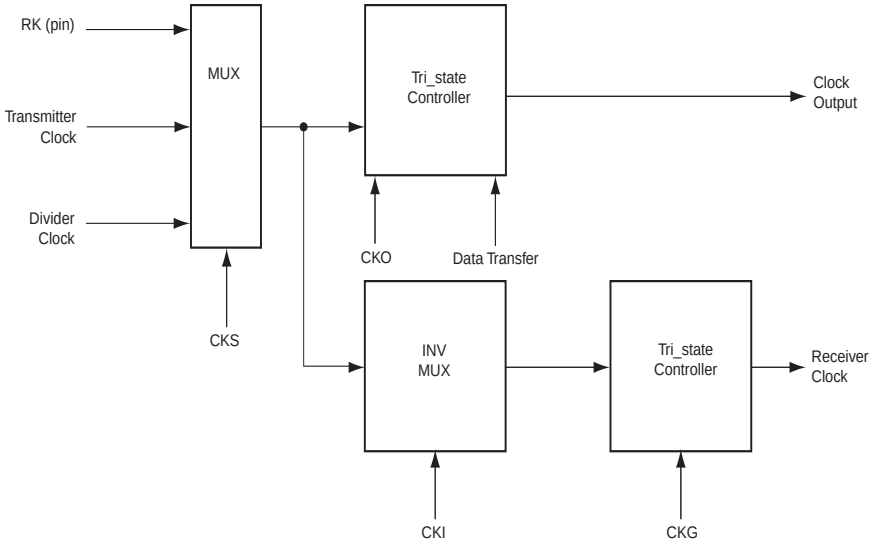


41.7.1.3 Receiver Clock Management

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 41-7. Receiver Clock Management



41.7.1.4 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

41.7.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

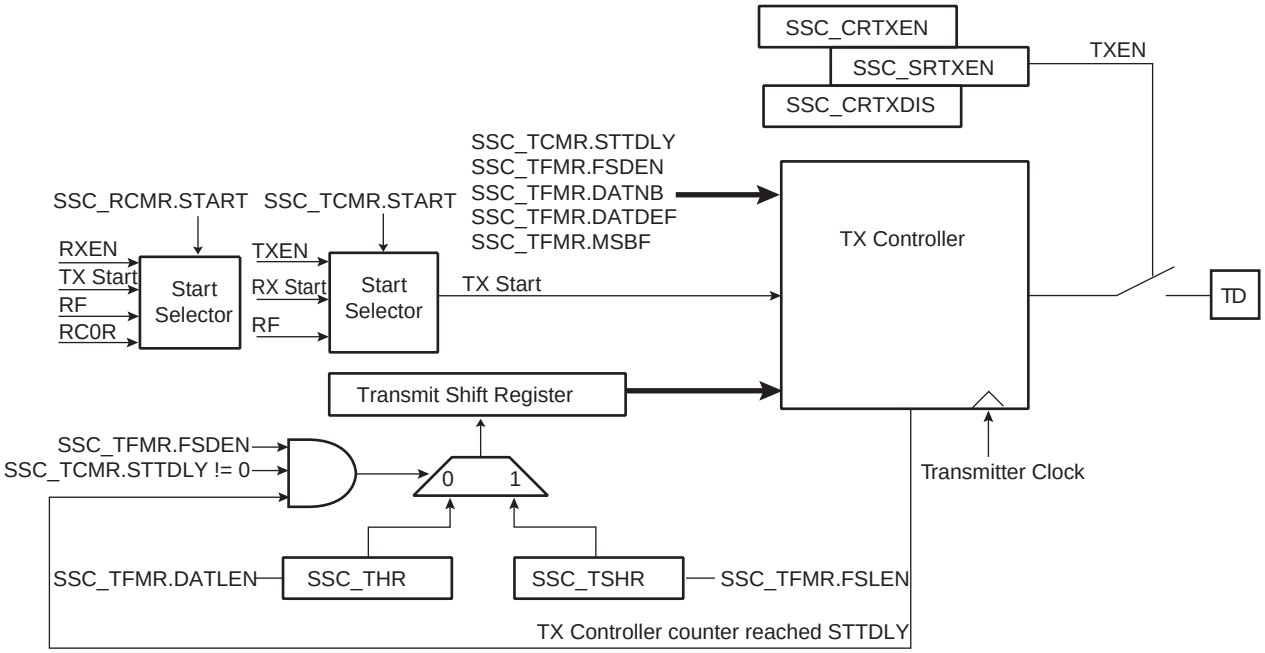
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See “Start” on page 1288.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See “Frame Sync” on page 1290.

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 41-8. Transmitter Block Diagram



41.7.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

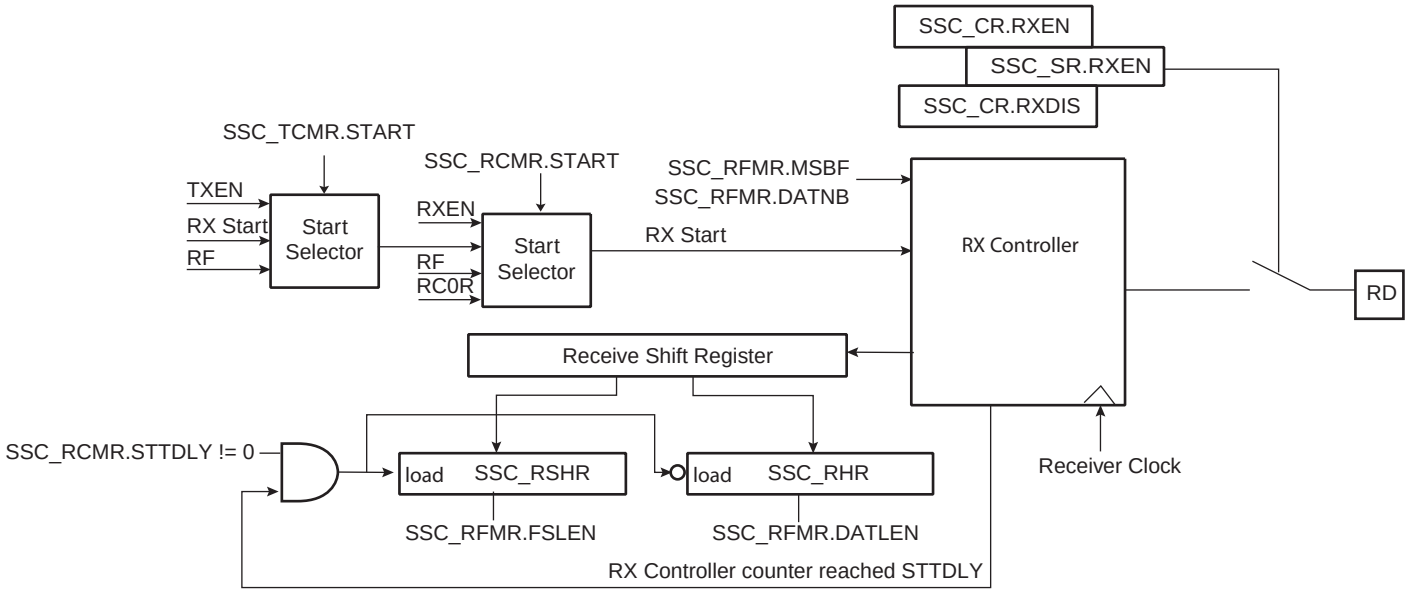
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See “Start” on page 1288.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See “Frame Sync” on page 1290.

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 41-9. Receiver Block Diagram



41.7.4 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 41-10.Transmit Start Mode

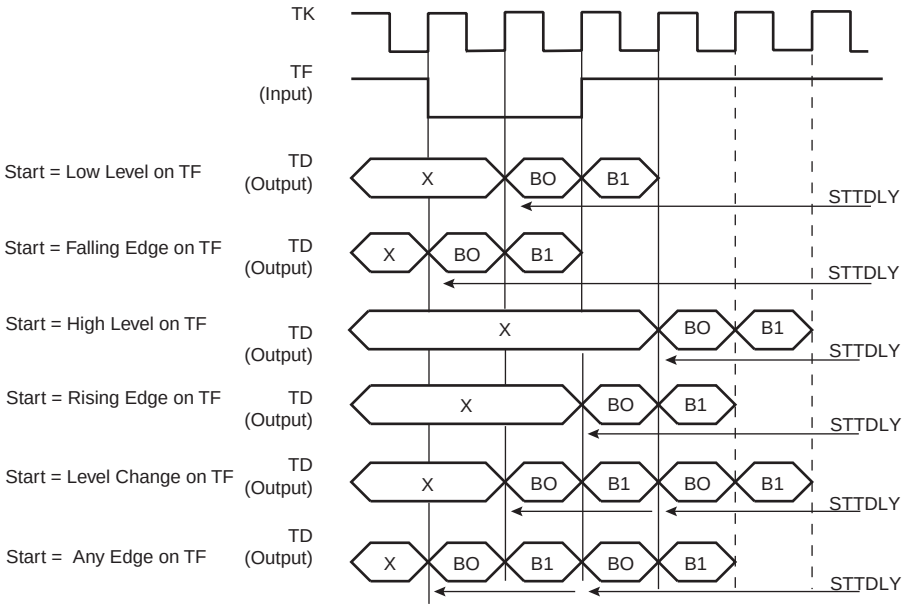
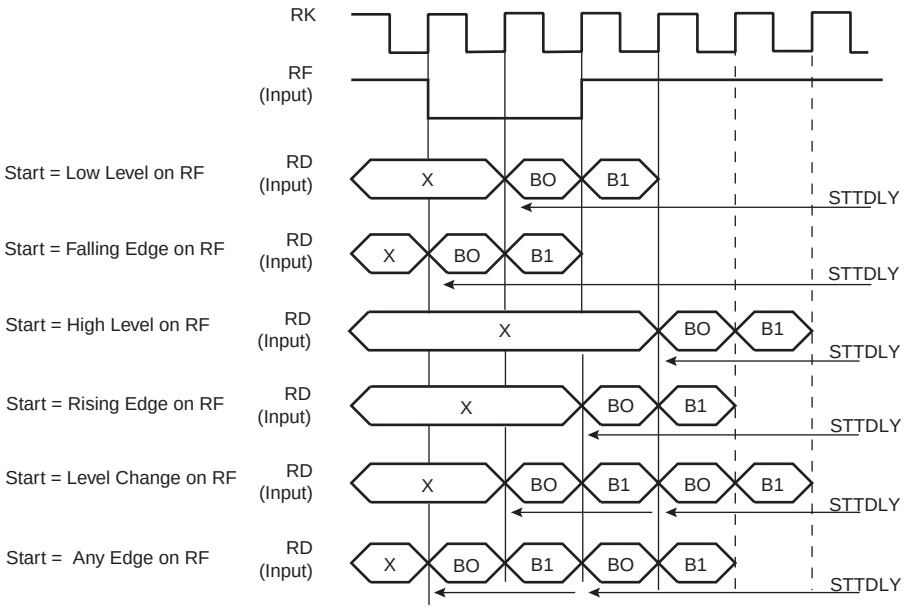


Figure 41-11.Receive Pulse/Edge Start Modes



41.7.5 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.

41.7.5.1 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.

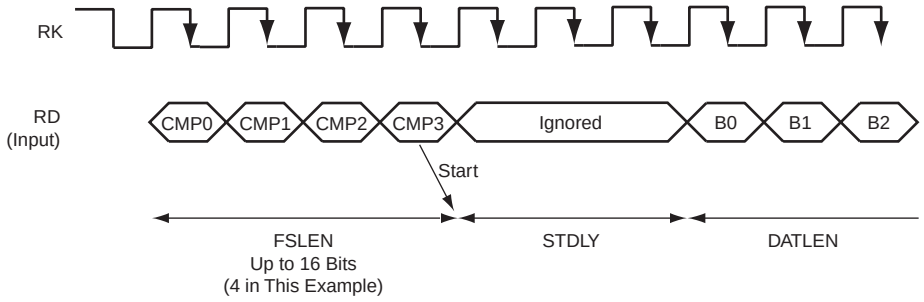
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

41.7.5.2 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).

41.7.6 Receive Compare Modes

Figure 41-12.Receive Compare Modes



41.7.6.1 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.

41.7.7 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:

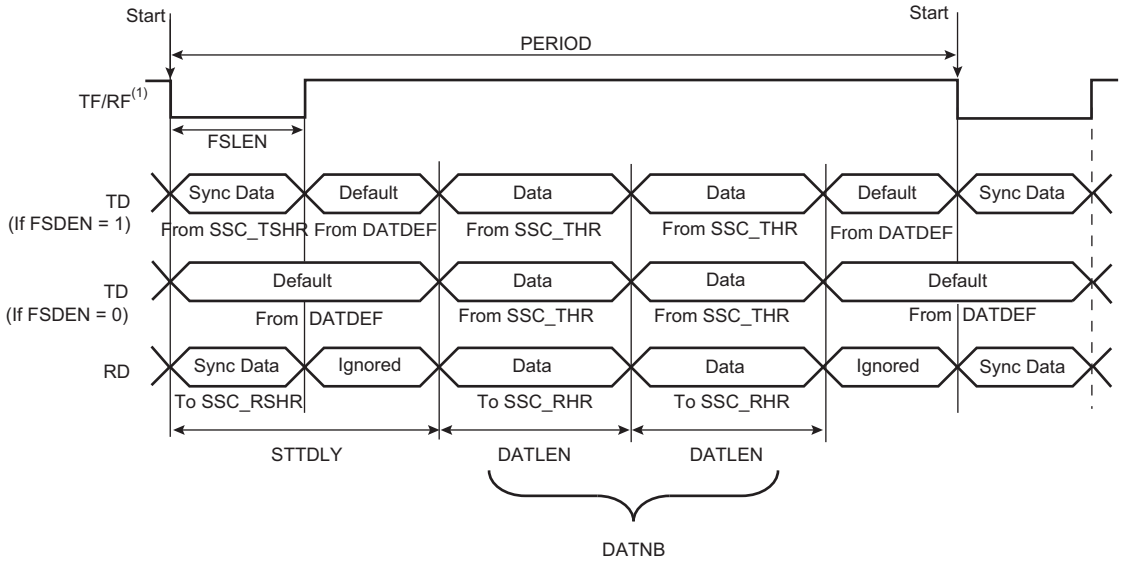
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start 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 41-4. Data Frame Registers

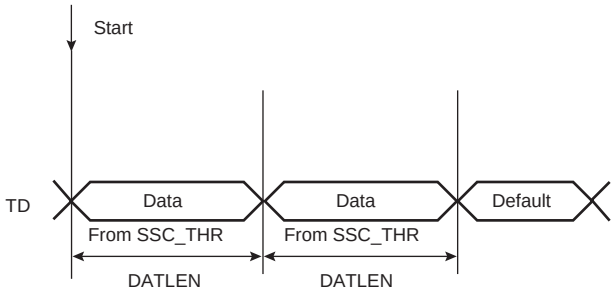
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data 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 41-13.Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

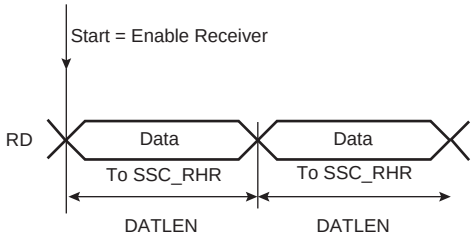
Figure 41-14.Transmit Frame Format in Continuous Mode



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 41-15.Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

41.7.8 Loop Mode

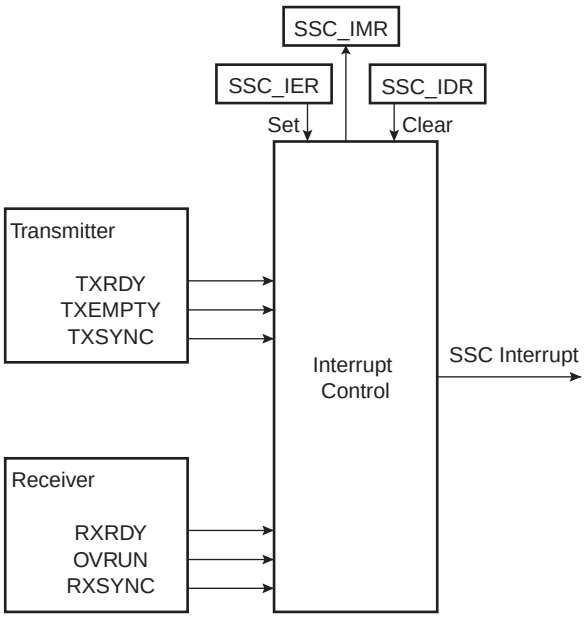
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

41.7.9 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 41-16.Interrupt Block Diagram



41.8 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 41-17.Audio Application Block Diagram

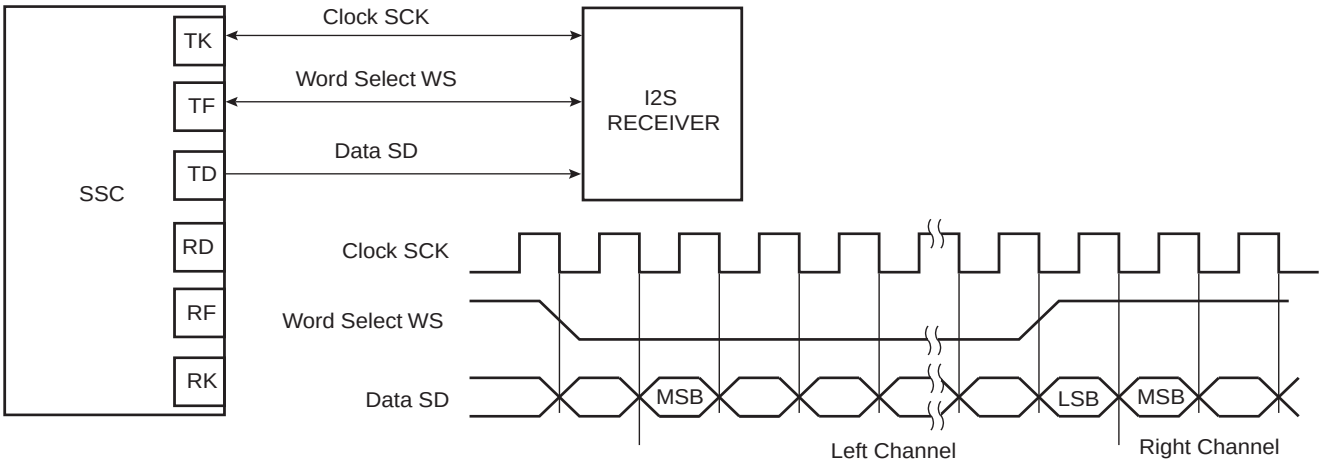


Figure 41-18.Codec Application Block Diagram

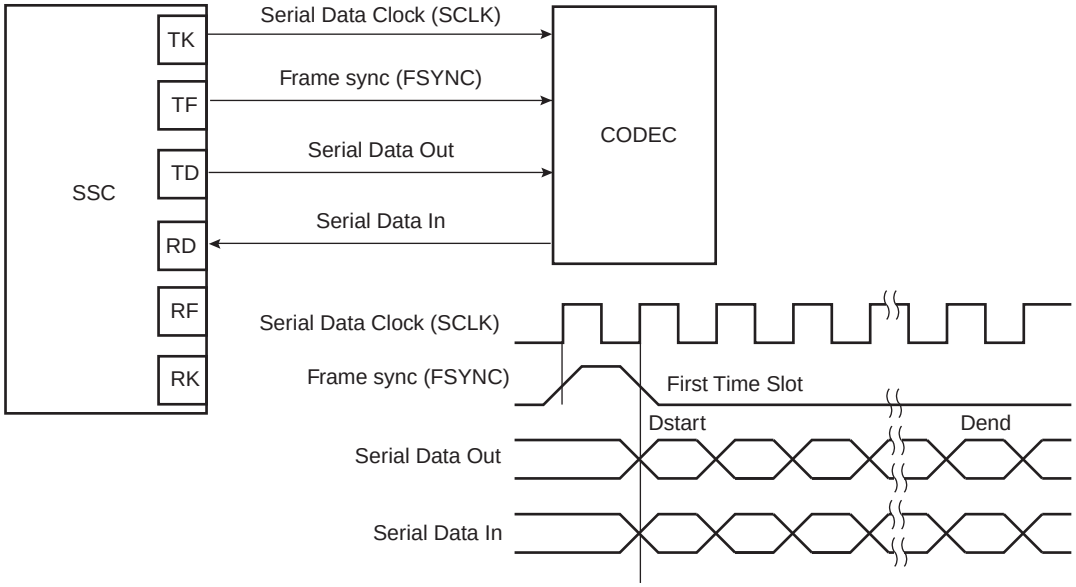
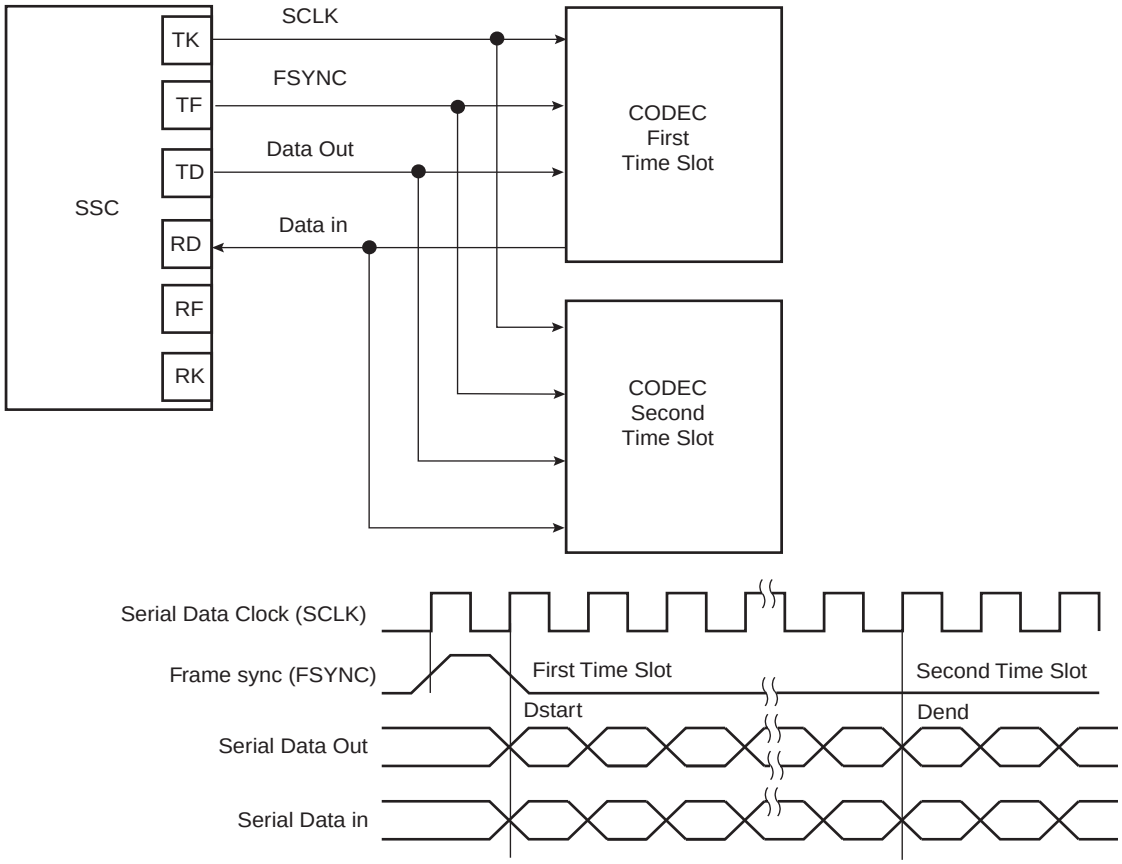


Figure 41-19.Time Slot Application Block Diagram



41.8.1 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 1298
- "SSC Receive Clock Mode Register" on page 1299
- "SSC Receive Frame Mode Register" on page 1301
- "SSC Transmit Clock Mode Register" on page 1303
- "SSC Transmit Frame Mode Register" on page 1305
- "SSC Receive Compare 0 Register" on page 1311
- "SSC Receive Compare 1 Register" on page 1312

41.9 Synchronous Serial Controller (SSC) User Interface

Table 41-5. 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	–	–	–

41.9.1 SSC Control Register

Name: SSC_CR:

Address: 0xF0008000 (0), 0xF800C000 (1)

Access: Write-only

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

- **RXEN: Receive Enable**

0 = No effect.

1 = Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0 = No effect.

1 = Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0 = No effect.

1 = Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0 = No effect.

1 = Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0 = No effect.

1 = Performs a software reset. Has priority on any other bit in SSC_CR.

41.9.2 SSC Clock Mode Register

Name: SSC_CMR

Address: 0xF0008004 (0), 0xF800C004 (1)

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

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.

41.9.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Address: 0xF0008010 (0), 0xF800C010 (1)

Access: Read-write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
–	–	–	STOP	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

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 Low
2	EN_RF_HIGH	Receive Clock enabled only if RF 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.

41.9.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Address: 0xF0008014 (0), 0xF800C014 (1)

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 1301.

41.9.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Address: 0xF0008018 (0), 0xF800C018 (1)

Access: Read-write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

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

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

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 Low
2	EN_TF_HIGH	Transmit Clock enabled only if TF 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.

41.9.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Address: 0xF000801C (0), 0xF800C01C (1)

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, 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
4	HIGH	Driven High during data transfer
5	TOGGLING	Toggling 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 1305.

41.9.7 SSC Receive Holding Register

Name: SSC_RHR

Address: 0xF0008020 (0), 0xF800C020 (1)

Access: Read-only

31	30	29	28	27	26	25	24
RDAT							

23	22	21	20	19	18	17	16
RDAT							

15	14	13	12	11	10	9	8
RDAT							

7	6	5	4	3	2	1	0
RDAT							

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

41.9.8 SSC Transmit Holding Register

Name: SSC_THR

Address: 0xF0008024 (0), 0xF800C024 (1)

Access: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

41.9.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

Address: 0xF0008030 (0), 0xF800C030 (1)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

41.9.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

Address: 0xF0008034 (0), 0xF800C034 (1)

Access: Read-write

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

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

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: Transmit Synchronization Data**

41.9.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Address: 0xF0008038 (0), 0xF800C038 (1)

Access: Read-write

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

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

15	14	13	12	11	10	9	8
CP0							

7	6	5	4	3	2	1	0
CP0							

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

- **CP0: Receive Compare Data 0**

41.9.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Address: 0xF000803C (0), 0xF800C03C (1)

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CP1							
7	6	5	4	3	2	1	0
CP1							

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

- **CP1: Receive Compare Data 1**

41.9.13 SSC Status Register

Name: SSC_SR

Address: 0xF0008040 (0), 0xF800C040 (1)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
–	–	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.

41.9.14 SSC Interrupt Enable Register

Name: SSC_IER

Address: 0xF0008044 (0), 0xF800C044 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	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.

41.9.15 SSC Interrupt Disable Register

Name: SSC_IDR

Address: 0xF0008048 (0), 0xF800C048 (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
–	–	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.

41.9.16 SSC Interrupt Mask Register

Name: SSC_IMR

Address: 0xF000804C (0), 0xF800C04C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	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.

41.9.17 SSC Write Protect Mode Register

Name: SSC_WPMR

Address: 0xF00080E4 (0), 0xF800C0E4 (1)

Access: Read-write

Reset: See Table 41-5

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 1298
- “SSC Receive Clock Mode Register” on page 1299
- “SSC Receive Frame Mode Register” on page 1301
- “SSC Transmit Clock Mode Register” on page 1303
- “SSC Transmit Frame Mode Register” on page 1305
- “SSC Receive Compare 0 Register” on page 1311
- “SSC Receive Compare 1 Register” on page 1312

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x535343	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

41.9.18 SSC Write Protect Status Register

Name: SSC_WPSR

Address: 0xF00080E8 (0), 0xF800C0E8 (1)

Access: Read-only

Reset: See Table 41-5

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.

42. Debug Unit (DBGU)

42.1 Description

The Debug Unit (DBGU) 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.

42.2 Embedded Characteristics

- System Peripheral to Facilitate Debug of Atmel® ARM®-based Systems
- Composed of Four Functions
  - Two-pin UART
  - Debug Communication Channel (DCC) Support
  - Chip ID Registers
  - ICE Access Prevention
- Two-pin UART
  - Implemented Features are USART Compatible
  - 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
- Debug Communication Channel Support
  - Offers Visibility of COMMRX and COMMTX Signals from the ARM Processor
  - Interrupt Generation
- Chip ID Registers
  - Identification of the Device Revision, Sizes of the Embedded Memories, Set of Peripherals
- ICE Access Prevention
  - Enables Software to Prevent System Access Through the ARM Processor's ICE
  - Prevention is Made by Asserting the NTRST Line of the ARM Processor's ICE

42.3 Block Diagram

Figure 42-1. Debug Unit Functional Block Diagram

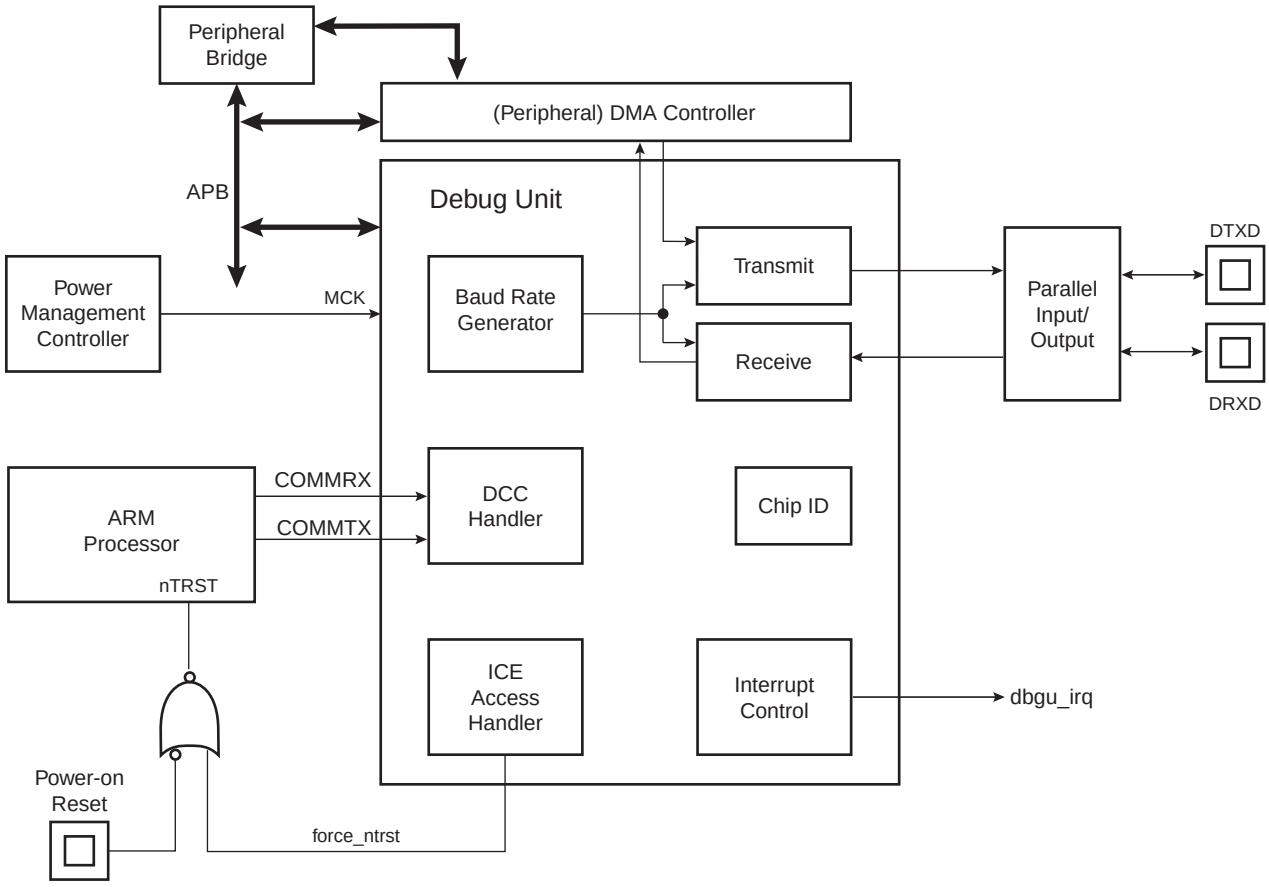
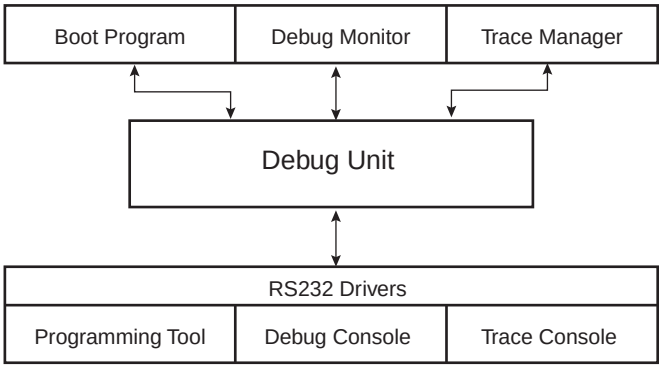


Table 42-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 42-2. Debug Unit Application Example



42.4 Product Dependencies

42.4.1 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 42-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
DBGU	DRXD	PB30	A
DBGU	DTXD	PB31	A

42.4.2 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.

42.4.3 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 42-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

42.5 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.

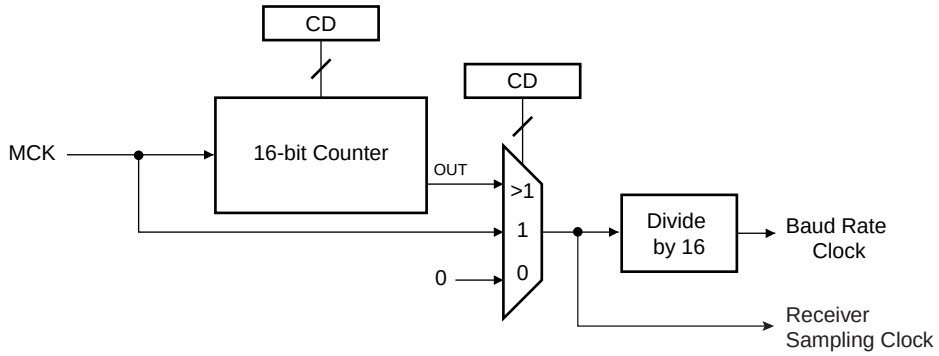
42.5.1 Baud Rate Generator

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 42-3. Baud Rate Generator



42.5.2 Receiver

42.5.2.1 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.

42.5.2.2 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 42-4. Start Bit Detection

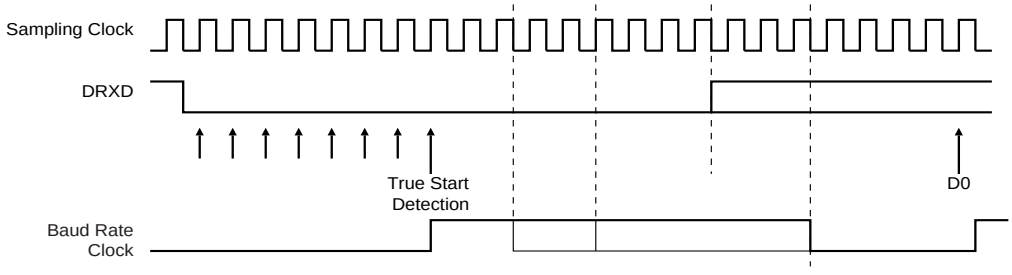
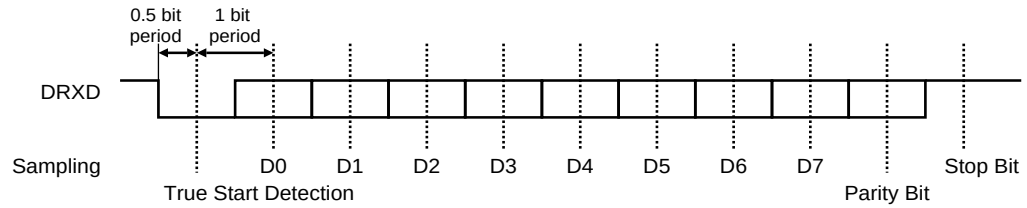


Figure 42-5. Character Reception

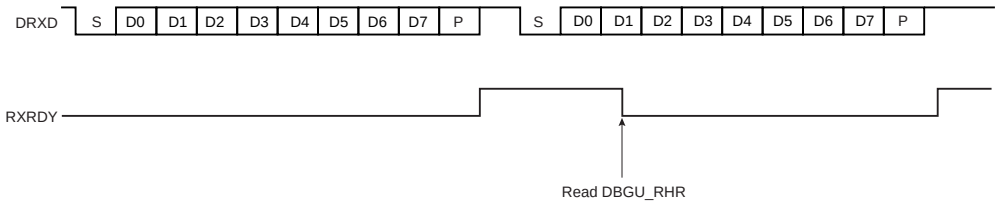
Example: 8-bit, parity enabled 1 stop



42.5.2.3 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.

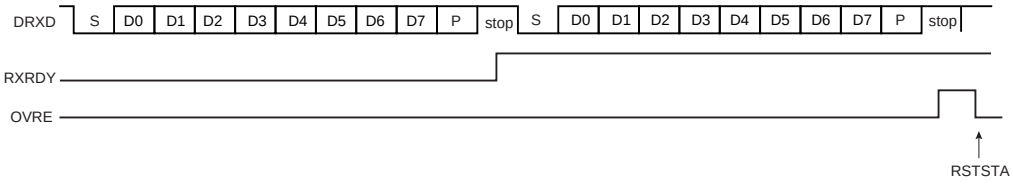
Figure 42-6. Receiver Ready



42.5.2.4 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.

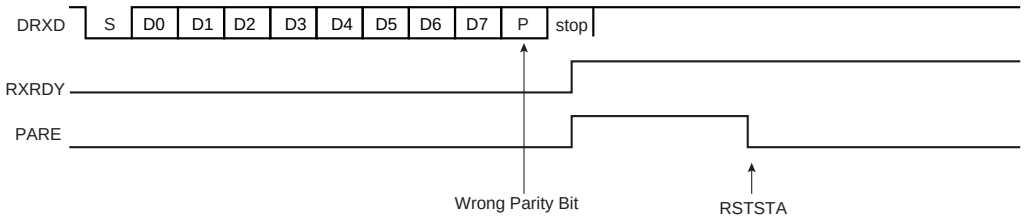
Figure 42-7. Receiver Overrun



42.5.2.5 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.

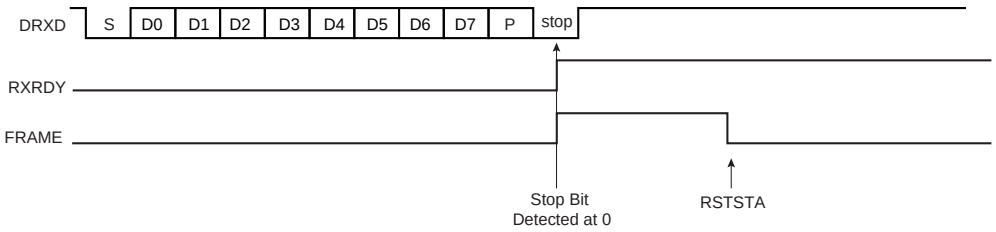
Figure 42-8. Parity Error



42.5.2.6 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 42-9. Receiver Framing Error



42.5.3 Transmitter

42.5.3.1 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.

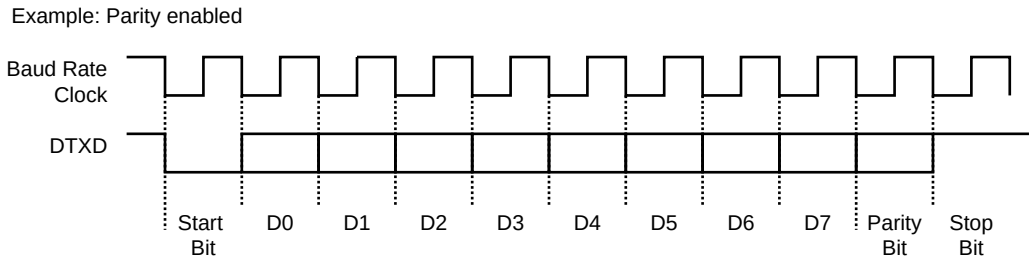
The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters 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.

42.5.3.2 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 42-10.Character Transmission

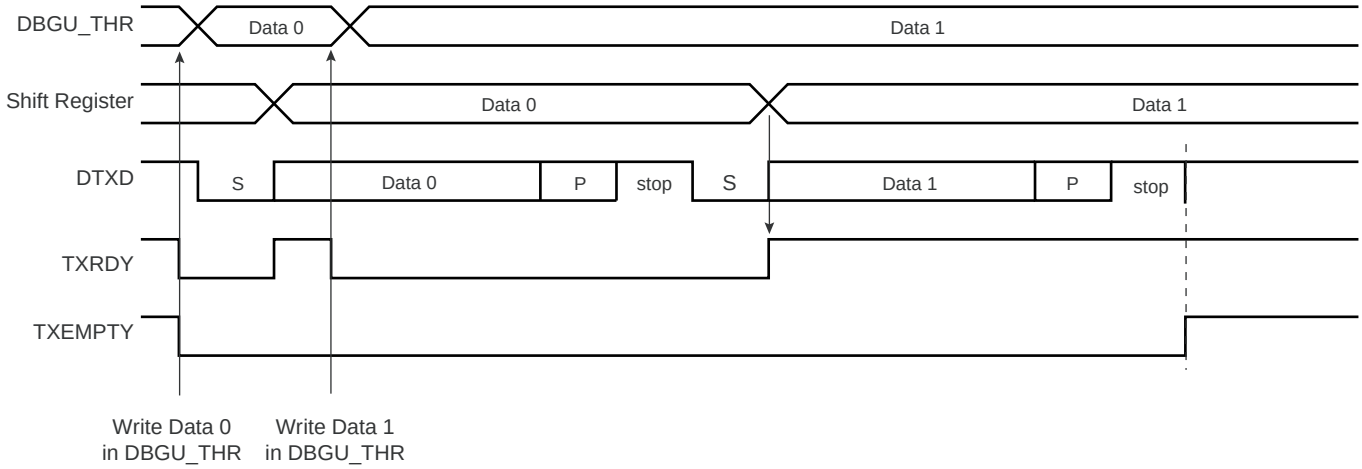


42.5.3.3 Transmitter Control

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 42-11.Transmitter Control



42.5.4 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.

42.5.5 Test Modes

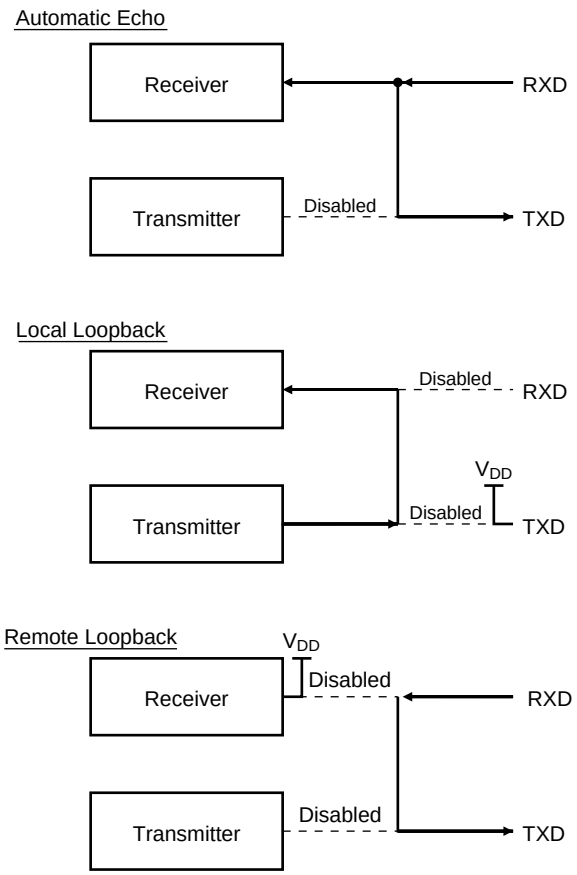
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

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 42-12.Test Modes



42.5.6 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.

42.5.7 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.

42.5.8 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.

42.6 Debug Unit (DBGU) User Interface

Table 42-3. 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	–	–	–

42.6.1 Debug Unit Control Register

Name: DBGU_CR

Address: 0xFFFFEE00

Access: Write-only

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

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME and OVRE in the DBGU_SR.

42.6.2 Debug Unit Mode Register

Name: DBGU_MR

Address: 0xFFFFEE04

Access: Read-write

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

- **PAR: Parity Type**

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

42.6.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Address: 0xFFFFEE08

Access: Write-only

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

42.6.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Address: 0xFFFFEE0C

Access: Write-only

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

42.6.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Address: 0xFFFFEE10

Access: Read-only

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

42.6.6 Debug Unit Status Register

Name: DBGU_SR

Address: 0xFFFFEE14

Access: Read-only

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

42.6.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Address: 0xFFFFEE18

Access: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

42.6.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Address: 0xFFFFEE1C

Access: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

42.6.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Address: 0xFFFFEE20

Access: Read-write

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

- **CD: Clock Divisor**

Value	Name	Description
0	DISABLED	DBGU Disabled
1	MCK	MCK
2 to 65535	–	MCK / (CD x 16)

42.6.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

Address: 0xFFFFEE40

Access: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**

Values depend upon the version of the device.

- **EPROC: Embedded Processor**

Value	Name	Description
6	CA5	Cortex-A5

- **NVPSIZ: Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6	–	Reserved
7	128K	128 Kbytes
8	–	Reserved
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

- **NVPSIZ2: Second Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6		Reserved
7	128K	128 Kbytes
8	–	Reserved
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

- **SRAMSIZ: Internal SRAM Size**

Value	Name	Description
0	–	Reserved
1	1K	1 Kbytes
2	2K	2 Kbytes
3	6K	6 Kbytes
4	112K	112 Kbytes
5	4K	4 Kbytes
6	80K	80 Kbytes
7	160K	160 Kbytes
8	8K	8 Kbytes
9	16K	16 Kbytes
10	32K	32 Kbytes
11	64K	64 Kbytes
12	128K	128 Kbytes
13	256K	256 Kbytes
14	96K	96 Kbytes
15	512K	512 Kbytes

- **ARCH: Architecture Identifier**

Value	Name	Description
0xA5	ATSAMA5xx	ATSAMA5xx 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.

42.6.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Address: 0xFFFFEE44

Access: Read-only

31	30	29	28	27	26	25	24
EXID							

23	22	21	20	19	18	17	16
EXID							

15	14	13	12	11	10	9	8
EXID							

7	6	5	4	3	2	1	0
EXID							

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

42.6.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Address: 0xFFFFEE48

Access: Read-write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor’s TAP controller is driven by the power_on_reset signal.

1 = NTRST of the ARM processor’s TAP controller is held low.

43. Universal Asynchronous Receiver Transmitter (UART)

43.1 Description

The Universal Asynchronous Receiver Transmitter (UART) 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 a DMA controller permits packet handling for these tasks with processor time reduced to a minimum.

43.2 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

43.3 Block Diagram

Figure 43-1. UART Functional Block Diagram

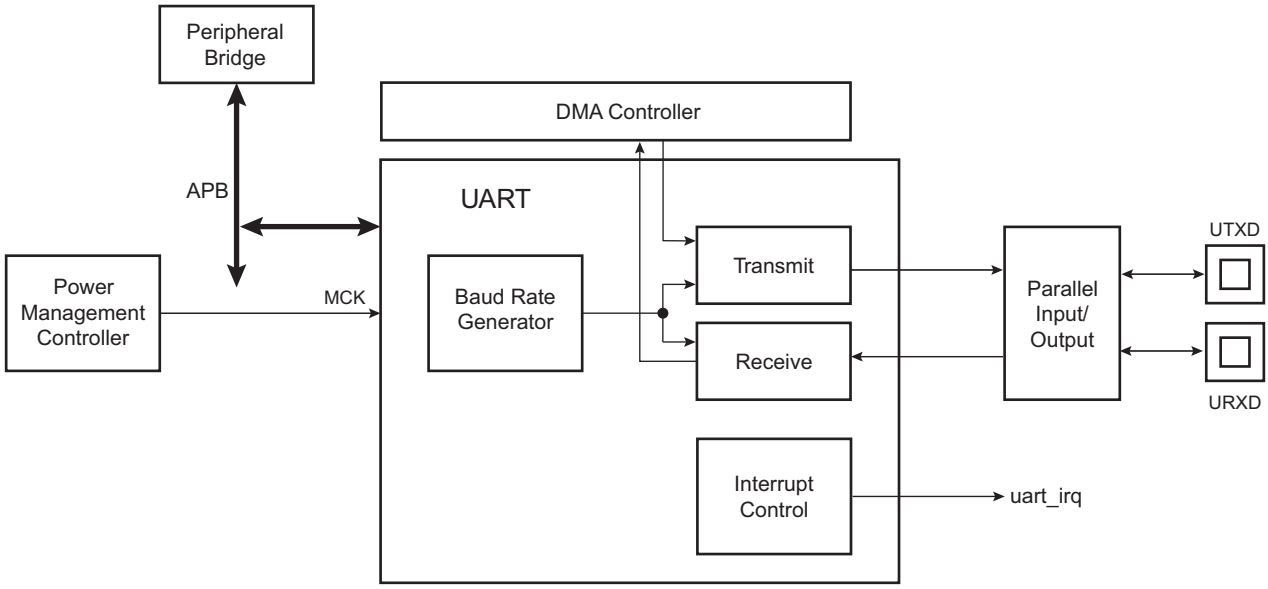


Table 43-1. UART Pin Description

Pin Name	Description	Type
URXD	UART Receive Data	Input
UTXD	UART Transmit Data	Output

43.4 Product Dependencies

43.4.1 I/O Lines

The UART pins are multiplexed with PIO lines. The user must first configure the corresponding PIO Controller to enable I/O line operations of the UART.

Table 43-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
UART0	URXD0	PC29	A
UART0	UTXD0	PC30	A
UART1	URXD1	PA30	B
UART1	UTXD1	PA31	B

43.4.2 Power Management

The UART clock can be controlled through the Power Management Controller (PMC). In this case, the user must first configure the PMC to enable the UART clock. Usually, the peripheral identifier used for this purpose is 1.

43.4.3 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.

43.5 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.

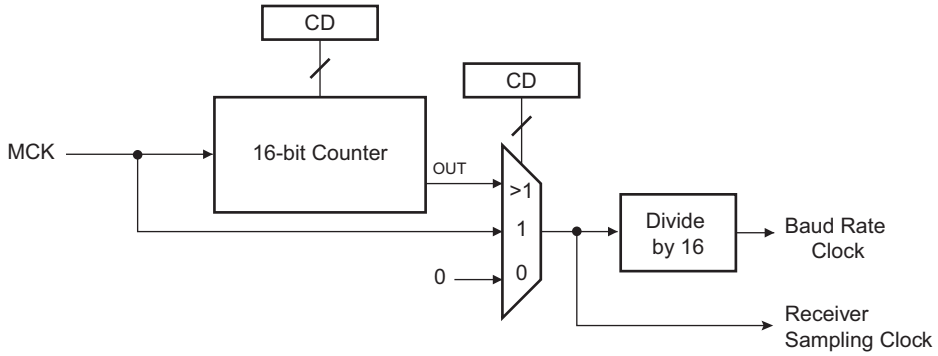
43.5.1 Baud Rate Generator

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 43-2. Baud Rate Generator



43.5.2 Receiver

43.5.2.1 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 receiver can be put in reset state by writing UART_CR with the bit RSTRX at 1. In this case, 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.

43.5.2.2 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 seven 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 detecting the falling edge of the start bit.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 43-3. Start Bit Detection

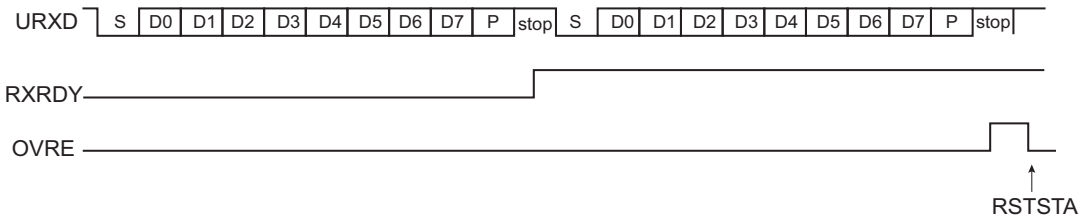
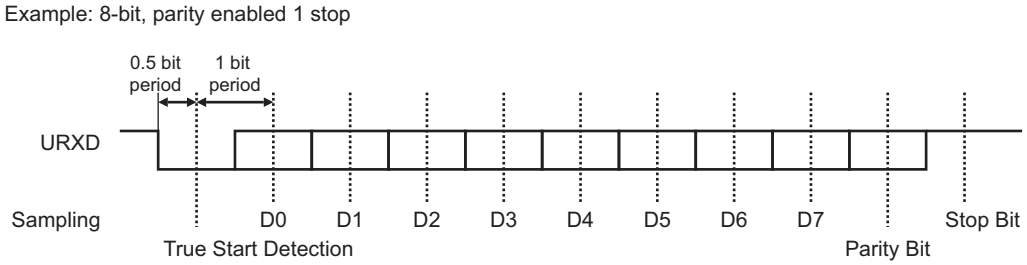


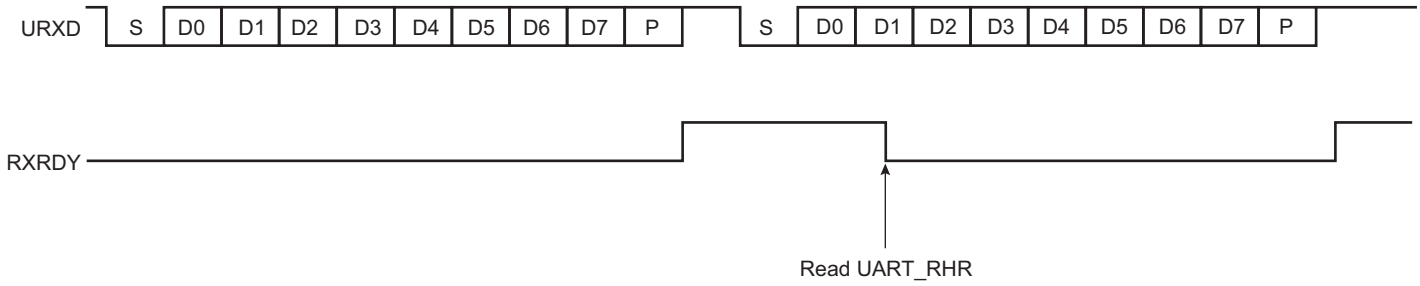
Figure 43-4. Character Reception



43.5.2.3 Receiver Ready

When a complete character is received, it is transferred to the Receive Holding register (UART_RHR) and the RXRDY status bit in the Status register (UART_SR) is set. The bit RXRDY is automatically cleared when UART_RHR is read.

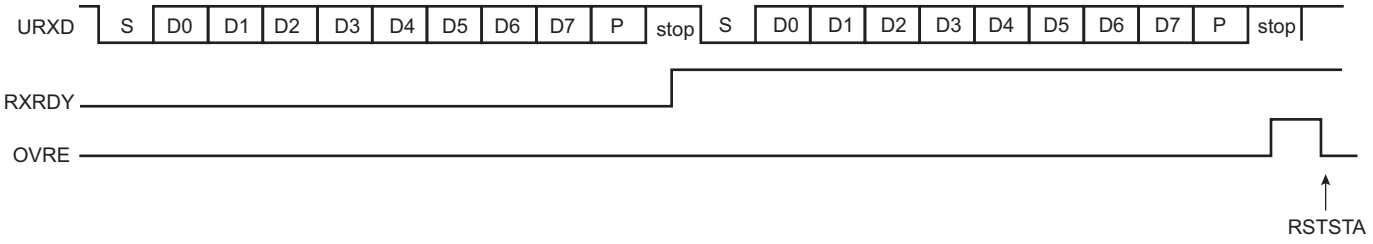
Figure 43-5. Receiver Ready



43.5.2.4 Receiver Overrun

The OVRE status bit in UART_SR is set 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. OVRE is cleared when the software writes a 1 to the bit RSTSTA (Reset Status) in UART_CR.

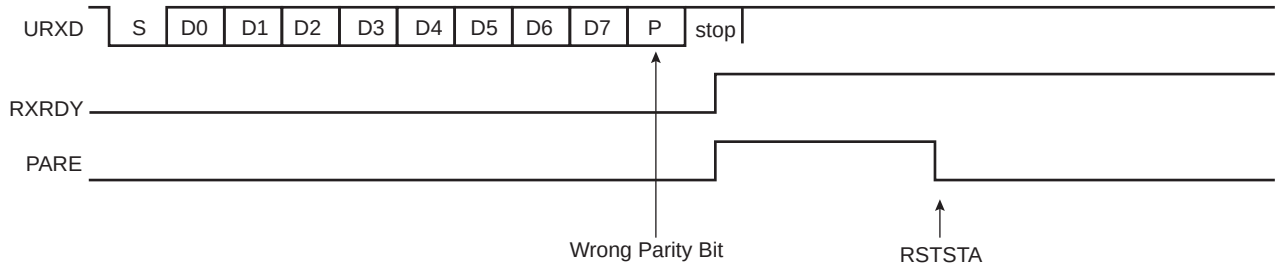
Figure 43-6. Receiver Overrun



43.5.2.5 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 the Mode register (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 RXRDY is set. The parity bit is cleared when 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.

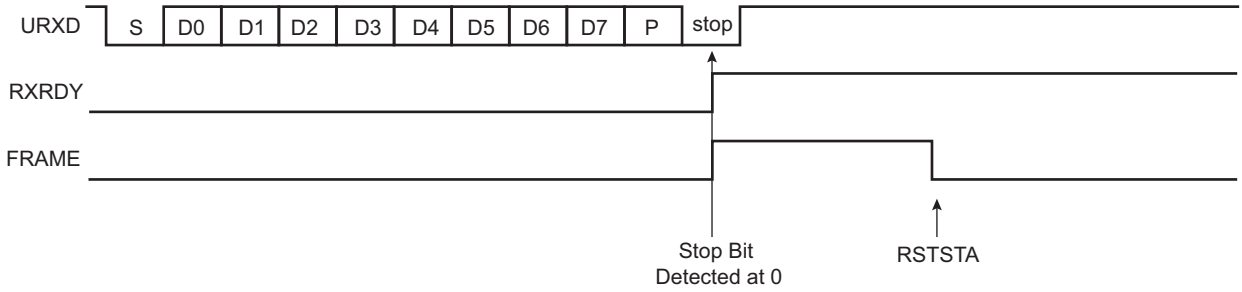
Figure 43-7. Parity Error



43.5.2.6 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 43-8. Receiver Framing Error



43.5.3 Transmitter

43.5.3.1 Transmitter Reset, Enable and Disable

After device reset, the UART transmitter is disabled and must be enabled before being used. The transmitter is enabled by writing 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.

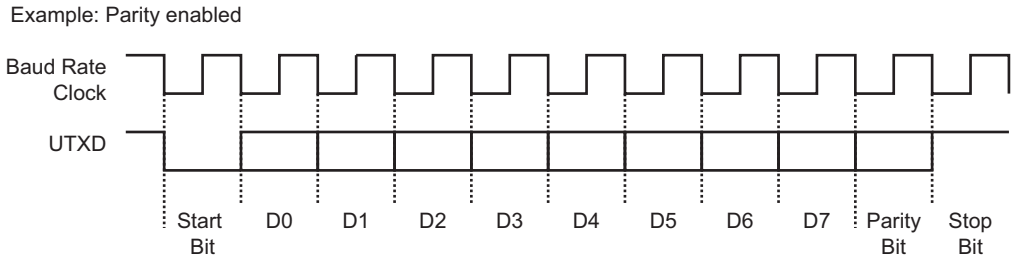
The programmer can disable the transmitter by writing UART_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters 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.

43.5.3.2 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 UART_MR 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 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 43-9. Character Transmission

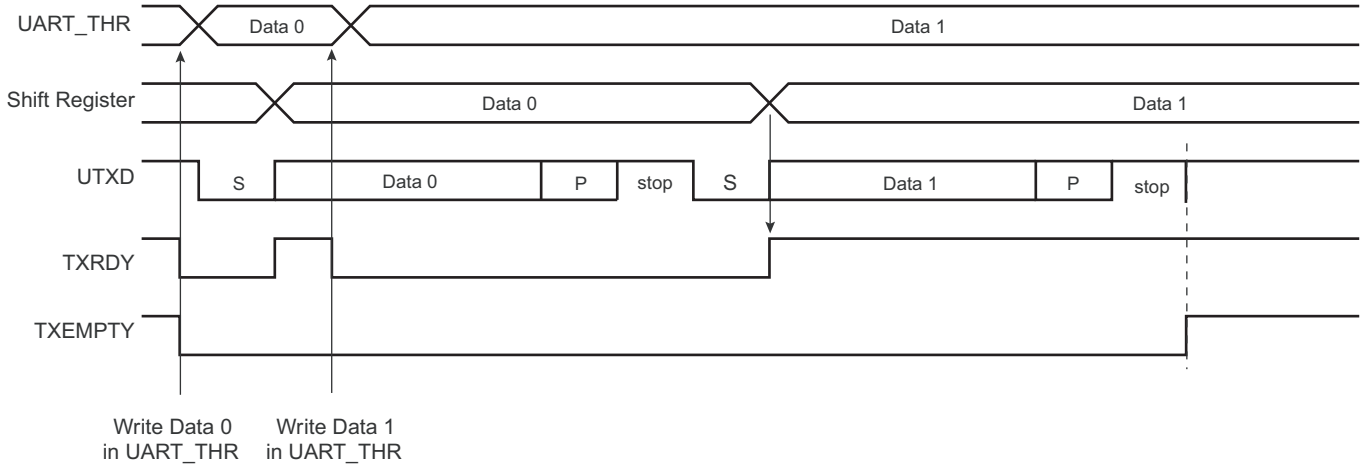


43.5.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in 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 43-10.Transmitter Control



43.5.4 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.

43.5.5 Test Modes

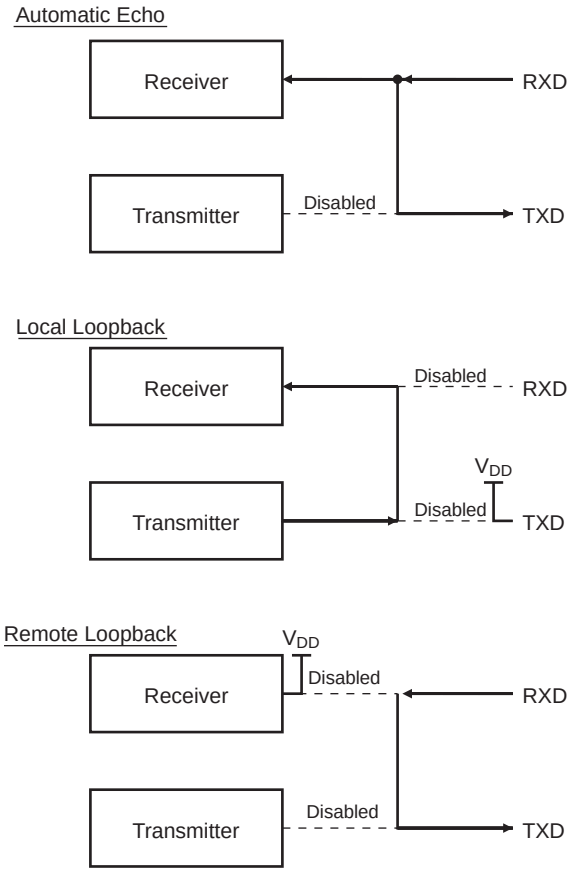
The UART supports three test modes. These modes of operation are programmed by using the CHMODE field in UART_MR.

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 43-11.Test Modes



43.6 Universal Asynchronous Receiver Transmitter (UART) User Interface

Table 43-3. 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	–	–	–
0x0040 - 0x00E8	Reserved	–	–	–
0x00EC - 0x00FC	Reserved	–	–	–

43.6.1 UART Control Register

Name: UART_CR

Address: 0xF0024000 (0), 0xF8028000 (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.

43.6.2 UART Mode Register

Name: UART_MR

Address: 0xF0024004 (0), 0xF8028004 (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

43.6.3 UART Interrupt Enable Register

Name: UART_IER

Address: 0xF0024008 (0), 0xF8028008 (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

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **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**

43.6.4 UART Interrupt Disable Register

Name: UART_IDR

Address: 0xF002400C (0), 0xF802800C (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

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **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**

43.6.5 UART Interrupt Mask Register

Name: UART_IMR

Address: 0xF0024010 (0), 0xF8028010 (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

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **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**

43.6.6 UART Status Register

Name: UART_SR

Address: 0xF0024014 (0), 0xF8028014 (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.

43.6.7 UART Receiver Holding Register

Name: UART_RHR

Address: 0xF0024018 (0), 0xF8028018 (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.

43.6.8 UART Transmit Holding Register

Name: UART_THR

Address: 0xF002401C (0), 0xF802801C (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.

43.6.9 UART Baud Rate Generator Register

Name: UART_BRGR

Address: 0xF0024020 (0), 0xF8028020 (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)

44. Universal Synchronous Asynchronous Receiver Transceiver (USART)

44.1 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 RS485and 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.

44.2 Embedded Characteristics

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Over-sampling Receiver Frequency
  - 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
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two DMA Controller Channels (DMAC)
- Offers Buffer Transfer without Processor Intervention

44.3 Block Diagram

Figure 44-1. USART Block Diagram

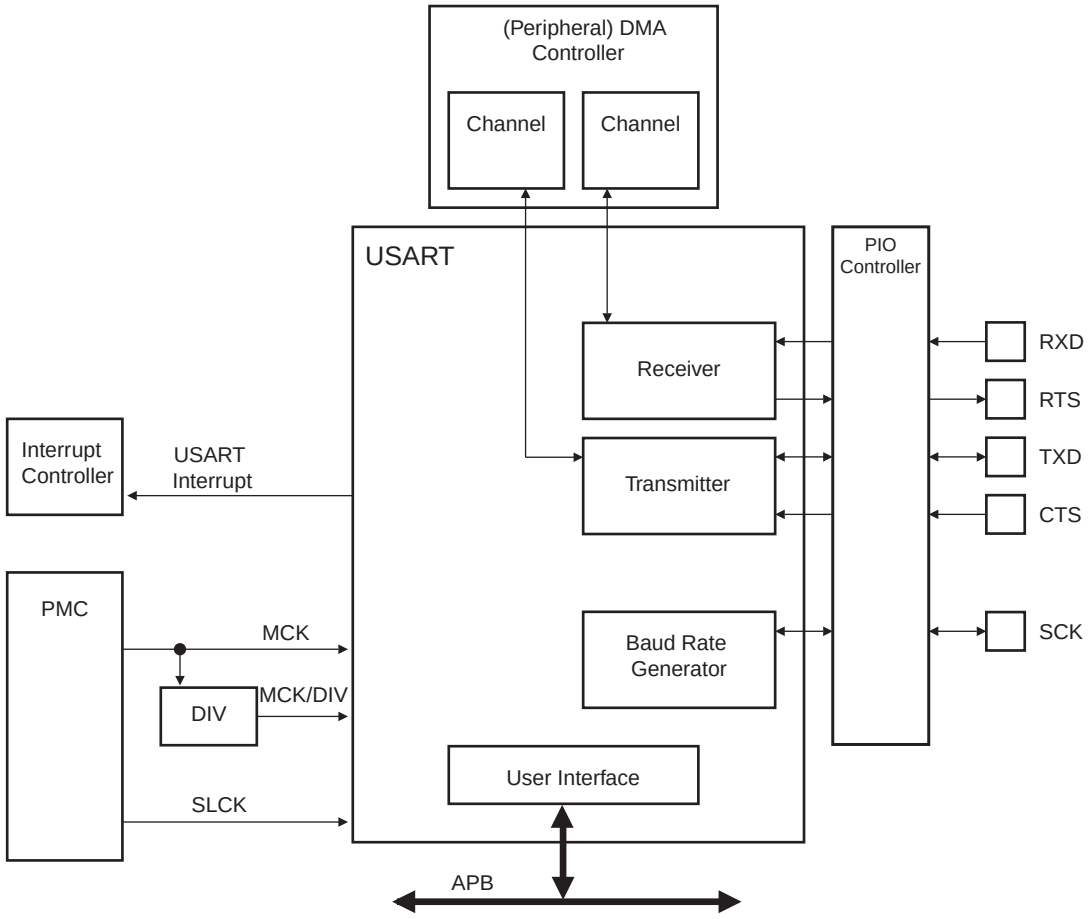
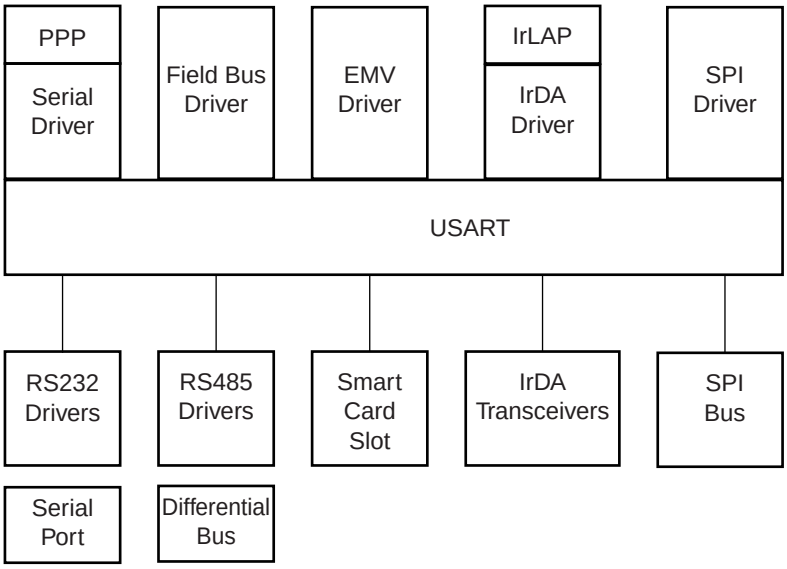


Table 44-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
RXD	RXD	MOSI	MISO
TXD	TXD	MISO	MOSI
RTS	RTS	–	CS
CTS	CTS	CS	–

44.4 Application Block Diagram

Figure 44-2. Application Block Diagram



44.5 I/O Lines Description

Table 44-2. 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

44.6 Product Dependencies

44.6.1 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 44-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PD15	A
USART0	RTS0	PD16	A
USART0	RXD0	PD17	A
USART0	SCK0	PD14	A
USART0	TXD0	PD18	A
USART1	CTS1	PB26	A
USART1	RTS1	PB27	A
USART1	RXD1	PB28	A
USART1	SCK1	PB25	A
USART1	TXD1	PB29	A
USART2	CTS2	PE23	B
USART2	RTS2	PE24	B
USART2	RXD2	PE25	B
USART2	SCK2	PE20	B
USART2	TXD2	PE26	B
USART3	CTS3	PE16	B
USART3	RTS3	PE17	B
USART3	RXD3	PE18	B
USART3	SCK3	PE15	B
USART3	TXD3	PE19	B

44.6.2 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.

44.6.3 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 44-4. Peripheral IDs

Instance	ID
USART0	12
USART1	13
USART2	14
USART3	15

44.7 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit, inverted data.
- 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
- Test modes
  - Remote loopback, local loopback, automatic echo

44.7.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

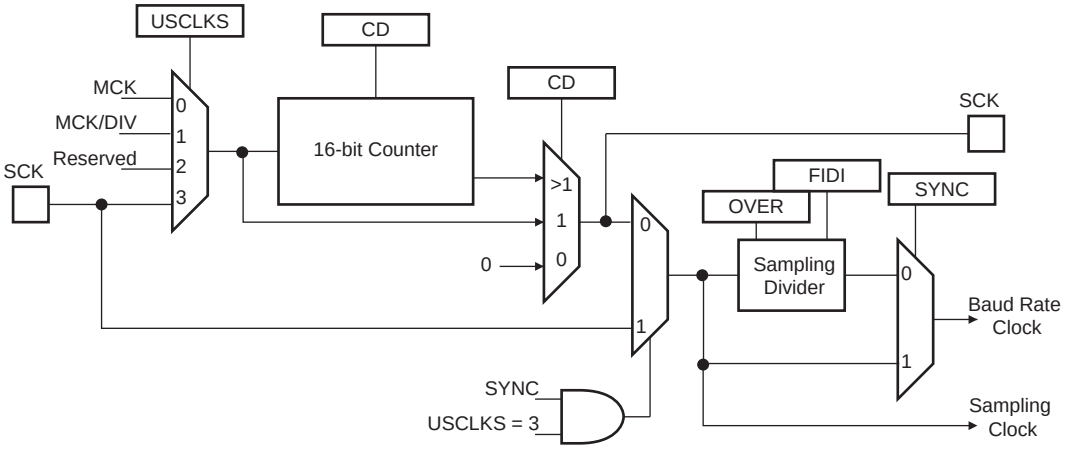
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- 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 a zero is written to CD, the Baud Rate Generator does not generate any clock. If a one is written to CD, 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 3 times lower than MCK in USART mode, or 6 times lower in SPI mode.

Figure 44-3. Baud Rate Generator



44.7.1.1 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 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 the US_MR.

If OVER is set, the receiver sampling is eight times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The baud rate is calculated as per the following formula:

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that the OVER bit is set.

Baud Rate Calculation Example

Table 44-5 shows calculations of CD to obtain a baud rate at 38,400 bit/s for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 44-5. Baud Rate Example (OVER = 0)

Source Clock (MHz)	Expected Baud Rate (Bit/s)	Calculation Result	CD	Actual Baud Rate (Bit/s)	Error
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%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

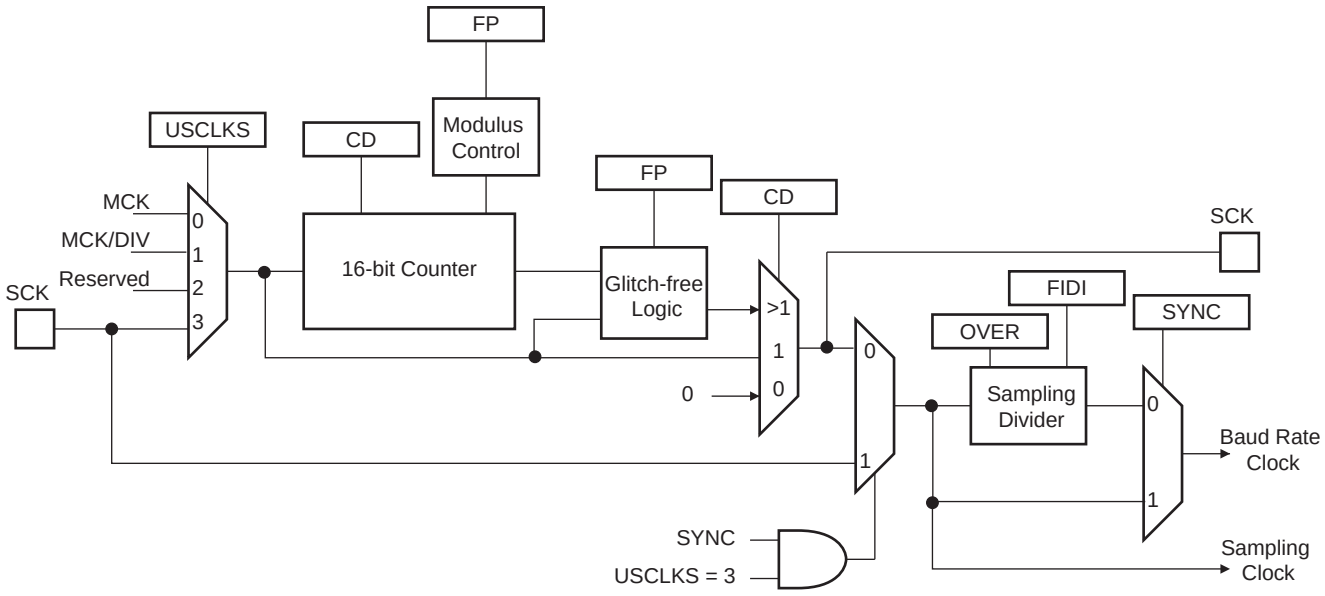
44.7.1.2 Fractional Baud Rate in Asynchronous Mode

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 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 44-4. Fractional Baud Rate Generator



44.7.1.3 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 the 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 3 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 toMCK/3 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.

44.7.1.4 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 44-6.

Table 44-6. 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 44-7.

Table 44-7. 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 44-8 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 44-8. 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

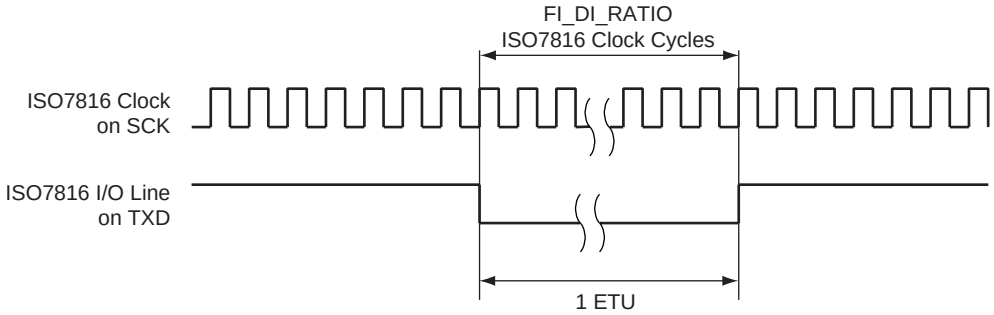
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the US_MR is first divided by the value programmed in the field CD in the US_BRGR. The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

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 44-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 44-5. Elementary Time Unit (ETU)



44.7.2 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 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 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 the 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.

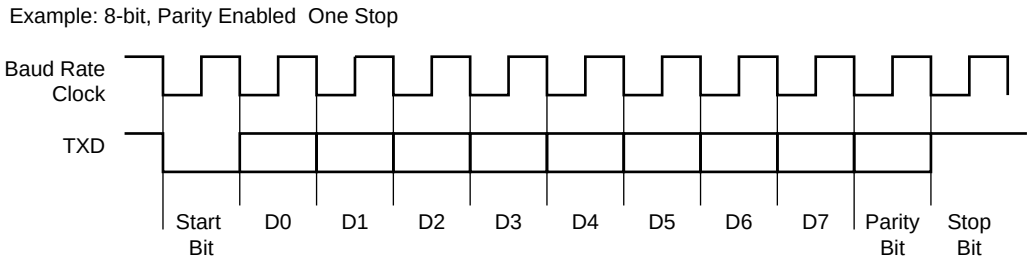
44.7.3 Synchronous and Asynchronous Modes

44.7.3.1 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 the 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 the US_MR. The 1.5 stop bit is supported in asynchronous mode only.

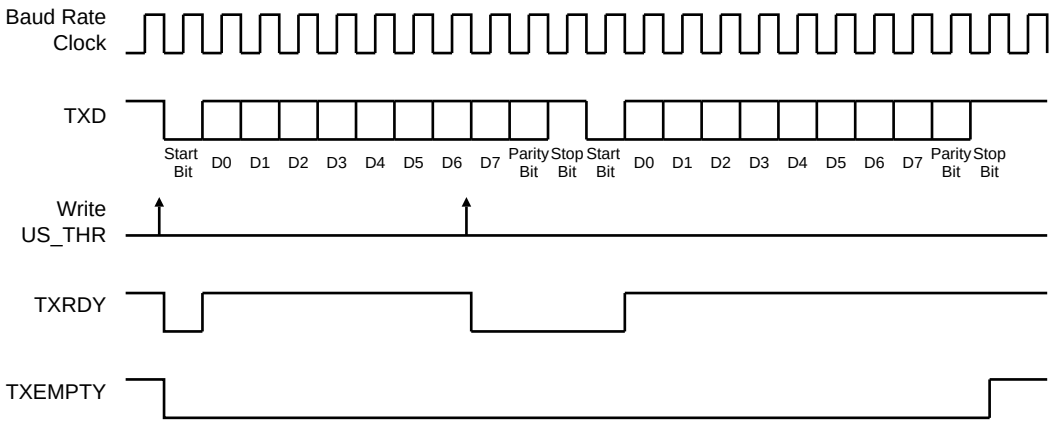
Figure 44-6. Character Transmit



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.

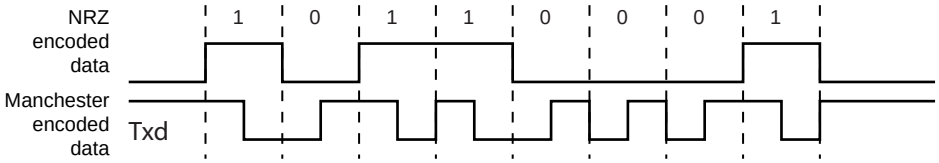
Figure 44-7. Transmitter Status



44.7.3.2 Manchester Encoder

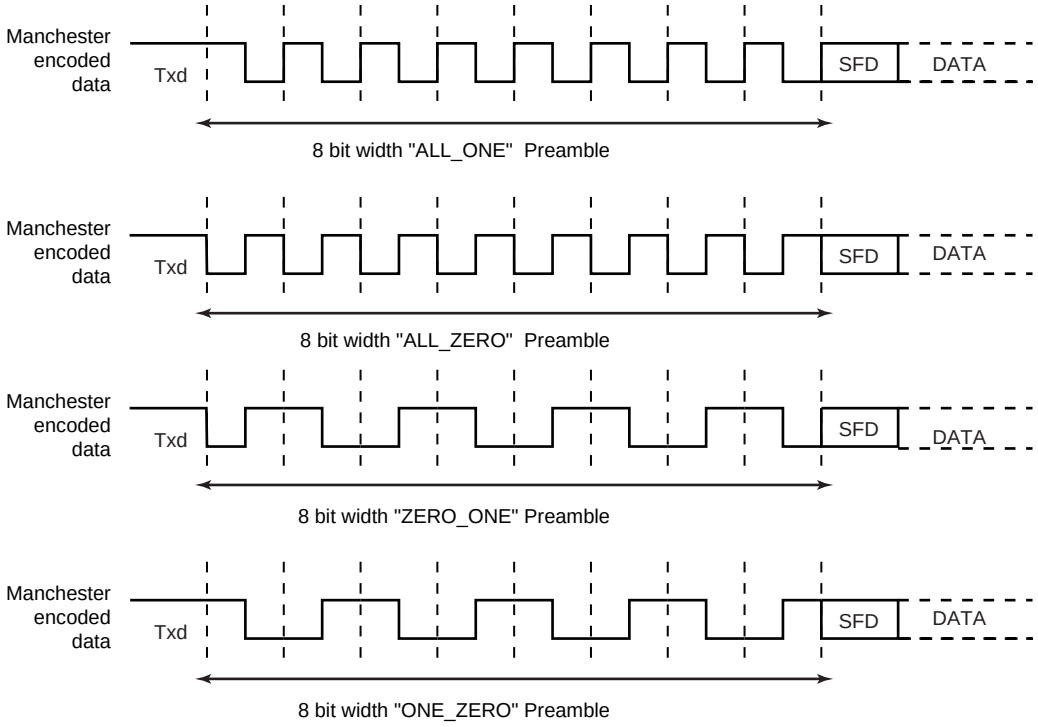
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero 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 44-8 illustrates this coding scheme.

Figure 44-8. NRZ to Manchester Encoding



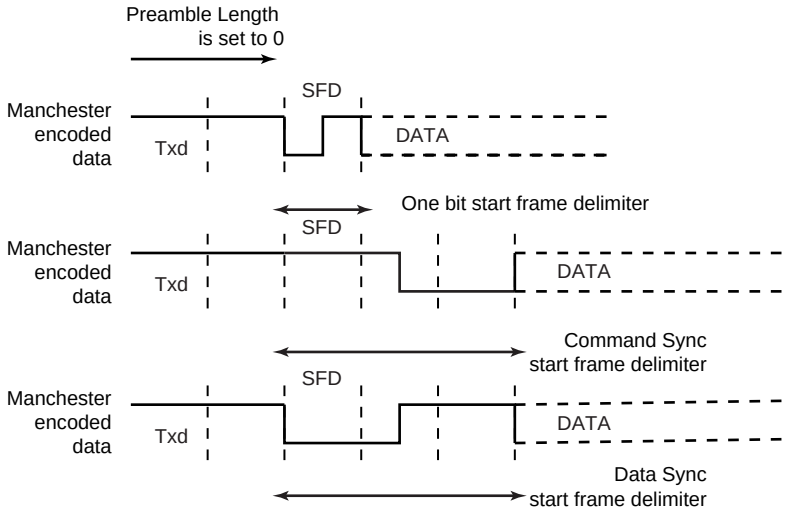
The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a predefined pattern with a programmable length from 1 to 15 bit times. If the preamble 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 44-9 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 44-9. 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 44-10 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 three 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 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 the US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR. The USART character format is modified and includes sync information.

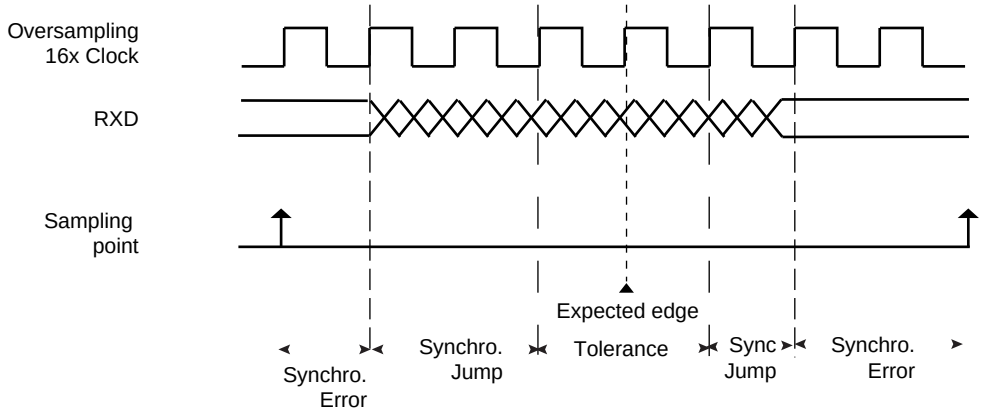
Figure 44-10.Start Frame Delimiter



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 44-11.Bit Resynchronization



44.7.3.3 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 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 44-12 and Figure 44-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 44-12.Asynchronous Start Detection

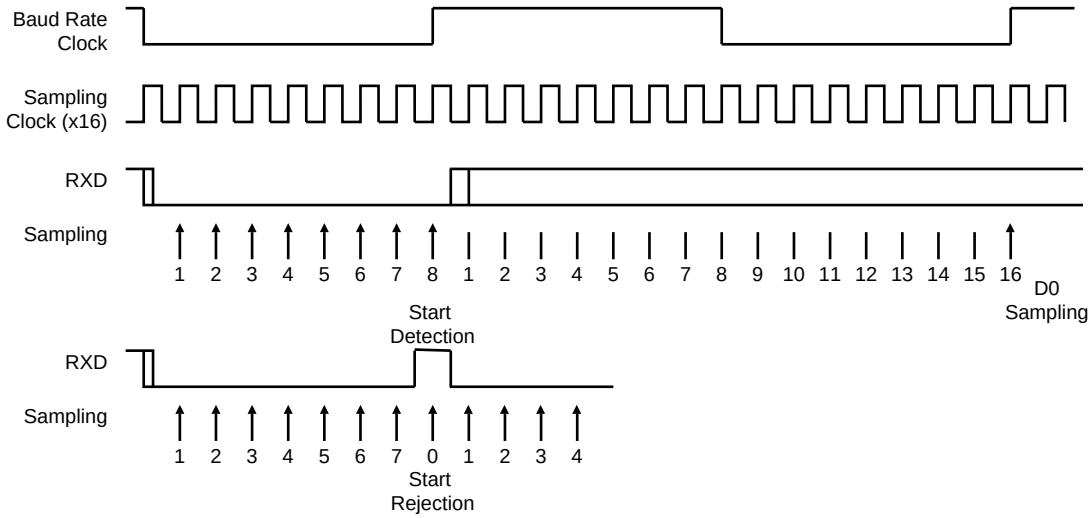
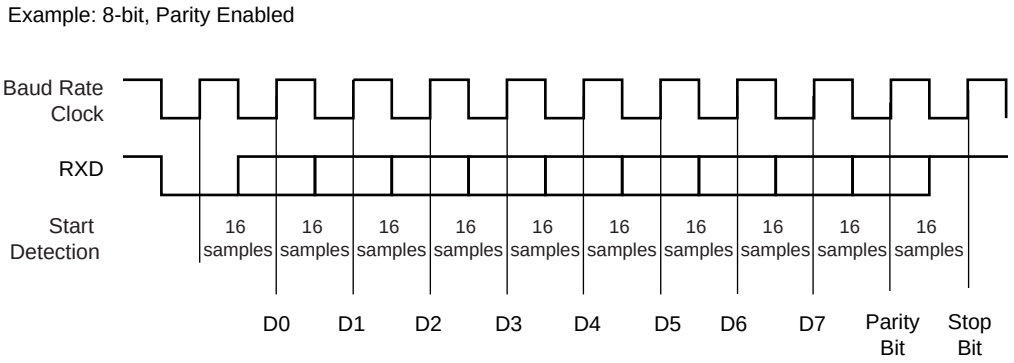


Figure 44-13.Asynchronous Character Reception



44.7.3.4 Manchester Decoder

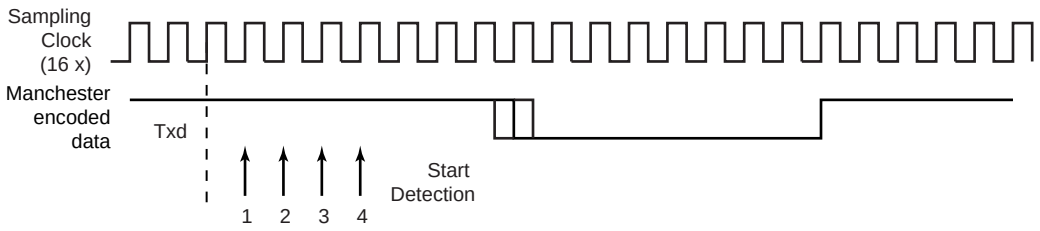
When the MAN field in the US_MR is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

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 44-9 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 44-14. 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 44-14.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 resynchronizes 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 44-15 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 the US_CSR is raised. It is cleared by writing the US_CR with the RSTSTA bit to 1. See Figure 44-16 for an example of Manchester error detection during data phase.

Figure 44-15.Preamble Pattern Mismatch

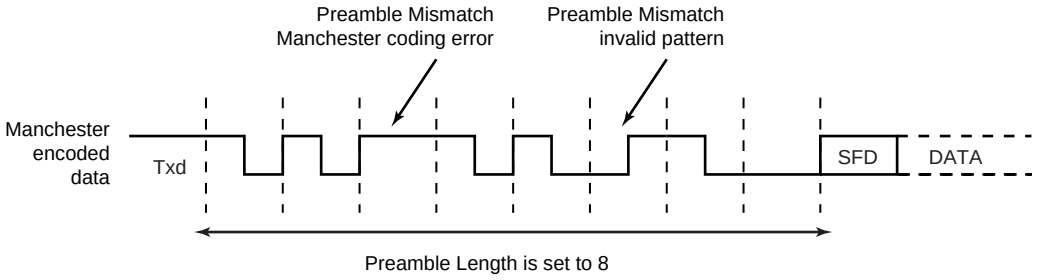
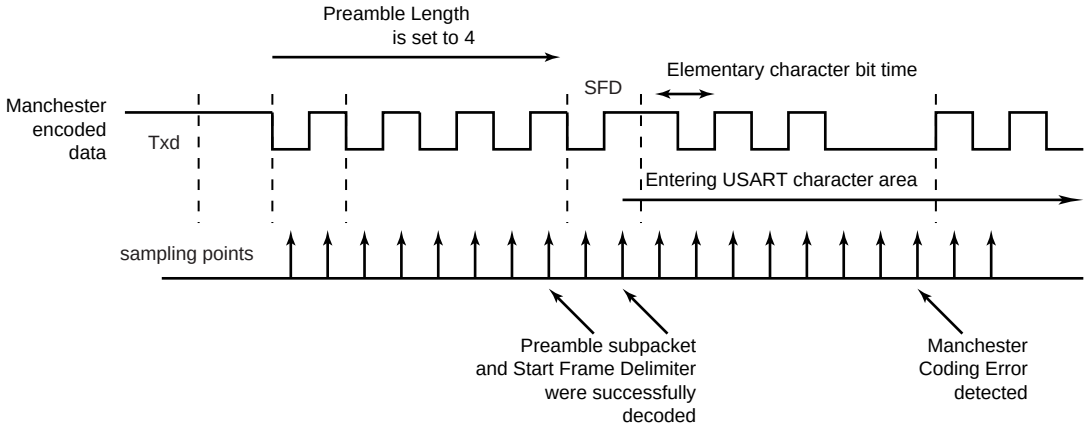


Figure 44-16.Manchester Error Flag



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.

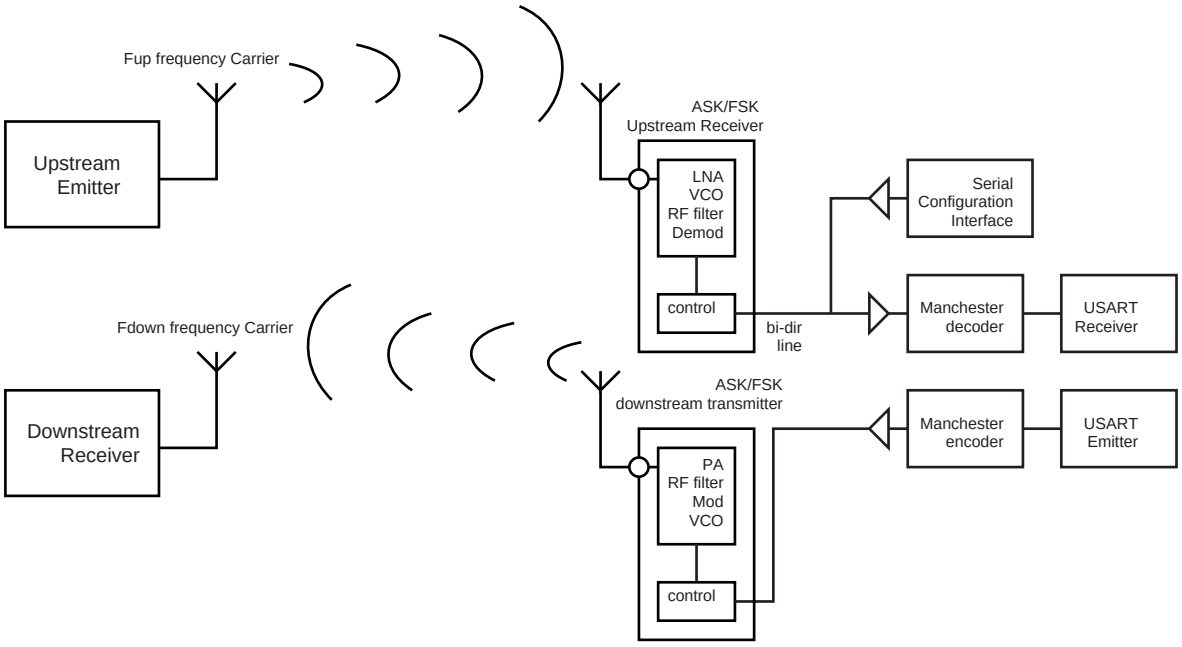
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

44.7.3.5 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 44-17.

Figure 44-17.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 44-18 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 44-19.

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 44-18.ASK Modulator Output

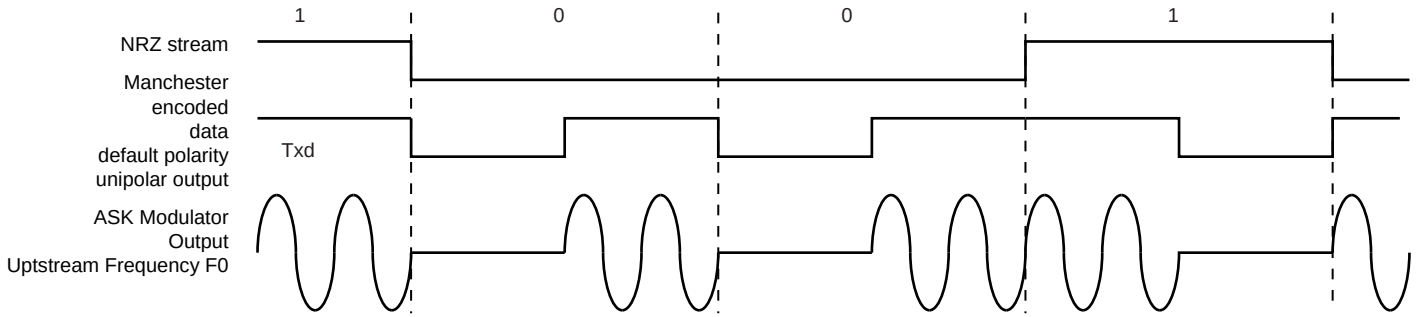
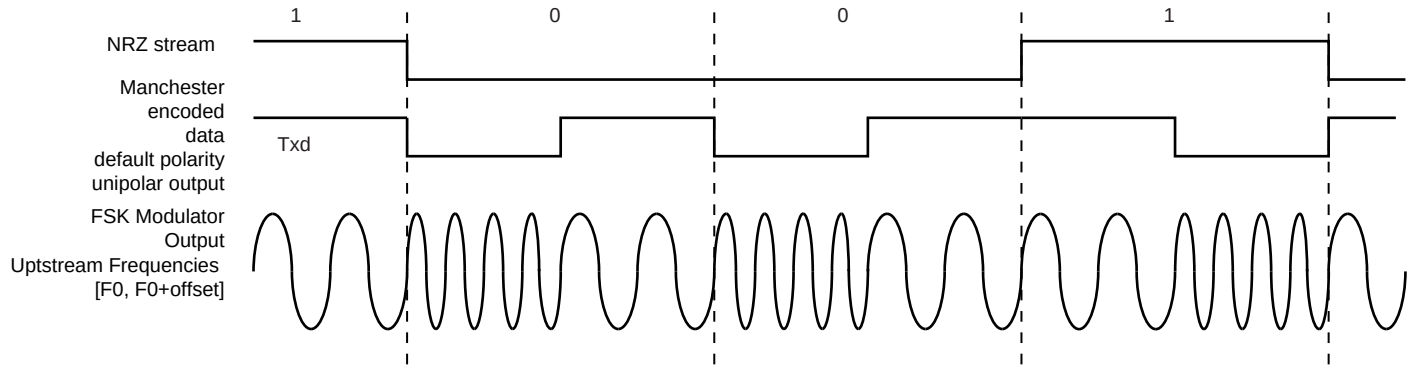


Figure 44-19.FSK Modulator Output



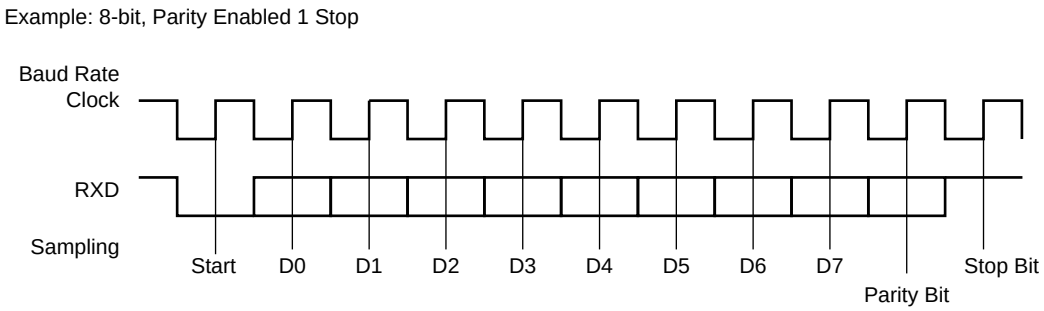
44.7.3.6 Synchronous Receiver

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 44-20 illustrates a character reception in synchronous mode.

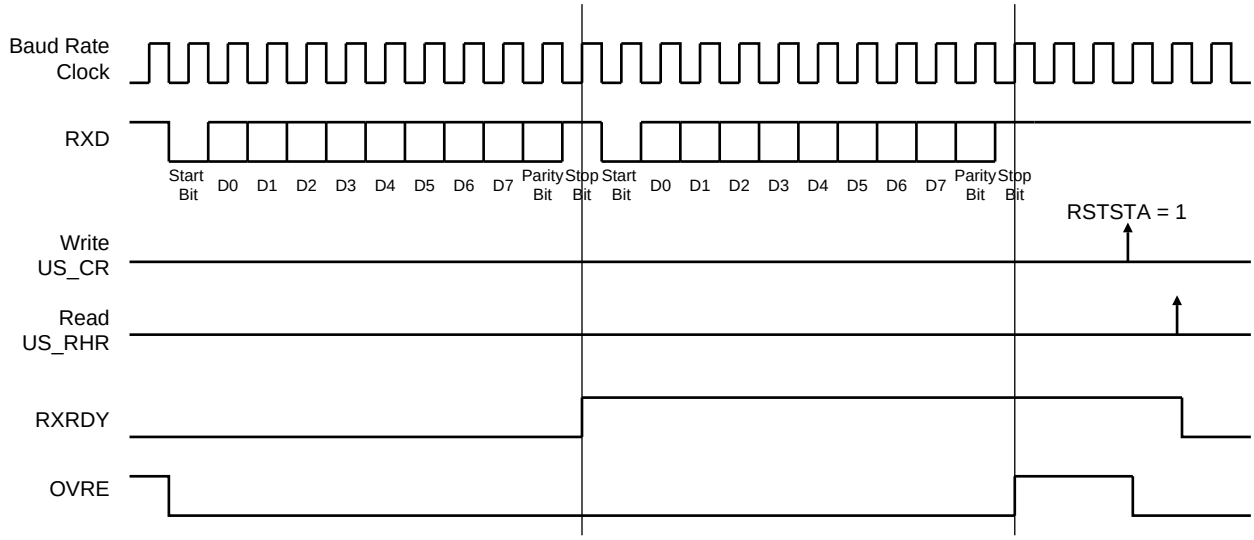
Figure 44-20.Synchronous Mode Character Reception



44.7.3.7 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the 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 US_CR with the RSTSTA (Reset Status) bit to 1.

Figure 44-21.Receiver Status



44.7.3.8 Parity

The USART supports five parity modes selected by writing to the PAR field in the US_MR. The PAR field also enables the Multidrop mode, see “Multidrop Mode” on page 1385. 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.

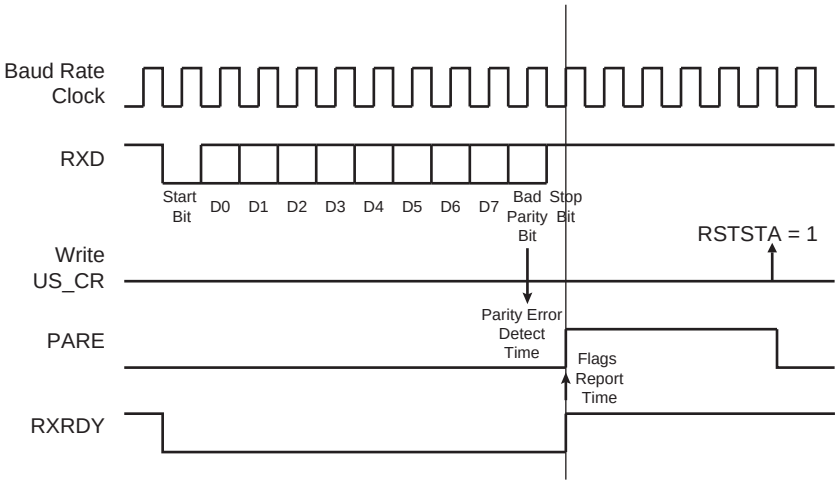
Table 44-9 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits to 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 44-9. Parity Bit Examples

Character	Hexadecimal	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 US_CSR. The PARE bit can be cleared by writing the US_CR with the RSTSTA bit to 1. Figure 44-22 illustrates the parity bit status setting and clearing.

Figure 44-22.Parity Error



44.7.3.9 Multidrop Mode

If the value 0x6 or 0x07 is written to the PAR field in the US_MR, 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 a one is written to the SENTA bit in the US_CR.

To handle parity error, the PARE bit is cleared when a one is written to the RSTSTA bit in the US_CR.

The transmitter sends an address byte (parity bit set) when SENDA is written to in the US_CR. In this case, the next byte written to the US_THR is transmitted as an address. Any character written in the US_THR without having written the command SENDA is transmitted normally with the parity to 0.

44.7.3.10 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 written 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 44-23, 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 44-23.Timeguard Operations

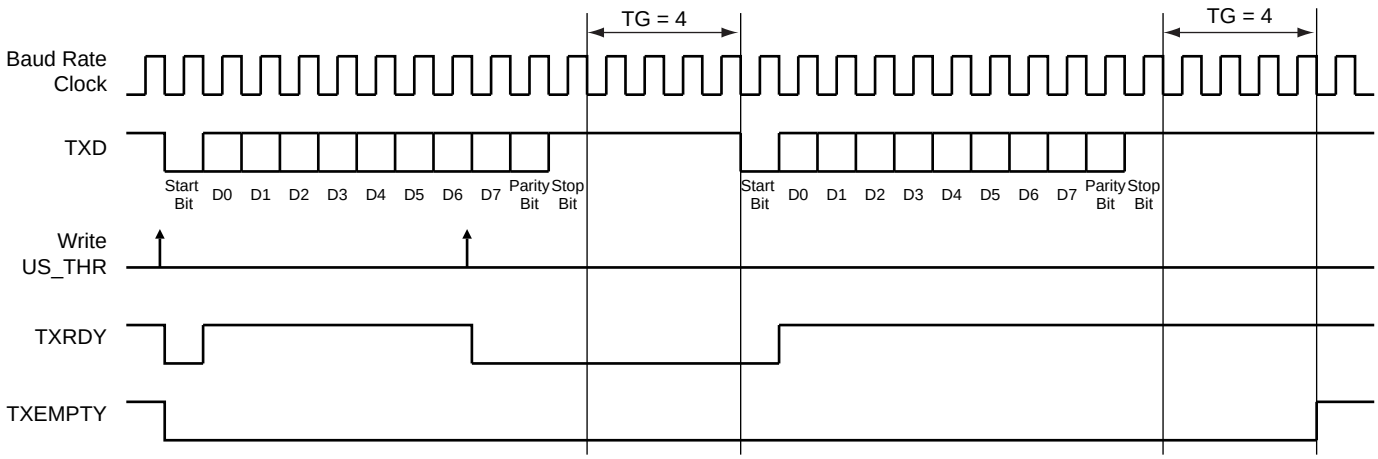


Table 44-10 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the baud rate.

Table 44-10. Maximum Timeguard Length Depending on Baud Rate

Baud Rate (Bit/s)	Bit Time (µs)	Timeguard (ms)
1,200	833	212.50
9,600	104	26.56
14,400	69.4	17.71
19,200	52.1	13.28
28,800	34.7	8.85
38,400	26	6.63
56,000	17.9	4.55
57,600	17.4	4.43
115,200	8.7	2.21

44.7.3.11 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 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 written to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in the US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the US_CSR rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing a one to the STTTO (Start Time-out) bit the US_CR. 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 a one to the RETTO (Reload and Start Time-out) bit in the US_CR. 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 44-24 shows the block diagram of the Receiver Time-out feature.

Figure 44-24.Receiver Time-out Block Diagram

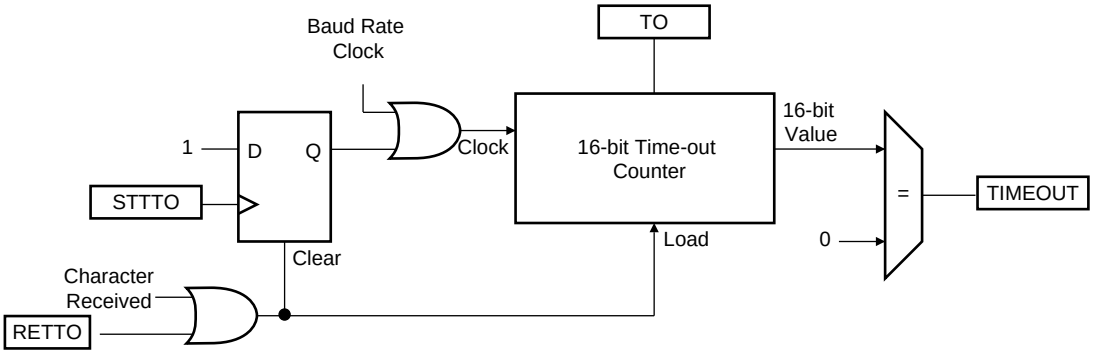


Table 44-11 gives the maximum time-out period for some standard baud rates.

Table 44-11. Maximum Time-out Period

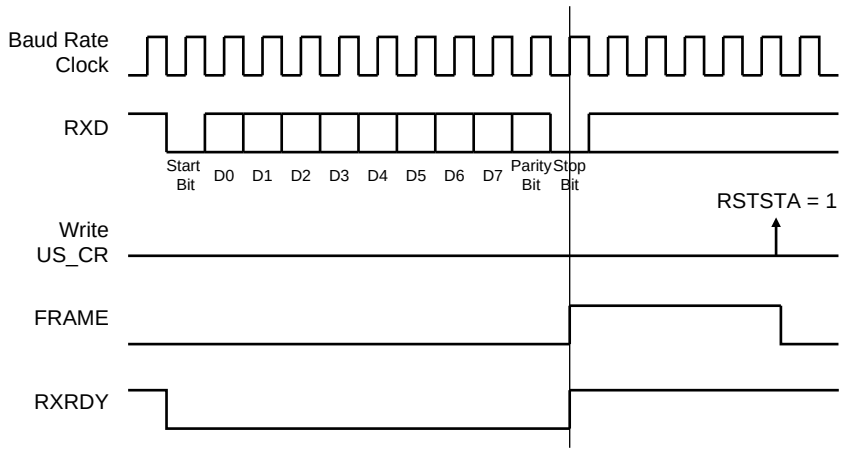
Baud Rate (Bit/s)	Bit Time (µs)	Time-out (ms)
600	1,667	109,225
1,200	833	54,613
2,400	417	27,306
4,800	208	13,653
9,600	104	6,827
14,400	69	4,551
19,200	52	3,413
28,800	35	2,276
38,400	26	1,704
56,000	18	1,170
57,600	17	1,138
200,000	5	328

44.7.3.12 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 44-25.Framing Error Status



44.7.3.13 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.

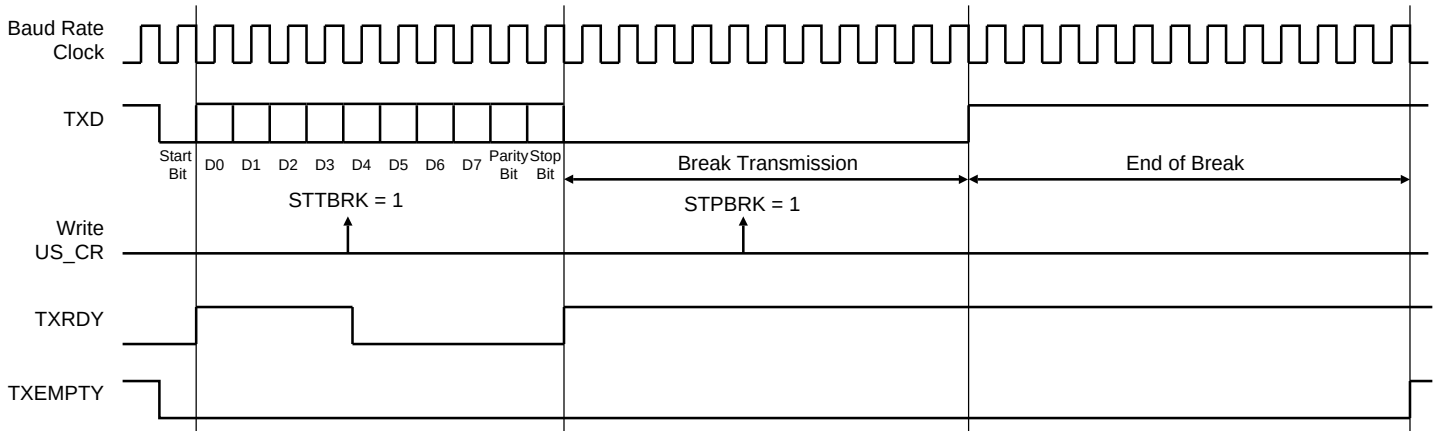
Writing US_CR with both STTBRK and STPBRK bits to 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 44-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 44-26.Break Transmission



44.7.3.14 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.

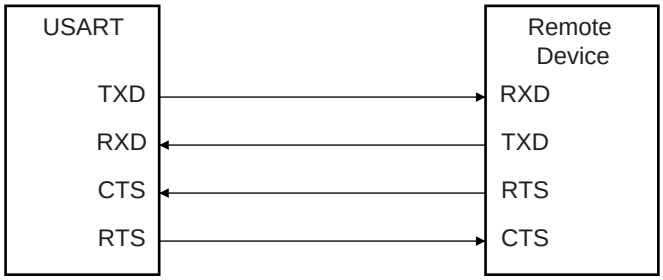
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA to 1.

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.

44.7.3.15 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 44-27.

Figure 44-27.Connection with a Remote Device for Hardware Handshaking

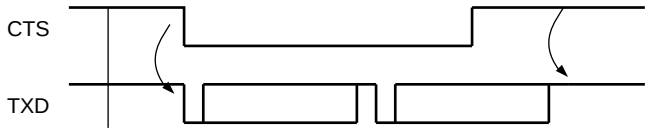


Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

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 44-28 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 44-28.Transmitter Behavior when Operating with Hardware Handshaking



44.7.4 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.

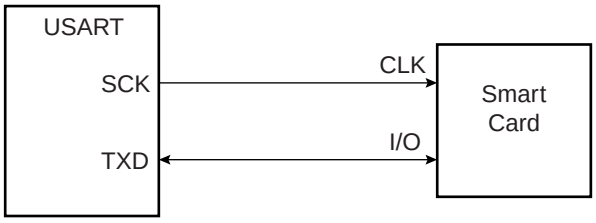
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

44.7.4.1 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 1372).

The USART connects to a smart card as shown in Figure 44-29. 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 44-29.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 1407 and “PAR: Parity Type” on page 1408.

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.

44.7.4.2 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 44-30.

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 44-31. 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 44-30.T = 0 Protocol without Parity Error

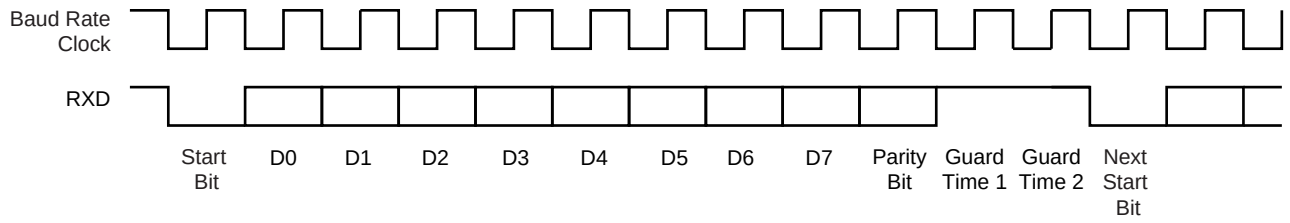
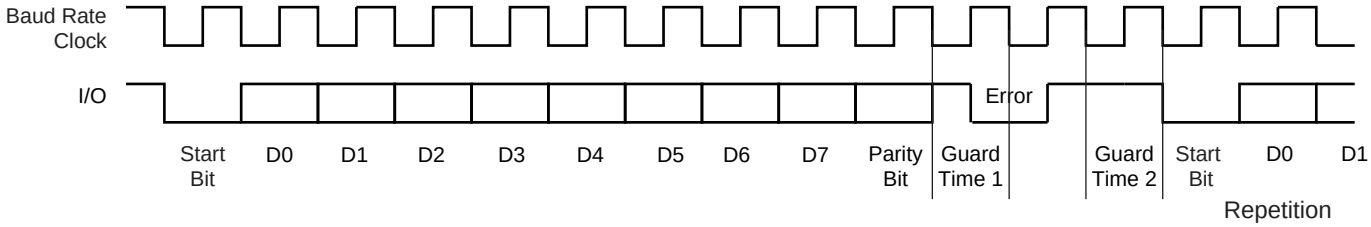


Figure 44-31.T = 0 Protocol with Parity Error



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 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 RSTIT 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 US_MR. The maximum number of NACKs 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 US_CSR is set.

44.7.4.3 Protocol T = 1

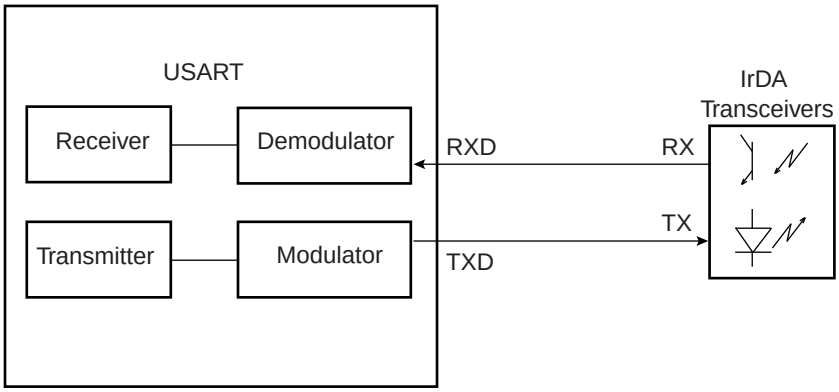
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the US_CSR.

44.7.5 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 44-32. 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 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 44-32.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

44.7.5.1 IrDA Modulation

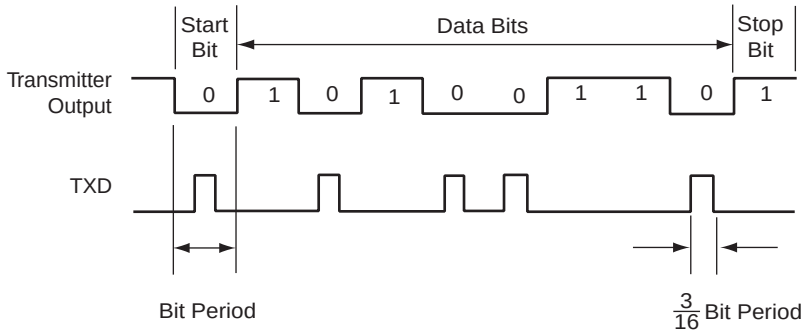
For baud rates up to and including 115.2 Kb/s, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 44-12.

Table 44-12. 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 44-33 shows an example of character transmission.

Figure 44-33.IrDA Modulation



44.7.5.2 IrDA Baud Rate

Table 44-13 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 44-13. IrDA Baud Rate Error

Peripheral Clock	Baud Rate (Bit/s)	CD	Baud Rate Error	Pulse Time (µs)
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

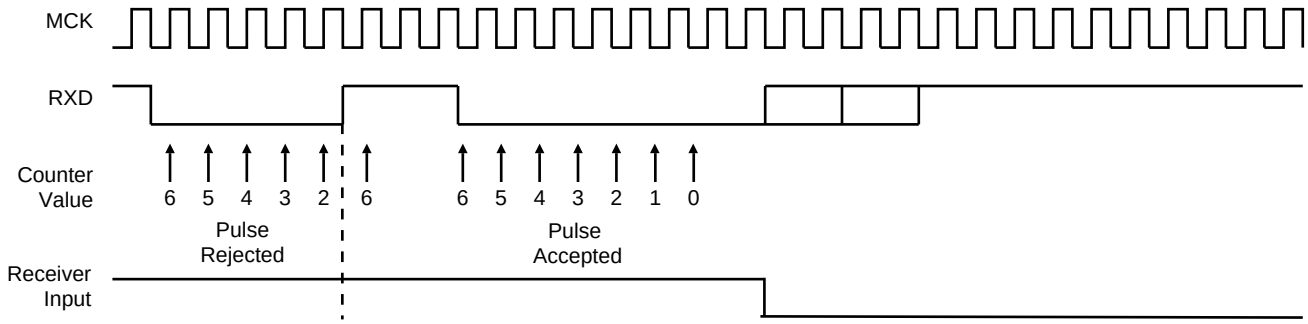
44.7.5.3 IrDA Demodulator

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 44-34 illustrates the operations of the IrDA demodulator.

Figure 44-34.IrDA Demodulator Operations



The programmed value in the US_IF register must always meet the following criteria:

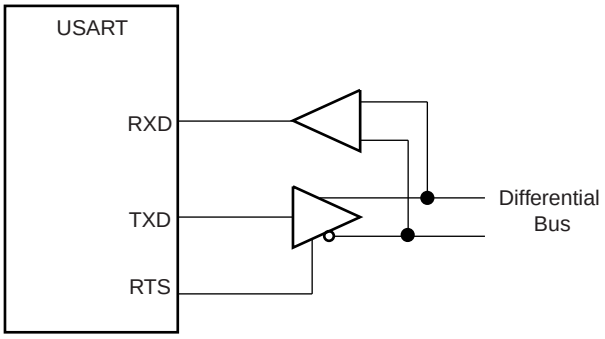
t_{MCK} * (IRDA_FILTER + 3) < 1.41 µs

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.

44.7.6 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 44-35.

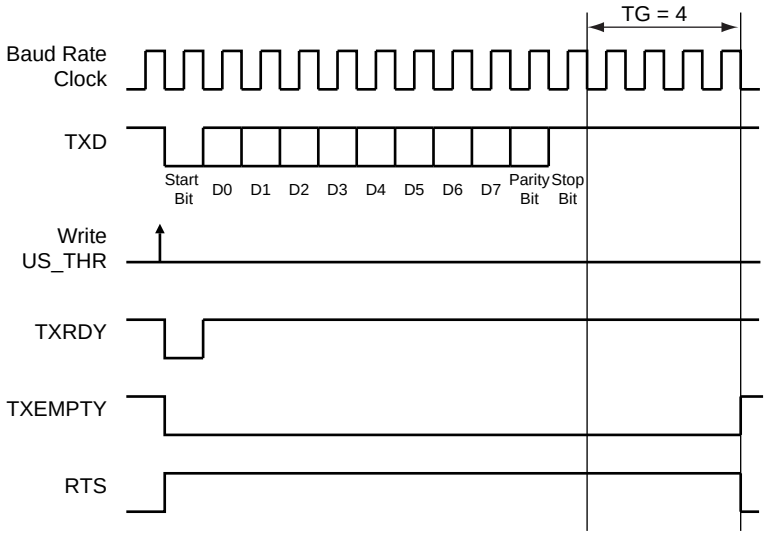
Figure 44-35.Typical Connection to a RS485 Bus



The USART is set in RS485 mode by writing the value 0x1 to the USART_MODE field in the Mode Register (US_MR).

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 44-36 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 44-36.Example of RTS Drive with Timeguard



44.7.7 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.

44.7.7.1 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 0xE to the USART_MODE field in the Mode Register (US_MR). 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 44.7.7.4).

44.7.7.2 Baud Rate

In SPI Mode, the baud rate generator operates in the same way as in USART synchronous mode: See “Baud Rate in Synchronous Mode or SPI Mode” on page 1374. 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 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 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.

44.7.7.3 Data Transfer

Up to nine 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 US_MR. The nine 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 US_MR. 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 44-14. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 44-37.SPI Transfer Format (CPHA = 1, 8 bits per transfer)

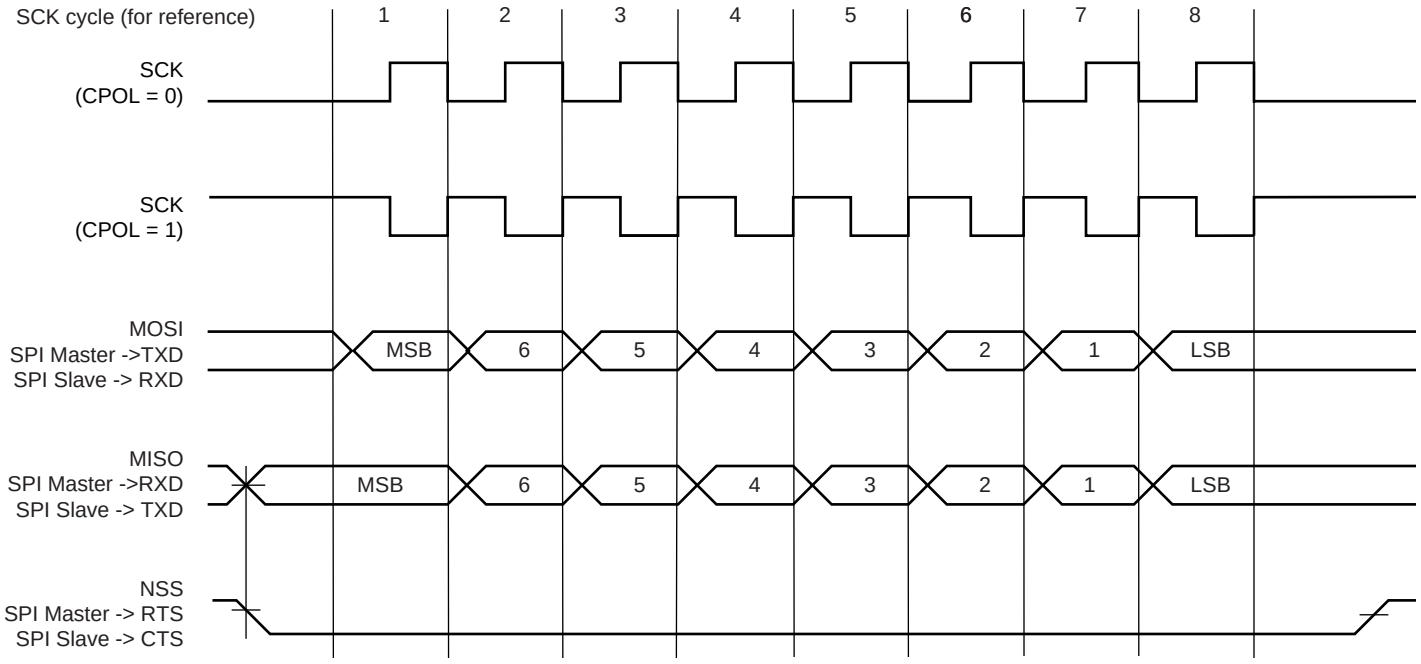
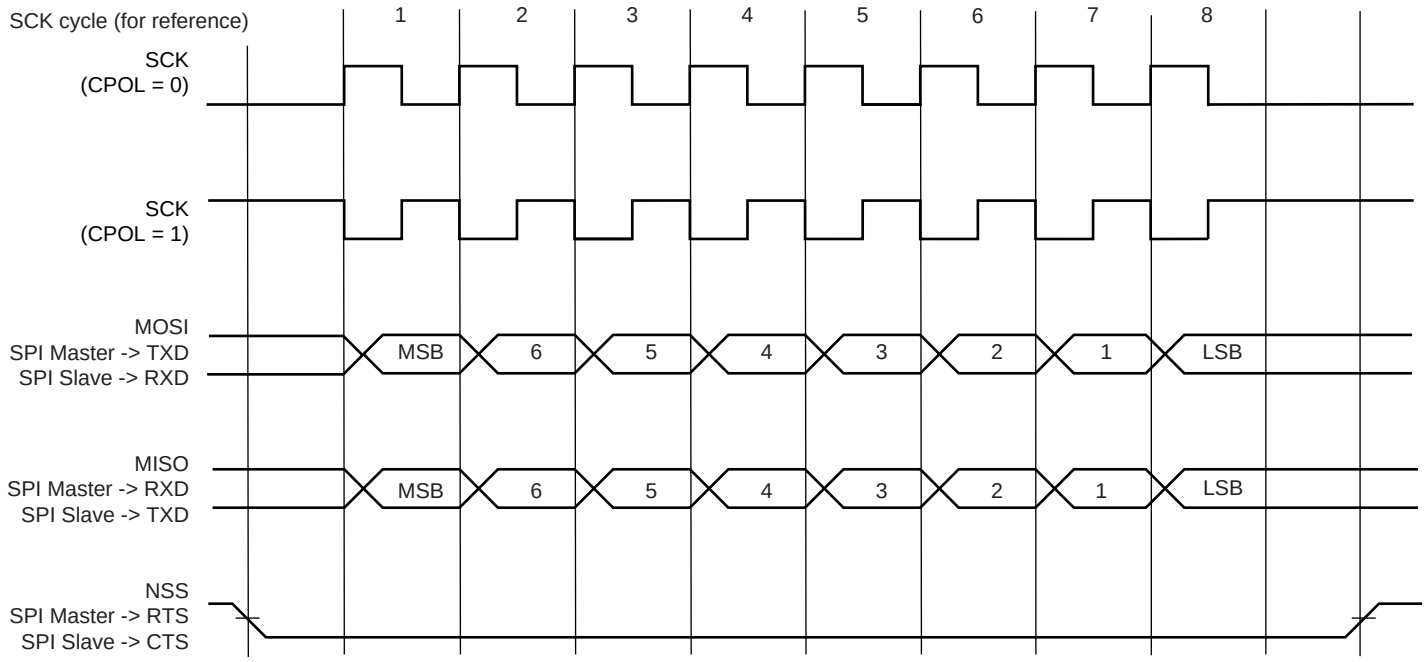


Figure 44-38.SPI Transfer Format (CPHA = 0, 8 bits per transfer)



44.7.7.4 Receiver and Transmitter Control

See “Receiver and Transmitter Control” on page 1375.

44.7.7.5 Character Transmission

The characters are sent by writing in the Transmit Holding Register (US_THR). An additional condition for transmitting a character can be added when the USART is configured in SPI master mode. In the USART_MR, the value configured on INACK field can prevent any character transmission (even if US_THR has been written) while the receiver side is not ready (character not read). When WRDBT equals 0, the character is transmitted whatever the receiver status. If WRDBT is set to 1, the transmitter waits for the receiver holding register to be read before transmitting the character (RXRDY flag cleared), thus preventing any overflow (character loss) on the receiver side.

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 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 a one to the RSTSTA (Reset Status) bit in the Control Register (US_CR).

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 a one to the RTSEN bit in the US_CR. The slave select line (NSS) can be released at high level only by writing a one to the RTSDIS bit in the US_CR (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.

44.7.7.6 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 a one to the RSTSTA (Reset Status) bit the US_CR.

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.

44.7.7.7 Receiver Timeout

Because the receiver baud rate 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).

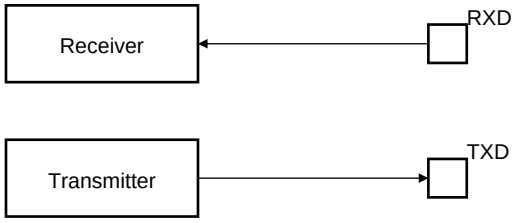
44.7.8 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

44.7.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

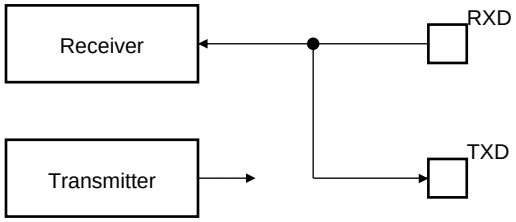
Figure 44-39.Normal Mode Configuration



44.7.8.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 44-40. 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.

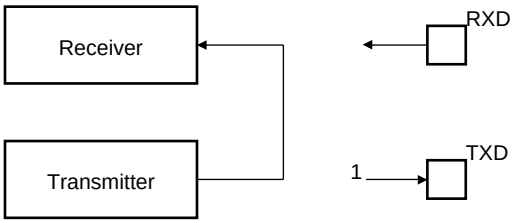
Figure 44-40.Automatic Echo Mode Configuration



44.7.8.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 44-41. 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.

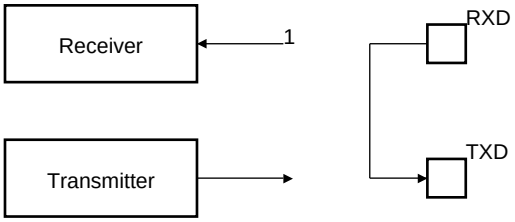
Figure 44-41.Local Loopback Mode Configuration



44.7.8.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 44-42. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 44-42.Remote Loopback Mode Configuration



44.7.9 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 field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically reset by reading the 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"

44.8 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 44-15. 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
0x0054–0x005C	Reserved	–	–	–
0x0060–0x00E0	Reserved	–	–	–
0x00E4	Write Protect Mode Register	US_WPMR	Read-write	0x0
0x00E8	Write Protect Status Register	US_WPSR	Read-only	0x0
0x00EC–0x00FC	Reserved	–	–	–

44.8.1 USART Control Register

Name: US_CR

Address: 0xF001C000 (0), 0xF0020000 (1), 0xF8020000 (2), 0xF8024000 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	–	–
15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

For SPI control, see “USART Control Register (SPI_MODE)” on page 1405.

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR and RXBRK in US_CSR.

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0: No effect.

1: Starts waiting for a character before clocking the time-out counter. Resets the status bit TIMEOUT in US_CSR.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop Mode only, the next character written to the US_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITERATION in US_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in US_CSR.

- **RETTO: Rearm Time-out**

0: No effect

1: Restart Time-out

- **RTSEN: Request to Send Enable**

0: No effect.

1: Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0: No effect.

1: Drives the pin RTS to 1.

44.8.2 USART Control Register (SPI_MODE)

Name: US_CR (SPI_MODE)

Address: 0xF001C000 (0), 0xF0020000 (1), 0xF8020000 (2), 0xF8024000 (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 1407.

- **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):

0: No effect.

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):

0: No effect.

1: Releases the Slave Select Line NSS (RTS pin).

44.8.3 USART Mode Register

Name: US_MR

Address: 0xF001C004 (0), 0xF0020004 (1), 0xF8020004 (2), 0xF8024004 (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
INVDATA	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 1431.

For SPI configuration, see “USART Mode Register (SPI_MODE)” on page 1410.

- **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
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=(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.

- **INVDATA: Inverted Data**

0: The data field transmitted on TXD line is the same as the one written in US_THR register or the content read in US_RHR is the same as RXD line. Normal mode of operation.

1: The data field transmitted on TXD line is inverted (voltage polarity only) compared to the value written on US_THR register or the content read in US_RHR is inverted compared to what is received on RXD line (or ISO7816 IO line). Inverted Mode of operation, useful for contactless card application. To be used with configuration bit MSBF.

- **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.

44.8.4 USART Mode Register (SPI_MODE)

Name: US_MR (SPI_MODE)

Address: 0xF001C004 (0), 0xF0020004 (1), 0xF8020004 (2), 0xF8024004 (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 1407.

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 1431.

- **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=(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):

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.

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):

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 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).

44.8.5 USART Interrupt Enable Register

Name: US_IER

Address: 0xF001C008 (0), 0xF0020008 (1), 0xF8020008 (2), 0xF8024008 (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 1413.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

- **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 AcknowledgeInterrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

44.8.6 USART Interrupt Enable Register (SPI_MODE)

Name: US_IER (SPI_MODE)

Address: 0xF001C008 (0), 0xF0020008 (1), 0xF8020008 (2), 0xF8024008 (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 1407.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **UNRE: SPI Underrun Error Interrupt Enable**

44.8.7 USART Interrupt Disable Register

Name: US_IDR

Address: 0xF001C00C (0), 0xF002000C (1), 0xF802000C (2), 0xF802400C (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 1415.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **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 AcknowledgeInterrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**
- **MANE: Manchester Error Interrupt Disable**

44.8.8 USART Interrupt Disable Register (SPI_MODE)

Name: US_IDR (SPI_MODE)

Address: 0xF001C00C (0), 0xF002000C (1), 0xF802000C (2), 0xF802400C (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 1407.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **UNRE: SPI Underrun Error Interrupt Disable**

44.8.9 USART Interrupt Mask Register

Name: US_IMR

Address: 0xF001C010 (0), 0xF0020010 (1), 0xF8020010 (2), 0xF8024010 (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 1417.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **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 AcknowledgeInterrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**
- **MANE: Manchester Error Interrupt Mask**

44.8.10 USART Interrupt Mask Register (SPI_MODE)

Name: US_IMR (SPI_MODE)

Address: 0xF001C010 (0), 0xF0020010 (1), 0xF8020010 (2), 0xF8024010 (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 1407.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **UNRE: SPI Underrun Error Interrupt Mask**

44.8.11 USART Channel Status Register

Name: US_CSR

Address: 0xF001C014 (0), 0xF0020014 (1), 0xF8020014 (2), 0xF8024014 (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 1420.

- **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: Max Number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSTSTA.

1: Maximum number of repetitions has been reached since the last RSTSTA.

- **NACK: Non AcknowledgeInterrupt**

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.

44.8.12 USART Channel Status Register (SPI_MODE)

Name: US_CSR (SPI_MODE)

Address: 0xF001C014 (0), 0xF0020014 (1), 0xF8020014 (2), 0xF8024014 (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 1407.

- **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.

44.8.13 USART Receive Holding Register

Name: US_RHR

Address: 0xF001C018 (0), 0xF0020018 (1), 0xF8020018 (2), 0xF8024018 (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.

44.8.14 USART Transmit Holding Register

Name: US_THR

Address: 0xF001C01C (0), 0xF002001C (1), 0xF802001C (2), 0xF802401C (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.

44.8.15 USART Baud Rate Generator Register

Name: US_BRGR

Address: 0xF001C020 (0), 0xF0020020 (1), 0xF8020020 (2), 0xF8024020 (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 1431.

- **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.

44.8.16 USART Receiver Time-out Register

Name: US_RTOR

Address: 0xF001C024 (0), 0xF0020024 (1), 0xF8020024 (2), 0xF8024024 (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
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 1431.

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1–65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

44.8.17 USART Transmitter Timeguard Register

Name: US_TTGR

Address: 0xF001C028 (0), 0xF0020028 (1), 0xF8020028 (2), 0xF8024028 (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 1431.

- **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.

44.8.18 USART FI DI RATIO Register

Name: US_FIDI

Address: 0xF001C040 (0), 0xF0020040 (1), 0xF8020040 (2), 0xF8024040 (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 1431.

- **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.

44.8.19 USART Number of Errors Register

Name: US_NER

Address: 0xF001C044 (0), 0xF0020044 (1), 0xF8020044 (2), 0xF8024044 (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 1407.

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

44.8.20 USART IrDA FILTER Register

Name: US_IF

Address: 0xF001C04C (0), 0xF002004C (1), 0xF802004C (2), 0xF802404C (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 1407.

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 1431.

- **IRDA_FILTER: IrDA Filter**

The IRDA_FILTER value must be defined to meet the following criteria:

t_{MCK} * (IRDA_FILTER + 3) < 1.41 µs

44.8.21 USART Manchester Configuration Register

Name: US_MAN

Address: 0xF001C050 (0), 0xF0020050 (1), 0xF8020050 (2), 0xF8024050 (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 1431.

- **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.

44.8.22 USART Write Protect Mode Register

Name: US_WPMR

Address: 0xF001C0E4 (0), 0xF00200E4 (1), 0xF80200E4 (2), 0xF80240E4 (3)

Access: Read-write

Reset: See Table 44-15

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 1407
- “USART Baud Rate Generator Register” on page 1423
- “USART Receiver Time-out Register” on page 1424
- “USART Transmitter Timeguard Register” on page 1425
- “USART FI DI RATIO Register” on page 1426
- “USART IrDA FILTER Register” on page 1428
- “USART Manchester Configuration Register” on page 1429

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x555341	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

44.8.23 USART Write Protect Status Register

Name: US_WPSR

Address: 0xF001C0E8 (0), 0xF00200E8 (1), 0xF80200E8 (2), 0xF80240E8 (3)

Access: Read-only

Reset: See Table 44-15

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.

1: A Write Protect Violation has occurred since the last read of the US_WPSR. 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.

45. Controller Area Network (CAN)

45.1 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/s.

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.

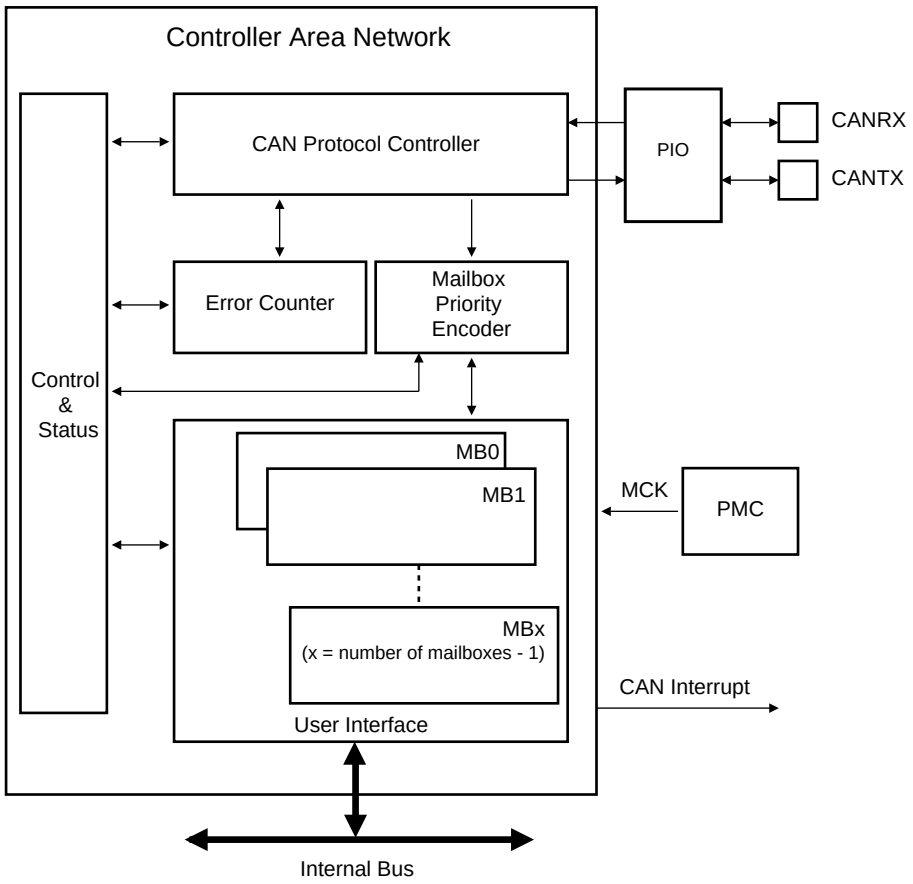
The CAN controller offers optimized features to support the Time Triggered Communication (TTC) protocol.

45.2 Embedded Characteristics

- Fully Compliant with CAN 2.0 Part A and 2.0 Part B
- Bit Rates up to 1 Mbit/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

45.3 Block Diagram

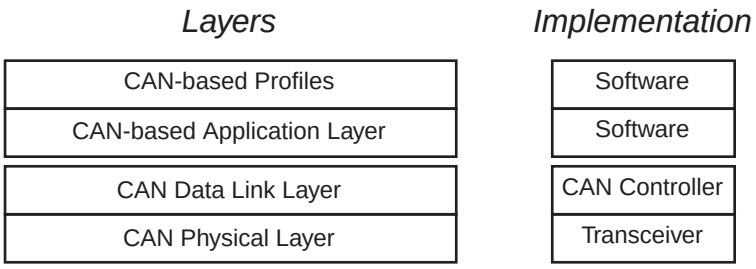
Figure 45-1. CAN Block Diagram



45.4 Application Block Diagram

Figure 45-2.

Application Block Diagram



45.5 I/O Lines Description

Table 45-1. I/O Lines Description

Name	Description	Type
CANRX	CAN Receive Serial Data	Input
CANTX	CAN Transmit Serial Data	Output

45.6 Product Dependencies

45.6.1 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 45-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
CAN0	CANRX0	PD14	C
CAN0	CANTX0	PD15	C
CAN1	CANRX1	PB14	B
CAN1	CANTX1	PB15	B

45.6.2 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.

45.6.3 Interrupt

The CAN interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the CAN interrupt requires the interrupt controller to be programmed first. Note that it is not recommended to use the CAN interrupt line in edge-sensitive mode.

Table 45-3. Peripheral IDs

Instance	ID
CAN0	40
CAN1	41

45.7 CAN Controller Features

45.7.1 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.

45.7.2 Mailbox Organization

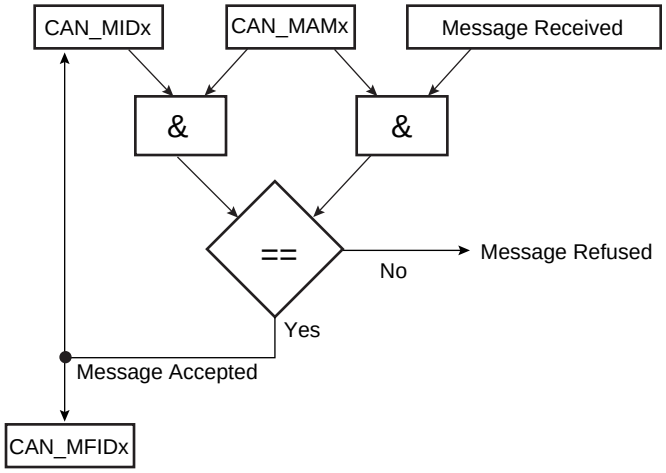
The CAN module has 8 buffers, also called channels or mailboxes. An identifier that corresponds to the CAN identifier is defined for each active mailbox. Message identifiers can match the standard frame identifier or the extended frame identifier. This identifier is defined for the first time during 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.

45.7.2.1 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 45-3. 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]();
```

45.7.2.2 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 45-4. Receive Mailbox Objects

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.

45.7.2.3 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).

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 45-5. Transmit Mailbox Objects

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 45.7.3). 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.

45.7.3 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). 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)
- 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. 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 is set. TOVF bit in the CAN_SR is cleared by reading the CAN_SR. Depending on the corresponding interrupt mask in the CAN_IMR, 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. 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 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, an interrupt is generated while TSTP is set in the CAN_SR. TSTP bit is cleared by reading the CAN_SR.

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. Time Triggered Mode is enabled by setting TTM field in the CAN_MR.

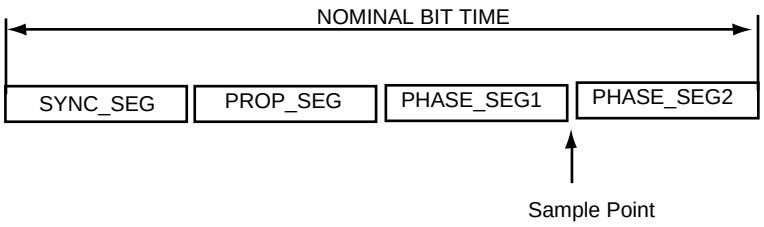
45.7.4 CAN 2.0 Standard Features

45.7.4.1 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 45-4. Partition of the CAN Bit Time



- 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 one 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".

- 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.

- 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 two 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 four TQ.

If the SMP field in the CAN_BR 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.

$$t_{PRS} = t_{CSC} \times (PROPAG + 1)$$

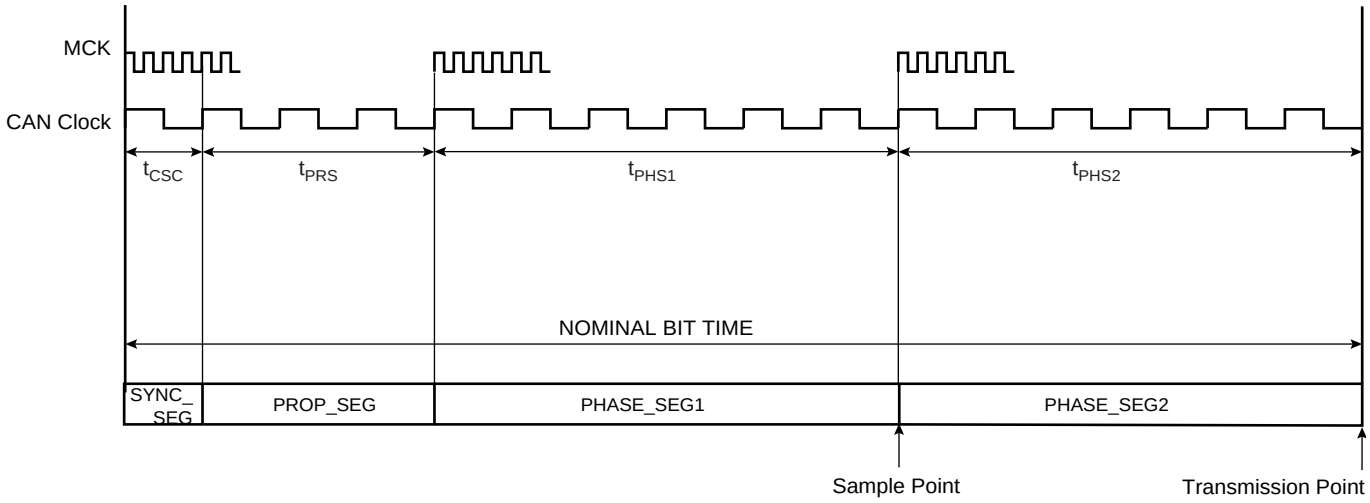
$$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$$

$$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$$

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 45-5. CAN Bit Timing



Example of bit timing determination for CAN baudrate of 500 Kbit/s:

MCK = 48 MHz

CAN baudrate = 500 Kbit/s => bit time = 2 µs

Delay of the bus driver: 50 ns

Delay of the receiver: 30 ns

Delay of the bus line (20 m): 110 ns

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:

t_{CSC} = 1 time quanta = bit time / 16 = 125 ns

=> BRP = (t_{CSC} 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:

t_{PRS} = 2 * (50+30+110) ns = 380 ns = 3 t_{CSC}

=> PROPAG = t_{PRS}/t_{CSC} - 1 = 2

The remaining time for the two phase segments is:

$t_{PHS}1 + t_{PHS}2$ = bit time - t_{CSC} - t_{PRS} = (16 - 1 - 3)t_{CSC}

$t_{PHS}1 + t_{PHS}2$ = 12 t_{CSC}

Because this number is even, we choose $t_{PHS}2 = t_{PHS}1$ (else we would choose $t_{PHS}2 = t_{PHS}1 + t_{CSC}$).

$t_{PHS}1 = t_{PHS}2$ = (12/2) t_{CSC} = 6 t_{CSC}

=> PHASE1 = PHASE2 = $t_{PHS}1/t_{CSC}$ - 1 = 5

The resynchronization jump width must comprise between one t_{CSC} and the minimum of four t_{CSC} and $t_{PHS}1$. We choose its maximum value:

t_{SJW} = Min(4 t_{CSC},$t_{PHS}1$) = 4 t_{CSC}

=> SJW = t_{SJW}/t_{CSC} - 1 = 3

Finally: CAN_BR = 0x00053255

CAN Bus Synchronization

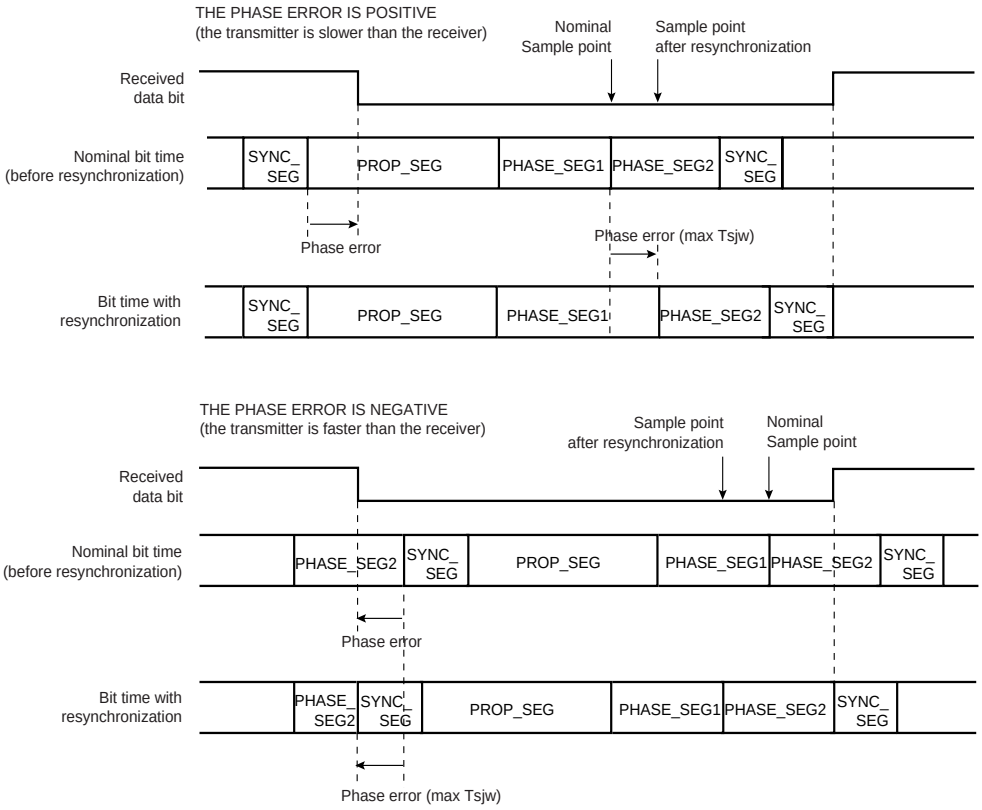
Two types of synchronization are distinguished: "hard synchronization" at the start of a frame and "resynchronization" inside a frame. After a hard synchronization, the bit time is restarted with the end of the SYNC_SEG segment, regardless of the phase error. Resynchronization causes a reduction or increase 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 45-6. CAN Resynchronization



Autobaud Mode

The autobaud feature is enabled by setting the ABM field in the CAN_MR. 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.

45.7.4.2 Error Detection

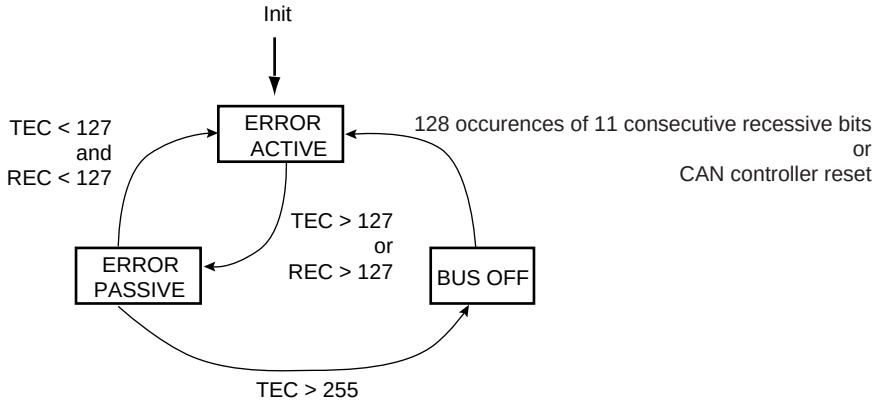
There are five different error types that are not mutually exclusive. Each error concerns only specific fields of the CAN data frame (refer to the Bosch CAN specification for their correspondence):

- CRC error (CERR bit in the CAN_SR): 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): 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): 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): 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): 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 45-7. 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. 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. The corresponding interrupt is pending while the interrupt is not masked in the CAN_IMR. If the CAN controller enters Error Passive Mode, then the ERRP bit is set in the CAN_SR and an interrupt remains pending while the ERRP bit is set in the CAN_IMR. If the CAN enters Bus Off Mode, then the BOFF bit is set in the CAN_SR. As for ERRP and ERRA, an interrupt is pending while the BOFF bit is set in the CAN_IMR.

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, but the node remains error active. The corresponding interrupt is pending while the interrupt is set in the CAN_IMR.

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 45-7 on page 1444. The transitions depend on the TEC and REC (CAN_ECR) values as described in Section "Fault Confinement" on page 1444.

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.

45.7.4.3 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.

Reactive overload frames are automatically handled by the CAN controller even if the OVL bit in the CAN_MR is not set. An overload flag is generated in the same way as an error flag, but error counters do not increment.

45.7.5 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 is set; otherwise, the WAKEUP signal in the CAN_SR 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 only after detection of 11 consecutive recessive bits on the bus.

45.7.5.1 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 is set. Depending on the corresponding mask in the CAN_IMR, an interrupt is generated while SLEEP is set.

The SLEEP signal in the CAN_SR 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. 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.

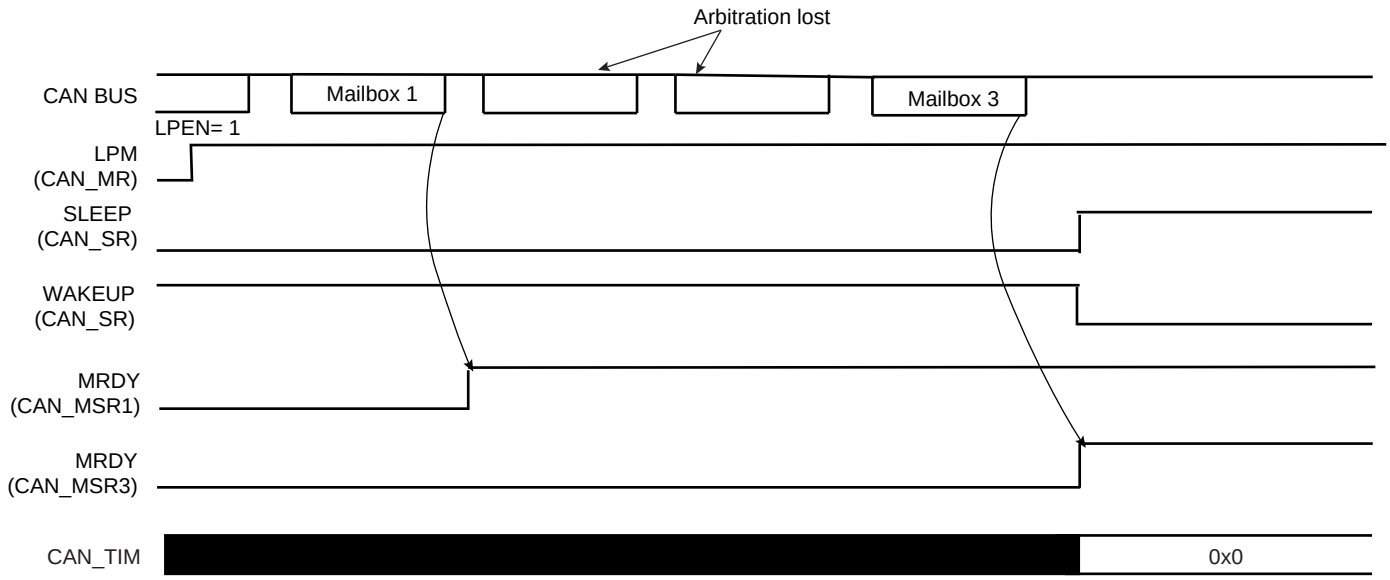
Error counters are disabled while the SLEEP signal is set to one.

Thus, to enter Low-power Mode, the software application must:

- Set LPM field in the CAN_MR
- Wait for SLEEP signal rising

Now the CAN Controller clock can be disabled. This is done by programming the Power Management Controller (PMC).

Figure 45-8. Enabling Low-power Mode



45.7.5.2 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

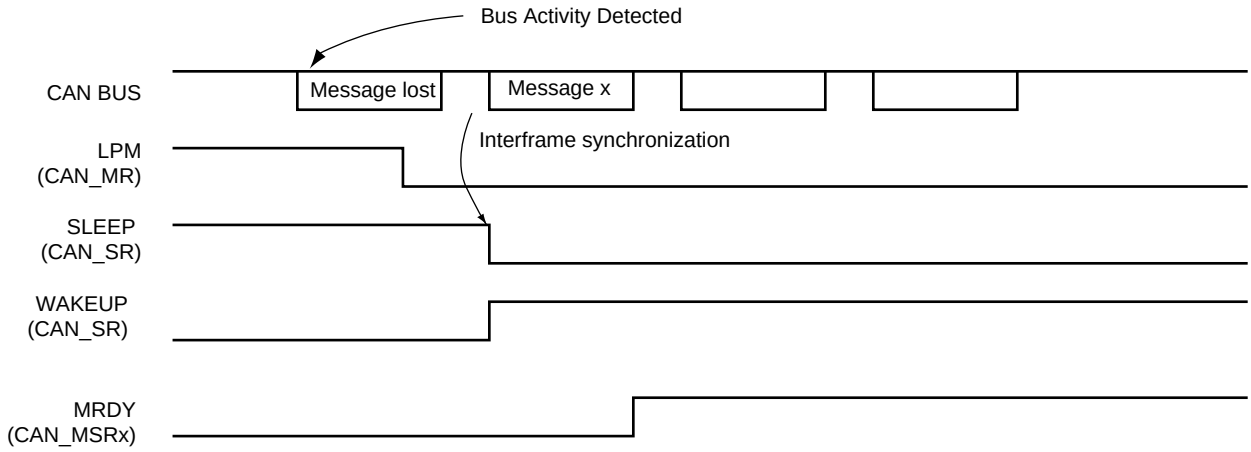
The CAN controller synchronizes itself with the bus activity by checking for eleven consecutive “recessive” bits. Once synchronized, the WAKEUP signal in the CAN_SR is set.

Depending on the corresponding mask in the CAN_IMR, an interrupt is generated while WAKEUP is set. The SLEEP signal in the CAN_SR 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 45-9).

Figure 45-9. Disabling Low-power Mode



45.8 Functional Description

45.8.1 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.

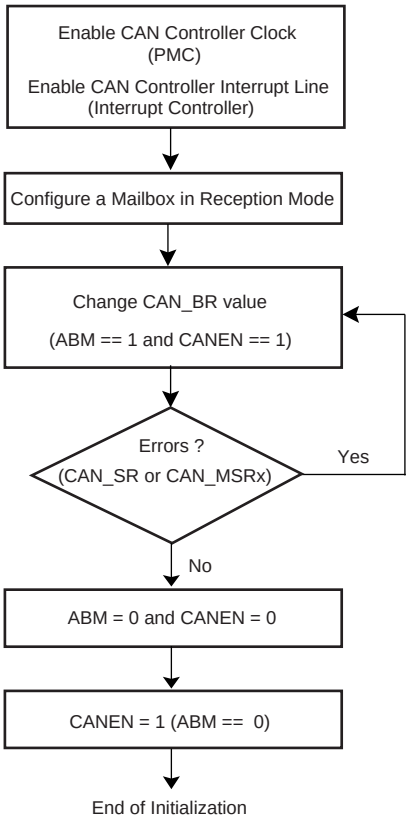
The CAN controller must be initialized with the CAN network parameters. The CAN_BR defines the sampling point in the bit time period. CAN_BR must be set before the CAN controller is enabled by setting the CANEN field in the CAN_MR.

The CAN controller is enabled by setting the CANEN flag in the CAN_MR. 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 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 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.

Figure 45-10.Possible Initialization Procedure



45.8.2 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. They can be unmasked by writing to the CAN_IER. After a power-up reset, all interrupt sources are disabled (masked). The current mask status can be checked by reading the CAN_IMR.

The CAN_SR 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.

45.8.3 CAN Controller Message Handling

45.8.3.1 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

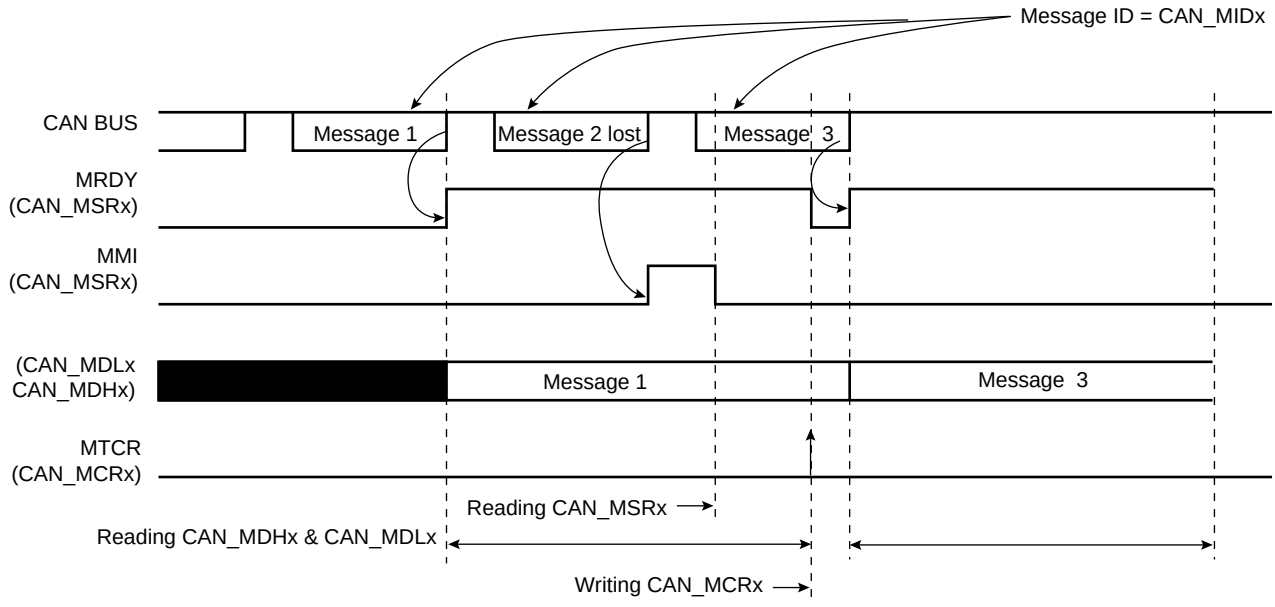
A mailbox is in Receive Mode once the MOT field in the CAN_MMRx has been configured. Message ID and Message Acceptance Mask must be set before the Receive Mode is enabled.

After Receive Mode is enabled, the MRDY flag in the CAN_MSR 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. This automatically clears the MRDY signal.

The MMI flag in the CAN_MSRx 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. 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. See Figure 45-11.

Figure 45-11.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

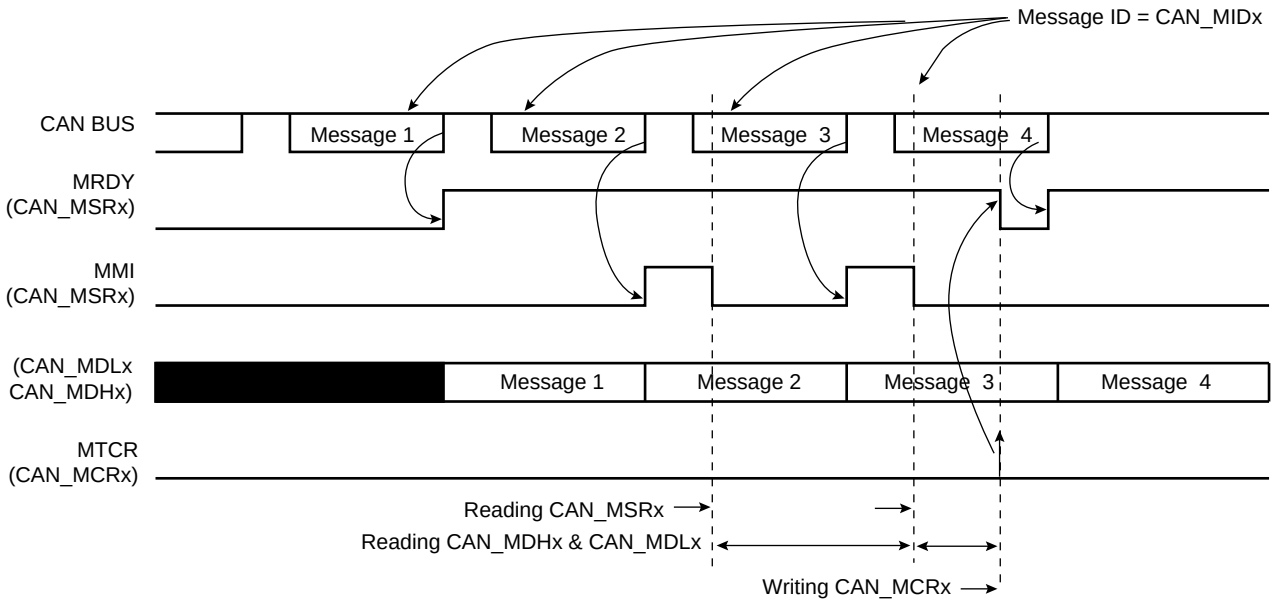
A mailbox is in Receive with Overwrite Mode once the MOT field in the CAN_MMRx has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

After Receive Mode is enabled, the MRDY flag in the CAN_MSR 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 notifies the software that a message has been dropped by the mailbox. This flag is cleared when reading the CAN_MSRx.

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 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 45-12).

Figure 45-12.Receive with Overwrite Mailbox

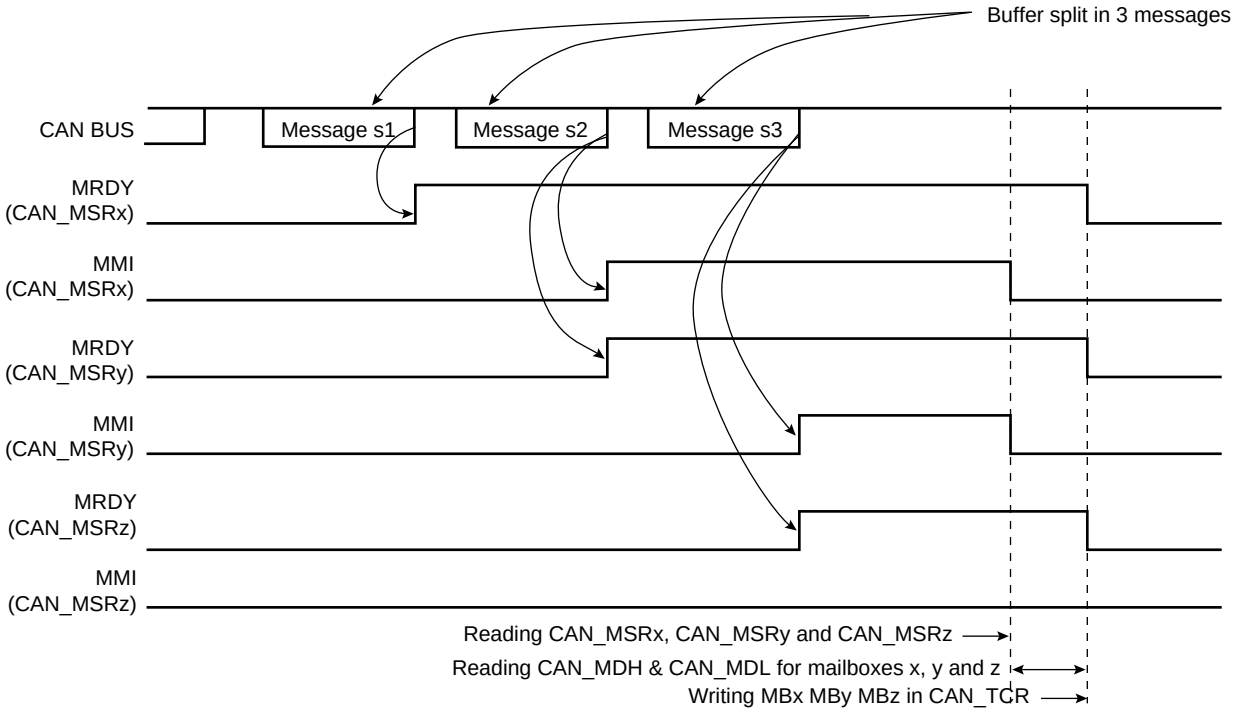


Chaining Mailboxes

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 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.

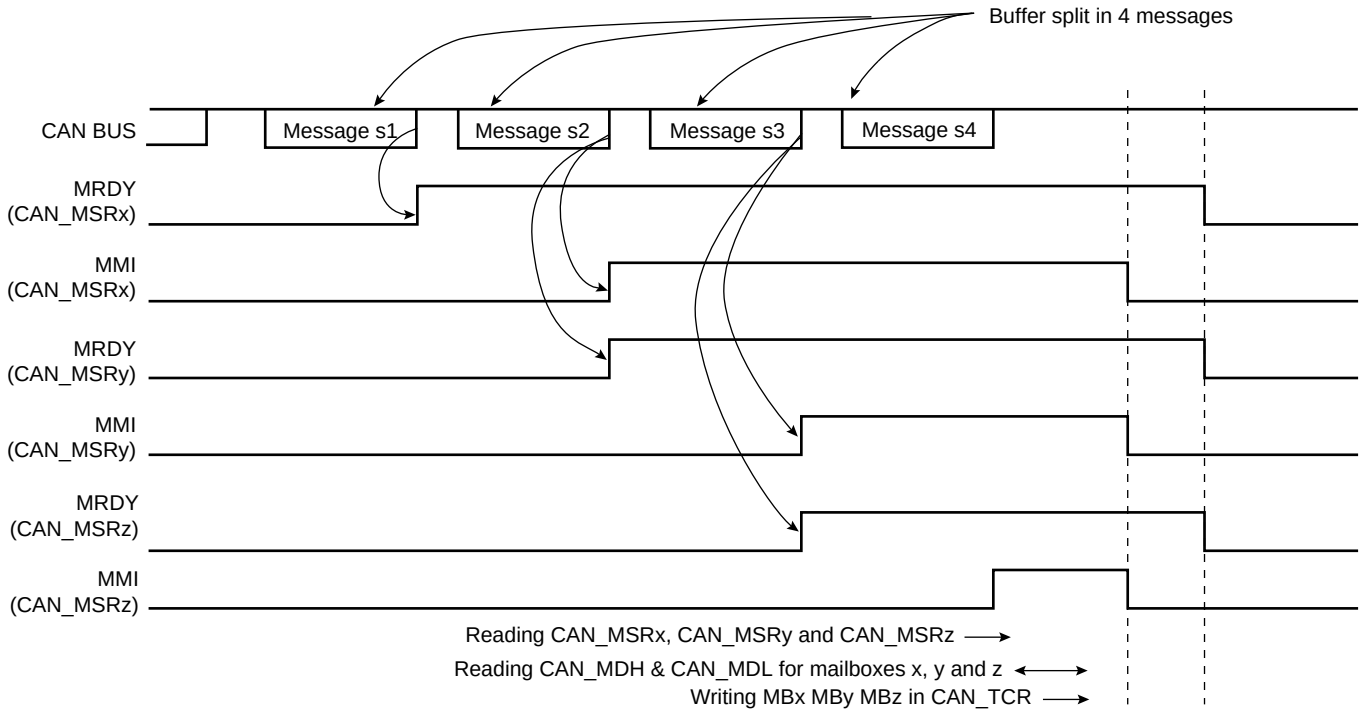
If several mailboxes are chained to receive a buffer split into several messages, all mailboxes except the last one (with the highest number) must be configured in Receive Mode. The first message received is handled by the first mailbox, the second one is refused by the first mailbox and accepted by the second mailbox, the last message is accepted by the last mailbox and refused by previous ones (see Figure 45-13).

Figure 45-13.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 45-14).

Figure 45-14.Chaining Three Mailboxes to Receive a Buffer Split into Four Messages



45.8.3.2 Transmission Handling

A mailbox is in Transmit Mode once the MOT field in the CAN_MMRx 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 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.

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 1453.

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. The priority is set in the PRIOR field of the CAN_MMRx. 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.

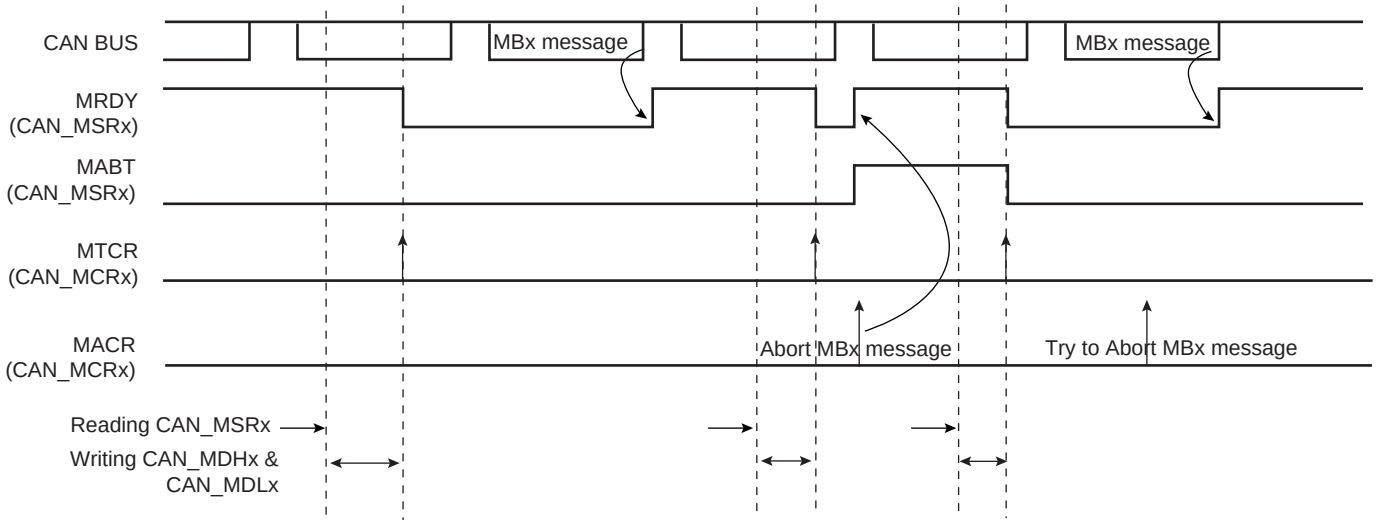
Setting the MACR bit in the CAN_MCRx aborts the transmission. Transmission for several mailboxes can be aborted by writing MBx fields in the CAN_MACR. If the message is being sent when the abort command is set, then the application is notified by the MRDY bit set and not the MABT in the CAN_MSRx. Otherwise, if the message has not been sent, then the MRDY and the MABT are set in the CAN_MSR.

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. 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 until the next transfer command.

Figure 45-15 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.

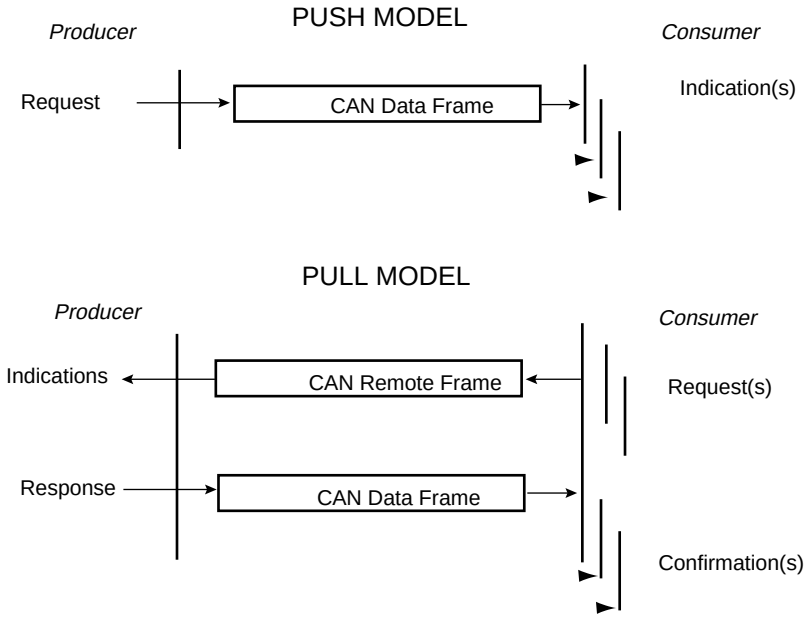
Figure 45-15.Transmitting Messages



45.8.3.3 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 45-16.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 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 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. Data is sent after the reception of a remote frame as soon as it wins the bus arbitration.

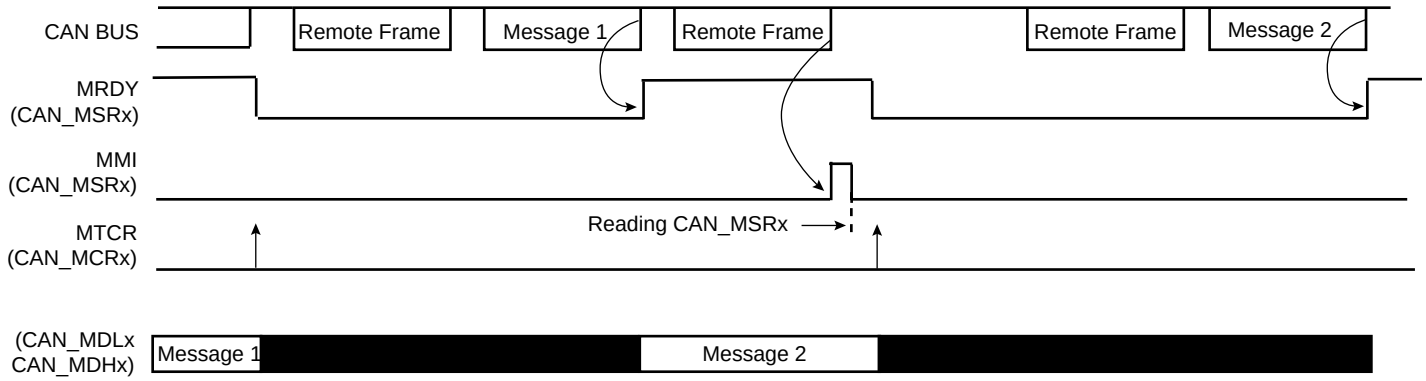
The MRDY flag remains at zero as long as the message has not been sent or aborted. No access to the mailbox data register can be done while MRDY flag is cleared. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked according to the mailbox flag in the CAN_IMR global register.

If a remote frame is received while no data are ready to be sent (signal MRDY set in the CAN_MSRx), then the MMI signal is set in the CAN_MSRx. This bit is cleared by reading the CAN_MSRx.

The MRTR field in the CAN_MSRx 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. Please refer to the section "Transmission Handling" on page 1452.

Figure 45-17.Producer Handling



Consumer Configuration

A mailbox is in Consumer Mode once the MOT field in the CAN_MMRx has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

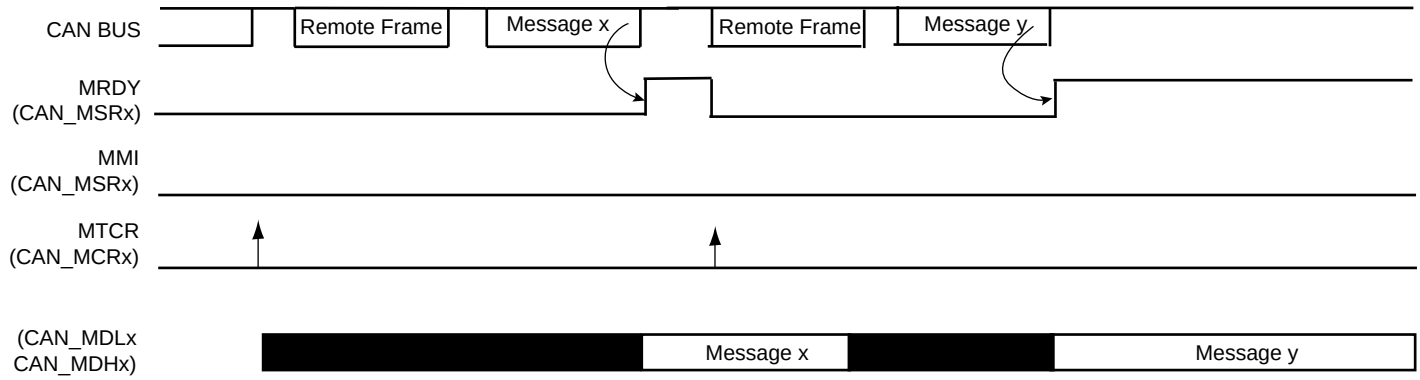
After Consumer Mode is enabled, the MRDY flag in the CAN_MSR is automatically cleared until the first transfer request command. The software application sends a remote frame by setting the MTCR bit in the CAN_MCRx or the MBx bit in the global CAN_TCR. The application is notified of the answer by the MRDY flag set in the CAN_MSRx. 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 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, they will be lost. The application is notified by reading the MMI field in the CAN_MSRx. 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.

Figure 45-18.Consumer Handling



45.8.4 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:

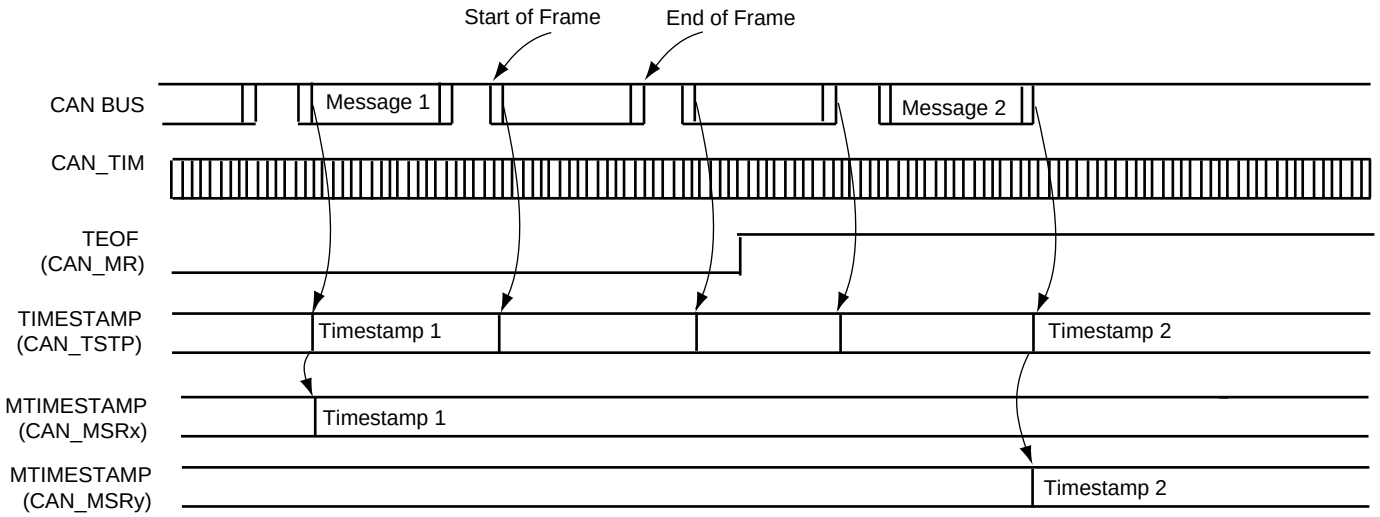
- Timestamping Mode: The value of the internal timer is captured at each Start Of Frame or each End Of Frame.
- Time Triggered Mode: The mailbox transfer operation is triggered when the internal timer reaches the mailbox trigger.

Timestamping Mode is enabled by clearing the TTM bit in the CAN_MR. Time Triggered Mode is enabled by setting the TTM bit in the CAN_MR.

45.8.4.1 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. The value read in the CAN_MSRx corresponds to the internal timer value at the Start Of Frame or the End Of Frame of the message handled by the mailbox.

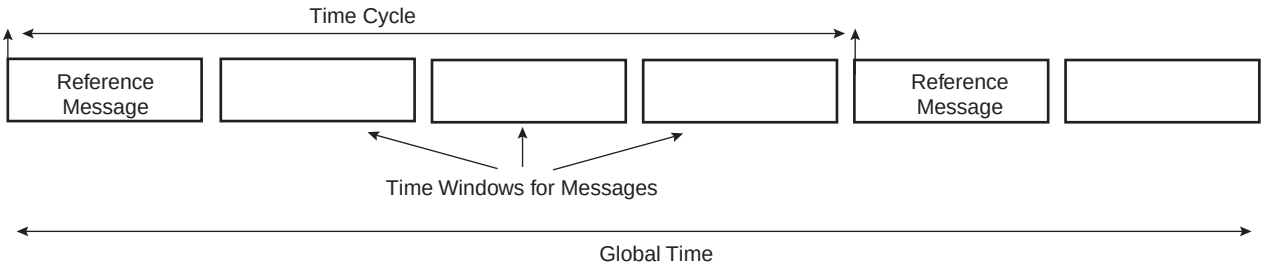
Figure 45-19.Mailbox Timestamp



45.8.4.2 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 45-20.Time Triggered Principle



Time Trigger Mode is enabled by setting the TTM field in the CAN_MR. 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. 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. 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.

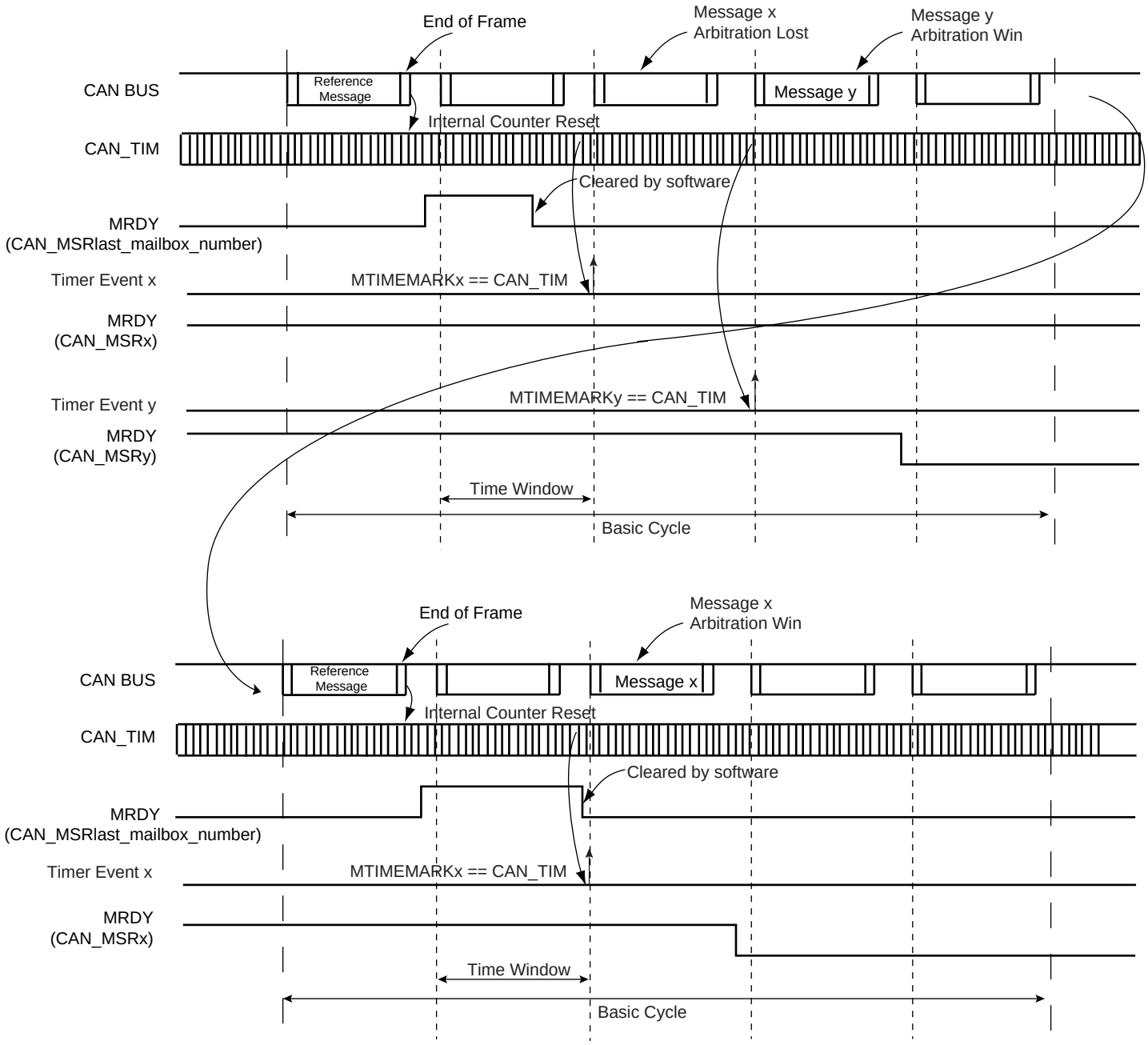
In Time Triggered Mode, transmit operations are delayed until the internal timer event for the mailbox. The application prepares a message to be sent by setting the MTCR in the CAN_MCRx. The message is not sent until the CAN_TIM value is less than the MTIMEMARK value defined in the CAN_MMRx.

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.

Freezing the Internal Timer Counter

The internal counter can be frozen by setting TIMFRZ in the CAN_MR. 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 is set when the counter is frozen. The TOVF bit in the CAN_SR is cleared by reading the CAN_SR. Depending on the corresponding interrupt mask in the CAN_IMR, an interrupt is generated when TOVF is set.

Figure 45-21.Time Triggered Operations



45.8.5 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_WPSR.

List of the write-protected registers:

Section 45.9.1 “CAN Mode Register” on page 1460

Section 45.9.6 “CAN Baudrate Register” on page 1471

Section 45.9.14 “CAN Message Mode Register” on page 1479

Section 45.9.15 “CAN Message Acceptance Mask Register” on page 1480

Section 45.9.16 “CAN Message ID Register” on page 1481

45.9 Controller Area Network (CAN) User Interface

Table 45-6. 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–x00E0	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	–

Notes: 1. Mailbox number ranges from 0 to 7.

45.9.1 CAN Mode Register

Name: CAN_MR

Address: 0xF000C000 (0), 0xF8010000 (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**

w Power Mode.

1: Enable Low Power M

CAN controller enters Low Power Mode once all pending messages have been transmitted.

- **ABM: Disable/Enable Autobaud/Listen mode**

0: Disable Autobaud/listen mode.

1: Enable Autobaud/listen mode.

- **OVL: Disable/Enable Overload Frame**

0: No overload frame is generated.

1: An overload frame is generated after each successful reception for mailboxes configured in Receive with/without overwrite Mode, Producer and Consumer.

- **TEOF: Timestamp messages at each end of Frame**

0: The value of CAN_TIM is captured in the CAN_TIMESTP register at each Start Of Frame.

1: The value of CAN_TIM is captured in the CAN_TIMESTP register at each End Of Frame.

- **TTM: Disable/Enable Time Triggered Mode**

0: Time Triggered Mode is disabled.

1: Time Triggered Mode is enabled.

- **TIMFRZ: Enable Timer Freeze**

0: The internal timer continues to be incremented after it reached 0xFFFF.

1: The internal timer stops incrementing after reaching 0xFFFF. It is restarted after a timer reset. See "Freezing the Internal Timer Counter" on page 1456.

- **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.

45.9.2 CAN Interrupt Enable Register

Name: CAN_IER

Address: 0xF000C004 (0), 0xF8010004 (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.

45.9.3 CAN Interrupt Disable Register

Name: CAN_IDR

Address: 0xF000C008 (0), 0xF8010008 (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.

45.9.4 CAN Interrupt Mask Register

Name: CAN_IMR

Address: 0xF000C00C (0), 0xF801000C (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.

45.9.5 CAN Status Register

Name: CAN_SR

Address: 0xF000C010 (0), 0xF8010010 (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.

- **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.

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.

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.

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.

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.

- **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).

This flag is automatically cleared by reading the CAN_SR.

- **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.

- **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.

- **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.

- **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.

- **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.

- **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.

45.9.6 CAN Baudrate Register

Name: CAN_BR

Address: 0xF000C014 (0), 0xF8010014 (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 must be done while CAN module is disabled.

To compute the different bit timings, please refer to the Section 45.7.4.1 "CAN Bit Timing Configuration" on page 1439.

- **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.

45.9.7 CAN Timer Register

Name: CAN_TIM

Address: 0xF000C018 (0), 0xF8010018 (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.

45.9.8 CAN Timestamp Register

Name: CAN_TIMESTP

Address: 0xF000C01C (0), 0xF801001C (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, 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. If the TSTP mask in the CAN_IMR is set, an interrupt is generated while TSTP flag is set in the CAN_SR. This flag is cleared by reading the CAN_SR.

Note: The CAN_TIMESTP register is reset when the CAN is disabled then enabled thanks to the CANEN bit in the CAN_MR.

45.9.9 CAN Error Counter Register

Name: CAN_ECR

Address: 0xF000C020 (0), 0xF8010020 (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.

45.9.10 CAN Transfer Command Register

Name: CAN_TCR

Address: 0xF000C024 (0), 0xF8010024 (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.

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.

45.9.11 CAN Abort Command Register

Name: CAN_ACR

Address: 0xF000C028 (0), 0xF8010028 (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 the MACR field (in the CAN_MCRx) for each mailbox.

45.9.12 CAN Write Protection Mode Register

Name: CAN_WPMR

Address: 0xF000C0E4 (0), 0xF80100E4 (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 if WPKEY is written with 0x43414E (“CAN” written in ASCII)

1: The Write Protection is Enabled if WPKEY is written with0x43414E (“CAN” written in ASCII)

Protects the registers:

Section 45.9.1 “CAN Mode Register” on page 1460

Section 45.9.6 “CAN Baudrate Register” on page 1471

Section 45.9.14 “CAN Message Mode Register” on page 1479

Section 45.9.15 “CAN Message Acceptance Mask Register” on page 1480

Section 45.9.16 “CAN Message ID Register” on page 1481

- **WPKEY: SPI Write Protection Key Password.**

Value	Name	Description
0x43414E	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

45.9.13 CAN Write Protection Status Register

Name: CAN_WPSR

Address: 0xF000C0E8 (0), 0xF80100E8 (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.

1: A Write Protect Violation has occurred since the last read of the CAN_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**

This Field indicates the offset of the register concerned by the violation

45.9.14 CAN Message Mode Register

Name: CAN_MMRx [x=0..7]

Address: 0xF000C200 (0)[0], 0xF000C220 (0)[1], 0xF000C240 (0)[2], 0xF000C260 (0)[3], 0xF000C280 (0)[4], 0xF000C2A0 (0)[5], 0xF000C2C0 (0)[6], 0xF000C2E0 (0)[7], 0xF8010200 (1)[0], 0xF8010220 (1)[1], 0xF8010240 (1)[2], 0xF8010260 (1)[3], 0xF8010280 (1)[4], 0xF80102A0 (1)[5], 0xF80102C0 (1)[6], 0xF80102E0 (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 1456.

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

45.9.15 CAN Message Acceptance Mask Register

Name: CAN_MAMx [x=0..7]

Address: 0xF000C204 (0)[0], 0xF000C224 (0)[1], 0xF000C244 (0)[2], 0xF000C264 (0)[3], 0xF000C284 (0)[4], 0xF000C2A4 (0)[5], 0xF000C2C4 (0)[6], 0xF000C2E4 (0)[7], 0xF8010204 (1)[0], 0xF8010224 (1)[1], 0xF8010244 (1)[2], 0xF8010264 (1)[3], 0xF8010284 (1)[4], 0xF80102A4 (1)[5], 0xF80102C4 (1)[6], 0xF80102E4 (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.

45.9.16 CAN Message ID Register

Name: CAN_MIDx [x=0..7]

Address: 0xF000C208 (0)[0], 0xF000C228 (0)[1], 0xF000C248 (0)[2], 0xF000C268 (0)[3], 0xF000C288 (0)[4], 0xF000C2A8 (0)[5], 0xF000C2C8 (0)[6], 0xF000C2E8 (0)[7], 0xF8010208 (1)[0], 0xF8010228 (1)[1], 0xF8010248 (1)[2], 0xF8010268 (1)[3], 0xF8010288 (1)[4], 0xF80102A8 (1)[5], 0xF80102C8 (1)[6], 0xF80102E8 (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**

45.9.17 CAN Message Family ID Register

Name: CAN_MFIDx [x=0..7]

Address: 0xF000C20C (0)[0], 0xF000C22C (0)[1], 0xF000C24C (0)[2], 0xF000C26C (0)[3], 0xF000C28C (0)[4], 0xF000C2AC (0)[5], 0xF000C2CC (0)[6], 0xF000C2EC (0)[7], 0xF801020C (1)[0], 0xF801022C (1)[1], 0xF801024C (1)[2], 0xF801026C (1)[3], 0xF801028C (1)[4], 0xF80102AC (1)[5], 0xF80102CC (1)[6], 0xF80102EC (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
```

45.9.18 CAN Message Status Register

Name: CAN_MSRx [x=0..7]

Address: 0xF000C210 (0)[0], 0xF000C230 (0)[1], 0xF000C250 (0)[2], 0xF000C270 (0)[3], 0xF000C290 (0)[4], 0xF000C2B0 (0)[5], 0xF000C2D0 (0)[6], 0xF000C2F0 (0)[7], 0xF8010210 (1)[0], 0xF8010230 (1)[1], 0xF8010250 (1)[2], 0xF8010270 (1)[3], 0xF8010290 (1)[4], 0xF80102B0 (1)[5], 0xF80102D0 (1)[6], 0xF80102F0 (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.

MRDY, MABT are cleared by writing MTCR or MACR in the CAN_MCRx.

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). If the TEOF field in the CAN_MR 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 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

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.

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.

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.

Mailbox Object Type	Description
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.

45.9.19 CAN Message Data Low Register

Name: CAN_MDLx [x=0..7]

Address: 0xF000C214 (0)[0], 0xF000C234 (0)[1], 0xF000C254 (0)[2], 0xF000C274 (0)[3], 0xF000C294 (0)[4], 0xF000C2B4 (0)[5], 0xF000C2D4 (0)[6], 0xF000C2F4 (0)[7], 0xF8010214 (1)[0], 0xF8010234 (1)[1], 0xF8010254 (1)[2], 0xF8010274 (1)[3], 0xF8010294 (1)[4], 0xF80102B4 (1)[5], 0xF80102D4 (1)[6], 0xF80102F4 (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, 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. In this mode, the software application must re-read CAN_MDH and CAN_MDL, while the MMI bit in the CAN_MSRx 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]

45.9.20 CAN Message Data High Register

Name: CAN_MDHx [x=0..7]

Address: 0xF000C218 (0)[0], 0xF000C238 (0)[1], 0xF000C258 (0)[2], 0xF000C278 (0)[3], 0xF000C298 (0)[4], 0xF000C2B8 (0)[5], 0xF000C2D8 (0)[6], 0xF000C2F8 (0)[7], 0xF8010218 (1)[0], 0xF8010238 (1)[1], 0xF8010258 (1)[2], 0xF8010278 (1)[3], 0xF8010298 (1)[4], 0xF80102B8 (1)[5], 0xF80102D8 (1)[6], 0xF80102F8 (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, 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. In this mode, the software application must re-read CAN_MDH and CAN_MDL, while the MMI bit in the CAN_MSRx 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]

45.9.21 CAN Message Control Register

Name: CAN_MCRx [x=0..7]

Address: 0xF000C21C (0)[0], 0xF000C23C (0)[1], 0xF000C25C (0)[2], 0xF000C27C (0)[3], 0xF000C29C (0)[4], 0xF000C2BC (0)[5], 0xF000C2DC (0)[6], 0xF000C2FC (0)[7], 0xF801021C (1)[0], 0xF801023C (1)[1], 0xF801025C (1)[2], 0xF801027C (1)[3], 0xF801029C (1)[4], 0xF80102BC (1)[5], 0xF80102DC (1)[6], 0xF80102FC (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 the MACR field for several mailboxes in the same time, setting several bits to the CAN_ACR.

- **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.

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.

46. Software Modem Device (SMD)

46.1 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 (LSD).

SMD and LSD are two parts of the “Transformer only” solution. The transformer is the only component connecting SMD and LSD and is used for power, clock and data transfers. Power and clock are supplied by the SMD and consumed by the LSD. The data flow is bidirectional. The data transfer is based on pulse width modulation for transmission from the SMD to the LSD, and for receiving from the LSD.

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 LSD 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.

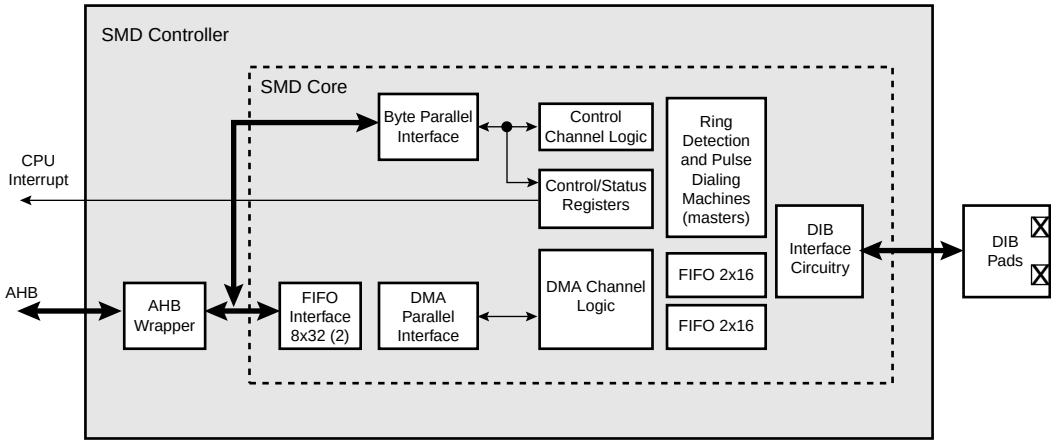
The SMD incorporates both RX and TX FIFOs, available through the DMAC interface. Each FIFO is able to hold eight 32-bit words (equivalent to 16 modem data samples).

46.2 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

46.3 Block Diagram

Figure 46-1. Software Modem Device Block Diagram



46.4 Software Modem Device (SMD) User Interface

The SMD presents a number of registers through the AHB interface for software control and status functions.

Table 46-1. Register Mapping

0x0C	SMD Drive register	SMD_DRIVE	Read-write	0x00000002

46.4.1 SMD Drive Register

Name: SMD_DRIVE

Address: 0x0040000C

Access: Read-write

Reset: 0x00000002

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
PWRCLKP_PCS		PWRCLKN_PCS2		PWRCLKP_PV	PWRCLKP_PV2	DC_PWRCLKPN	MIE

- **PWRCLKP_PCS: PWRCLKP Pin Control Select.**

When DC_PWRCLKPN is a 1, the usage of PWRCLKP_PCS bits for direct control of PWRCLKP pin is enabled as follows:

X1 = High impedance on PWRCLKP pin.

00 = Drive low on PWRCLKP pin.

10 = Drive high on PWRCLKP pin.

When DC_PWRCLKPN is a 0, the protocol logic controls PWRCLKP pin.

If PWRCLKPN_FS bit is a 1, the above information is applied to PWRCLKN pin because of swapping with PWRCLKP.

- **PWRCLKN_PCS2: PWRCLKN Pin Control Select.**

When DC_PWRCLKPN is a 1, the usage of PWRCLKN_PCS2 bits for direct control of PWRCLKN pin is enabled as follows:

X1 = High impedance on PWRCLKN pin.

00 = Drive low on PWRCLKN pin.

10 = Drive high on PWRCLKN pin.

When DC_PWRCLKPN is a 0, the protocol logic controls PWRCLKN pin.

If PWRCLKPN_FS bit is a 1, the above information is applied to PWRCLKP pin because of swapping with PWRCLKN.

- **PWRCLKP_PV: PWRCLKP Pin Value.**

This bit reflects the PWRCLKP pin value if PWRCLKPN_FS = 0, or the PWRCLKN pin value if PWRCLKPN_FS = 1 (because of swapping with PWRCLKP).

- **PWRCLKP_PV2: PWRCLKP Pin Value.**

This bit reflects the PWRCLKN pin value if PWRCLKPN_FS = 0, or the PWRCLKP pin value if PWRCLKPN_FS = 1 (because of swapping with PWRCLKN).

- **DC_PWRCLKPN: Direct Control of PWRCLKP, PWRCLKN Pins Enable.**

0 = Enables protocol logic control of PWRCLKP, PWRCLKN pins.

1 = Enables the use of PWRCLKP_PCS and PWRCLKN_PCS2 bits for direct control of PWRCLKP, PWRCLKN pins making them general purpose input/outputs (GPIOs).

- **MIE: MADCVS Interrupt Enable.**

0 = Disables smd_irq interrupt generation for MADCVS flag.

1 = Enables smd_irq interrupt generation for MADCVS flag.

47. Timer Counter (TC)

47.1 Description

The Timer Counter (TC) includes 6 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 (TC) embeds a quadrature decoder logic connected in front of the timers and driven by TIOA0, TIOB0 and TIOB1 inputs. When enabled, the quadrature decoder performs the input lines filtering, decoding of quadrature signals and connects to the timers/counters in order to read the position and speed of the motor through the user interface.

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 47-1 gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2.

Table 47-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	div2
TIMER_CLOCK2	div8
TIMER_CLOCK3	div32
TIMER_CLOCK4	div128
TIMER_CLOCK5[(1)]	slow_clock

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.

47.2 Embedded Characteristics

- Provides 6 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
  - Quadrature decoder logic
  - 2-bit gray up/down count for stepper motor
- 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
- Read of the Capture registers by the DMAC
- Configuration registers can be write protected

47.3 Block Diagram

Figure 47-1. Timer Counter Block Diagram

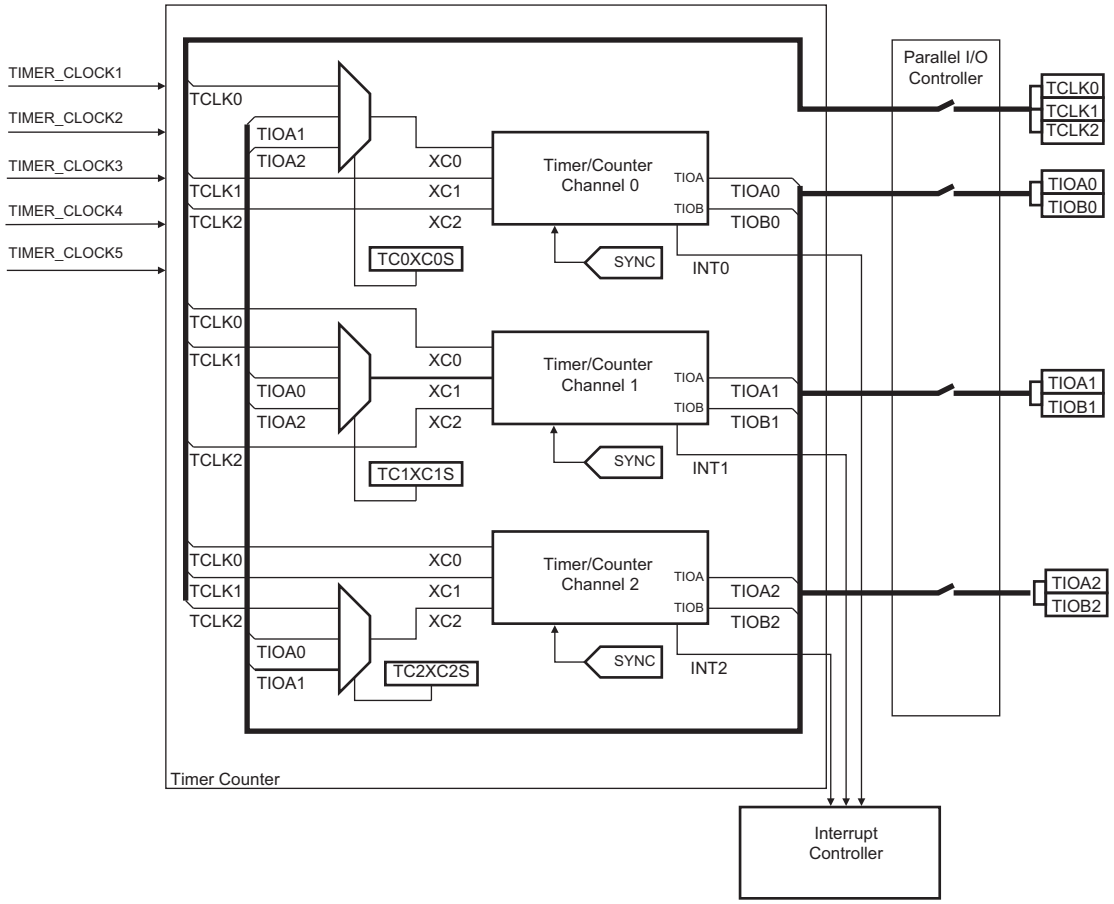


Table 47-2. Signal Name Description

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)

47.4 Pin Name List

Table 47-3. 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

47.5 Product Dependencies

47.5.1 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 47-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TC0	TCLK0	PD7	B
TC0	TCLK1	PC14	B
TC0	TCLK2	PE29	B
TC0	TIOA0	PD5	B
TC0	TIOA1	PC12	B
TC0	TIOA2	PE27	B
TC0	TIOB0	PD6	B
TC0	TIOB1	PC13	B
TC0	TIOB2	PE28	B
TC1	TCLK3	PC2	B
TC1	TCLK4	PC5	B
TC1	TCLK5	PC8	B
TC1	TIOA3	PC0	B
TC1	TIOA4	PC3	B
TC1	TIOA5	PC6	B
TC1	TIOB3	PC1	B
TC1	TIOB4	PC4	B
TC1	TIOB5	PC7	B

47.5.2 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.

47.5.3 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.

47.6 Functional Description

47.6.1 TC Description

The 6 channels of the Timer Counter are independent and identical in operation except when quadrature decoder is enabled. The registers for channel programming are listed in Table 47-5 on page 1517.

47.6.2 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 the TC Status Register (TC_SR) is set.

The current value of the counter is accessible in real time by reading the TC 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.

47.6.3 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 Block Mode Register (TC_BMR). See Figure 47-2 “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 (TC_CMR).

The selected clock can be inverted with the CLKI bit in the 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 TC_CMR defines this signal (none, XC0, XC1, XC2). See Figure 47-3 “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 47-2. Clock Chaining Selection

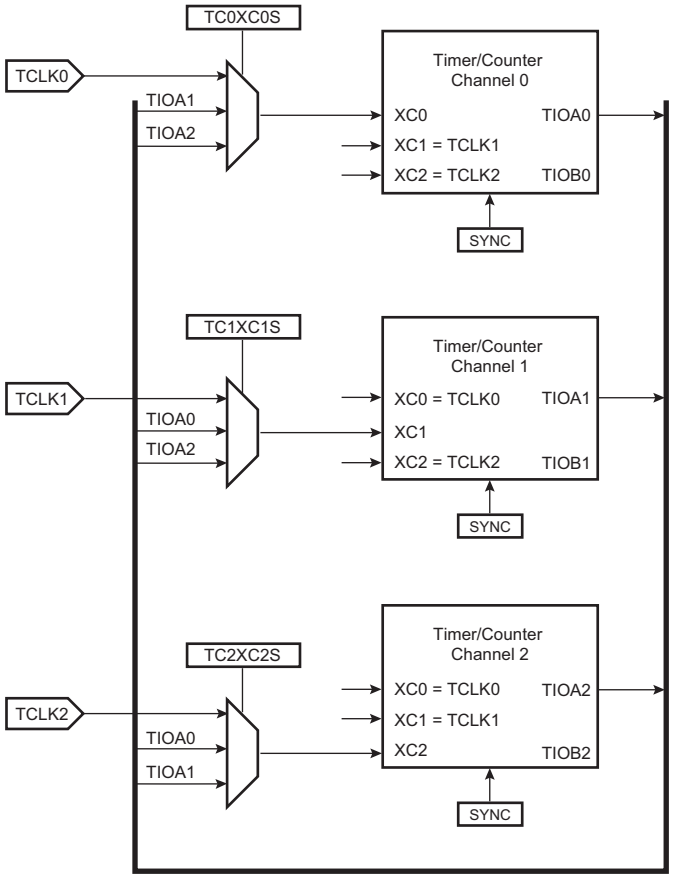
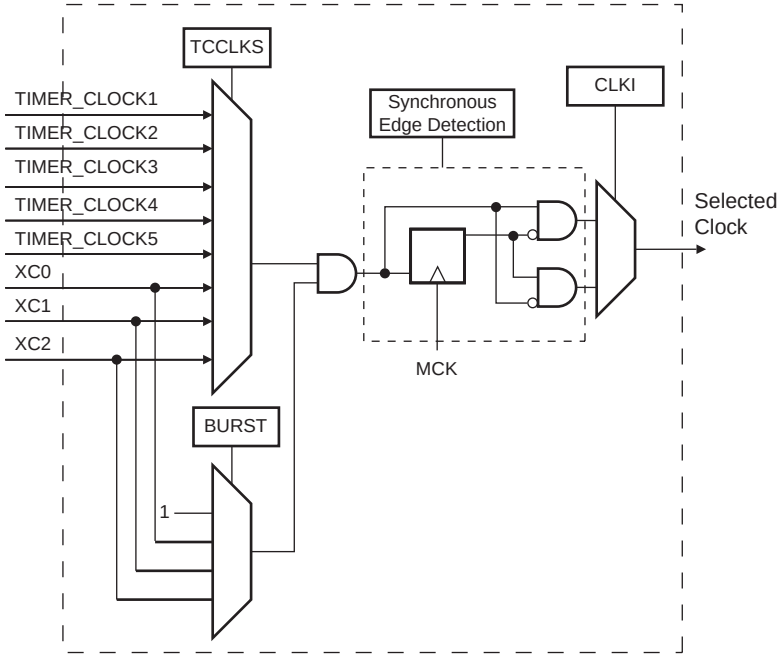


Figure 47-3. Clock Selection

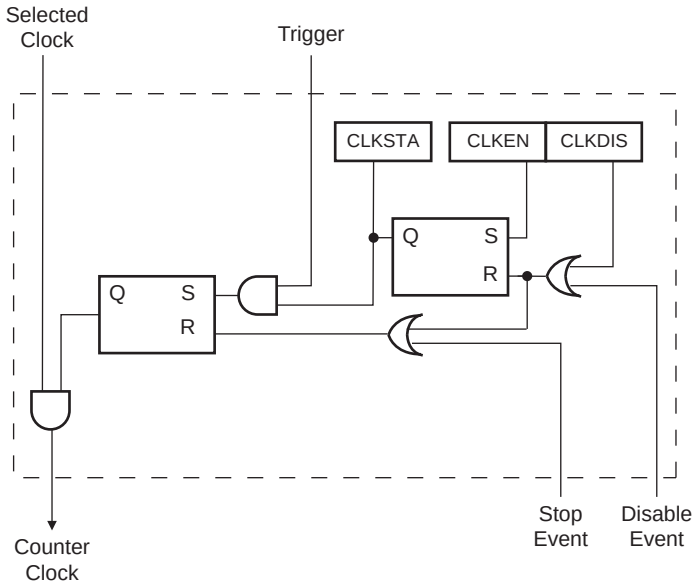


47.6.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 47-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the TC Channel Control Register (TC_CCR). In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in the 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 TC_CCR can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the TC_SR.
- 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 47-4. Clock Control



47.6.5 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.

47.6.6 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 the 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 bit ENETRG in the 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.

47.6.7 Capture Operating Mode

This mode is entered by clearing the WAVE bit in the TC_CMR.

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 47-6shows the configuration of the TC channel when programmed in Capture Mode.

47.6.8 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 field in the TC_CMR defines the TIOA selected edge for the loading of register A, and the LDRB field 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 bit) in the TC_SR. In this case, the old value is overwritten.

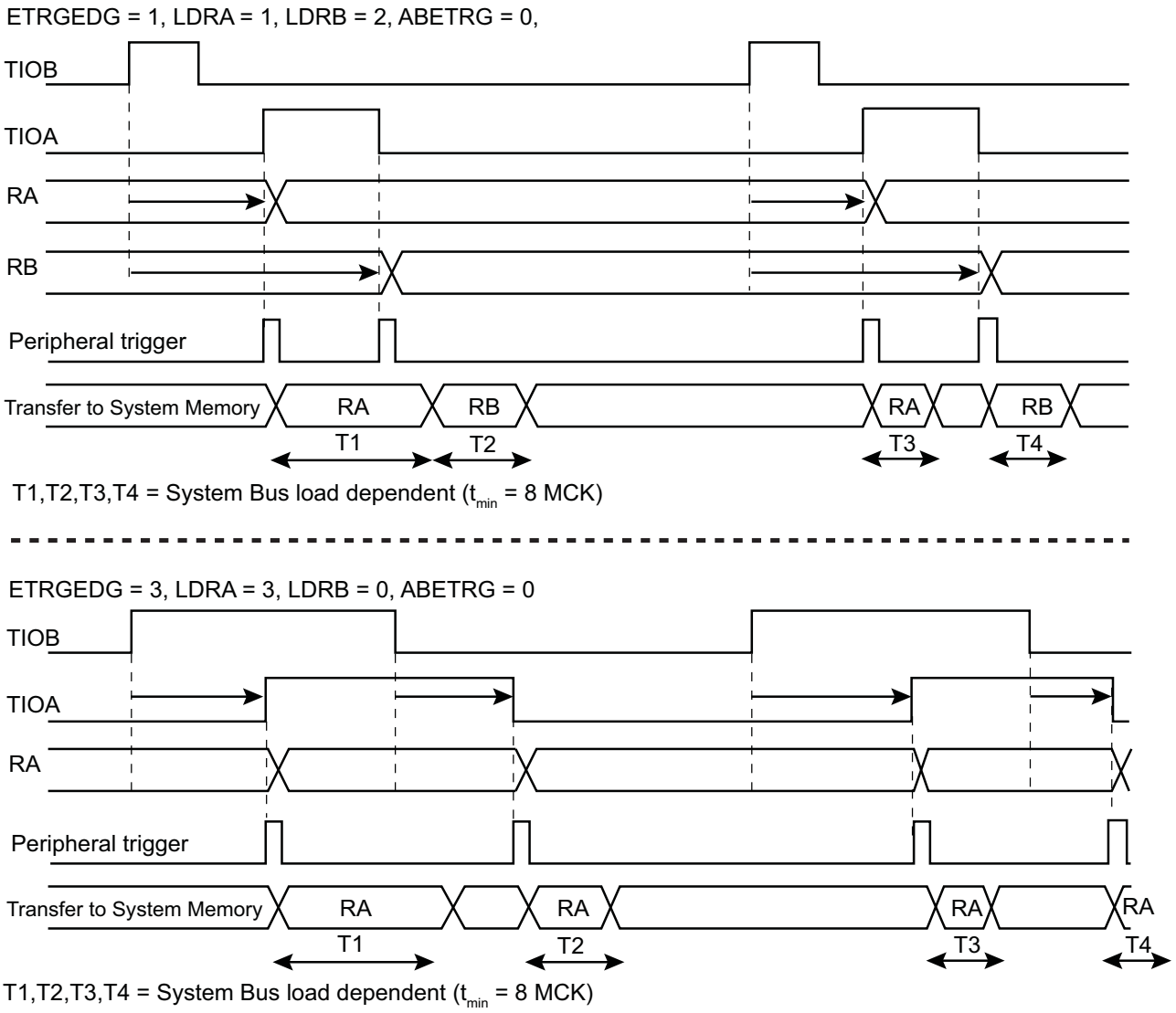
When DMA is used, the RAB register address must be configured as source address of the transfer. The RAB register provides the next unread value from Register A and Register B. It may be read by the DMA after a request has been triggered upon loading Register A or Register B.

47.6.9 Transfer with DMAC

The DMAC can only perform access from timer to system memory.

Figure 47-5 “Example of Transfer with DMAC” illustrates how the RA and RB registers can be loaded in the system memory without CPU intervention.

Figure 47-5. Example of Transfer with DMAC

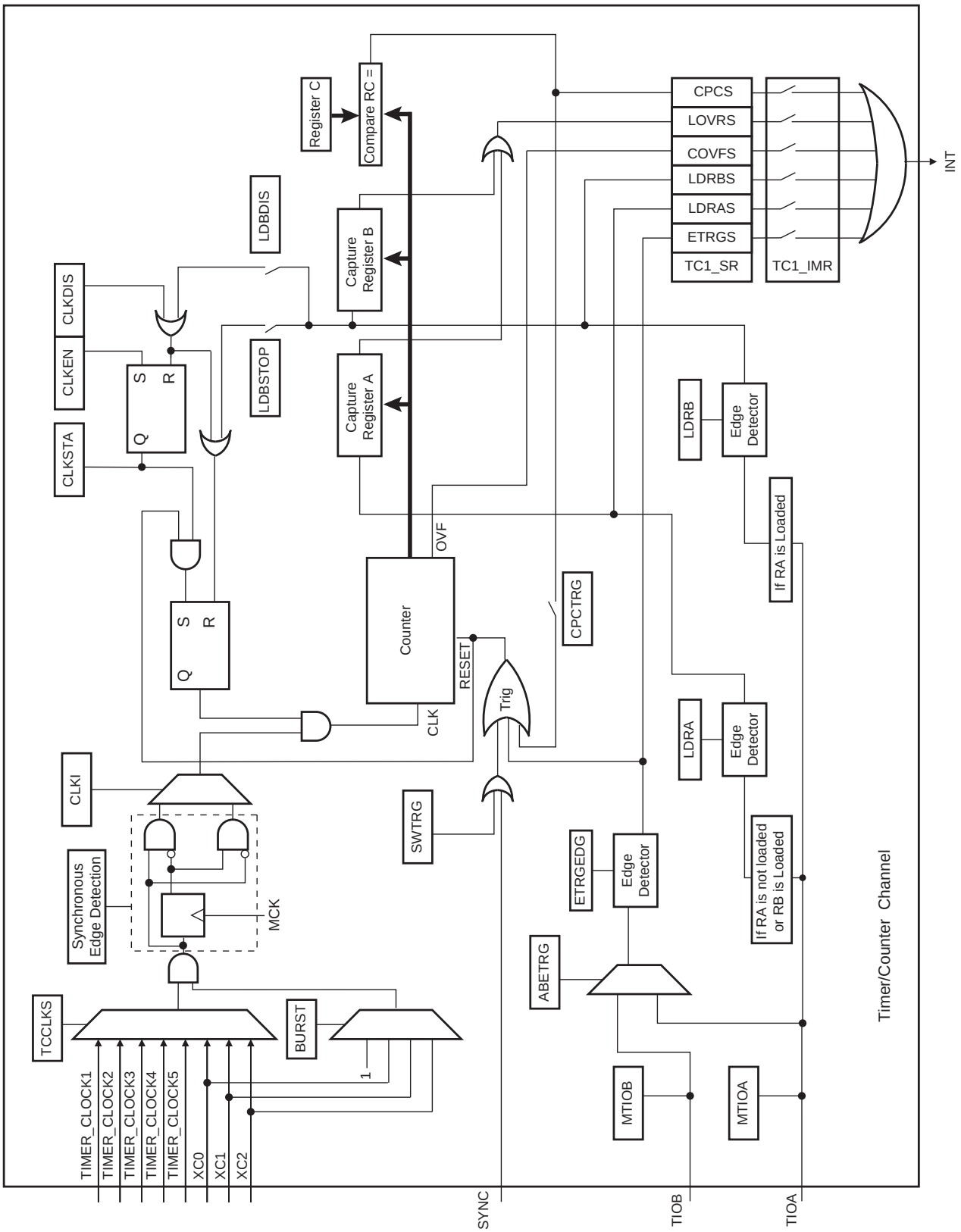


47.6.10 Trigger Conditions

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 selects TIOA or TIOB input signal as an external trigger . The External Trigger Edge Selection parameter (ETRGEDG field in TC_CMR) defines the edge (rising, falling, or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Figure 47-6. Capture Mode



47.6.11 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 47-7 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

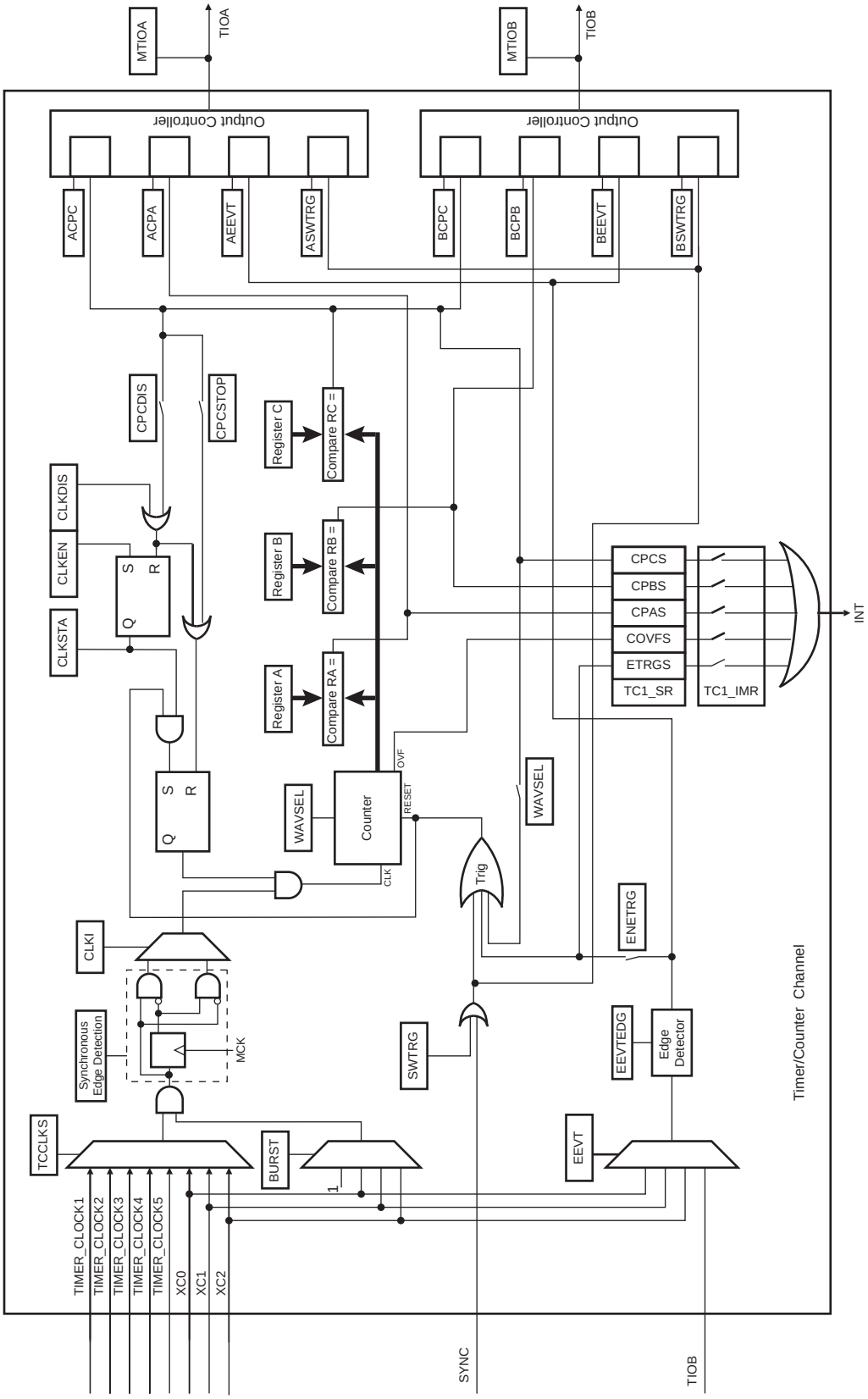
47.6.12 Waveform Selection

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 47-7. Waveform Mode



47.6.12.1 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 47-8.

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 47-9.

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 47-8. WAVSEL = 00 without trigger

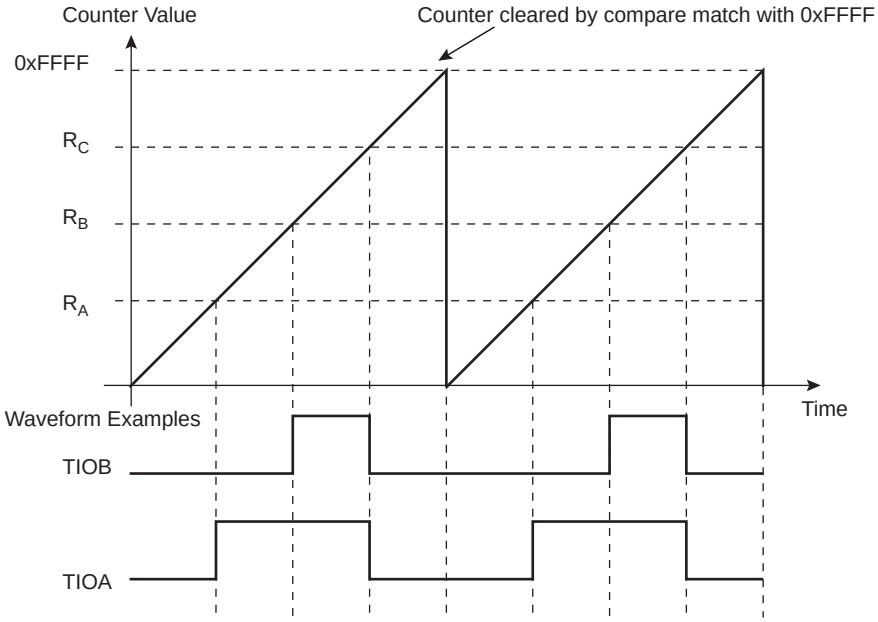
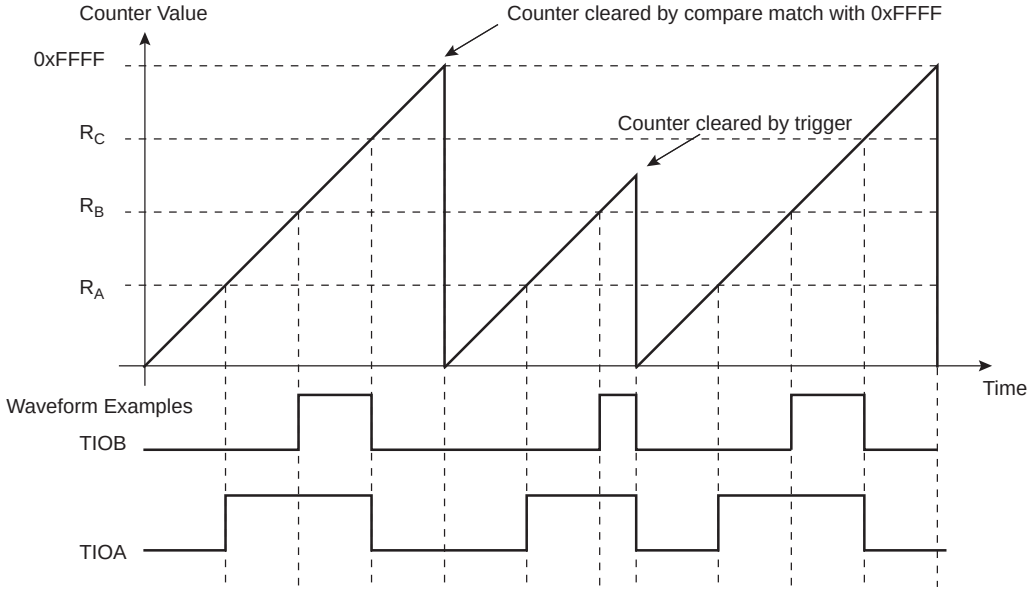


Figure 47-9. WAVSEL= 00 with Trigger



47.6.12.2 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 47-10.

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 47-11.

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 47-10.WAVSEL = 10 without Trigger

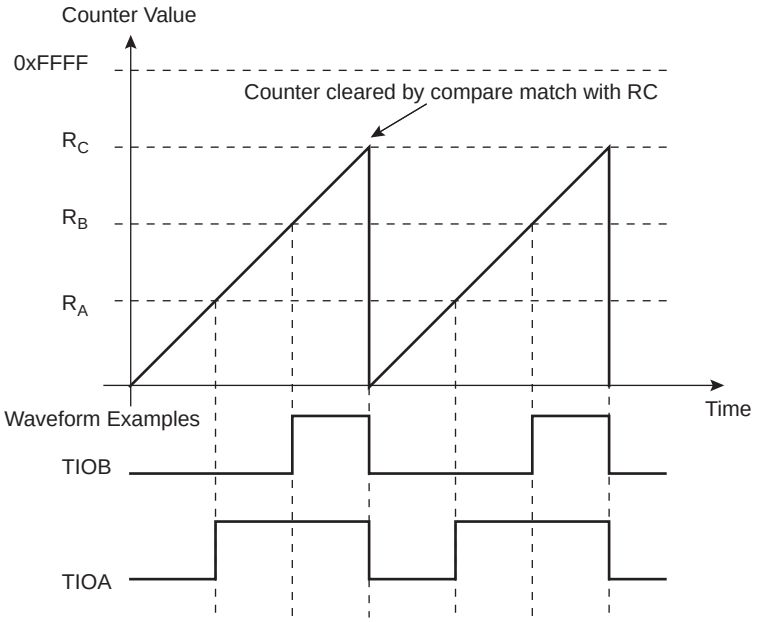
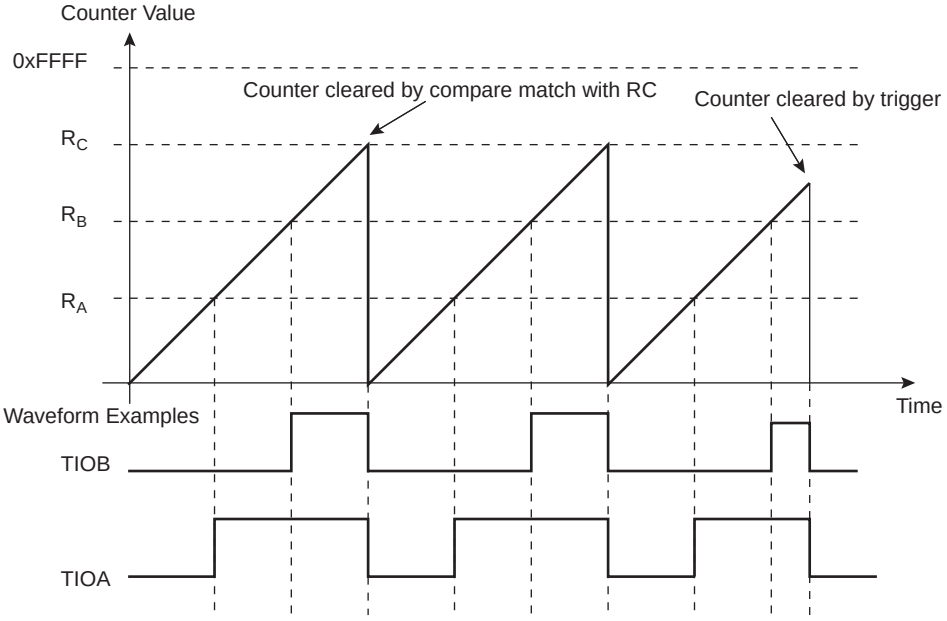


Figure 47-11.WAVSEL = 10 with Trigger



47.6.12.3 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 47-12.

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 47-13.

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 47-12.WAVSEL = 01 without Trigger

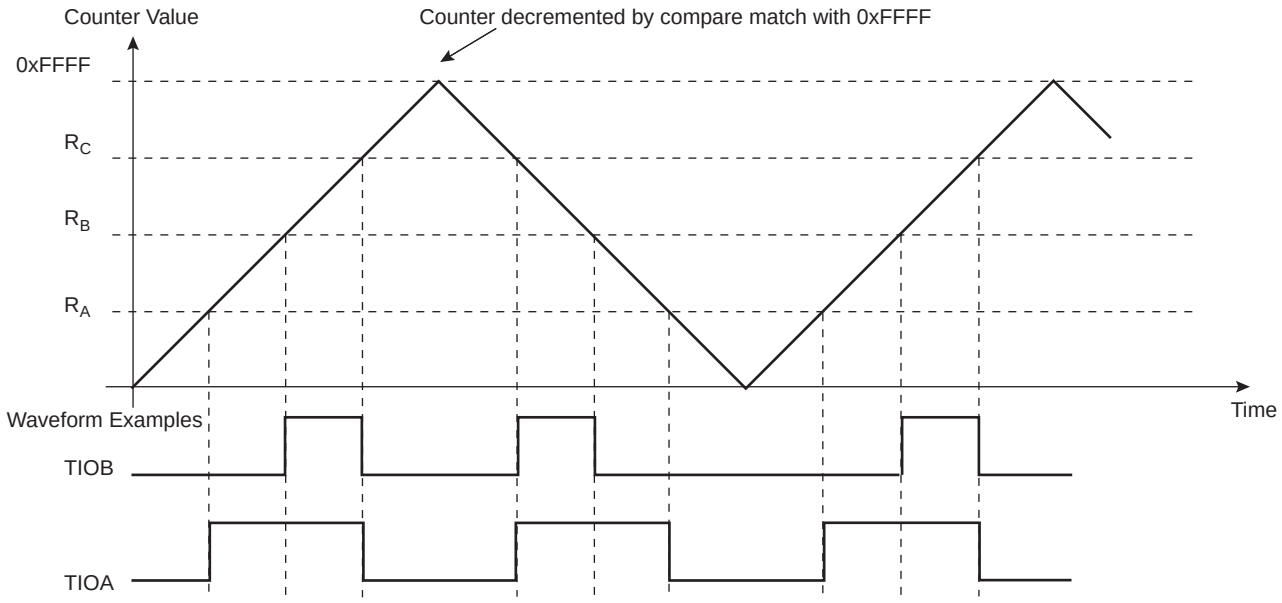
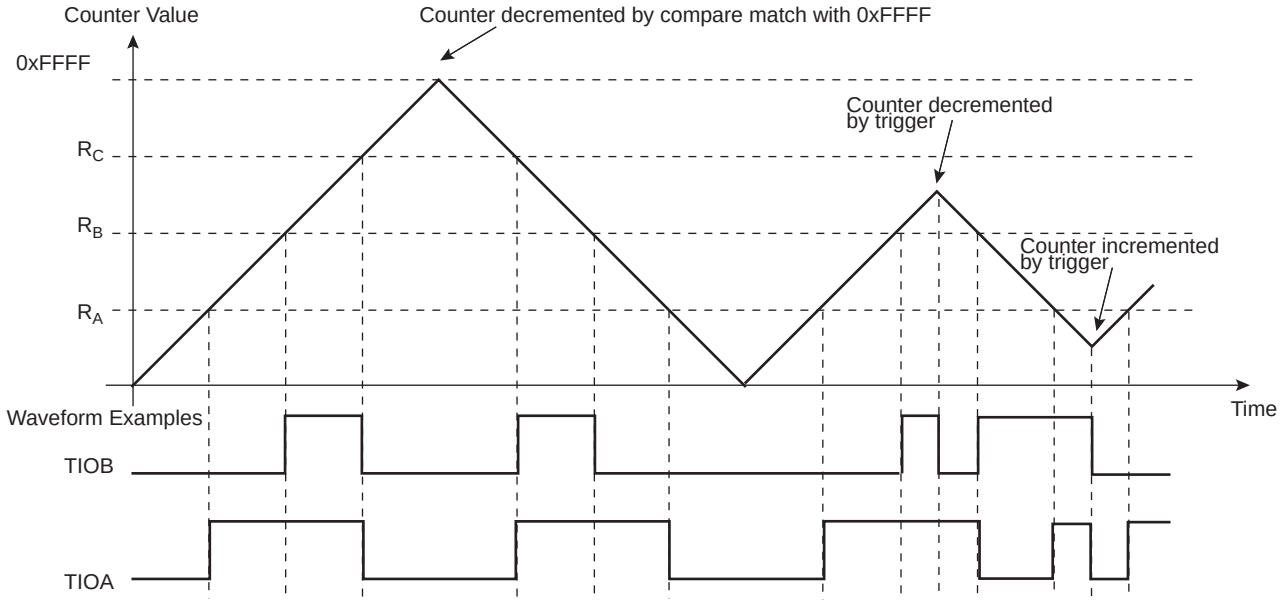


Figure 47-13.WAVSEL = 01 with Trigger



47.6.12.4 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 47-14.

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 47-15.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 47-14.WAVSEL = 11 without Trigger

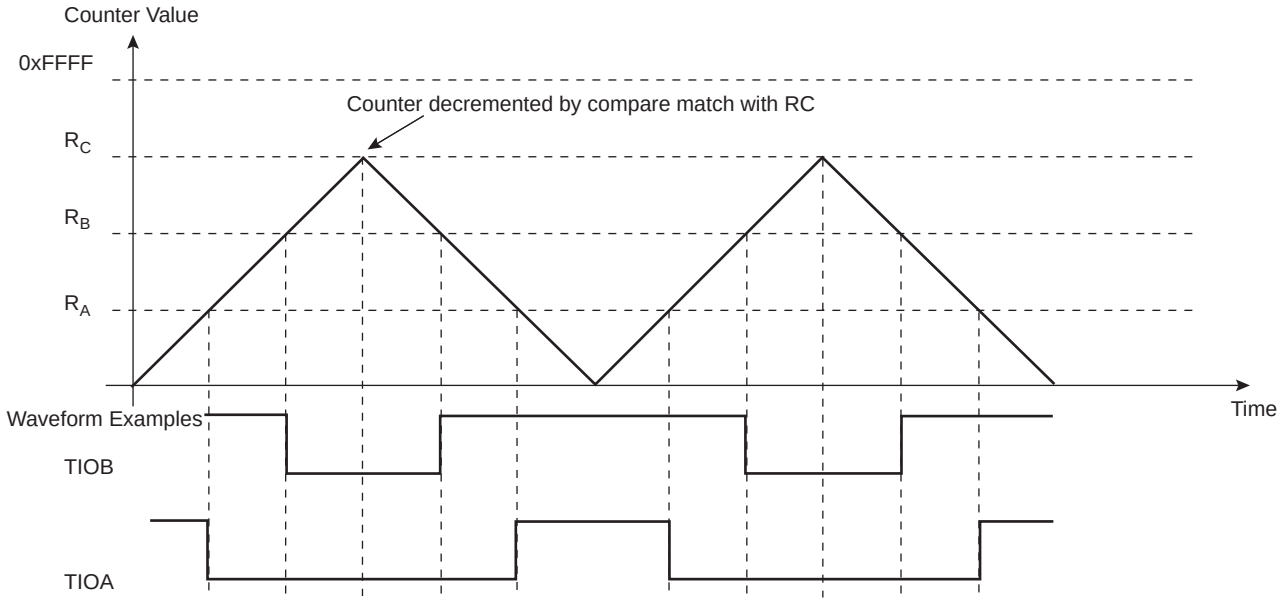
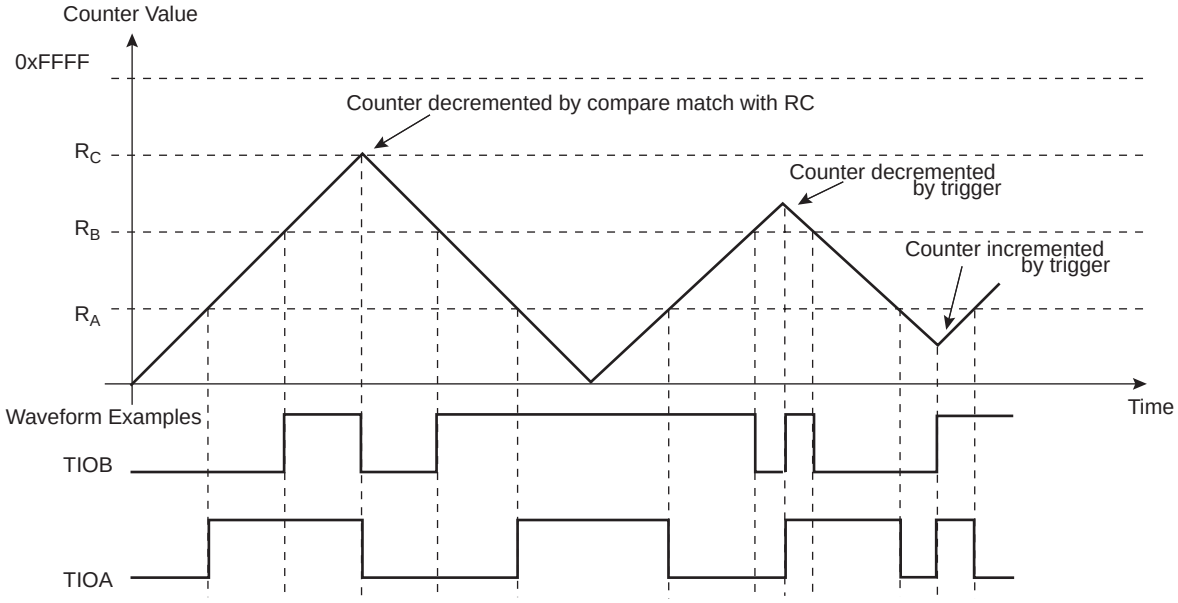


Figure 47-15.WAVSEL = 11 with Trigger



47.6.13 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 the 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.

47.6.14 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.

47.6.15 Quadrature Decoder Logic

47.6.15.1 Description

The quadrature decoder logic is driven by TIOA0, TIOB0, TIOB1 input pins and drives the timer/counter of channel 0 and 1. Channel 2 can be used as a time base in case of speed measurement requirements (refer to Figure 47-16 “Predefined Connection of the Quadrature Decoder with Timer Counters”).

When writing a 0 to bit QDEN of the TC_BMR, the quadrature decoder logic is totally transparent.

TIOA0 and TIOB0 are to be driven by the two dedicated quadrature signals from a rotary sensor mounted on the shaft of the off-chip motor.

A third signal from the rotary sensor can be processed through pin TIOB1 and is typically dedicated to be driven by an index signal if it is provided by the sensor. This signal is not required to decode the quadrature signals PHA, PHB.

Field TCCLKS of TC_CMRx must be configured to select XC0 input (i.e., 0x101). Field TC0XC0S has no effect as soon as quadrature decoder is enabled.

Either speed or position/revolution can be measured. Position channel 0 accumulates the edges of PHA, PHB input signals giving a high accuracy on motor position whereas channel 1 accumulates the index pulses of the sensor, therefore the number of rotations. Concatenation of both values provides a high level of precision on motion system position.

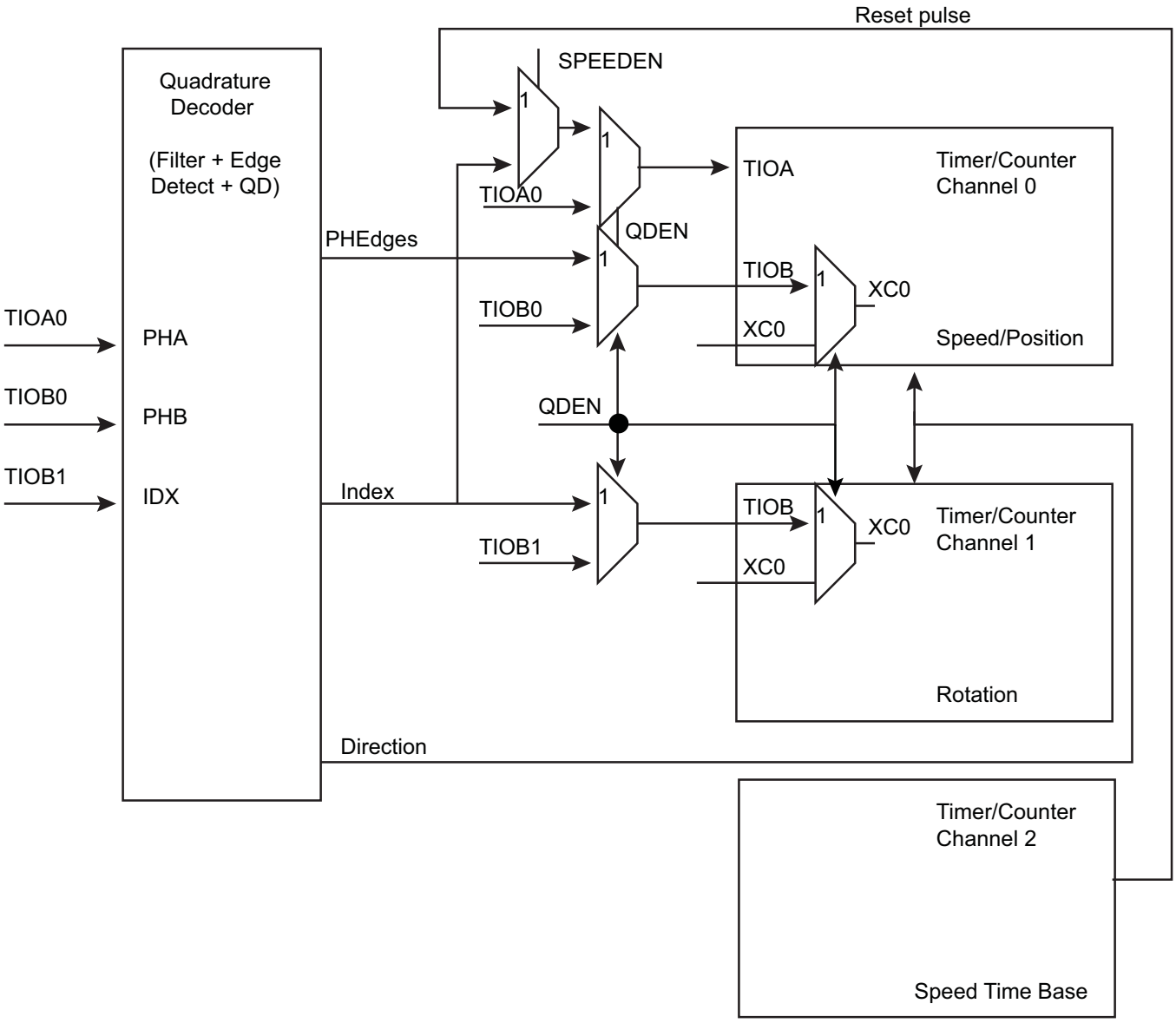
In speed mode, position cannot be measured but revolution can be measured.

Inputs from the rotary sensor can be filtered prior to down-stream processing. Accommodation of input polarity, phase definition and other factors are configurable.

Interruptions can be generated on different events.

A compare function (using TC_RC register) is available on channel 0 (speed/position) or channel 1 (rotation) and can generate an interrupt by means of the CPCS flag in the TC_SRx.

Figure 47-16.Predefined Connection of the Quadrature Decoder with Timer Counters



47.6.15.2 Input Pre-processing

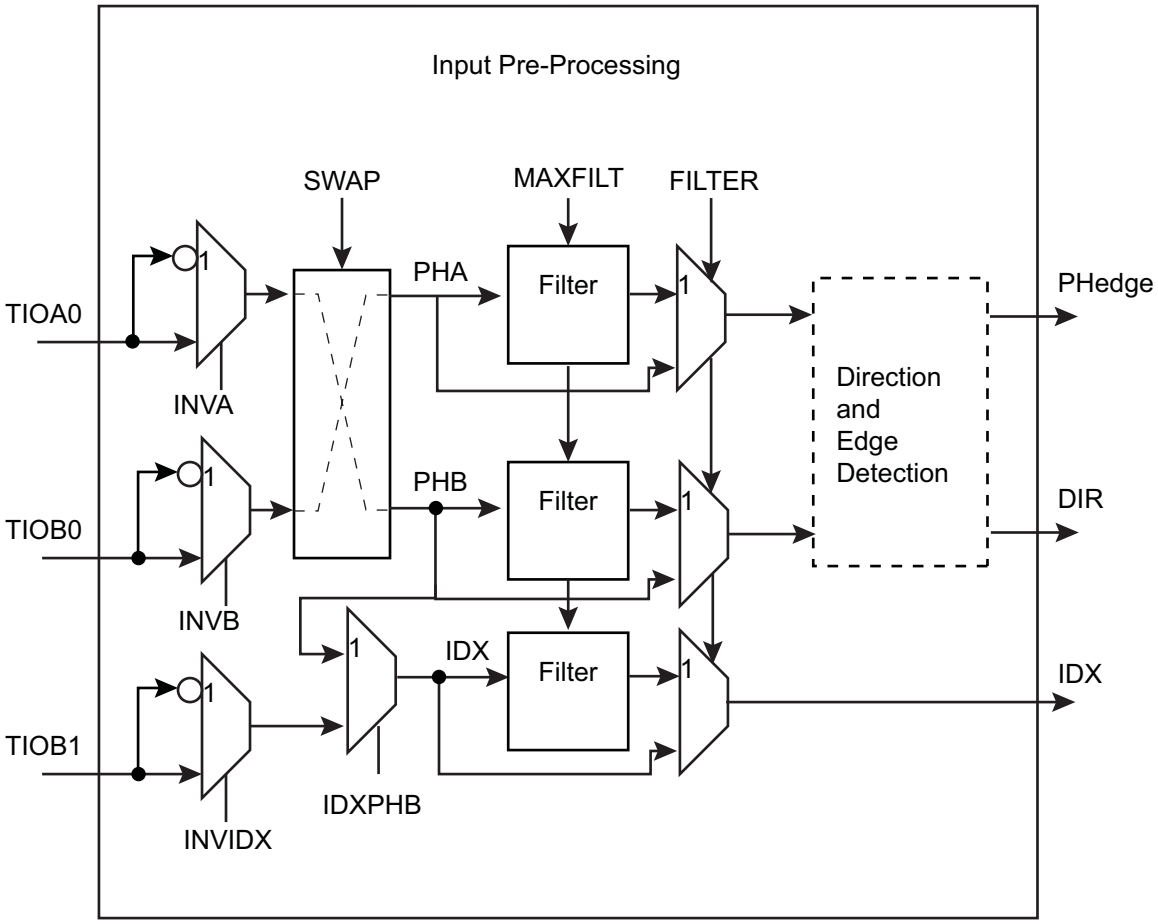
Input pre-processing consists of capabilities to take into account rotary sensor factors such as polarities and phase definition followed by configurable digital filtering.

Each input can be negated and swapping PHA, PHB is also configurable.

The MAXFILT field in the TC_BMR is used to configure a minimum duration for which the pulse is stated as valid. When the filter is active, pulses with a duration lower than MAXFILT +1 * tMCK ns are not passed to down-stream logic.

Filters can be disabled using the FILTER bit in the TC_BMR.

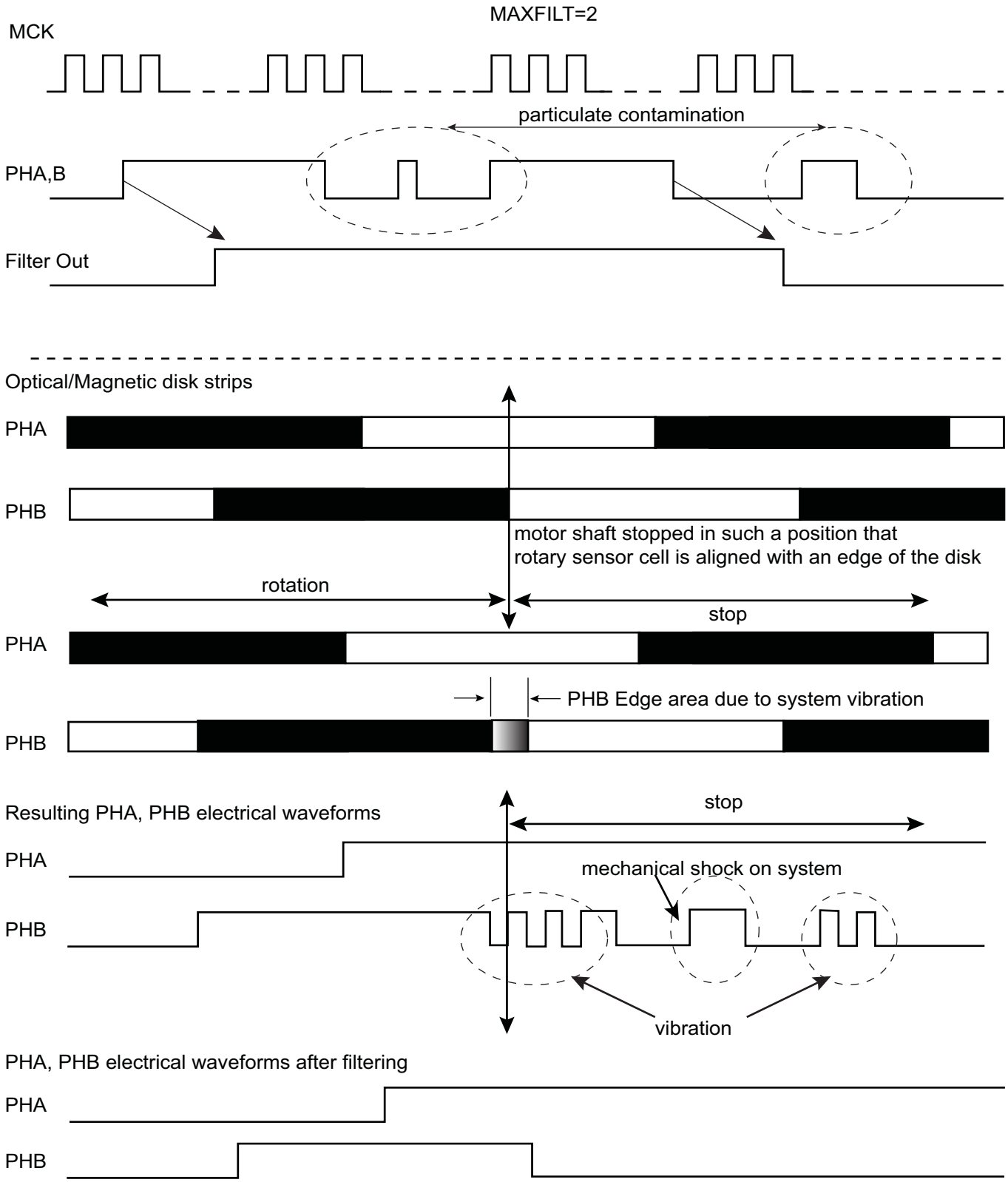
Figure 47-17.Input Stage



Input filtering can efficiently remove spurious pulses that might be generated by the presence of particulate contamination on the optical or magnetic disk of the rotary sensor.

Spurious pulses can also occur in environments with high levels of electro-magnetic interference. Or, simply if vibration occurs even when rotation is fully stopped and the shaft of the motor is in such a position that the beginning of one of the reflective or magnetic bars on the rotary sensor disk is aligned with the light or magnetic (Hall) receiver cell of the rotary sensor. Any vibration can make the PHA, PHB signals toggle for a short duration.

Figure 47-18.Filtering Examples



47.6.15.3 Direction Status and Change Detection

After filtering, the quadrature signals are analyzed to extract the rotation direction and edges of the two quadrature signals detected in order to be counted by timer/counter logic downstream.

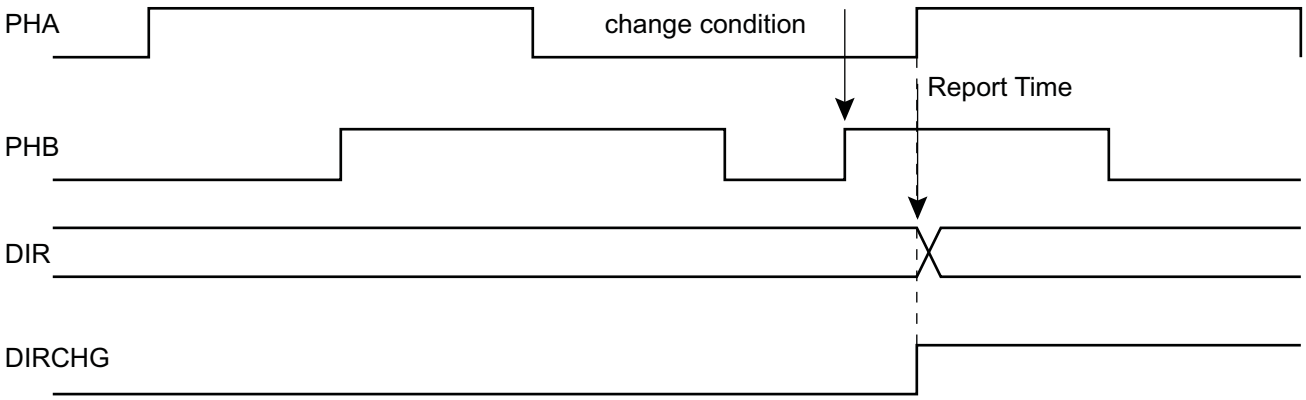
The direction status can be directly read at anytime in the TC_QISR. The polarity of the direction flag status depends on the configuration written in TC_BMR. INVA, INVB, INVIDX, SWAP modify the polarity of DIR flag.

Any change in rotation direction is reported in theTC_QISR and can generate an interrupt.

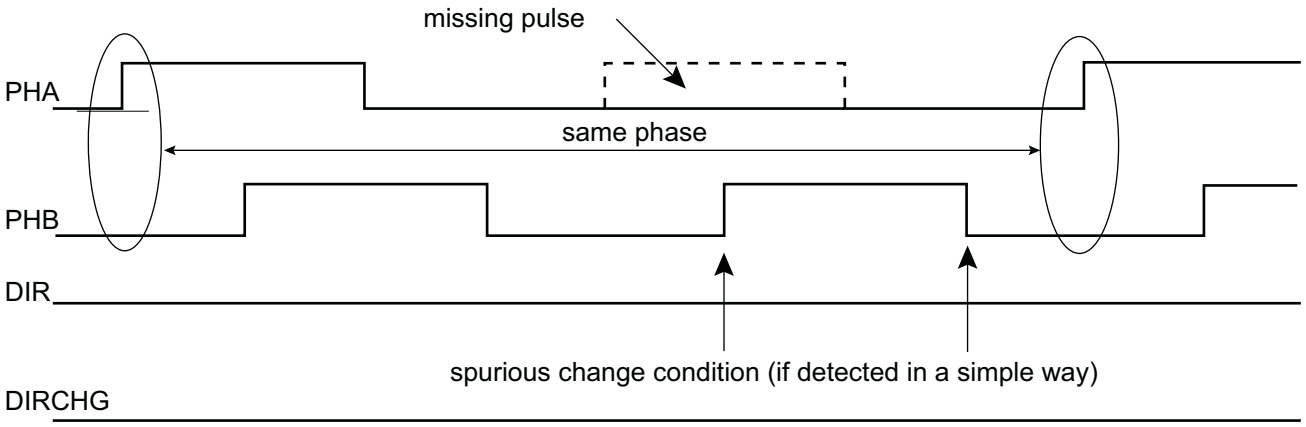
The direction change condition is reported as soon as two consecutive edges on a phase signal have sampled the same value on the other phase signal and there is an edge on the other signal. The two consecutive edges of one phase signal sampling the same value on other phase signal is not sufficient to declare a direction change, for the reason that particulate contamination may mask one or more reflective bars on the optical or magnetic disk of the sensor. (Refer to Figure 47-19 “Rotation Change Detection” for waveforms.)

Figure 47-19.Rotation Change Detection

Direction Change under normal conditions



No direction change due to particulate contamination masking a reflective bar

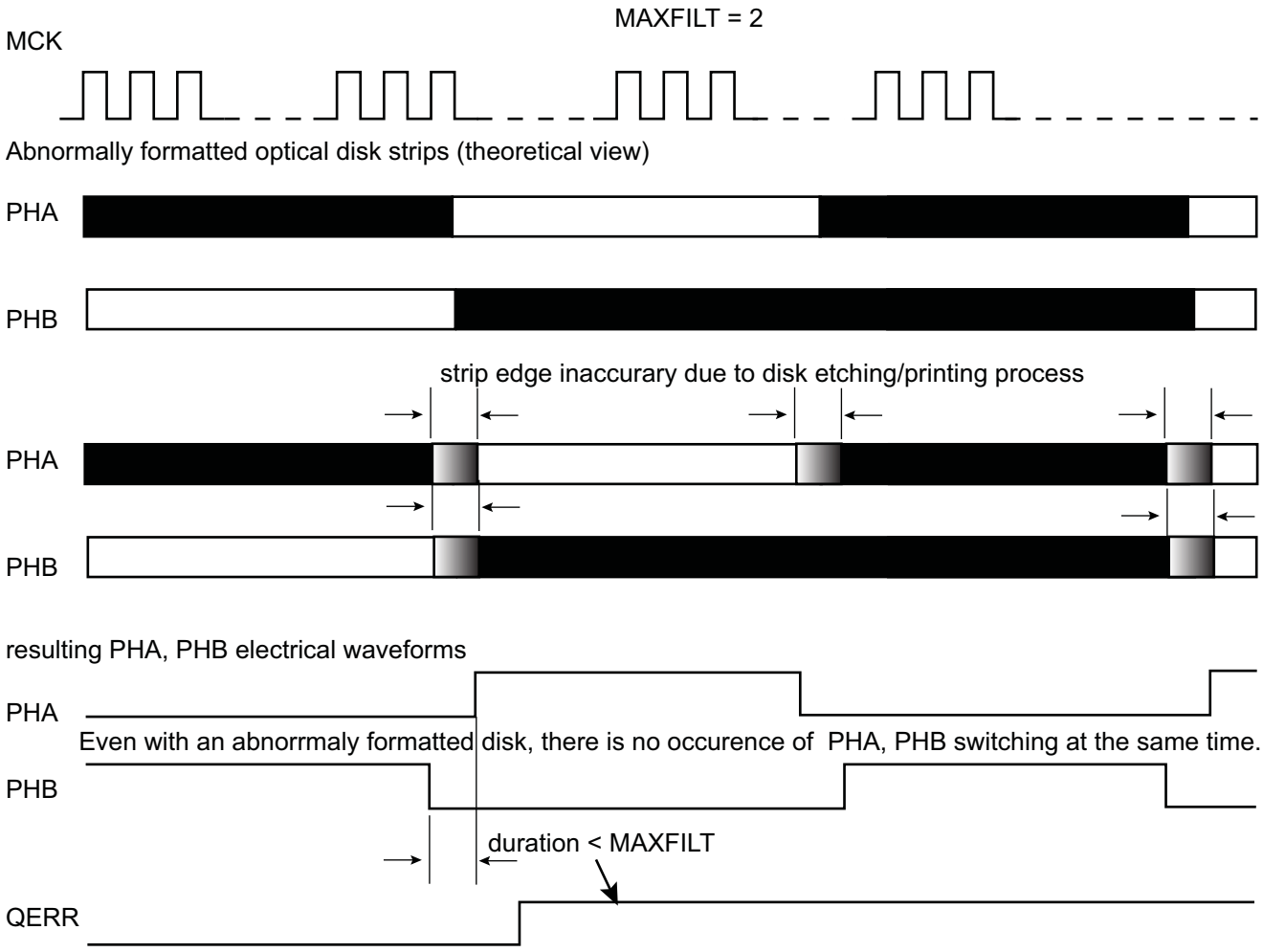


The direction change detection is disabled when QDTRANS is set in the TC_BMR. In this case the DIR flag report must not be used.

A quadrature error is also reported by the quadrature decoder logic via the QERR flag in the TC_QISR. This error is reported if the time difference between two edges on PHA, PHB is lower than a predefined value. This predefined value

is configurable and corresponds to (MAXFILT + 1) * tMCK ns. After being filtered there is no reason to have two edges closer than (MAXFILT + 1) * tMCK ns under normal mode of operation.

Figure 47-20.Quadrature Error Detection



MAXFILT must be tuned according to several factors such as the system clock frequency (MCK), type of rotary sensor and rotation speed to be achieved.

47.6.15.4 Position and Rotation Measurement

When the POSEN bit is set in the TC_BMR, the motor axis position is processed on channel 0 (by means of the PHA, PHB edge detections) and the number of motor revolutions are recorded on channel 1 if the IDX signal is provided on the TIOB1 input. The position measurement can be read in the TC_CV0 register and the rotation measurement can be read in the TC_CV1 register.

Channel 0 and 1 must be configured in capture mode (WAVE = 0 in TC_CMR0).

In parallel, the number of edges are accumulated on timer/counter channel 0 and can be read on the TC_CV0 register.

Therefore, the accurate position can be read on both TC_CV registers and concatenated to form a 32-bit word.

The timer/counter channel 0 is cleared for each increment of IDX count value.

Depending on the quadrature signals, the direction is decoded and allows to count up or down in timer/counter channels 0 and 1. The direction status is reported on TC_QISR.

47.6.15.5 Speed Measurement

When SPEEDEN is set in the TC_BMR, the speed measure is enabled on channel 0.

A time base must be defined on channel 2 by writing the TC_RC2 period register. Channel 2 must be configured in waveform mode (WAVE bit set) in TC_CMR2 register. The WAVSEL field must be defined with 0x10 to clear the counter by comparison and matching with TC_RC value. Field ACPC must be defined at 0x11 to toggle TIOA output.

This time base is automatically fed back to TIOA of channel 0 when QDEN and SPEEDEN are set.

Channel 0 must be configured in capture mode (WAVE = 0 in TC_CMR0). The ABETRG bit of TC_CMR0 must be configured at 1 to select TIOA as a trigger for this channel.

EDGTRG can be set to 0x01, to clear the counter on a rising edge of the TIOA signal and field LDRA must be set accordingly to 0x01, to load TC_RA0 at the same time as the counter is cleared (LDRB must be set to 0x01). As a consequence, at the end of each time base period the differentiation required for the speed calculation is performed.

The process must be started by configuring bits CLKEN and SWTRG in the TC_CCR.

The speed can be read on field RA in register TC_RA0.

Channel 1 can still be used to count the number of revolutions of the motor.

47.6.16 2-bit Gray Up/Down Counter for Stepper Motor

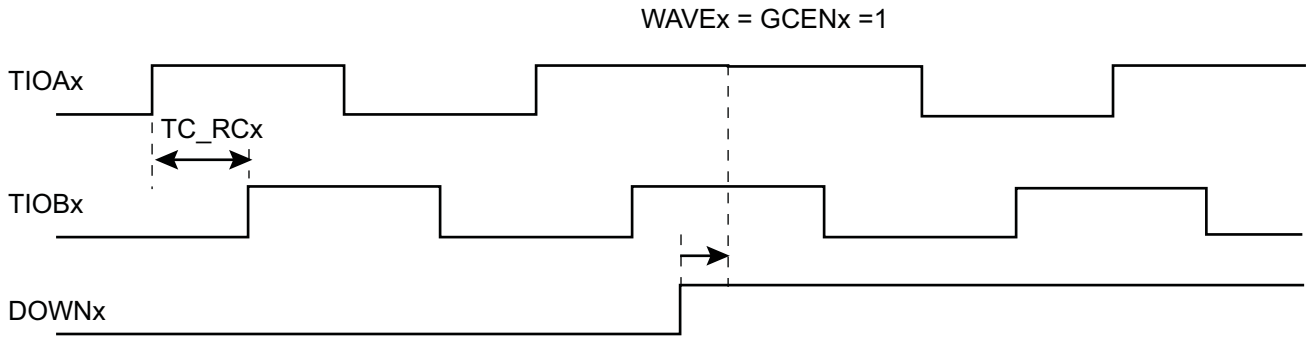
Each channel can be independently configured to generate a 2-bit gray count waveform on corresponding TIOA, TIOB outputs by means of the GCEN bit in TC_SMMRx.

Up or Down count can be defined by writing bit DOWN in TC_SMMRx.

It is mandatory to configure the channel in WAVE mode in the TC_CMR.

The period of the counters can be programmed in TC_RCx.

Figure 47-21.2-bit Gray Up/Down Counter



47.6.17 Register Write Protection

To prevent any single software error from corrupting TC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the TC Write Protection Mode Register (TC_WPMR).

The following registers can be protected:

- TC Block Mode Register
- TC Channel Mode Register: Capture Mode
- TC Channel Mode Register: Waveform Mode
- TC Stepper Motor Mode Register
- TC Register A
- TC Register B
- TC Register C

47.7 Timer Counter (TC) User Interface

Table 47-5. 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	Stepper Motor Mode Register	TC_SMMR	Read/Write	0
0x00 + channel * 0x40 + 0x0C	Register AB	TC_RAB	Read-only	0
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	QDEC Interrupt Enable Register	TC_QIER	Write-only	–
0xCC	QDEC Interrupt Disable Register	TC_QIDR	Write-only	–
0xD0	QDEC Interrupt Mask Register	TC_QIMR	Read-only	0
0xD4	QDEC Interrupt Status Register	TC_QISR	Read-only	0
0xD8	Reserved	–	–	–
0xE4	Write Protection Mode Register	TC_WPMR	Read/Write	0
0xE8–0xFC	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if WAVE = 0

47.7.1 TC Channel Control Register

Name: TC_CCRx [x=0..2]

Address: 0xF0010000 (0)[0], 0xF0010040 (0)[1], 0xF0010080 (0)[2], 0xF8014000 (1)[0], 0xF8014040 (1)[1], 0xF8014080 (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.

47.7.2 TC Channel Mode Register: Capture Mode

Name: TC_CMRx [x=0..2] (WAVE = 0)

Address: 0xF0010004 (0)[0], 0xF0010044 (0)[1], 0xF0010084 (0)[2], 0xF8014004 (1)[0], 0xF8014044 (1)[1], 0xF8014084 (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		

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **TCCLKS: Clock Selection**

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: internal TIMER_CLOCK1 clock signal (from PMC)
1	TIMER_CLOCK2	Clock selected: internal TIMER_CLOCK2 clock signal (from PMC)
2	TIMER_CLOCK3	Clock selected: internal TIMER_CLOCK3 clock signal (from PMC)
3	TIMER_CLOCK4	Clock selected: internal TIMER_CLOCK4 clock signal (from PMC)
4	TIMER_CLOCK5	Clock selected: internal TIMER_CLOCK5 clock signal (from PMC)
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

47.7.3 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		

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **TCCLKS: Clock Selection**

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: internal TIMER_CLOCK1 clock signal (from PMC)
1	TIMER_CLOCK2	Clock selected: internal TIMER_CLOCK2 clock signal (from PMC)
2	TIMER_CLOCK3	Clock selected: internal TIMER_CLOCK3 clock signal (from PMC)
3	TIMER_CLOCK4	Clock selected: internal TIMER_CLOCK4 clock signal (from PMC)
4	TIMER_CLOCK5	Clock selected: internal TIMER_CLOCK5 clock signal (from PMC)
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.

1: The external event resets the counter and starts the counter clock.

Note: Whatever the value programmed in ENETRG, the selected external event only controls the TIOA output and TIOB if not used as input (trigger event input or other input used).

- **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

47.7.4 TC Stepper Motor Mode Register

Name: TC_SMMRx [x=0..2]

Address: 0xF0010008 (0)[0], 0xF0010048 (0)[1], 0xF0010088 (0)[2], 0xF8014008 (1)[0], 0xF8014048 (1)[1], 0xF8014088 (1)[2]

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **GCEN: Gray Count Enable**

0: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by internal counter of channel x.

1: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by a 2-bit gray counter.

- **DOWN: DOWN Count**

0: Up counter.

1: Down counter.

47.7.5 TC Register AB

Name: TC_RABx [x=0..2]

Address: 0xF001000C (0)[0], 0xF001004C (0)[1], 0xF001008C (0)[2], 0xF801400C (1)[0], 0xF801404C (1)[1], 0xF801408C (1)[2]

Access: Read-only

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

- **RAB: Register A or Register B**

RAB contains the next unread capture Register A or Register B value in real time. It is usually read by the DMA after a request due to a valid load edge on TIOA.

When DMA is used, the RAB register address must be configured as source address of the transfer.

47.7.6 TC Counter Value Register

Name: TC_CVx [x=0..2]

Address: 0xF0010010 (0)[0], 0xF0010050 (0)[1], 0xF0010090 (0)[2], 0xF8014010 (1)[0], 0xF8014050 (1)[1], 0xF8014090 (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.

47.7.7 TC Register A

Name: TC_RAx [x=0..2]

Address: 0xF0010014 (0)[0], 0xF0010054 (0)[1], 0xF0010094 (0)[2], 0xF8014014 (1)[0], 0xF8014054 (1)[1], 0xF8014094 (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							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RA: Register A**

RA contains the Register A value in real time.

47.7.8 TC Register B

Name: TC_RBx [x=0..2]

Address: 0xF0010018 (0)[0], 0xF0010058 (0)[1], 0xF0010098 (0)[2], 0xF8014018 (1)[0], 0xF8014058 (1)[1], 0xF8014098 (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							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RB: Register B**

RB contains the Register B value in real time.

47.7.9 TC Register C

Name: TC_RCx [x=0..2]

Address: 0xF001001C (0)[0], 0xF001005C (0)[1], 0xF001009C (0)[2], 0xF801401C (1)[0], 0xF801405C (1)[1], 0xF801409C (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							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RC: Register C**

RC contains the Register C value in real time.

47.7.10 TC Status Register

Name: TC_SRx [x=0..2]

Address: 0xF0010020 (0)[0], 0xF0010060 (0)[1], 0xF00100A0 (0)[2], 0xF8014020 (1)[0], 0xF8014060 (1)[1], 0xF80140A0 (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.

47.7.11 TC Interrupt Enable Register

Name: TC_IERx [x=0..2]

Address: 0xF0010024 (0)[0], 0xF0010064 (0)[1], 0xF00100A4 (0)[2], 0xF8014024 (1)[0], 0xF8014064 (1)[1], 0xF80140A4 (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.

47.7.12 TC Interrupt Disable Register

Name: TC_IDRx [x=0..2]

Address: 0xF0010028 (0)[0], 0xF0010068 (0)[1], 0xF00100A8 (0)[2], 0xF8014028 (1)[0], 0xF8014068 (1)[1], 0xF80140A8 (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.

47.7.13 TC Interrupt Mask Register

Name: TC_IMRx [x=0..2]

Address: 0xF001002C (0)[0], 0xF001006C (0)[1], 0xF00100AC (0)[2], 0xF801402C (1)[0], 0xF801406C (1)[1], 0xF80140AC (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.

47.7.14 TC Block Control Register

Name: TC_BCR

Address: 0xF00100C0 (0), 0xF80140C0 (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.

47.7.15 TC Block Mode Register

Name: TC_BMR

Address: 0xF00100C4 (0), 0xF80140C4 (1)

Access: Read/Write

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

23	22	21	20	19	18	17	16
MAXFILT				FILTER	–	IDXPHB	SWAP

15	14	13	12	11	10	9	8
INVIDX	INVB	INVA	EDGPHA	QDTRANS	SPEEDEN	POSEN	QDEN

7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **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	TIOA0	Signal connected to XC2: TIOA0
3	TIOA1	Signal connected to XC2: TIOA1

- **QDEN: Quadrature Decoder ENabled**

0: Disabled.

1: Enables the quadrature decoder logic (filter, edge detection and quadrature decoding).

Quadrature decoding (direction change) can be disabled using QDTRANS bit.

One of the POSEN or SPEEDEN bits must be also enabled.

- **POSEN: POSition ENabled**

0: Disable position.

1: Enables the position measure on channel 0 and 1.

- **SPEEDEN: SPEED ENabled**

0: Disabled.

1: Enables the speed measure on channel 0, the time base being provided by channel 2.

- **QDTRANS: Quadrature Decoding TRANSparent**

0: Full quadrature decoding logic is active (direction change detected).

1: Quadrature decoding logic is inactive (direction change inactive) but input filtering and edge detection are performed.

- **EDGPHA: EDGe on PHA count mode**

0: Edges are detected on PHA only.

1: Edges are detected on both PHA and PHB.

- **INVA: INVerted phA**

0: PHA (TIOA0) is directly driving quadrature decoder logic.

1: PHA is inverted before driving quadrature decoder logic.

- **INVB: INVerted phB**

0: PHB (TIOB0) is directly driving quadrature decoder logic.

1: PHB is inverted before driving quadrature decoder logic.

- **SWAP: SWAP PHA and PHB**

0: No swap between PHA and PHB.

1: Swap PHA and PHB internally, prior to driving quadrature decoder logic.

- **INVIDX: INVerted InDeX**

0: IDX (TIOA1) is directly driving quadrature logic.

1: IDX is inverted before driving quadrature logic.

- **IDXPHB: InDeX pin is PHB pin**

0: IDX pin of the rotary sensor must drive TIOA1.

1: IDX pin of the rotary sensor must drive TIOB0.

- **FILTER: Glitch Filter**

0: IDX,PHA, PHB input pins are not filtered.

1: IDX,PHA, PHB input pins are filtered using MAXFILT value.

- **MAXFILT: MAXimum FILTer**

1.. 63: Defines the filtering capabilities.

Pulses with a period shorter than MAXFILT+1 MCK clock cycles are discarded.

47.7.16 TC QDEC Interrupt Enable Register

Name: TC_QIER

Address: 0xF00100C8 (0), 0xF80140C8 (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
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: InDeX**

0: No effect.

1: Enables the interrupt when a rising edge occurs on IDX input.

- **DIRCHG: DIRection CHanGe**

0: No effect.

1: Enables the interrupt when a change on rotation direction is detected.

- **QERR: Quadrature ERRor**

0: No effect.

1: Enables the interrupt when a quadrature error occurs on PHA,PHB.

47.7.17 TC QDEC Interrupt Disable Register

Name: TC_QIDR

Address: 0xF00100CC (0), 0xF80140CC (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
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: InDeX**

0: No effect.

1: Disables the interrupt when a rising edge occurs on IDX input.

- **DIRCHG: DIRection CHanGe**

0: No effect.

1: Disables the interrupt when a change on rotation direction is detected.

- **QERR: Quadrature ERRor**

0: No effect.

1: Disables the interrupt when a quadrature error occurs on PHA, PHB.

47.7.18 TC QDEC Interrupt Mask Register

Name: TC_QIMR

Address: 0xF00100D0 (0), 0xF80140D0 (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
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: InDeX**

0: The interrupt on IDX input is disabled.

1: The interrupt on IDX input is enabled.

- **DIRCHG: DIRection CHanGe**

0: The interrupt on rotation direction change is disabled.

1: The interrupt on rotation direction change is enabled.

- **QERR: Quadrature ERRor**

0: The interrupt on quadrature error is disabled.

1: The interrupt on quadrature error is enabled.

47.7.19 TC QDEC Interrupt Status Register

Name: TC_QISR

Address: 0xF00100D4 (0), 0xF80140D4 (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
–	–	–	–	–	–	–	DIR
7	6	5	4	3	2	1	0
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: InDeX**

0: No Index input change since the last read of TC_QISR.

1: The IDX input has changed since the last read of TC_QISR.

- **DIRCHG: DIRection CHanGe**

0: No change on rotation direction since the last read of TC_QISR.

1: The rotation direction changed since the last read of TC_QISR.

- **QERR: Quadrature ERRor**

0: No quadrature error since the last read of TC_QISR.

1: A quadrature error occurred since the last read of TC_QISR.

- **DIR: DIRection**

Returns an image of the actual rotation direction.

47.7.20 TC Write Protection Mode Register

Name: TC_WPMR

Address: 0xF00100E4 (0), 0xF80140E4 (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 Protect Enable**

0: Disables the Write Protect if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

1: Enables the Write Protect if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

See Section 47.6.17 “Register Write Protection” for list of write-protected registers.

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x54494D	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

48. Pulse Width Modulation Controller (PWM)

48.1 Description

The PWM macrocell controls 4 channels independently. Each channel controls two complementary square output waveforms. Characteristics of the output waveforms such as period, duty-cycle, polarity and dead-times (also called dead-bands or non-overlapping times) are configured 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 master clock (MCK).

All PWM macrocell accesses are made through registers mapped on the peripheral bus. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period, the duty-cycle or the dead-times.

Channels can be linked together as synchronous channels to be able to update their duty-cycle or dead-times at the same time.

The PWM macrocell provides 8 independent comparison units capable of comparing a programmed value to the counter of the synchronous channels (counter of channel 0). These comparisons are intended to generate software interrupts, to trigger pulses on the 2 independent event lines (in order to synchronize ADC conversions with a lot of flexibility independently of the PWM outputs).

The PWM outputs can be overridden synchronously or asynchronously to their channel counter.

The PWM block provides a fault protection mechanism with 1 fault inputs**,** capable to detect a fault condition and to override the PWM outputs asynchronously (outputs forced to ‘0’, ‘1’).

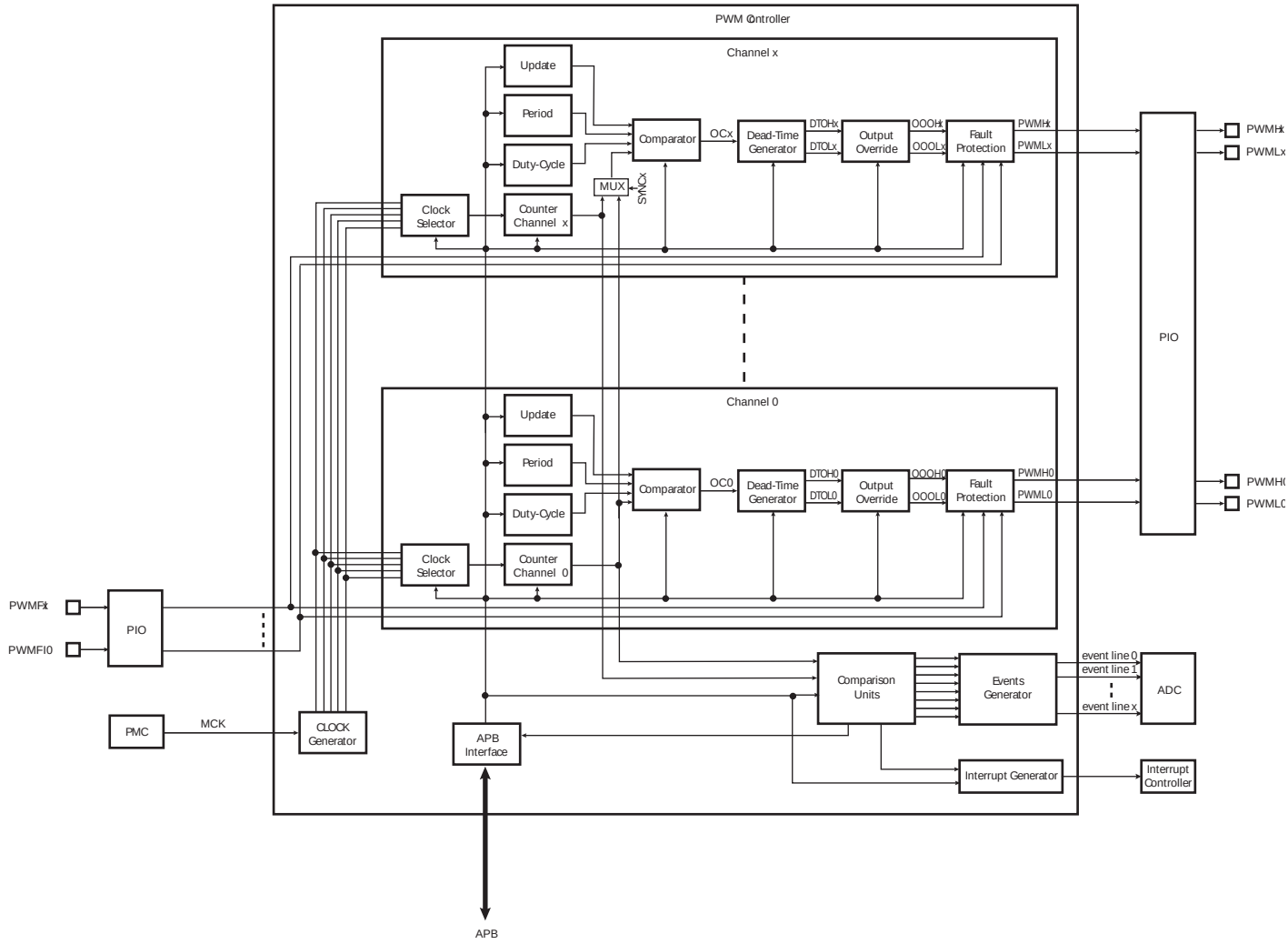
For safety usage, some configuration registers are write-protected.

48.2 Embedded Characteristics

- 4 Channels
- 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 16-bit Counter for Each Channel
  - Independent Complementary Outputs with 12-bit Dead-Time Generator (Also Called Dead-Band or Non-Overlapping Time) for Each Channel
  - Independent Enable Disable Command for Each Channel
  - Independent Clock Selection for Each Channel
  - Independent Period, Duty-Cycle and Dead-Time for Each Channel
  - Independent Double Buffering of Period, Duty-Cycle and Dead-Times for Each Channel
  - Independent Programmable Selection of The Output Waveform Polarity for Each Channel
  - Independent Programmable Center or Left Aligned Output Waveform for Each Channel
  - Independent Output Override for Each Channel
  - Independent Interrupt for Each Channel, at Each Period for Left-Aligned or Center-Aligned Configuration
- 2 2-bit Gray Up/Down Channels for Stepper Motor Control
- Synchronous Channel Mode
  - Synchronous Channels Share the Same Counter
  - Mode to Update the Synchronous Channels Registers after a Programmable Number of Periods
- 2 Independent Events Lines Intended to Synchronize ADC Conversions
  - Programmable delay for Events Lines to delay ADC measurements
- 8 Comparison Units Intended to Generate Interrupts, Pulses on Event Lines
- 1 Programmable Fault/Break Inputs Providing an Asynchronous Protection of PWM Outputs
  - 4 User Driven through PIO inputs
  - PMC Driven when Crystal Oscillator Clock Fails
  - ADC Controller Driven through Configurable Comparison Function
- Write Protected Registers

48.3 Block Diagram

Figure 48-1. Pulse Width Modulation Controller Block Diagram



48.4 I/O Lines Description

Each channel outputs two complementary external I/O lines.

Table 48-1. I/O Line Description

Name	Description	Type
PWMHx	PWM Waveform Output High for channel x	Output
PWMLx	PWM Waveform Output Low for channel x	Output
PWMFIx	PWM Fault Input x	Input

48.5 Product Dependencies

48.5.1 I/O Lines

The pins used for interfacing the PWM are 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 48-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
PWM	PWMFI0	PC28	B
PWM	PWMFI1	PC31	B
PWM	PWMFI2	PC29	B
PWM	PWMFI3	PD16	C
PWM	PWMH0	PA20	B
PWM	PWMH0	PB0	B
PWM	PWMH1	PA22	B
PWM	PWMH1	PB4	B
PWM	PWMH1	PB27	C
PWM	PWMH2	PB8	B
PWM	PWMH2	PD5	C
PWM	PWMH3	PB12	B
PWM	PWMH3	PD7	C
PWM	PWML0	PA21	B
PWM	PWML0	PB1	B
PWM	PWML1	PA23	B
PWM	PWML1	PB5	B
PWM	PWML1	PE31	B
PWM	PWML2	PB9	B
PWM	PWML2	PD6	C
PWM	PWML3	PB13	B
PWM	PWML3	PD8	C

48.5.2 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.

In the PWM description, Master Clock (MCK) is the clock of the peripheral bus to which the PWM is connected.

48.5.3 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 48-3. Peripheral IDs

Instance	ID
PWM	28

48.5.4 Fault Inputs

The PWM has the FAULT inputs connected to the different modules. Please refer to the implementation of these module within the product for detailed information about the fault generation procedure. The PWM receives faults from PIO inputs, PMC, ADC controller, .

Table 48-4. Fault Inputs

Fault Generator	External PWM Fault Input Number	Polarity Level(1)	Fault Input ID
PC28	PWMFI0	User Defined	0
PC31	PWMFI1	User Defined	1
PC29	PWMFI2	User Defined	2
PD16	PWMFI3	User Defined	3
Main OSC (PMC)	–	To be configured to 1	4
ADC	–	To be configured to 1	5

Note: 1. FPOL field in PWMC_FMR.

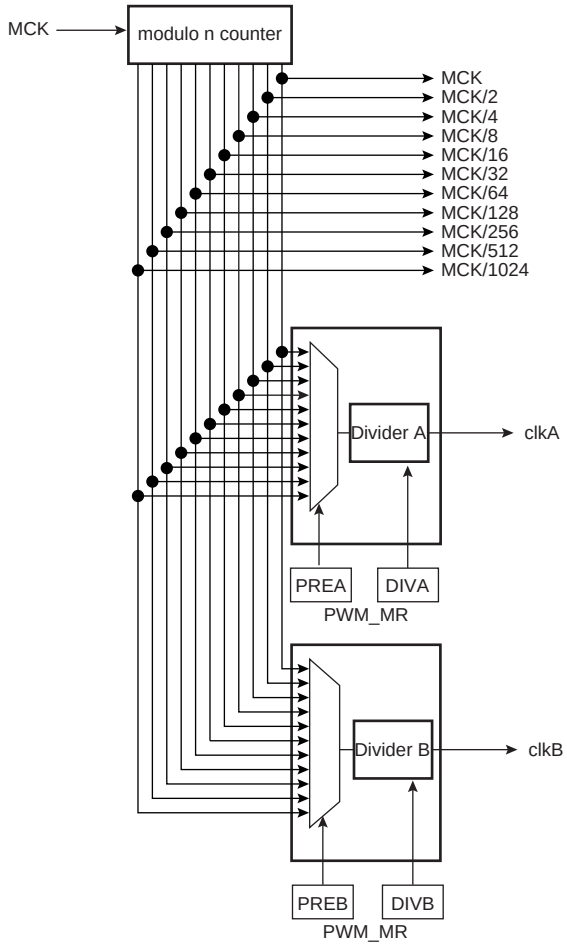
48.6 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

- Clocked by the master 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.

48.6.1 PWM Clock Generator

Figure 48-2. Functional View of the Clock Generator Block Diagram



The PWM 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 Clock register (PWM_CLK). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value.

After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) 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 “MCK”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

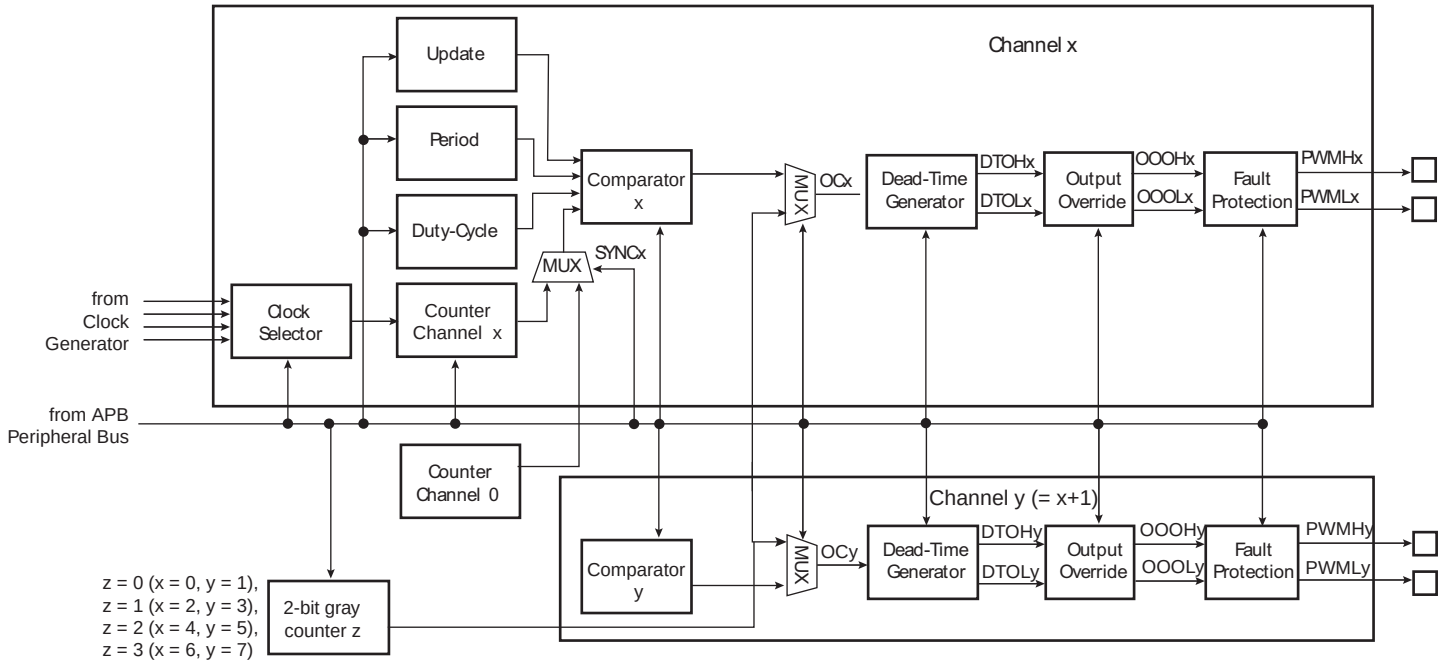
CAUTION:

- Before using the PWM macrocell, the programmer must first enable the PWM clock in the Power Management Controller (PMC).

48.6.2 PWM Channel

48.6.2.1 Channel Block Diagram

Figure 48-3. Functional View of the Channel Block Diagram



Each of the 4 channels is composed of six blocks:

- A clock selector which selects one of the clocks provided by the clock generator (described in Section 48.6.1 on page 1550).
- A counter clocked by the output of the clock selector. This counter is incremented or decremented according to the channel configuration and comparators matches. The size of the counter is 16 bits.
- A comparator used to compute the OCx output waveform according to the counter value and the configuration. The counter value can be the one of the channel counter or the one of the channel 0 counter according to SYNCx bit in the "PWM Sync Channels Mode Register" (PWM_SCM).
- A 2-bit configurable gray counter enables the stepper motor driver. One gray counter drives 2 channels.
- A dead-time generator providing two complementary outputs (DTOHx/DTOLx) which allows to drive external power control switches safely.
- An output override block that can force the two complementary outputs to a programmed value (OOOHx/OOOLx).
- An asynchronous fault protection mechanism that has the highest priority to override the two complementary outputs (PWMHx/PWMLx) in case of fault detection (outputs forced to '0', '1').

48.6.2.2 Comparator

The comparator continuously compares its counter value with the channel period defined by CPRD in the "PWM Channel Period Register" (PWM_CPRDx) and the duty-cycle defined by CDTY in the "PWM Channel Duty Cycle Register" (PWM_CDTYx) to generate an output signal OCx accordingly.

The different properties of the waveform of the output OCx are:

- the ***clock selection***. The 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 Channel Mode Register" (PWM_CMRx). 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 PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula will be:

  $$\frac{(X \times CPRD)}{MCK}$$

  By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

  $$\frac{(X \times CRPD \times DIVA)}{MCK} \quad or$$
  $$\frac{(X \times CRPD \times DIVB)}{MCK}$$

  If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
  By using the PWM master clock (MCK) divided by an X given prescaler value
  (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

  $$\frac{(2 \times X \times CPRD)}{MCK}$$

  By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

  $$\frac{(2 \times X \times CPRD \times DIVA)}{MCK} \quad or$$
  $$\frac{(2 \times X \times CPRD \times DIVB)}{MCK}$$

- the ***waveform duty-cycle***. This channel parameter is defined in the CDTY field of the PWM_CDTYx register.
  If the waveform is left aligned then:

  $$\text{duty cycle} = \frac{(period - 1/\text{fchannel\_x\_clock} \times CDTY)}{period}$$

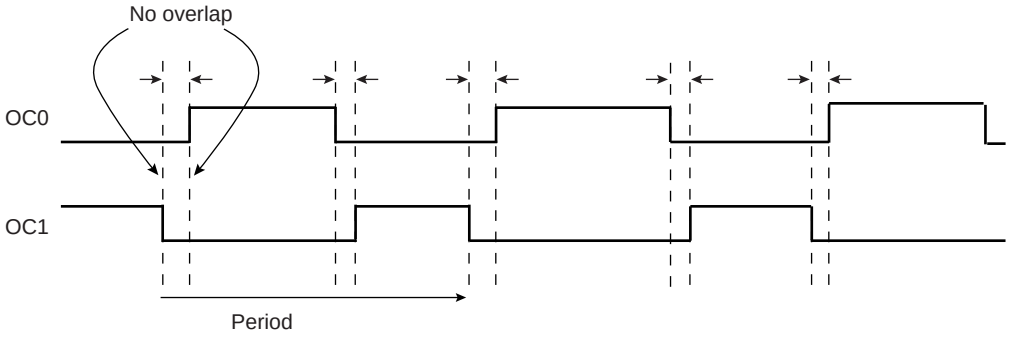
  If the waveform is center aligned, then:

  $$\text{duty cycle} = \frac{((period/2) - 1/\text{fchannel\_x\_clock} \times CDTY))}{(period/2)}$$

- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL bit of the PWM_CMRx. 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 bit of the PWM_CMRx. The default mode is left aligned.

Figure 48-4. Non Overlapped Center Aligned Waveforms



Note: 1. See Figure 48-5 on page 1554 for a detailed description of center aligned waveforms.

When center aligned, the channel counter increases up to CPRD and decreases down to 0. This ends the period.

When left aligned, the 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

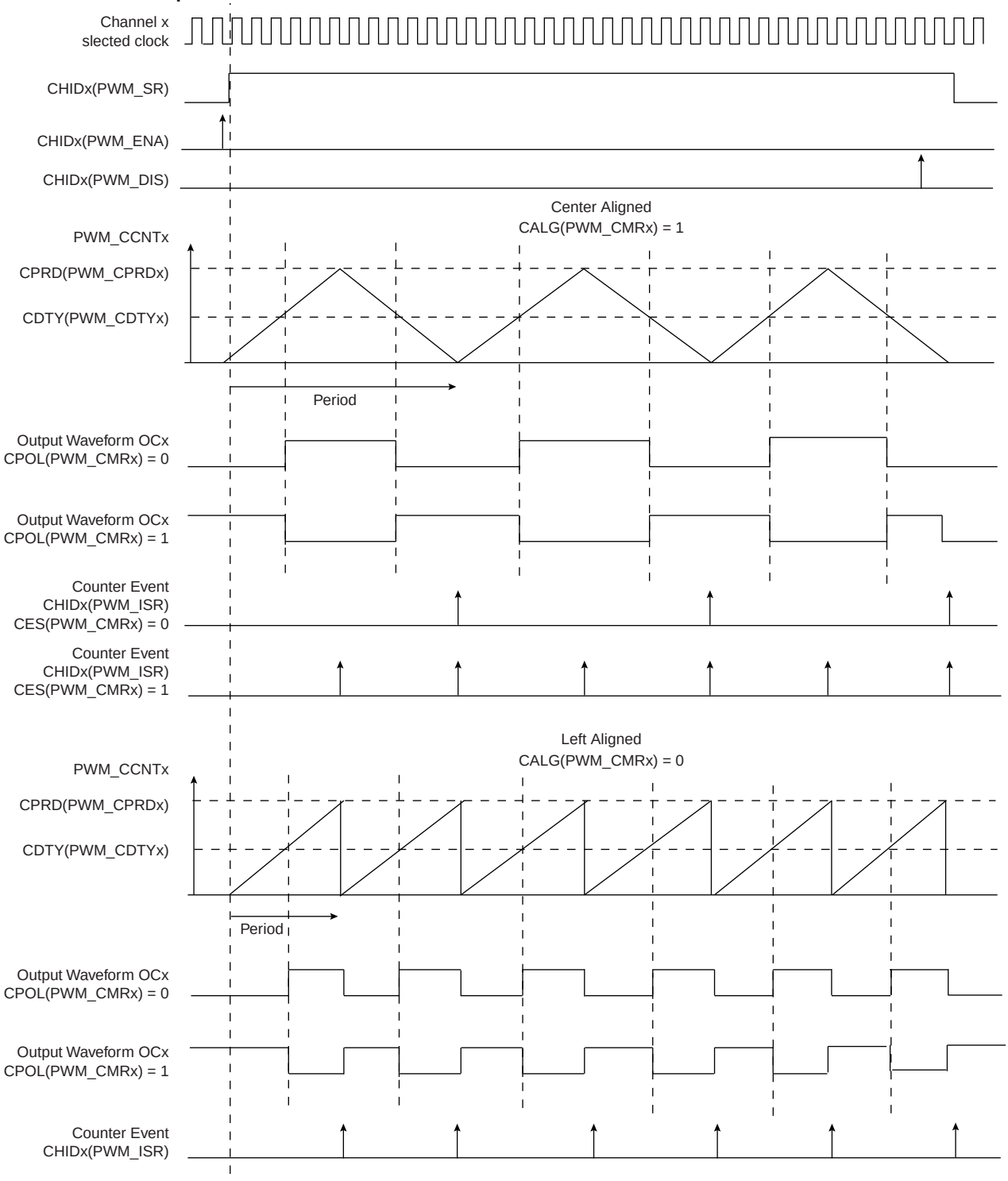
The waveform polarity must be set before enabling the channel. This immediately affects the channel output level.

Modifying CPOL in “PWM Channel Mode Register” while the channel is enabled can lead to an unexpected behavior of the device being driven by PWM.

Besides generating output signals OCx, the comparator generates interrupts in function of the counter value. When the output waveform is left aligned, the interrupt occurs at the end of the counter period. When the output waveform is center aligned, the bit CES of the PWM_CMRx defines when the channel counter interrupt occurs. If CES is set to ‘0’, the interrupt occurs at the end of the counter period. If CES is set to ‘1’, the interrupt occurs at the end of the counter period and at half of the counter period.

Figure 48-5 “Waveform Properties” illustrates the counter interrupts in function of the configuration.

Figure 48-5. Waveform Properties



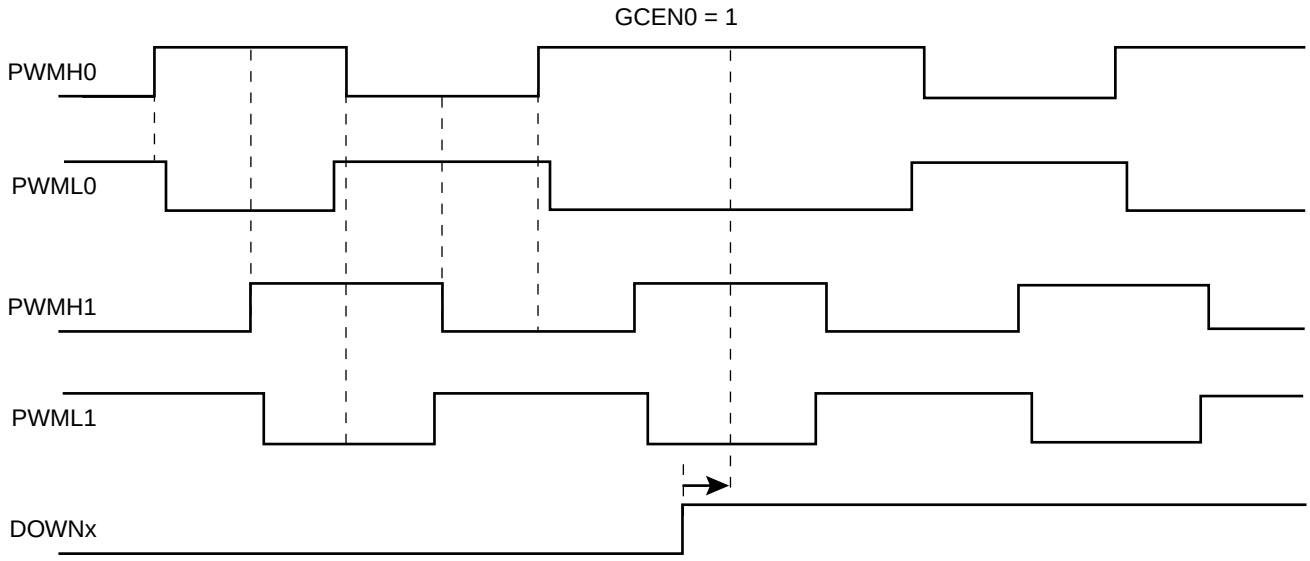
48.6.2.3 2-bit Gray Up/Down Counter for Stepper Motor

It is possible to configure a couple of channels to provide a 2-bit gray count waveform on two outputs. Dead-Time Generator and other downstream logic can be configured on these channels.

Up or down count mode can be configured on-the-fly by means of PWM_SMMR configuration registers.

When GCEN0 is set to ‘1’, channels 0 and 1 outputs are driven with gray counter.

Figure 48-6. 2-bit Gray Up/Down Counter



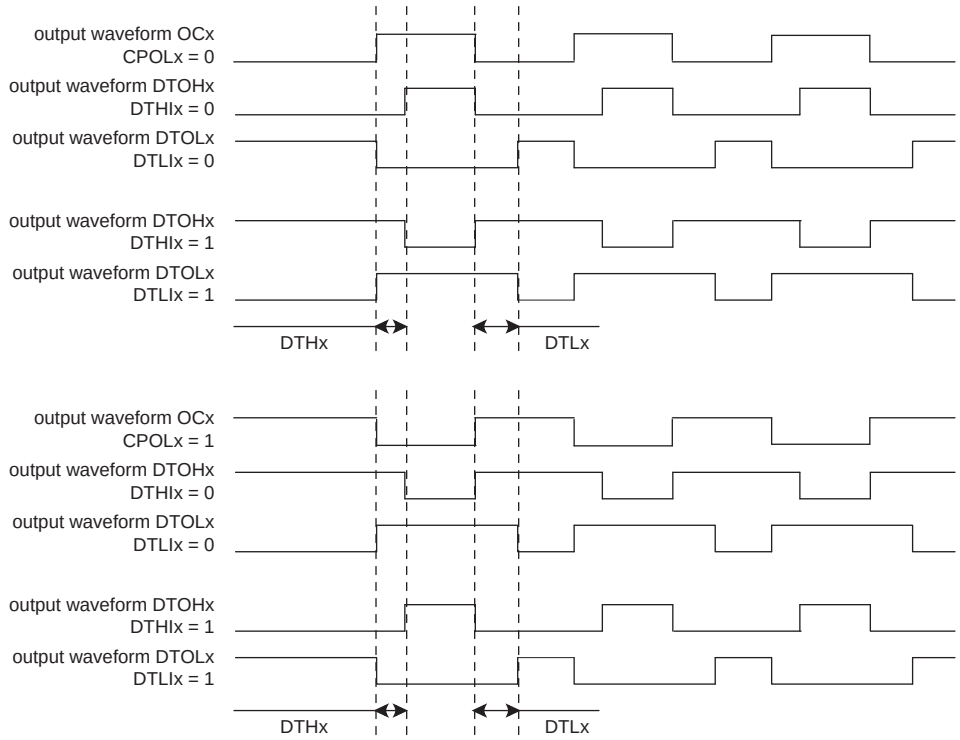
48.6.2.4 Dead-Time Generator

The dead-time generator uses the comparator output OCx to provide the two complementary outputs DTOHx and DTOLx, which allows the PWM macrocell to drive external power control switches safely. When the dead-time generator is enabled by setting the bit DTE to ‘1’ or 0 in the “PWM Channel Mode Register” (PWM_CMRx), dead-times (also called dead-bands or non-overlapping times) are inserted between the edges of the two complementary outputs DTOHx and DTOLx. Note that enabling or disabling the dead-time generator is allowed only if the channel is disabled.

The dead-time is adjustable by the “PWM Channel Dead Time Register” (PWM_DTx). Both outputs of the dead-time generator can be adjusted separately by DTH and DTL. The dead-time values can be updated synchronously to the PWM period by using the “PWM Channel Dead Time Update Register” (PWM_DTUPDx).

The dead-time is based on a specific counter which uses the same selected clock that feeds the channel counter of the comparator. Depending on the edge and the configuration of the dead-time, DTOHx and DTOLx are delayed until the counter has reached the value defined by DTH or DTL. An inverted configuration bit (DTHI and DTLI bit in the PWM_CMRx) is provided for each output to invert the dead-time outputs. The following figure shows the waveform of the dead-time generator.

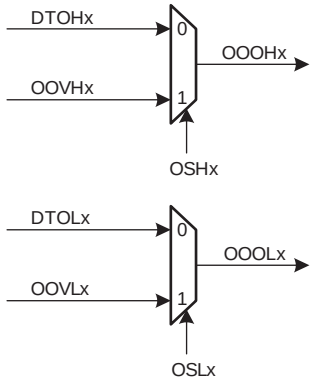
Figure 48-7. Complementary Output Waveforms



48.6.2.5 Output Override

The two complementary outputs DTOHx and DTOLx of the dead-time generator can be forced to a value defined by the software.

Figure 48-8. Override Output Selection



The fields OSHx and OSLx in the "PWM Output Selection Register" (PWM_OS) allow the outputs of the dead-time generator DTOHx and DTOLx to be overridden by the value defined in the fields OOVHx and OOVLx in the"PWM Output Override Value Register" (PWM_OOV).

The set registers "PWM Output Selection Set Register" (PWM_OSS) and "PWM Output Selection Set Update Register" (PWM_OSSUPD) enable the override of the outputs of a channel regardless of other channels. In the same way, the clear registers "PWM Output Selection Clear Register" (PWM_OSC) and "PWM Output Selection Clear Update Register" (PWM_OSCUPD) disable the override of the outputs of a channel regardless of other channels.

By using buffer registers PWM_OSSUPD and PWM_OSCUPD, the output selection of PWM outputs is done synchronously to the channel counter, at the beginning of the next PWM period.

By using registers PWM_OSS and PWM_OSC, the output selection of PWM outputs is done asynchronously to the channel counter, as soon as the register is written.

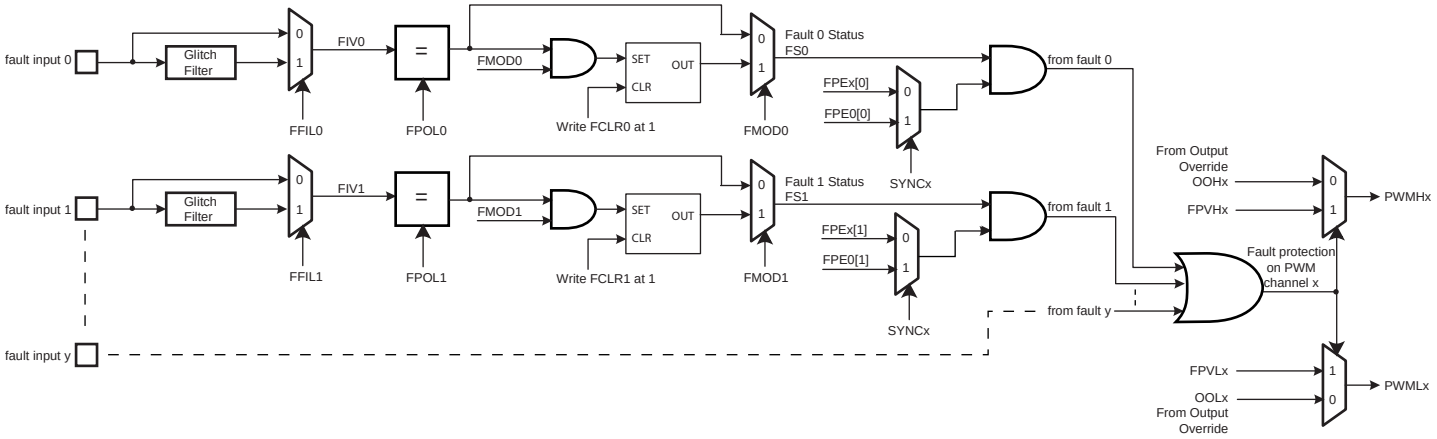
The value of the current output selection can be read in PWM_OS.

While overriding PWM outputs, the channel counters continue to run, only the PWM outputs are forced to user defined values.

48.6.2.6 Fault Protection

1 inputs provide fault protection which can force any of the PWM output pairs to a programmable value. This mechanism has priority over output overriding.

Figure 48-9. Fault Protection



The polarity level of the fault inputs is configured by the FPOL field in the “PWM Fault Mode Register” (PWM_FMR). For fault inputs coming from internal peripherals such as ADC, Timer Counter, to name but a few, the polarity level must be FPOL = 1. For fault inputs coming from external GPIO pins the polarity level depends on the user's implementation.

The configuration of the Fault Activation Mode (FMOD field in PWMC_FMR) depends on the peripheral generating the fault. If the corresponding peripheral does not have “Fault Clear” management, then the FMOD configuration to use must be FMOD = 1, to avoid spurious fault detection. Check the corresponding peripheral documentation for details on handling fault generation.

The fault inputs can be glitch filtered or not in function of the FFIL field in the PWM_FMR. When the filter is activated, glitches on fault inputs with a width inferior to the PWM master clock (MCK) period are rejected.

A fault becomes active as soon as its corresponding fault input has a transition to the programmed polarity level. If the corresponding bit FMOD is set to ‘0’ in the PWM_FMR, the fault remains active as long as the fault input is at this polarity level. If the corresponding FMOD field is set to ‘1’, the fault remains active until the fault input is not at this polarity level anymore and until it is cleared by writing the corresponding bit FCLR in the “PWM Fault Clear Register” (PWM_FCR). By reading the “PWM Fault Status Register” (PWM_FSR), the user can read the current level of the fault inputs by means of the field FIV, and can know which fault is currently active thanks to the FS field.

Each fault can be taken into account or not by the fault protection mechanism in each channel. To be taken into account in the channel x, the fault y must be enabled by the bit FPEx[y] in the “PWM Fault Protection Enable Registers” (PWM_FPE1). However the synchronous channels (see Section 48.6.2.7 “Synchronous Channels”) do not use their own fault enable bits, but those of the channel 0 (bits FPE0[y]).

The fault protection on a channel is triggered when this channel is enabled and when any one of the faults that are enabled for this channel is active. It can be triggered even if the PWM master clock (MCK) is not running but only by a fault input that is not glitch filtered.

When the fault protection is triggered on a channel, the fault protection mechanism resets the counter of this channel and forces the channel outputs to the values defined by the fields FPVHx and FPVLx in the “PWM Fault Protection Value Register” (PWM_FPV) . The output forcing is made asynchronously to the channel counter.

CAUTION:

- To prevent an unexpected activation of the status flag FSy in the PWM_FSR, the FMODy bit can be set to ‘1’ only if the FPOLy bit has been previously configured to its final value.
- To prevent an unexpected activation of the Fault Protection on the channel x, the bit FPEx[y] can be set to ‘1’ only if the FPOLy bit has been previously configured to its final value.

If a comparison unit is enabled (see Section 48.6.3 “PWM Comparison Units”) and if a fault is triggered in the channel 0, in this case the comparison cannot match.

As soon as the fault protection is triggered on a channel, an interrupt (different from the interrupt generated at the end of the PWM period) can be generated but only if it is enabled and not masked. The interrupt is reset by reading the interrupt status register, even if the fault which has caused the trigger of the fault protection is kept active.

48.6.2.7 Synchronous Channels

Some channels can be linked together as synchronous channels. They have the same source clock, the same period, the same alignment and are started together. In this way, their counters are synchronized together.

The synchronous channels are defined by the SYNCx bits in the “PWM Sync Channels Mode Register” (PWM_SCM). Only one group of synchronous channels is allowed.

When a channel is defined as a synchronous channel, the channel 0 is automatically defined as a synchronous channel too, because the channel 0 counter configuration is used by all the synchronous channels.

If a channel x is defined as a synchronous channel, it uses the following configuration fields of the channel 0 instead of its own:

- CPRE0 field in PWM_CMR0 instead of CPREx field in PWM_CMRx (same source clock)
- CPRD0 field in PWM_CMR0 instead of CPRDx field in PWM_CMRx (same period)
- CALG0 field in PWM_CMR0 instead of CALGx field in PWM_CMRx (same alignment)

Thus writing these fields of a synchronous channel has no effect on the output waveform of this channel (except channel 0 of course).

Because counters of synchronous channels must start at the same time, they are all enabled together by enabling the channel 0 (by the CHID0 bit in PWM_ENA register). In the same way, they are all disabled together by disabling channel 0 (by the CHID0 bit in PWM_DIS register). However, a synchronous channel x different from channel 0 can be enabled or disabled independently from others (by the CHIDx bit in PWM_ENA and PWM_DIS registers).

Defining a channel as a synchronous channel while it is an asynchronous channel (by writing the bit SYNCx to ‘1’ while it was at ‘0’) is allowed only if the channel is disabled at this time (CHIDx = 0 in PWM_SR). In the same way, defining a channel as an asynchronous channel while it is a synchronous channel (by writing the SYNCx bit to ‘0’ while it was 1) is allowed only if the channel is disabled at this time.

The field UPDM (Update Mode) in the PWM_SCM register allow to select one of the three methods to update the registers of the synchronous channels:

- Method 1 (UPDM = 0): The period value, the duty-cycle values and the dead-time values must be written by the CPU in their respective update registers (respectively PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPDx).The update is triggered at the next PWM period as soon as the bit UPDULOCK in the “PWM Sync Channels Update Control Register” (PWM_SCUC) is set to ‘1’ (see “Method 1: Manual write of duty-cycle values and manual trigger of the update” on page 1559).
- Method 2 (UPDM = 1): The period value, the duty-cycle values, the dead-time values and the update period value must be written by the CPU in their respective update registers (respectively PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPD). The update of the period value and of the dead-time values is triggered at the next PWM period as soon as the bit UPDULOCK in the PWM_SCUC register is set to ‘1’. The update of the duty-cycle values and the update period value is triggered automatically after an update period defined by the field UPR in the “PWM Sync Channels Update Period Register” (PWM_SCUP) (see “Method 2: Manual write of duty-cycle values and automatic trigger of the update” on page 1560).

Table 48-5. Summary of the Update of Registers of Synchronous Channels

	UPDM=0	UPDM=1
Period Value (PWM_CPRDUPDx)	Write by the CPU	
	Update is triggered at the next PWM period as soon as the bit UPDULOCK is set to ‘1’	
Dead-Time Values (PWM_DTUPDx)	Write by the CPU	
	Update is triggered at the next PWM period as soon as the bit UPDULOCK is set to ‘1’	
Duty-Cycle Values (PWM_CDTYUPDx)	Write by the CPU	Write by the CPU
	Update is triggered at the next PWM period as soon as the bit UPDULOCK is set to ‘1’	Update is triggered at the next PWM period as soon as the update period counter has reached the value UPR
Update Period Value (PWM_SCUPUPD)	Not applicable	Write by the CPU
	Not applicable	Update is triggered at the next PWM period as soon as the update period counter has reached the value UPR

Method 1: Manual write of duty-cycle values and manual trigger of the update

In this mode, the update of the period value, the duty-cycle values and the dead-time values must be done by writing in their respective update registers with the CPU (respectively PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPDx).

To trigger the update, the user must use the bit UPDULOCK in the PWM_SCUC register which allows to update synchronously (at the same PWM period) the synchronous channels:

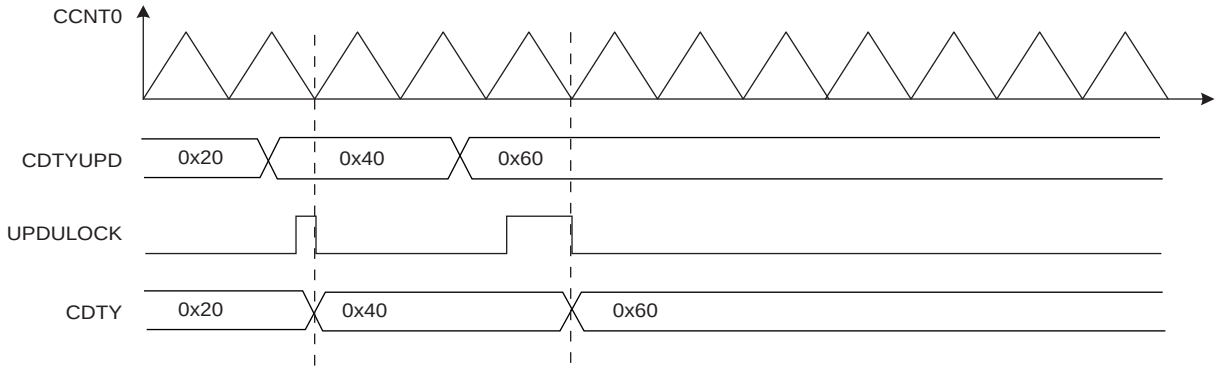
- If the bit UPDULOCK is set to ‘1’, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to ‘1’, the update is locked and cannot be performed.

After writing the UPDULOCK bit to ‘1’, it is held at this value until the update occurs, then it is read 0.

Sequence for Method 1:

1. Select the manual write of duty-cycle values and the manual update by setting the UPDM field to ‘0’ in the PWM_SCM register
2. Define the synchronous channels by the SYNCx bits in the PWM_SCM register.
3. Enable the synchronous channels by writing CHID0 in the PWM_ENA register.
4. If an update of the period value and/or the duty-cycle values and/or the dead-time values is required, write registers that need to be updated (PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPDx).
5. Set UPDULOCK to ‘1’ in PWM_SCUC.
6. The update of the registers will occur at the beginning of the next PWM period. At this moment the UPDULOCK bit is reset, go to Step 4.) for new values.

Figure 48-10.Method 1 (UPDM = 0)



Method 2: Manual write of duty-cycle values and automatic trigger of the update

In this mode, the update of the period value, the duty-cycle values, the dead-time values and the update period value must be done by writing in their respective update registers with the CPU (respectively PWM_CPRDUPDx, PWM_CDTYUPDx, PWM_DTUPDx and PWM_SCUPUPD).

To trigger the update of the period value and the dead-time values, the user must use the bit UPDULOCK in the PWM_SCUC register, which allows to update synchronously (at the same PWM period) the synchronous channels:

- If the bit UPDULOCK is set to ‘1’, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to ‘1’, the update is locked and cannot be performed.

After writing the UPDULOCK bit to ‘1’, it is held at this value until the update occurs, then it is read 0.

The update of the duty-cycle values and the update period is triggered automatically after an update period.

To configure the automatic update, the user must define a value for the Update Period by the UPR field in the PWM_SCUP register. The PWM controller waits UPR+1 period of synchronous channels before updating automatically the duty values and the update period value.

The status of the duty-cycle value write is reported in the “PWM Interrupt Status Register 2” (PWM_ISR2) by the following flags:

- WRDY: this flag is set to ‘1’ when the PWM Controller is ready to receive new duty-cycle values and a new update period value. It is reset to ‘0’ when the PWM_ISR2 register is read.

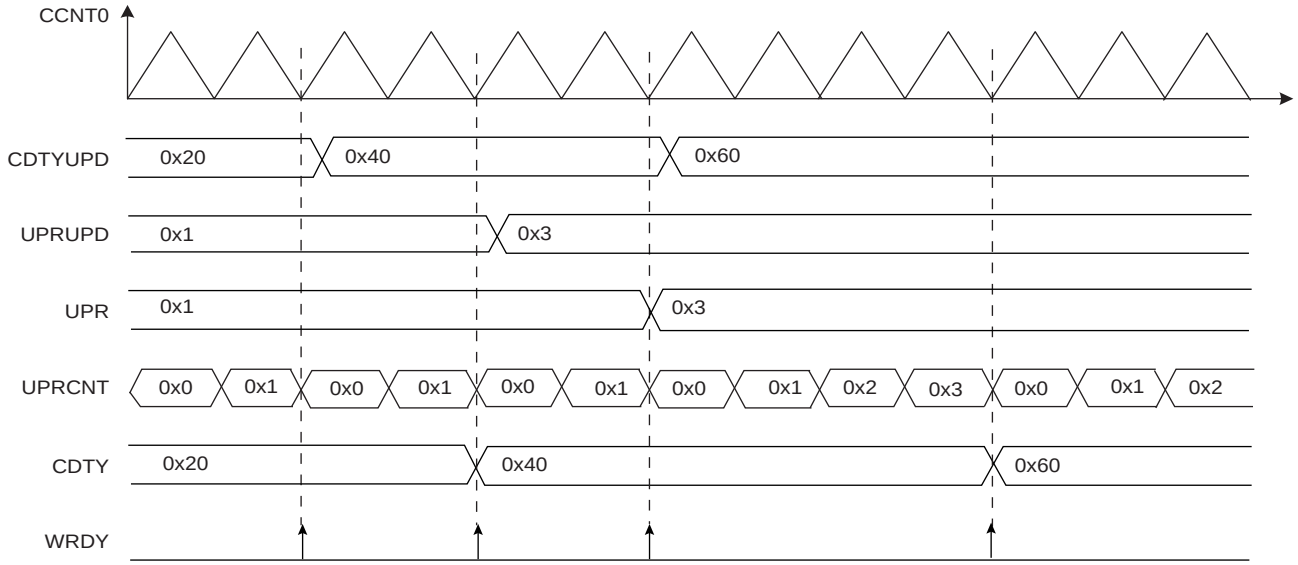
Depending on the interrupt mask in the “PWM Interrupt Mask Register 2” (PWM_IMR2), an interrupt can be generated by these flags.

Sequence for Method 2:

1. Select the manual write of duty-cycle values and the automatic update by setting the field UPDM to ‘1’ in the PWM_SCM register
2. Define the synchronous channels by the bits SYNCx in the PWM_SCM register.
3. Define the update period by the field UPR in the PWM_SCUP register.
4. Enable the synchronous channels by writing CHID0 in the PWM_ENA register.
5. If an update of the period value and/or of the dead-time values is required, write registers that need to be updated (PWM_CPRDUPDx, PWM_DTUPDx), else go to Step 8.
6. Set UPDULOCK to ‘1’ in PWM_SCUC.
7. The update of these registers will occur at the beginning of the next PWM period. At this moment the bit UPDULOCK is reset, go to Step 5. for new values.
8. If an update of the duty-cycle values and/or the update period is required, check first that write of new update values is possible by polling the flag WRDY (or by waiting for the corresponding interrupt) in the PWM_ISR2.
9. Write registers that need to be updated (PWM_CDTYUPDx, PWM_SCUPUPD).

10. The update of these registers will occur at the next PWM period of the synchronous channels when the Update Period is elapsed. Go to Step 8. for new values.

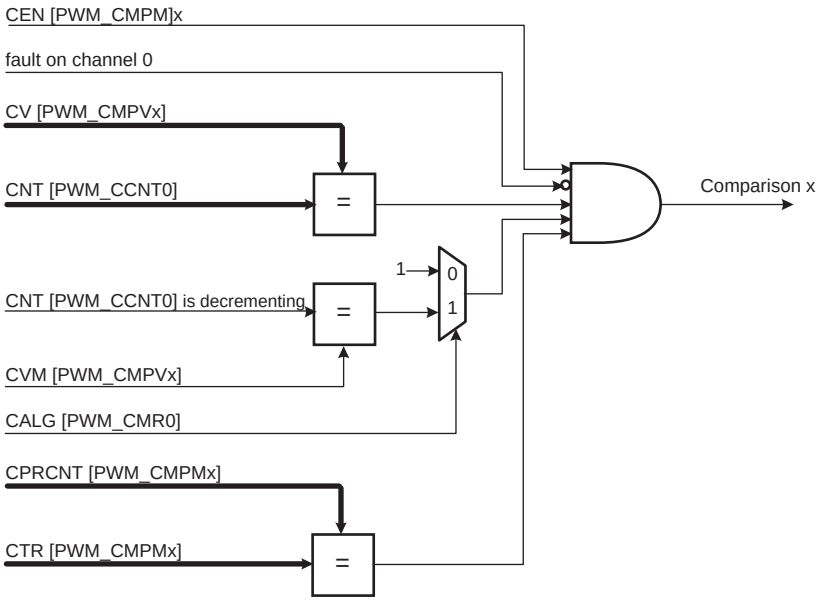
Figure 48-11.Method 2 (UPDM=1)



48.6.3 PWM Comparison Units

The PWM provides 8 independent comparison units able to compare a programmed value with the current value of the channel 0 counter (which is the channel counter of all synchronous channels, Section 48.6.2.7 “Synchronous Channels”). These comparisons are intended to generate pulses on the event lines (used to synchronize ADC, see Section 48.6.4 “PWM Event Lines”), to generate software interrupts.

Figure 48-12.Comparison Unit Block Diagram



The comparison x matches when it is enabled by the bit CEN in the “PWM Comparison x Mode Register” (PWM_CMPMx for the comparison x) and when the counter of the channel 0 reaches the comparison value defined by the field CV in “PWM Comparison x Value Register” (PWM_CMPVx for the comparison x). If the counter of the channel 0 is center aligned (CALG = 1 in “PWM Channel Mode Register”), the bit CVM (in PWM_CMPVx) defines if the

comparison is made when the counter is counting up or counting down (in left alignment mode CALG = 0, this bit is useless).

If a fault is active on the channel 0, the comparison is disabled and cannot match (see Section 48.6.2.6 “Fault Protection”).

The user can define the periodicity of the comparison x by the fields CTR and CPR (in PWM_CMPVx). The comparison is performed periodically once every CPR+1 periods of the counter of the channel 0, when the value of the comparison period counter CPRCNT (in PWM_CMPMx) reaches the value defined by CTR. CPR is the maximum value of the comparison period counter CPRCNT. If CPR=CTR=0, the comparison is performed at each period of the counter of the channel 0.

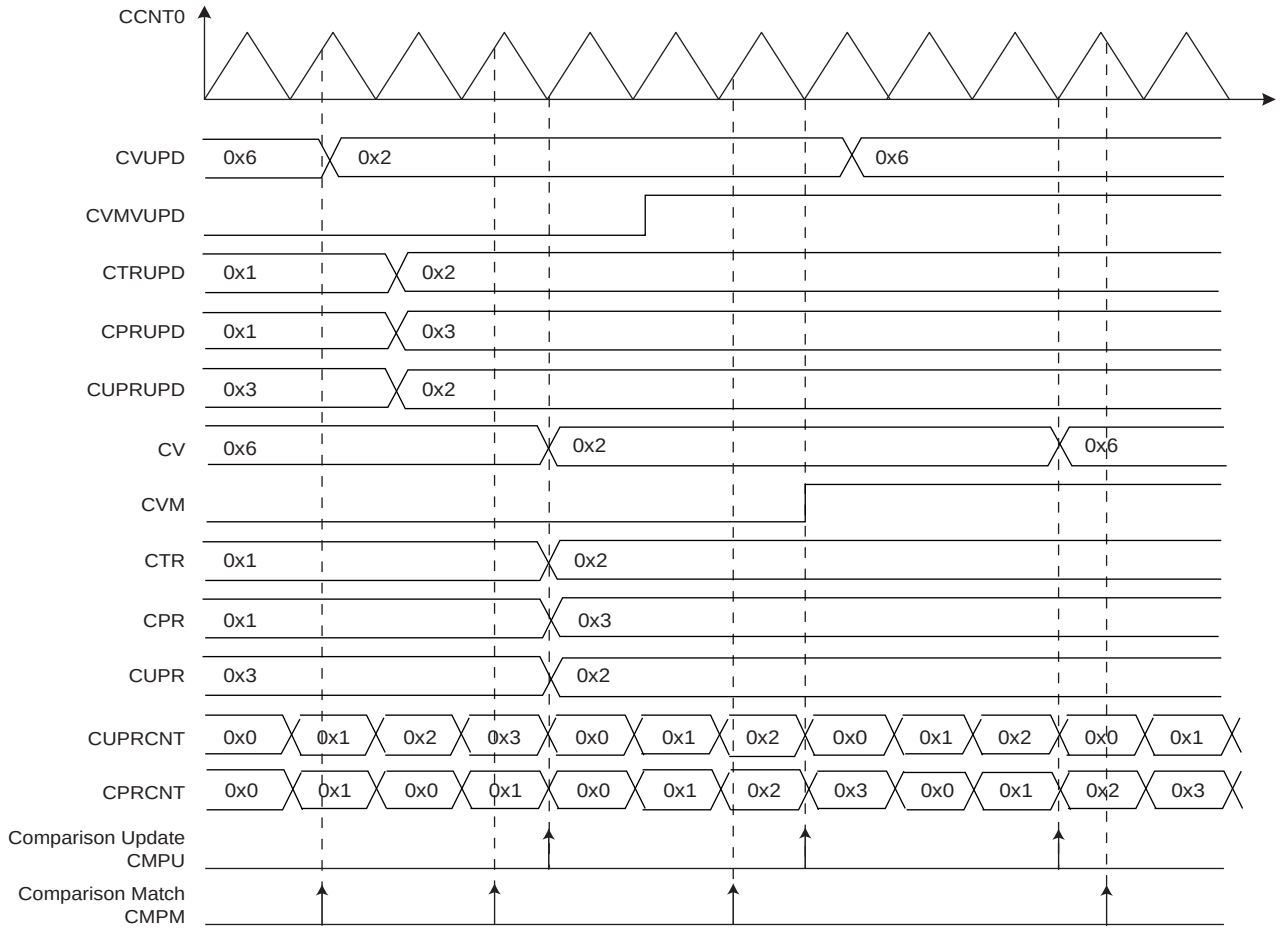
The comparison x configuration can be modified while the channel 0 is enabled by using the “PWM Comparison x Mode Update Register” (PWM_CMPMUPDx registers for the comparison x). In the same way, the comparison x value can be modified while the channel 0 is enabled by using the “PWM Comparison x Value Update Register” (PWM_CMPVUPDx registers for the comparison x).

The update of the comparison x configuration and the comparison x value is triggered periodically after the comparison x update period. It is defined by the field CUPR in the PWM_CMPMx. The comparison unit has an update period counter independent from the period counter to trigger this update. When the value of the comparison update period counter CUPRCNT (in PWM_CMPMx) reaches the value defined by CUPR, the update is triggered. The comparison x update period CUPR itself can be updated while the channel 0 is enabled by using the PWM_CMPMUPDx register.

CAUTION: to be taken into account, the write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

The comparison match and the comparison update can be source of an interrupt, but only if it is enabled and not masked. These interrupts can be enabled by the “PWM Interrupt Enable Register 2” and disabled by the “PWM Interrupt Disable Register 2” . The comparison match interrupt and the comparison update interrupt are reset by reading the “PWM Interrupt Status Register 2” .

Figure 48-13.Comparison Waveform

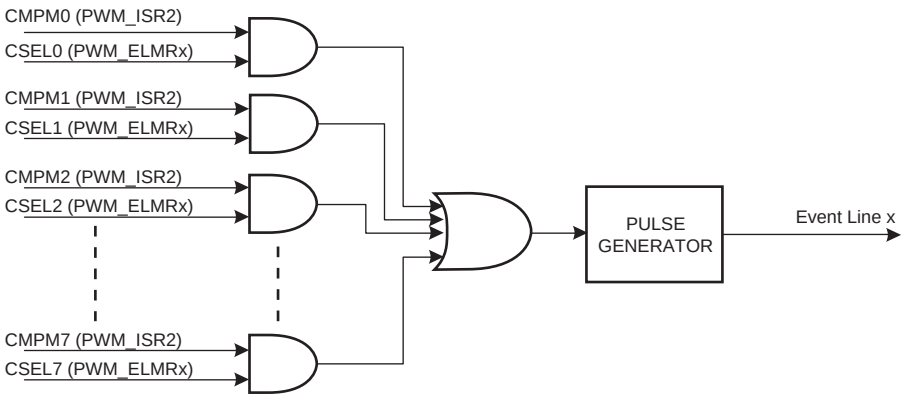


48.6.4 PWM Event Lines

The PWM provides 2 independent event lines intended to trigger actions in other peripherals (in particular for ADC (Analog-to-Digital Converter)).

A pulse (one cycle of the master clock (MCK)) is generated on an event line, when at least one of the selected comparisons is matching. The comparisons can be selected or unselected independently by the CSEL bits in the "PWM Event Line x Register" (PWM_ELMRx for the Event Line x).

Figure 48-14.Event Line Block Diagram



48.6.5 PWM Controller Operations

48.6.5.1 Initialization

Before enabling the channels, they must have been configured by the software application:

- Unlock User Interface by writing the WPCMD field in the PWM_WPCR.
- Configuration of the clock generator (DIVA, PREA, DIVB, PREB in the PWM_CLK register if required).
- Selection of the clock for each channel (CPRE field in PWM_CMRx)
- Configuration of the waveform alignment for each channel (CALG field in PWM_CMRx)
- Selection of the counter event selection (if CALG = 1) for each channel (CES field in PWM_CMRx)
- Configuration of the output waveform polarity for each channel (CPOL bit in PWM_CMRx)
- 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_CPRDUPDx 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_CDTYUPDx register to update PWM_CDTYx as explained below.
- Configuration of the dead-time generator for each channel (DTH and DTL in PWM_DTx) if enabled (DTE bit in the PWM_CMRx). Writing in the PWM_DTx register is possible while the channel is disabled. After validation of the channel, the user must use PWM_DTUPDx register to update PWM_DTx
- Selection of the synchronous channels (SYNCx in the PWM_SCM register)
- Configuration of the update mode (UPDM in PWM_SCM register)
- Configuration of the update period (UPR in PWM_SCUP register) if needed
- Configuration of the comparisons (PWM_CMPVx and PWM_CMPMx)
- Configuration of the event lines (PWM_ELMRx)
- Configuration of the fault inputs polarity (FPOL in PWM_FMR)
- Configuration of the fault protection (FMOD and FFIL in PWM_FMR, PWM_FPV and PWM_FPE1)
- Enable of the Interrupts (writing CHIDx and FCHIDx in PWM_IER1, and writing WRDYE, ENDTXE, TXBUFE, UNRE, CMPMx and CMPUx in PWM_IER2)
- Enable of the PWM channels (writing CHIDx in the PWM_ENA register)

48.6.5.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the "PWM Channel Period Register" (PWM_CPRDx) and the "PWM Channel 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 $1/CPRDx$ 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 from between 1 up to 14 in PWM_CDTYx. The resulting duty-cycle quantum cannot be lower than 1/15 of the PWM period.

48.6.5.3 Changing the Duty-Cycle, the Period and the Dead-Times

It is possible to modulate the output waveform duty-cycle, period and dead-times.

To prevent unexpected output waveform, the user must use the "PWM Channel Duty Cycle Update Register" (PWM_CDTYUPDx), the "PWM Channel Period Update Register" (PWM_CPRDUPDx) and the "PWM Channel Dead Time Update Register" (PWM_DTUPDx) to change waveform parameters while the channel is still enabled.

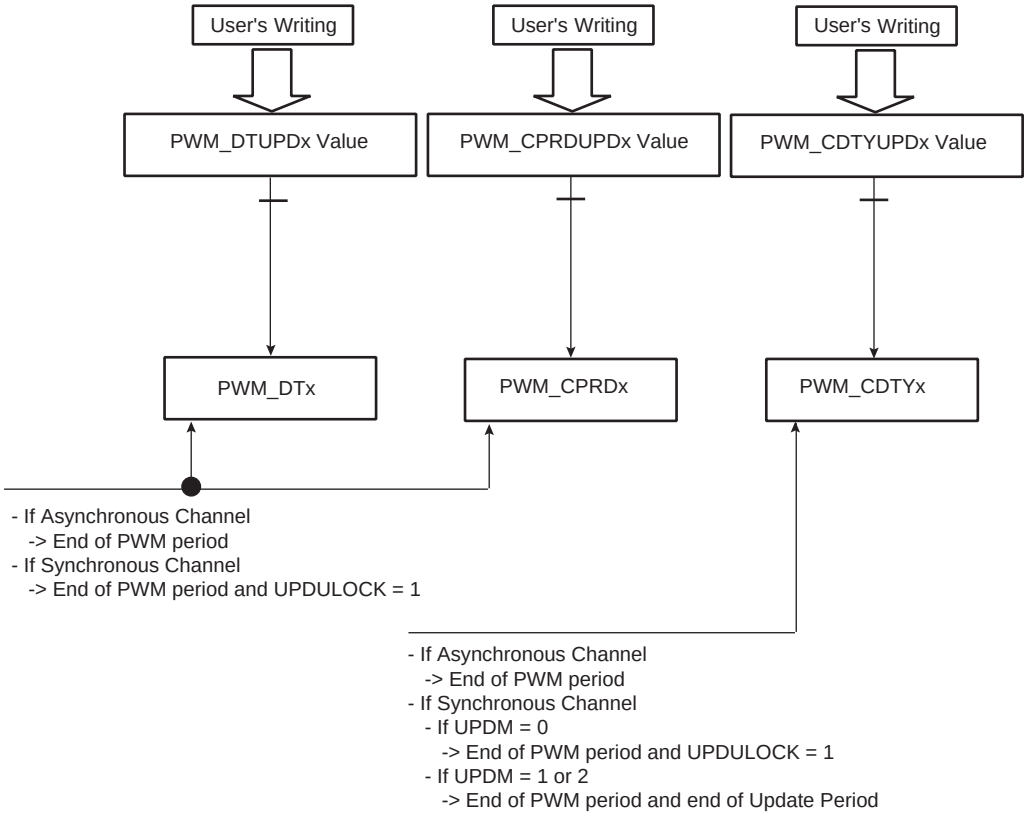
- If the channel is an asynchronous channel (SYNCx = 0 in "PWM Sync Channels Mode Register" (PWM_SCM)), these registers hold the new period, duty-cycle and dead-times values until the end of the current PWM period and update the values for the next period.
- If the channel is a synchronous channel and update method 0 is selected (SYNCx = 1 and UPDM = 0 in PWM_SCM register), these registers hold the new period, duty-cycle and dead-times values until the bit

UPDULOCK is written at ‘1’ (in “PWM Sync Channels Update Control Register” (PWM_SCUC)) and the end of the current PWM period, then update the values for the next period.

- If the channel is a synchronous channel and update method 1 or 2 is selected (SYNCx = 1 and UPDM = 1 or 2 in PWM_SCM register):
  - registers PWM_CPRDUPDx and PWM_DTUPDx hold the new period and dead-times values until the bit UPDULOCK is written at ‘1’ (in PWM_SCUC) and the end of the current PWM period, then update the values for the next period.
  - register PWM_CDTYUPDx holds the new duty-cycle value until the end of the update period of synchronous channels (when UPRCNT is equal to UPR in “PWM Sync Channels Update Period Register” (PWM_SCUP)) and the end of the current PWM period, then updates the value for the next period.

Note: If the update registers PWM_CDTYUPDx, PWM_CPRDUPDx and PWM_DTUPDx are written several times between two updates, only the last written value is taken into account.

Figure 48-15.Synchronized Period, Duty-Cycle and Dead-Times Update



48.6.5.4 Changing the Synchronous Channels Update Period

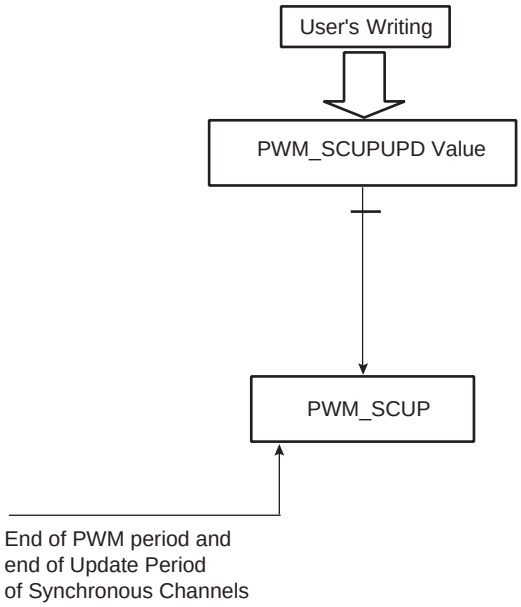
It is possible to change the update period of synchronous channels while they are enabled. (See “Method 2: Manual write of duty-cycle values and automatic trigger of the update” on page 1560)

To prevent an unexpected update of the synchronous channels registers, the user must use the “PWM Sync Channels Update Period Update Register” (PWM_SCUPUPD) to change the update period of synchronous channels while they are still enabled. This register holds the new value until the end of the update period of synchronous channels (when UPRCNT is equal to UPR in PWM_SCUP) and the end of the current PWM period, then updates the value for the next period.

Note: If the update register PWM_SCUPUPD is written several times between two updates, only the last written value is taken into account.

Note: Changing the update period does make sense only if there is one or more synchronous channels and if the update method 1 or 2 is selected (UPDM = 1 or 2 in “PWM Sync Channels Mode Register”).

Figure 48-16.Synchronized Update of Update Period Value of Synchronous Channels



48.6.5.5 Changing the Comparison Value and the Comparison Configuration

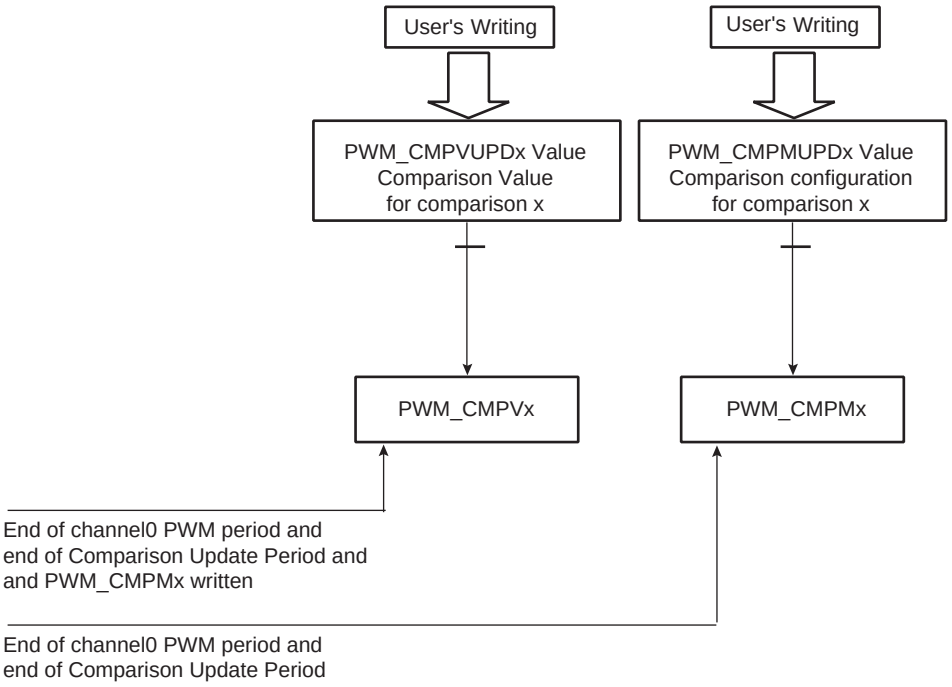
It is possible to change the comparison values and the comparison configurations while the channel 0 is enabled (see Section 48.6.3 “PWM Comparison Units”).

To prevent unexpected comparison match, the user must use the “PWM Comparison x Value Update Register” (PWM_CMPVUPDx) and the “PWM Comparison x Mode Update Register” (PWM_CMPMUPDx) to change respectively the comparison values and the comparison configurations while the channel 0 is still enabled. These registers hold the new values until the end of the comparison update period (when CUPRCNT is equal to CUPR in “PWM Comparison x Mode Register” (PWM_CMPMx) and the end of the current PWM period, then update the values for the next period.

CAUTION: The write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

Note: If the update registers PWM_CMPVUPDx and PWM_CMPMUPDx are written several times between two updates, only the last written value are taken into account.

Figure 48-17.Synchronized Update of Comparison Values and Configurations



48.6.5.6 Interrupts

Depending on the interrupt mask in the PWM_IMR1 and PWM_IMR2, an interrupt can be generated at the end of the corresponding channel period (CHIDx in the PWM Interrupt Status Register 1 (PWM_ISR1)), after a fault event (FCHIDx in the PWM_ISR1), after a comparison match (CMPMx in the PWM_ISR2), after a comparison update (CMPUx in the PWM_ISR2) or according to the transfer mode of the synchronous channels (WRDY, ENDTX, TXBUFE and UNRE in the PWM_ISR2).

If the interrupt is generated by the flags CHIDx or FCHIDx, the interrupt remains active until a read operation in the PWM_ISR1 occurs.

If the interrupt is generated by the flags WRDY or UNRE or CMPMx or CMPUx, the interrupt remains active until a read operation in the PWM_ISR2 occurs.

A channel interrupt is enabled by setting the corresponding bit in PWM_IER1 and PWM_IER2. A channel interrupt is disabled by setting the corresponding bit in PWM_IDR1 and PWM_IDR2.

48.6.5.7 Register Write Protection

To prevent any single software error that may corrupt PWM behavior, the registers listed below can be write-protected by writing the field WPCMD in the "PWM Write Protection Control Register" (PWM_WPCR). They are divided into six groups:

- Register group 0:
  - "PWM Clock Register" on page 1572
- Register group 1:
  - "PWM Disable Register" on page 1574
- Register group 2:
  - "PWM Sync Channels Mode Register" on page 1580
  - "PWM Channel Mode Register" on page 1608
  - "PWM Stepper Motor Mode Register" on page 1600
- Register group 3:
  - "PWM Channel Period Register" on page 1612
  - "PWM Channel Period Update Register" on page 1613

- Register group 4:
  - “PWM Channel Dead Time Register” on page 1615
  - “PWM Channel Dead Time Update Register” on page 1616
- Register group 5:
  - “PWM Fault Mode Register” on page 1594
  - “PWM Fault Protection Value Register” on page 1597

There are two types of Write Protection:

- SW Write Protect—can be enabled or disabled by software
- HW Write Protect—can be enabled by software but only disabled by a hardware reset of the PWM controller

Both types of Write Protect can be applied independently to a particular register group by means of the WPCMD and WPRGx fields in the PWM_WPCR. If at least one Write Protect is active, the register group is write-protected. The value of field WPCMD defines the action to be performed:

0: disables SW Write Protect of the register groups of which the bit WPRGx is at ‘1’

1: enables SW Write Protect of the register groups of which the bit WPRGx is at ‘1’

2: enables HW Write Protect of the register groups of which the bit WPRGx is at ‘1’

At any time, the user can determine which Write Protect is active in which register group by the fields WPSWS and WPHWS in the “PWM Write Protection Status Register” (PWM_WPSR).

If a write access in a write-protected register is detected, then the WPVS flag in the PWM_WPSR is set and the field WPVSRC indicates in which register the write access has been attempted, through its address offset without the two LSBs.

The WPVS and WPVSRC fields are automatically cleared after reading the PWM_WPSR.

48.7 Pulse Width Modulation Controller (PWM) User Interface

Table 48-6. Register Mapping

Offset	Register	Name	Access	Reset
0x00	PWM Clock Register	PWM_CLK	Read/Write	0x0
0x04	PWM Enable Register	PWM_ENA	Write-only	–
0x08	PWM Disable Register	PWM_DIS	Write-only	–
0x0C	PWM Status Register	PWM_SR	Read-only	0x0
0x10	PWM Interrupt Enable Register 1	PWM_IER1	Write-only	–
0x14	PWM Interrupt Disable Register 1	PWM_IDR1	Write-only	–
0x18	PWM Interrupt Mask Register 1	PWM_IMR1	Read-only	0x0
0x1C	PWM Interrupt Status Register 1	PWM_ISR1	Read-only	0x0
0x20	PWM Sync Channels Mode Register	PWM_SCM	Read/Write	0x0
0x24	Reserved	–	–	–
0x28	PWM Sync Channels Update Control Register	PWM_SCUC	Read/Write	0x0
0x2C	PWM Sync Channels Update Period Register	PWM_SCUP	Read/Write	0x0
0x30	PWM Sync Channels Update Period Update Register	PWM_SCUPUPD	Write-only	0x0
0x34	PWM Interrupt Enable Register 2	PWM_IER2	Write-only	–
0x38	PWM Interrupt Disable Register 2	PWM_IDR2	Write-only	–
0x3C	PWM Interrupt Mask Register 2	PWM_IMR2	Read-only	0x0
0x40	PWM Interrupt Status Register 2	PWM_ISR2	Read-only	0x0
0x44	PWM Output Override Value Register	PWM_OOV	Read/Write	0x0
0x48	PWM Output Selection Register	PWM_OS	Read/Write	0x0
0x4C	PWM Output Selection Set Register	PWM_OSS	Write-only	–
0x50	PWM Output Selection Clear Register	PWM_OSC	Write-only	–
0x54	PWM Output Selection Set Update Register	PWM_OSSUPD	Write-only	–
0x58	PWM Output Selection Clear Update Register	PWM_OSCUPD	Write-only	–
0x5C	PWM Fault Mode Register	PWM_FMR	Read/Write	0x0
0x60	PWM Fault Status Register	PWM_FSR	Read-only	0x0
0x64	PWM Fault Clear Register	PWM_FCR	Write-only	–
0x68	PWM Fault Protection Value Register	PWM_FPV	Read/Write	0x0
0x6C	PWM Fault Protection Enable Register	PWM_FPE	Read/Write	0x0
0x70–0x78	Reserved	–	–	–
0x7C	PWM Event Line 0 Mode Register	PWM_ELMR0	Read/Write	0x0
0x80	PWM Event Line 1 Mode Register	PWM_ELMR1	Read/Write	0x0
0x84–0x9C	Reserved	–	–	–
0xA0–0xAC	Reserved	–	–	–
0xB0	PWM Stepper Motor Mode Register	PWM_SMMR	Read/Write	0x0
0xB4–0xBC	Reserved	–	–	–

Table 48-6. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0xC0–0xE0	Reserved	–	–	–
0xE4	PWM Write Protection Control Register	PWM_WPCR	Write-only	–
0xE8	PWM Write Protection Status Register	PWM_WPSR	Read-only	0x0
0xEC–0xFC	Reserved	–	–	–
0x130	PWM Comparison 0 Value Register	PWM_CMPV0	Read/Write	0x0
0x134	PWM Comparison 0 Value Update Register	PWM_CMPVUPD0	Write-only	–
0x138	PWM Comparison 0 Mode Register	PWM_CMPM0	Read/Write	0x0
0x13C	PWM Comparison 0 Mode Update Register	PWM_CMPMUPD0	Write-only	–
0x140	PWM Comparison 1 Value Register	PWM_CMPV1	Read/Write	0x0
0x144	PWM Comparison 1 Value Update Register	PWM_CMPVUPD1	Write-only	–
0x148	PWM Comparison 1 Mode Register	PWM_CMPM1	Read/Write	0x0
0x14C	PWM Comparison 1 Mode Update Register	PWM_CMPMUPD1	Write-only	–
0x150	PWM Comparison 2 Value Register	PWM_CMPV2	Read/Write	0x0
0x154	PWM Comparison 2 Value Update Register	PWM_CMPVUPD2	Write-only	–
0x158	PWM Comparison 2 Mode Register	PWM_CMPM2	Read/Write	0x0
0x15C	PWM Comparison 2 Mode Update Register	PWM_CMPMUPD2	Write-only	–
0x160	PWM Comparison 3 Value Register	PWM_CMPV3	Read/Write	0x0
0x164	PWM Comparison 3 Value Update Register	PWM_CMPVUPD3	Write-only	–
0x168	PWM Comparison 3 Mode Register	PWM_CMPM3	Read/Write	0x0
0x16C	PWM Comparison 3 Mode Update Register	PWM_CMPMUPD3	Write-only	–
0x170	PWM Comparison 4 Value Register	PWM_CMPV4	Read/Write	0x0
0x174	PWM Comparison 4 Value Update Register	PWM_CMPVUPD4	Write-only	–
0x178	PWM Comparison 4 Mode Register	PWM_CMPM4	Read/Write	0x0
0x17C	PWM Comparison 4 Mode Update Register	PWM_CMPMUPD4	Write-only	–
0x180	PWM Comparison 5 Value Register	PWM_CMPV5	Read/Write	0x0
0x184	PWM Comparison 5 Value Update Register	PWM_CMPVUPD5	Write-only	–
0x188	PWM Comparison 5 Mode Register	PWM_CMPM5	Read/Write	0x0
0x18C	PWM Comparison 5 Mode Update Register	PWM_CMPMUPD5	Write-only	–
0x190	PWM Comparison 6 Value Register	PWM_CMPV6	Read/Write	0x0
0x194	PWM Comparison 6 Value Update Register	PWM_CMPVUPD6	Write-only	–
0x198	PWM Comparison 6 Mode Register	PWM_CMPM6	Read/Write	0x0
0x19C	PWM Comparison 6 Mode Update Register	PWM_CMPMUPD6	Write-only	–
0x1A0	PWM Comparison 7 Value Register	PWM_CMPV7	Read/Write	0x0
0x1A4	PWM Comparison 7 Value Update Register	PWM_CMPVUPD7	Write-only	–
0x1A8	PWM Comparison 7 Mode Register	PWM_CMPM7	Read/Write	0x0
0x1AC	PWM Comparison 7 Mode Update Register	PWM_CMPMUPD7	Write-only	–
0x1B0–0x1FC	Reserved	–	–	–

Table 48-6. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x200 + ch_num * 0x20 + 0x00	PWM Channel Mode Register[1]	PWM_CMR	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x04	PWM Channel Duty Cycle Register[1]	PWM_CDTY	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x08	PWM Channel Duty Cycle Update Register[1]	PWM_CDTYUPD	Write-only	–
0x200 + ch_num * 0x20 + 0x0C	PWM Channel Period Register[1]	PWM_CPRD	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x10	PWM Channel Period Update Register[1]	PWM_CPRDUPD	Write-only	–
0x200 + ch_num * 0x20 + 0x14	PWM Channel Counter Register[1]	PWM_CCNT	Read-only	0x0
0x200 + ch_num * 0x20 + 0x18	PWM Channel Dead Time Register[1]	PWM_DT	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x1C	PWM Channel Dead Time Update Register[1]	PWM_DTUPD	Write-only	–

Notes: 1. Some registers are indexed with “ch_num” index ranging from 0 to 3.

48.7.1 PWM Clock Register

Name: PWM_CLK

Address: 0xF002C000

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							

This register can only be written if bits WPSWS0 and WPHWS0 are cleared in the “PWM Write Protection Status Register” .

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

DIVA, DIVB	CLKA, CLKB
0	CLKA, CLKB clock is turned off
1	CLKA, CLKB clock is clock selected by PREA, PREB
2–255	CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor.

- **PREA, PREB: CLKA, CLKB Source Clock Selection**

PREA, PREB				Divider Input Clock
0	0	0	0	MCK
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
Other				Reserved

48.7.2 PWM Enable Register

Name: PWM_ENA

Address: 0xF002C004

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.

48.7.3 PWM Disable Register

Name: PWM_DIS

Address: 0xF002C008

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

This register can only be written if bits WPSWS1 and WPHWS1 are cleared in the “PWM Write Protection Status Register” .

- **CHIDx: Channel ID**

0: No effect.

1: Disable PWM output for channel x.

48.7.4 PWM Status Register

Name: PWM_SR

Address: 0xF002C00C

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.

48.7.5 PWM Interrupt Enable Register 1

Name: PWM_IER1

Address: 0xF002C010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Enable**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Enable**

48.7.6 PWM Interrupt Disable Register 1

Name: PWM_IDR1

Address: 0xF002C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Disable**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Disable**

48.7.7 PWM Interrupt Mask Register 1

Name: PWM_IMR1

Address: 0xF002C018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Mask**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Mask**

48.7.8 PWM Interrupt Status Register 1

Name: PWM_ISR1

Address: 0xF002C01C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x**

0: No new counter event has occurred since the last read of the PWM_ISR1.

1: At least one counter event has occurred since the last read of the PWM_ISR1.

- **FCHIDx: Fault Protection Trigger on Channel x**

0: No new trigger of the fault protection since the last read of the PWM_ISR1.

1: At least one trigger of the fault protection since the last read of the PWM_ISR1.

Note: Reading PWM_ISR1 automatically clears CHIDx and FCHIDx flags.

48.7.9 PWM Sync Channels Mode Register

Name: PWM_SCM

Address: 0xF002C020

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	UPDM	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	SYNC3	SYNC2	SYNC1	SYNC0

This register can only be written if bits WPSWS2 and WPHWS2 are cleared in the "PWM Write Protection Status Register" .

- **SYNCx: Synchronous Channel x**

0: Channel x is not a synchronous channel.

1: Channel x is a synchronous channel.

- **UPDM: Synchronous Channels Update Mode**

Value	Name	Description
0	MODE0	Manual write of double buffer registers and manual update of synchronous channels(1)
1	MODE1	Manual write of double buffer registers and automatic update of synchronous channels(2)

Notes: 1. The update occurs at the beginning of the next PWM period, when the UPDULOCK bit in "PWM Sync Channels Update Control Register" is set.

2. The update occurs when the Update Period is elapsed.

48.7.10 PWM Sync Channels Update Control Register

Name: PWM_SCUC

Address: 0xF002C028

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
–	–	–	–	–	–	–	UPDULOCK

- **UPDULOCK: Synchronous Channels Update Unlock**

0: No effect

1: If the UPDM field is set to ‘0’ in “PWM Sync Channels Mode Register” , writing the UPDULOCK bit to ‘1’ triggers the update of the period value, the duty-cycle and the dead-time values of synchronous channels at the beginning of the next PWM period. If the field UPDM is set to ‘1’ or ‘2’, writing the UPDULOCK bit to ‘1’ triggers only the update of the period value and of the dead-time values of synchronous channels.

This bit is automatically reset when the update is done.

48.7.11 PWM Sync Channels Update Period Register

Name: PWM_SCUP

Address: 0xF002C02C

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
UPRCNT				UPR			

- **UPR: Update Period**

Defines the time between each update of the synchronous channels if automatic trigger of the update is activated (UPDM = 1 or UPDM = 2 in “PWM Sync Channels Mode Register”). This time is equal to UPR+1 periods of the synchronous channels.

- **UPRCNT: Update Period Counter**

Reports the value of the Update Period Counter.

48.7.12 PWM Sync Channels Update Period Update Register

Name: PWM_SCUPUPD

Address: 0xF002C030

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
–	–	–	–	UPRUPD			

This register acts as a double buffer for the UPR value. This prevents an unexpected automatic trigger of the update of synchronous channels.

- **UPRUPD: Update Period Update**

Defines the wanted time between each update of the synchronous channels if automatic trigger of the update is activated (UPDM = 1 or UPDM = 2 in “PWM Sync Channels Mode Register”). This time is equal to UPR+1 periods of the synchronous channels.

48.7.13 PWM Interrupt Enable Register 2

Name: PWM_IER2

Address: 0xF002C034

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
–	–	–	–	UNRE			WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Enable**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Enable**
- **CMPMx: Comparison x Match Interrupt Enable**
- **CMPUx: Comparison x Update Interrupt Enable**

48.7.14 PWM Interrupt Disable Register 2

Name: PWM_IDR2

Address: 0xF002C038

Access: Write-only

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

23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0

15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0

7	6	5	4	3	2	1	0
–	–	–	–	UNRE			WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Disable**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Disable**
- **CMPMx: Comparison x Match Interrupt Disable**
- **CMPUx: Comparison x Update Interrupt Disable**

48.7.15 PWM Interrupt Mask Register 2

Name: PWM_IMR2

Address: 0xF002C03C

Access: Read-only

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

23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0

15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0

7	6	5	4	3	2	1	0
–	–	–	–	UNRE			WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Mask**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Mask**
- **CMPMx: Comparison x Match Interrupt Mask**
- **CMPUx: Comparison x Update Interrupt Mask**

48.7.16 PWM Interrupt Status Register 2

Name: PWM_ISR2

Address: 0xF002C040

Access: Read-only

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

23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0

15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0

7	6	5	4	3	2	1	0
–	–	–	–	UNRE			WRDY

- **WRDY: Write Ready for Synchronous Channels Update**

0: New duty-cycle and dead-time values for the synchronous channels cannot be written.

1: New duty-cycle and dead-time values for the synchronous channels can be written.

- **UNRE: Synchronous Channels Update Underrun Error**

0: No Synchronous Channels Update Underrun has occurred since the last read of the PWM_ISR2 register.

1: At least one Synchronous Channels Update Underrun has occurred since the last read of the PWM_ISR2 register.

- **CMPMx: Comparison x Match**

0: The comparison x has not matched since the last read of the PWM_ISR2 register.

1: The comparison x has matched at least one time since the last read of the PWM_ISR2 register.

- **CMPUx: Comparison x Update**

0: The comparison x has not been updated since the last read of the PWM_ISR2 register.

1: The comparison x has been updated at least one time since the last read of the PWM_ISR2 register.

Note: Reading PWM_ISR2 automatically clears flags WRDY, UNRE and CMPSx.

48.7.17 PWM Output Override Value Register

Name: PWM_OOV

Address: 0xF002C044

Access: Read/Write

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

23	22	21	20	19	18	17	16
–	–	–	–	OOVL3	OOVL2	OOVL1	OOVL0

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	OOVH3	OOVH2	OOVH1	OOVH0

- **OOVHx: Output Override Value for PWMH output of the channel x**

0: Override value is 0 for PWMH output of channel x.

1: Override value is 1 for PWMH output of channel x.

- **OOVLx: Output Override Value for PWML output of the channel x**

0: Override value is 0 for PWML output of channel x.

1: Override value is 1 for PWML output of channel x.

48.7.18 PWM Output Selection Register

Name: PWM_OS

Address: 0xF002C048

Access: Read/Write

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

23	22	21	20	19	18	17	16
–	–	–	–	OSL3	OSL2	OSL1	OSL0

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	OSH3	OSH2	OSH1	OSH0

- **OSHx: Output Selection for PWMH output of the channel x**

0: Dead-time generator output DTOHx selected as PWMH output of channel x.

1: Output override value OOVHx selected as PWMH output of channel x.

- **OSLx: Output Selection for PWML output of the channel x**

0: Dead-time generator output DTOLx selected as PWML output of channel x.

1: Output override value OOVLx selected as PWML output of channel x.

48.7.19 PWM Output Selection Set Register

Name: PWM_OSS

Address: 0xF002C04C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSSL3	OSSL2	OSSL1	OSSL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSSH3	OSSH2	OSSH1	OSSH0

- **OSSHx: Output Selection Set for PWMH output of the channel x**

0: No effect.

1: Output override value OOVHx selected as PWMH output of channel x.

- **OSSLx: Output Selection Set for PWML output of the channel x**

0: No effect.

1: Output override value OOVLx selected as PWML output of channel x.

48.7.20 PWM Output Selection Clear Register

Name: PWM_OSC

Address: 0xF002C050

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSCL3	OSCL2	OSCL1	OSCL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCH3	OSCH2	OSCH1	OSCH0

- **OSCHx: Output Selection Clear for PWMH output of the channel x**

0: No effect.

1: Dead-time generator output DTOHx selected as PWMH output of channel x.

- **OSCLx: Output Selection Clear for PWML output of the channel x**

0: No effect.

1: Dead-time generator output DTOLx selected as PWML output of channel x.

48.7.21 PWM Output Selection Set Update Register

Name: PWM_OSSUPD

Address: 0xF002C054

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSSUPL3	OSSUPL2	OSSUPL1	OSSUPL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSSUPH3	OSSUPH2	OSSUPH1	OSSUPH0

- **OSSUPHx: Output Selection Set for PWMH output of the channel x**

0: No effect.

1: Output override value OOVHx selected as PWMH output of channel x at the beginning of the next channel x PWM period.

- **OSSUPLx: Output Selection Set for PWML output of the channel x**

0: No effect.

1: Output override value OOVLx selected as PWML output of channel x at the beginning of the next channel x PWM period.

48.7.22 PWM Output Selection Clear Update Register

Name: PWM_OSCUPD

Address: 0xF002C058

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSCUPL3	OSCUPL2	OSCUPL1	OSCUPL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCUPH3	OSCUPH2	OSCUPH1	OSCUPH0

- **OSCUPHx: Output Selection Clear for PWMH output of the channel x**

0: No effect.

1: Dead-time generator output DTOHx selected as PWMH output of channel x at the beginning of the next channel x PWM period.

- **OSCUPLx: Output Selection Clear for PWML output of the channel x**

0: No effect.

1: Dead-time generator output DTOLx selected as PWML output of channel x at the beginning of the next channel x PWM period.

48.7.23 PWM Fault Mode Register

Name: PWM_FMR

Address: 0xF002C05C

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
FFIL							

15	14	13	12	11	10	9	8
FMOD							

7	6	5	4	3	2	1	0
FPOL							

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the “PWM Write Protection Status Register” .

- **FPOL: Fault Polarity**

For each field bit y (fault input number):

0: The fault y becomes active when the fault input y is at 0.

1: The fault y becomes active when the fault input y is at 1.

- **FMOD: Fault Activation Mode**

For each field bit y (fault input number):

0: The fault y is active until the Fault condition is removed at the peripheral(1) level.

1: The fault y stays active until the Fault condition is removed at the peripheral(1) level AND until it is cleared in the “PWM Fault Clear Register” .

Note: 1. The Peripheral generating the fault.

- **FFIL: Fault Filtering**

For each field bit y (fault input number):

0: The fault input y is not filtered.

1: The fault input y is filtered.

CAUTION: To prevent an unexpected activation of the status flag FSy in the “PWM Fault Status Register” , the bit FMODy can be set to ‘1’ only if the FPOLy bit has been previously configured to its final value.

48.7.24 PWM Fault Status Register

Name: PWM_FSR

Address: 0xF002C060

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
FS							

7	6	5	4	3	2	1	0
FIV							

- **FIV: Fault Input Value**

For each field bit y (fault input number):

0: The current sampled value of the fault input y is 0 (after filtering if enabled).

1: The current sampled value of the fault input y is 1 (after filtering if enabled).

- **FS: Fault Status**

For each field bit y (fault input number):

0: The fault y is not currently active.

1: The fault y is currently active.

48.7.25 PWM Fault Clear Register

Name: PWM_FCR

Address: 0xF002C064

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
FCLR							

- **FCLR: Fault Clear**

For each field bit y (fault input number):

0: No effect.

1: If bit y of FMOD field is set to ‘1’ and if the fault input y is not at the level defined by the bit y of FPOL field, the fault y is cleared and becomes inactive (FMOD and FPOL fields belong to “PWM Fault Mode Register”), else writing this bit to ‘1’ has no effect.

48.7.26 PWM Fault Protection Value Register

Name: PWM_FPV

Address: 0xF002C068

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FPVL3	FPVL2	FPVL1	FPVL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	FPVH3	FPVH2	FPVH1	FPVH0

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the “PWM Write Protection Status Register” .

- **FPVHx: Fault Protection Value for PWMH output on channel x**

0: PWMH output of channel x is forced to ‘0’ when fault occurs.

1: PWMH output of channel x is forced to ‘1’ when fault occurs.

- **FPVLx: Fault Protection Value for PWML output on channel x**

0: PWML output of channel x is forced to ‘0’ when fault occurs.

1: PWML output of channel x is forced to ‘1’ when fault occurs.

48.7.27 PWM Fault Protection Enable Register

Name: PWM_FPE

Address: 0xF002C06C

Access: Read/Write

31	30	29	28	27	26	25	24
FPE3							
23	22	21	20	19	18	17	16
FPE2							
15	14	13	12	11	10	9	8
FPE1							
7	6	5	4	3	2	1	0
FPE0							

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the “PWM Write Protection Status Register” .

Only the first 1 bits (number of fault input pins) of fields FPE0, FPE1, FPE2 and FPE3 are significant.

- **FPEx: Fault Protection Enable for channel x**

For each field bit y (fault input number):

0: Fault y is not used for the Fault Protection of channel x.

1: Fault y is used for the Fault Protection of channel x.

CAUTION: To prevent an unexpected activation of the Fault Protection, the bit y of FPEx field can be set to ‘1’ only if the corresponding FPOL field has been previously configured to its final value in “PWM Fault Mode Register” .

48.7.28 PWM Event Line x Register

Name: PWM_ELMRx

Address: 0xF002C07C

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
CSEL7	CSEL6	CSEL5	CSEL4	CSEL3	CSEL2	CSEL1	CSEL0

- **CSELy: Comparison y Selection**

0: A pulse is not generated on the event line x when the comparison y matches.

1: A pulse is generated on the event line x when the comparison y match.

48.7.29 PWM Stepper Motor Mode Register

Name: PWM_SMMR

Address: 0xF002C0B0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	DOWN1	DOWN0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	GCEN1	GCEN0

- **GCENx: Gray Count ENable**

0: Disable gray count generation on PWML[2*x], PWMH[2*x], PWML[2*x +1], PWMH[2*x +1]

1: enable gray count generation on PWML[2*x], PWMH[2*x], PWML[2*x +1], PWMH[2*x +1.

- **DOWNx: DOWN Count**

0: Up counter.

1: Down counter.

48.7.30 PWM Write Protection Control Register

Name: PWM_WPCR

Address: 0xF002C0E4

Access: Write-only

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
WPRG5	WPRG4	WPRG3	WPRG2	WPRG1	WPRG0	WPCMD	

- **WPCMD: Write Protect Command**

This command is performed only if the WPKEY value is correct (0x50574D, “PWM” in ASCCII).

Value	Name	Description
0	DISABLE_SW_PROT	Disable the Software Write Protect of the register groups of which the bit WPRGx is at ‘1’.
1	ENABLE_SW_PROT	Enable the Software Write Protect of the register groups of which the bit WPRGx is at ‘1’.
2	ENABLE_HW_PROT	Enable the Hardware Write Protect of the register groups of which the bit WPRGx is at ‘1’. Only a hardware reset of the PWM controller can disable the hardware write protect. Moreover, to meet security requirements, the PIO lines associated with PWM can not be configured through the PIO interface.

- **WPRGx: Write Protect Register Group x**

0: The WPCMD command has no effect on the register group x.

1: The WPCMD command is applied to the register group x.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x50574D	PASSWD	Writing any other value in this field aborts the write operation of the WPCMD field. Always reads as 0

List of register groups:

- Register group 0:
  - “PWM Clock Register” on page 1572
- Register group 1:
  - “PWM Disable Register” on page 1574
- Register group 2:
  - “PWM Sync Channels Mode Register” on page 1580
  - “PWM Channel Mode Register” on page 1608
  - “PWM Stepper Motor Mode Register” on page 1600
- Register group 3:

  - “PWM Channel Period Register” on page 1612
  - “PWM Channel Period Update Register” on page 1613
- Register group 4:
  - “PWM Channel Dead Time Register” on page 1615
  - “PWM Channel Dead Time Update Register” on page 1616
- Register group 5:
  - “PWM Fault Mode Register” on page 1594
  - “PWM Fault Protection Value Register” on page 1597

48.7.31 PWM Write Protection Status Register

Name: PWM_WPSR

Address: 0xF002C0E8

Access: Read-only

31	30	29	28	27	26	25	24
WPVSRC							

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
–	–	WPHWS5	WPHWS4	WPHWS3	WPHWS2	WPHWS1	WPHWS0

7	6	5	4	3	2	1	0
WPVS	–	WPSWS5	WPSWS4	WPSWS3	WPSWS2	WPSWS1	WPSWS0

- **WPSWSx: Write Protect SW Status**

0: The Write Protect SW x of the register group x is disabled.

1: The Write Protect SW x of the register group x is enabled.

- **WPHWSx: Write Protect HW Status**

0: The Write Protect HW x of the register group x is disabled.

1: The Write Protect HW x of the register group x is enabled.

- **WPVS: Write Protect Violation Status**

0: No Write Protect violation has occurred since the last read of the PWM_WPSR.

1: At least one Write Protect violation has occurred since the last read of the PWM_WPSR. 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) in which a write access has been attempted.

Note: The two LSBs of the address offset of the write-protected register are not reported

Note: Reading PWM_WPSR automatically clears WPVS and WPVSRC fields.

48.7.32 PWM Comparison x Value Register

Name: PWM_CMPVx

Address: 0xF002C130 [0], 0xF002C140 [1], 0xF002C150 [2], 0xF002C160 [3], 0xF002C170 [4], 0xF002C180 [5], 0xF002C190 [6], 0xF002C1A0 [7]

Access: Read/Write

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

Only the first 16 bits (channel counter size) of field CV are significant.

- **CV: Comparison x Value**

Define the comparison x value to be compared with the counter of the channel 0.

- **CVM: Comparison x Value Mode**

0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.

1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decrementing.

Note: This bit is useless if the counter of the channel 0 is left aligned (CALG = 0 in “PWM Channel Mode Register” on page 1608)

48.7.33 PWM Comparison x Value Update Register

Name: PWM_CMPVUPDx

Address: 0xF002C134 [0], 0xF002C144 [1], 0xF002C154 [2], 0xF002C164 [3], 0xF002C174 [4], 0xF002C184 [5], 0xF002C194 [6], 0xF002C1A4 [7]

Access: Write-only

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

23	22	21	20	19	18	17	16
CVUPD							

15	14	13	12	11	10	9	8
CVUPD							

7	6	5	4	3	2	1	0
CVUPD							

This register acts as a double buffer for the CV and CVM values. This prevents an unexpected comparison x match.

Only the first 16 bits (channel counter size) of field CVUPD are significant.

- **CVUPD: Comparison x Value Update**

Define the comparison x value to be compared with the counter of the channel 0.

- **CVMUPD: Comparison x Value Mode Update**

0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.

1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decrementing.

Note: This bit is useless if the counter of the channel 0 is left aligned (CALG = 0 in “PWM Channel Mode Register”)

CAUTION: to be taken into account, the write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

48.7.34 PWM Comparison x Mode Register

Name: PWM_CMPMx

Address: 0xF002C138 [0], 0xF002C148 [1], 0xF002C158 [2], 0xF002C168 [3], 0xF002C178 [4], 0xF002C188 [5], 0xF002C198 [6], 0xF002C1A8 [7]

Access: Read/Write

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

23	22	21	20	19	18	17	16
CUPRCNT				CUPR			

15	14	13	12	11	10	9	8
CPRCNT				CPR			

7	6	5	4	3	2	1	0
CTR				–	–	–	CEN

- **CEN: Comparison x Enable**

0: The comparison x is disabled and can not match.

1: The comparison x is enabled and can match.

- **CTR: Comparison x Trigger**

The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.

- **CPR: Comparison x Period**

CPR defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.

- **CPRCNT: Comparison x Period Counter**

Reports the value of the comparison x period counter.

Note: The field CPRCNT is read-only

- **CUPR: Comparison x Update Period**

Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.

- **CUPRCNT: Comparison x Update Period Counter**

Reports the value of the comparison x update period counter.

Note: The field CUPRCNT is read-only

48.7.35 PWM Comparison x Mode Update Register

Name: PWM_CMPMUPDx

Address: 0xF002C13C [0], 0xF002C14C [1], 0xF002C15C [2], 0xF002C16C [3], 0xF002C17C [4], 0xF002C18C [5], 0xF002C19C [6], 0xF002C1AC [7]

Access: Write-only

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

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

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

7	6	5	4	3	2	1	0
CTRUPD				–	–	–	CENUPD

This register acts as a double buffer for the CEN, CTR, CPR and CUPR values. This prevents an unexpected comparison x match.

- **CENUPD: Comparison x Enable Update**

0: The comparison x is disabled and can not match.

1: The comparison x is enabled and can match.

- **CTRUPD: Comparison x Trigger Update**

The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.

- **CPRUPD: Comparison x Period Update**

CPR defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.

- **CUPRUPD: Comparison x Update Period Update**

Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.

48.7.36 PWM Channel Mode Register

Name: PWM_CMRx [x=0..3]

Address: 0xF002C200 [0], 0xF002C220 [1], 0xF002C240 [2], 0xF002C260 [3]

Access: Read/Write

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

This register can only be written if bits WPSWS2 and WPHWS2 are cleared in the “PWM Write Protection Status Register” .

- **CPRE: Channel Pre-scaler**

Value	Name	Description
0b0000	MCK	Master clock
0b0001	MCK_DIV_2	Master clock/2
0b0010	MCK_DIV_4	Master clock/4
0b0011	MCK_DIV_8	Master clock/8
0b0100	MCK_DIV_16	Master clock/16
0b0101	MCK_DIV_32	Master clock/32
0b0110	MCK_DIV_64	Master clock/64
0b0111	MCK_DIV_128	Master clock/128
0b1000	MCK_DIV_256	Master clock/256
0b1001	MCK_DIV_512	Master clock/512
0b1010	MCK_DIV_1024	Master clock/1024
0b1011	CLKA	Clock A
0b1100	CLKB	Clock B

- **CALG: Channel Alignment**

0: The period is left aligned.

1: The period is center aligned.

- **CPOL: Channel Polarity**

0: The OCx output waveform (output from the comparator) starts at a low level.

1: The OCx output waveform (output from the comparator) starts at a high level.

- **CES: Counter Event Selection**

The bit CES defines when the channel counter event occurs when the period is center aligned (flag CHIDx in “PWM Interrupt Status Register 1”).

CALG = 0 (Left Alignment):

0/1: The channel counter event occurs at the end of the PWM period.

CALG = 1 (Center Alignment):

0: The channel counter event occurs at the end of the PWM period.

1: The channel counter event occurs at the end of the PWM period and at half the PWM period.

- **DTE: Dead-Time Generator Enable**

0: The dead-time generator is disabled.

1: The dead-time generator is enabled.

- **DTHI: Dead-Time PWMHx Output Inverted**

0: The dead-time PWMHx output is not inverted.

1: The dead-time PWMHx output is inverted.

- **DTLI: Dead-Time PWMLx Output Inverted**

0: The dead-time PWMLx output is not inverted.

1: The dead-time PWMLx output is inverted.

48.7.37 PWM Channel Duty Cycle Register

Name: PWM_CDTYx [x=0..3]

Address: 0xF002C204 [0], 0xF002C224 [1], 0xF002C244 [2], 0xF002C264 [3]

Access: Read/Write

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

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 (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).

48.7.38 PWM Channel Duty Cycle Update Register

Name: PWM_CDTYUPDx [x=0..3]

Address: 0xF002C208 [0], 0xF002C228 [1], 0xF002C248 [2], 0xF002C268 [3]

Access: Write-only.

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

23	22	21	20	19	18	17	16
CDTYUPD							

15	14	13	12	11	10	9	8
CDTYUPD							

7	6	5	4	3	2	1	0
CDTYUPD							

This register acts as a double buffer for the CDTY value. This prevents an unexpected waveform when modifying the waveform duty-cycle.

Only the first 16 bits (channel counter size) are significant.

- **CDTYUPD: Channel Duty-Cycle Update**

Defines the waveform duty-cycle. This value must be defined between 0 and CPRD (PWM_CPRx).

48.7.39 PWM Channel Period Register

Name: PWM_CPRDx [x=0..3]

Address: 0xF002C20C [0], 0xF002C22C [1], 0xF002C24C [2], 0xF002C26C [3]

Access: Read/Write

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

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							

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the “PWM Write Protection Status Register” .

Only the first 16 bits (channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

– By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

48.7.40 PWM Channel Period Update Register

Name: PWM_CPRDUPDx [x=0..3]

Address: 0xF002C210 [0], 0xF002C230 [1], 0xF002C250 [2], 0xF002C270 [3]

Access: Write-only

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

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the “PWM Write Protection Status Register” .

This register acts as a double buffer for the CPRD value. This prevents an unexpected waveform when modifying the waveform period.

Only the first 16 bits (channel counter size) are significant.

- **CPRDUPD: Channel Period Update**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRDUPD)}{MCK}$$

– By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPDUPD \times DIVA)}{MCK} \text{ or } \frac{(CRPDUPD \times DIVB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRDUPD)}{MCK}$$

– By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRDUPD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRDUPD \times DIVB)}{MCK}$$

48.7.41 PWM Channel Counter Register

Name: PWM_CCNTx [x=0..3]

Address: 0xF002C214 [0], 0xF002C234 [1], 0xF002C254 [2], 0xF002C274 [3]

Access: Read-only

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

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							

Only the first 16 bits (channel counter size) are significant.

- **CNT: Channel Counter Register**

Channel counter value. This register is reset when:

- the channel is enabled (writing CHIDx in the PWM_ENA register).
- the channel counter reaches CPRD value defined in the PWM_CPRDx register if the waveform is left aligned.

48.7.42 PWM Channel Dead Time Register

Name: PWM_DTx [x=0..3]

Address: 0xF002C218 [0], 0xF002C238 [1], 0xF002C258 [2], 0xF002C278 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
DTL							

23	22	21	20	19	18	17	16
DTL							

15	14	13	12	11	10	9	8
DTH							

7	6	5	4	3	2	1	0
DTH							

This register can only be written if bits WPSWS4 and WPHWS4 are cleared in the “PWM Write Protection Status Register” .

Only the first 12 bits (dead-time counter size) of fields DTH and DTL are significant.

- **DTH: Dead-Time Value for PWMHx Output**

Defines the dead-time value for PWMHx output. This value must be defined between 0 and CPRD-CDTY (PWM_CPRx and PWM_CDTYx).

- **DTL: Dead-Time Value for PWMLx Output**

Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (PWM_CDTYx).

48.7.43 PWM Channel Dead Time Update Register

Name: PWM_DTUPDx [x=0..3]

Address: 0xF002C21C [0], 0xF002C23C [1], 0xF002C25C [2], 0xF002C27C [3]

Access: Write-only

31	30	29	28	27	26	25	24
DTLUPD							

23	22	21	20	19	18	17	16
DTLUPD							

15	14	13	12	11	10	9	8
DTHUPD							

7	6	5	4	3	2	1	0
DTHUPD							

This register can only be written if bits WPSWS4 and WPHWS4 are cleared in the “PWM Write Protection Status Register” .

This register acts as a double buffer for the DTH and DTL values. This prevents an unexpected waveform when modifying the dead-time values.

Only the first 12 bits (dead-time counter size) of fields DTHUPD and DTLUPD are significant.

- **DTHUPD: Dead-Time Value Update for PWMHx Output**

Defines the dead-time value for PWMHx output. This value must be defined between 0 and CPRD-CDTY (PWM_CPRx and PWM_CDTYx). This value is applied only at the beginning of the next channel x PWM period.

- **DTLUPD: Dead-Time Value Update for PWMLx Output**

Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (PWM_CDTYx). This value is applied only at the beginning of the next channel x PWM period.

49. Analog-to-Digital Converter (ADC)

49.1 Description

The ADC is based on a 12-bit Analog-to-Digital Converter (ADC) managed by an ADC Controller. Refer to Figure 49-1, "Analog-to-Digital Converter Block Diagram". 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 the voltage carried on pin ADVREF.

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 Controller internal fault output is directly connected to PWM Fault input. This input can be asserted by means of comparison circuitry in order to immediately put the PWM outputs in a safe state (pure combinational path).

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.

This ADC has a selectable single-ended or fully differential input and benefits from a 2-bit programmable gain.

A digital error correction circuit based on the multi-bit redundant signed digit (RSD) algorithm is employed in order to reduce INL and DNL errors.

Finally, the user can configure ADC timings, such as Startup Time and Tracking Time.

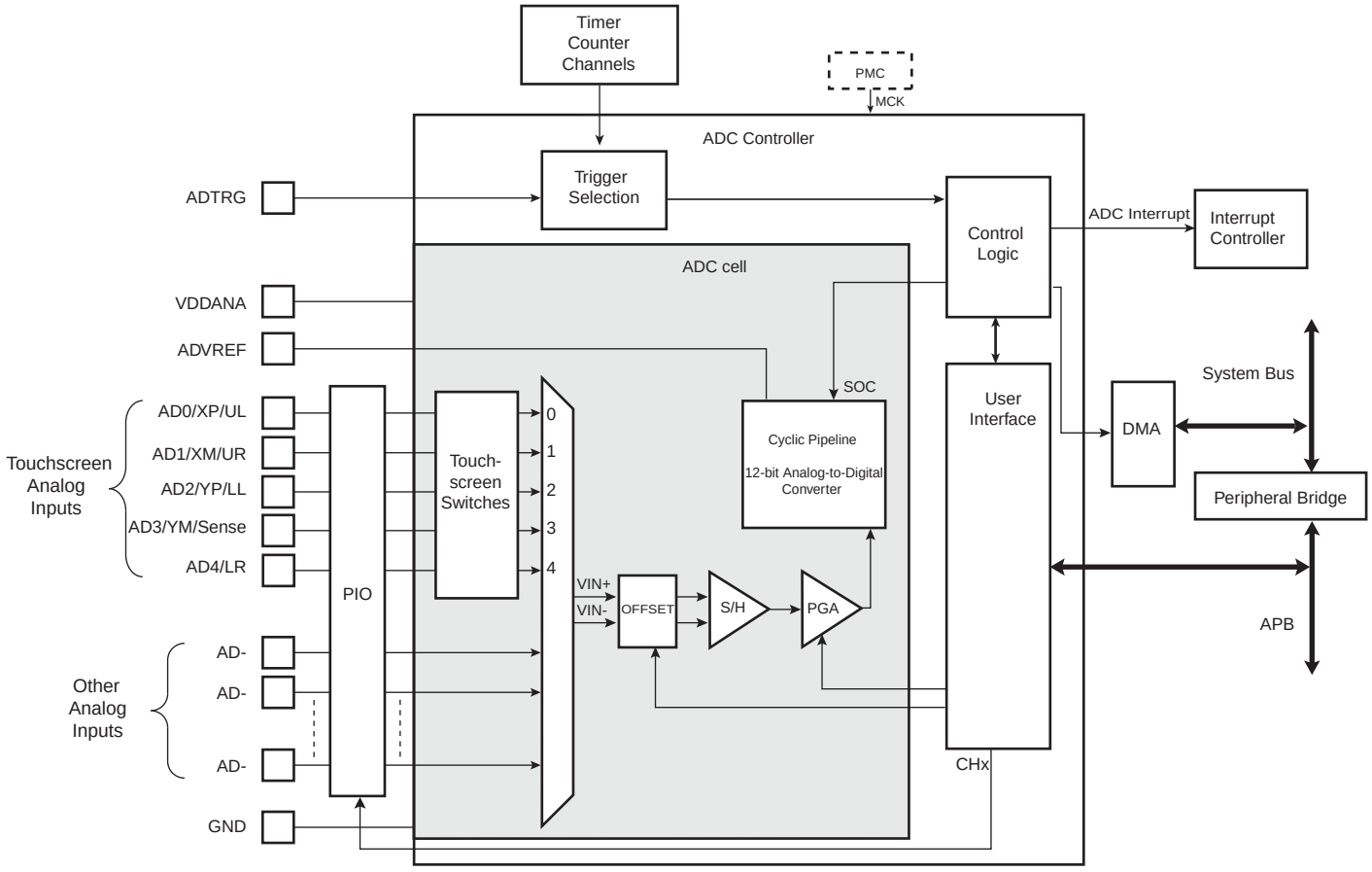
This ADC Controller includes a Resistive Touchscreen Controller. It supports 4-wire and 5-wire technologies.

49.2 Embedded Characteristics

- 12-bit Resolution
- 1 MHz Conversion Rate
- Wide Range Power Supply Operation
- Selectable Single Ended or Differential Input Voltage
- Programmable Gain For Maximum Full Scale Input Range 0 - VDD
- Resistive 4-wire and 5-wire Touchscreen Controller
  - Position and Pressure Measurement for 4-wire screens
  - Position Measurement for 5-wire screens
  - Average of up to 8 measures for noise filtering
- Programmable Pen Detection sensitivity
- Integrated Multiplexer Offering Up to 12 Independent Analog Inputs
- Individual Enable and Disable of Each Channel
- Hardware or Software Trigger
  - External Trigger Pin
  - Timer Counter Outputs (Corresponding TIOA Trigger)
  - Internal Trigger Counter
  - Trigger on Pen Contact Detection
  - PWM Event Line
- Drive of PWM Fault Input
- 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

49.3 Block Diagram

Figure 49-1. Analog-to-Digital Converter Block Diagram



49.4 Signal Description

Table 49-1. ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
ADVREF	Reference voltage
AD0 - AD11	Analog input channels
ADTRG	External trigger

49.5 Product Dependencies

49.5.1 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.

49.5.2 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 49-2. Peripheral IDs

Instance	ID
ADC	29

49.5.3 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.

49.5.4 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 49-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
ADC	ADTRG	PD19	A
ADC	AD0	PD20	A
ADC	AD1	PD21	A
ADC	AD2	PD22	A
ADC	AD3	PD23	A
ADC	AD4	PD24	A
ADC	AD5	PD25	A
ADC	AD6	PD26	A
ADC	AD7	PD27	A
ADC	AD8	PD28	A
ADC	AD9	PD29	A
ADC	AD10	PD30	A
ADC	AD11	PD31	A

49.5.5 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.

49.5.6 PWM Event Line

PWM Event Lines may or may not be used as hardware triggers depending on user requirements.

49.5.7 Fault Output

The ADC Controller has the FAULT output connected to the FAULT input of PWM. Please refer to Section 49.7.11 "Fault Output" and implementation of the PWM in the product.

49.5.8 Conversion Performances

For performance and electrical characteristics of the ADC, see the product electrical characteristics.

49.6 Functional Description

49.6.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 12-bit digital data requires Tracking Clock cycles as defined in the field TRACKTIM of the "ADC Mode Register" on page 1646. 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 49-2. Sequence of ADC conversions when Tracking time > Conversion time

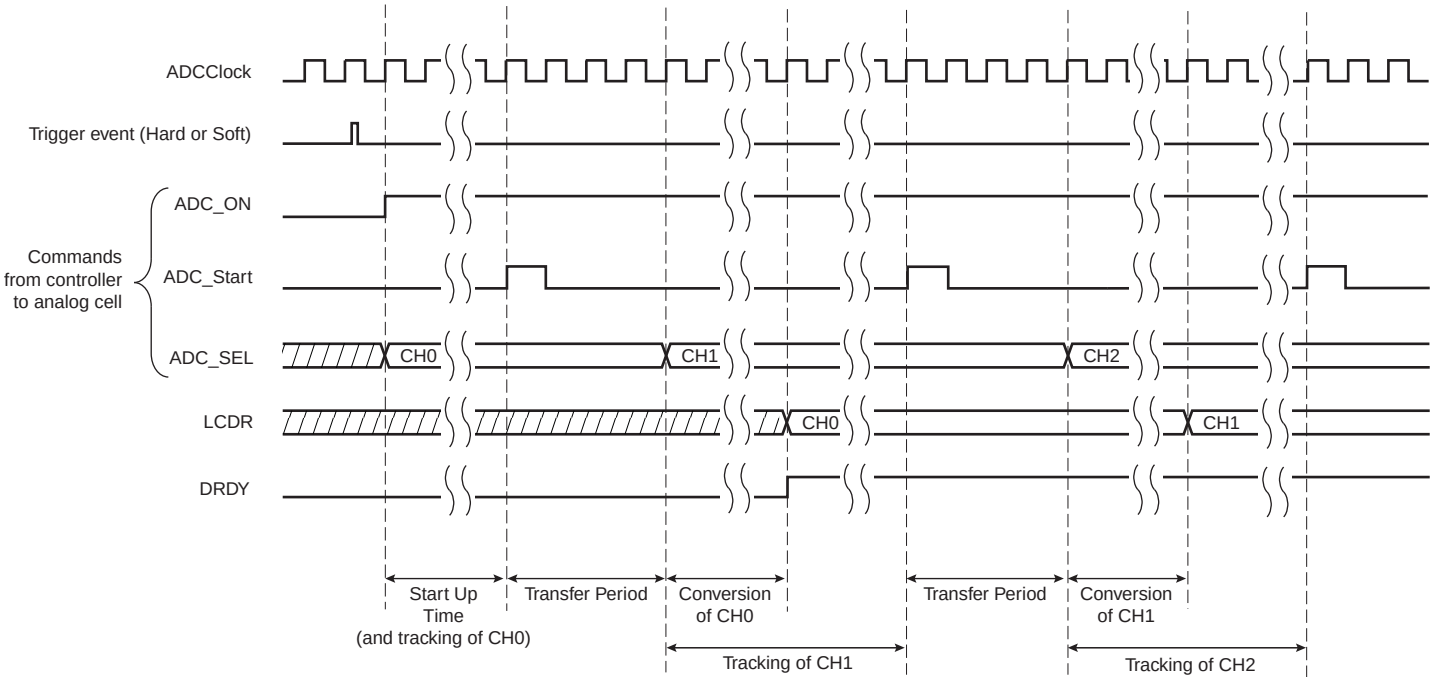
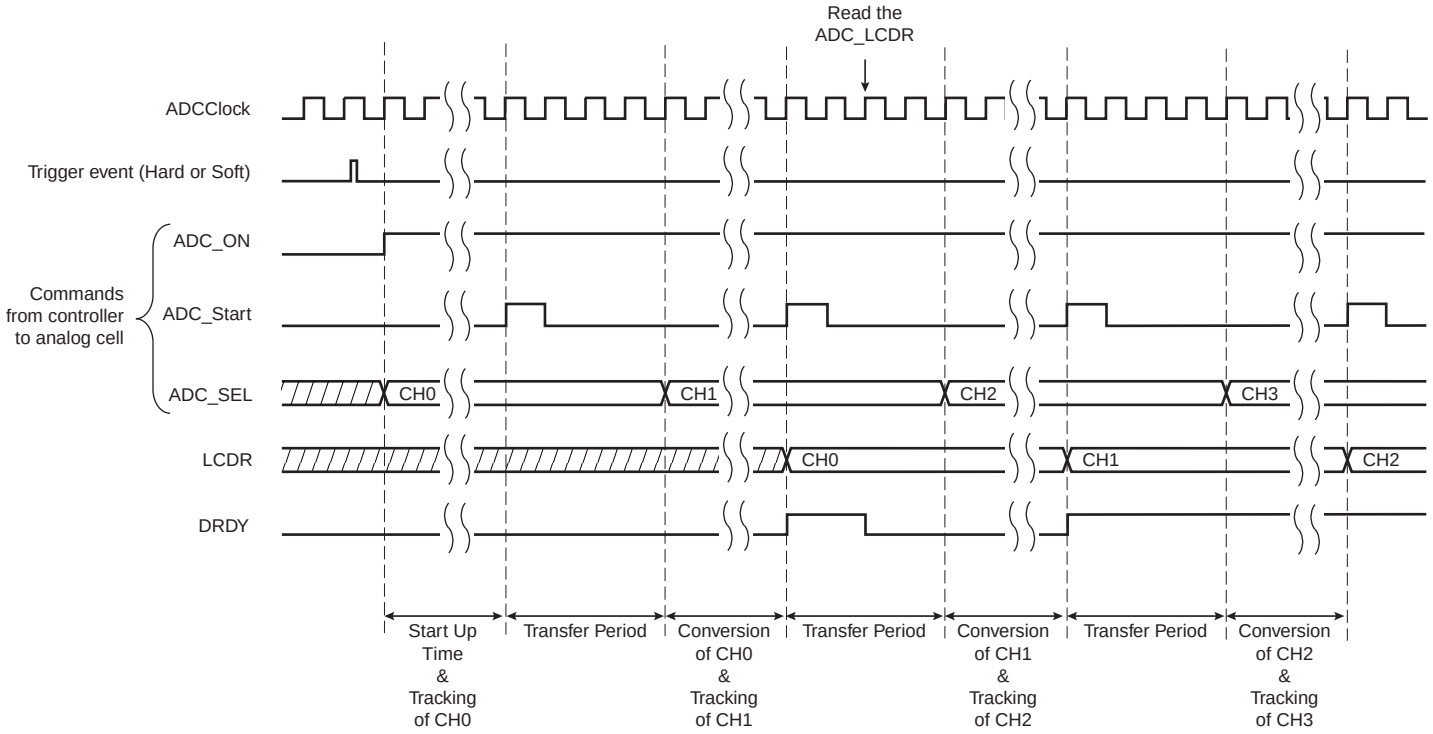


Figure 49-3. Sequence of ADC conversions when Tracking time < Conversion time



49.6.2 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.

49.6.3 Conversion Resolution

The ADC supports 12-bit resolutions.

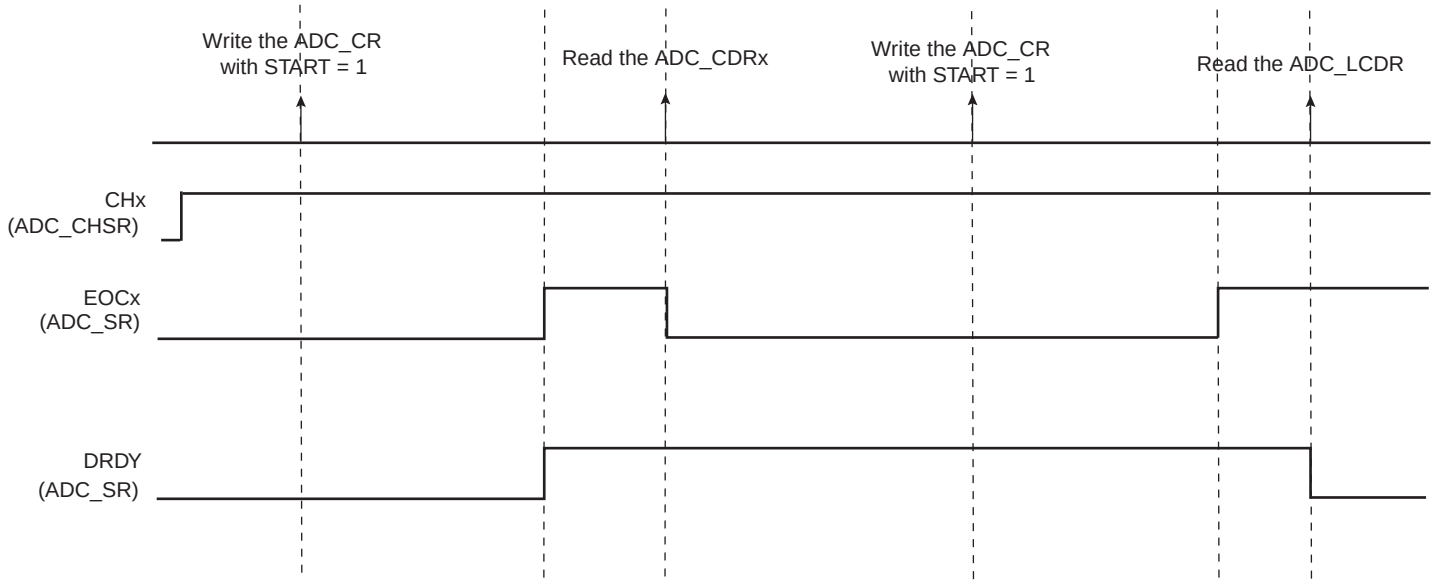
49.6.4 Conversion Results

When a conversion is completed, the resulting 12-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 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 49-4. EOCx and DRDY Flag Behavior

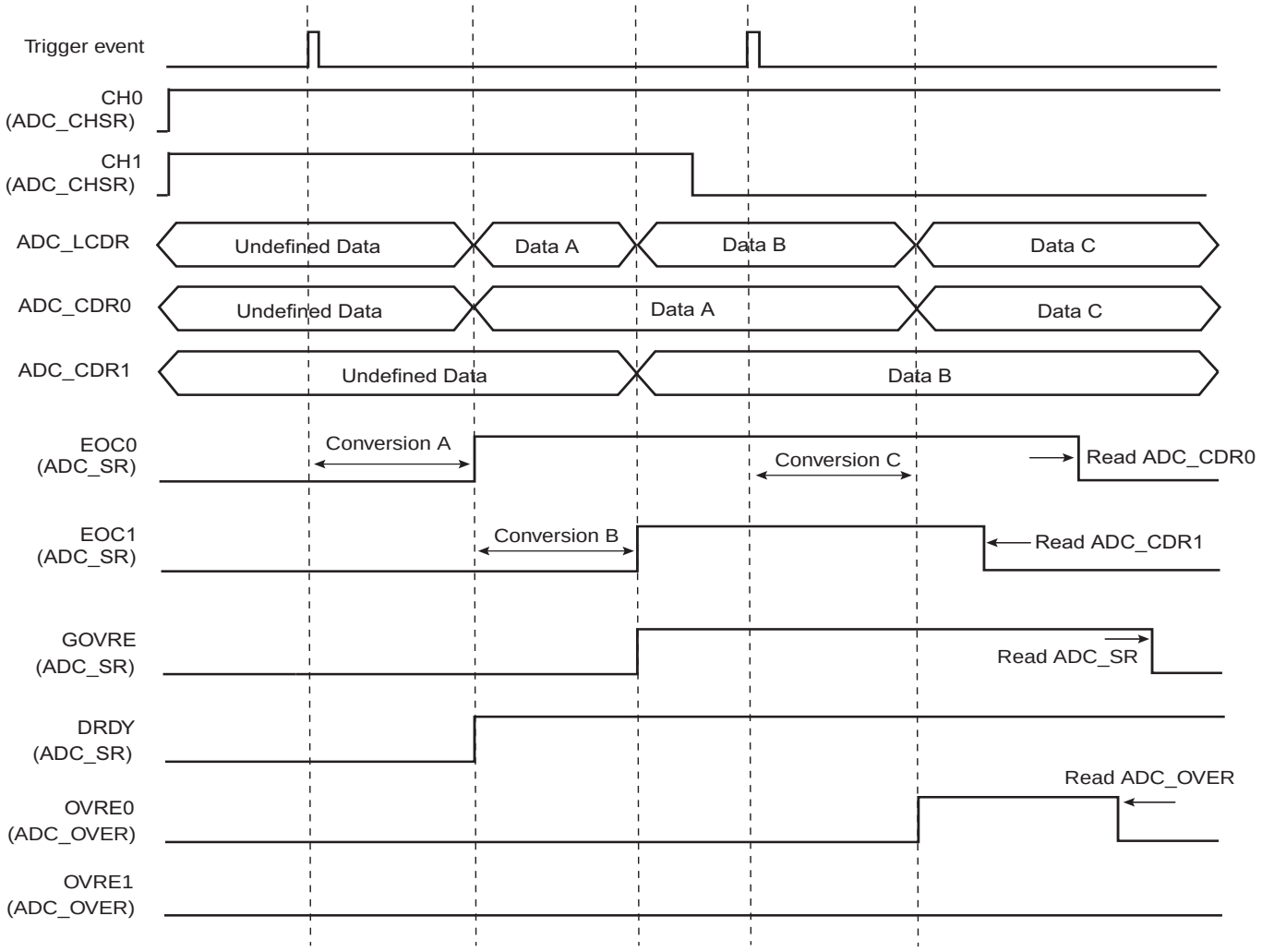


If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVREx) flag is set in the Overrun Status Register (ADC_OVER).

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 49-5. 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.

49.6.5 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 one of the TIOA outputs of the Timer Counter channels, PWM Event line, or the external trigger input of the ADC (ADTRG). The hardware trigger is selected with the TRGSEL field in the Mode Register (ADC_MR). The selected hardware trigger is enabled if TRGMOD=1,2 or 3 in “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.

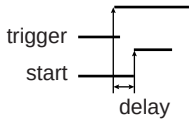
The hardware trigger can be selected by the TRGMOD field in the “ADC Trigger Register” between:

- any edge, either rising or falling or both, detected on the external trigger pin, TSADTRG. The hardware trigger source is selected with the TRGSEL field in the Mode Register (ADC_MR). The selected hardware trigger is enabled if TRGMOD=1,2 or 3 in “ADC Trigger Register”.
- the Pen Detect, depending on how the PENDET bit is set in the “ADC Touchscreen Mode Register”.
- 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, ADC_TSMR.

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.

Figure 49-6. Hardware Trigger Delay



If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform Mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers 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.

49.6.6 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.

A fast wake-up mode is available in the ADC Mode Register (ADC_MR) as a compromise between power saving strategy and responsiveness. Setting the FWUP bit to ‘1’ enables the fast wake-up mode. In fast wake-up mode the ADC cell is not fully deactivated while no conversion is requested, thereby providing less power saving but faster wakeup.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using the internal timer (ADC_TRGR register) or the PWM event line. 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.

49.6.7 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.

49.6.8 Differential Inputs

The ADC can be used either as a single ended ADC (DIFF bit equal to 0) or as a fully differential ADC (DIFF bit equal to 1) as shown in Figure 49-7. By default, after a reset, the ADC is in single ended mode.

If ANACH is set in ADC_MR the ADC can apply a different mode on each channel. Otherwise the parameters of CH0 are applied to all channels.

The same inputs are used in single ended or differential mode.

In single ended mode, inputs are managed by a 16:1 channels analog multiplexer. In the fully differential mode, inputs are managed by an 8:1 channels analog multiplexer. See Table 49-4 and Table 49-5.

Table 49-4. Input Pins and Channel Number in Single Ended Mode

Input Pins	Channel Number
AD0	CH0
AD1	CH1
AD2	CH2
AD3	CH3
AD4	CH4
AD5	CH5
AD6	CH6
AD7	CH7
AD8	CH8
AD9	CH9
AD10	CH10
AD11	CH11
AD12	CH12
AD13	CH13
AD14	CH14
AD15	CH15

Table 49-5. Input Pins and Channel Number In Differential Mode

Input Pins	Channel Number
AD0-AD1	CH0
AD2-AD3	CH2
AD4-AD5	CH4
AD6-AD7	CH6
AD8-AD9	CH8
AD10-AD11	CH10
AD12-AD13	CH12
AD14-AD15	CH14

49.6.9 Input Gain and Offset

The ADC has a built in Programmable Gain Amplifier (PGA) and Programmable Offset.

The Programmable Gain Amplifier can be set to gains of 1/2, 1, 2 and 4. The Programmable Gain Amplifier can be used either for single ended applications or for fully differential applications.

If ANACH is set in ADC_MR the ADC can apply different gain and offset on each channel. Otherwise the parameters of CH0 are applied to all channels.

The gain is configurable through the GAIN bit of the Channel Gain Register (ADC_CGR) as shown in Table 49-6.

Table 49-6. Gain of the Sample and Hold Unit: GAIN Bits and DIFF Bit.

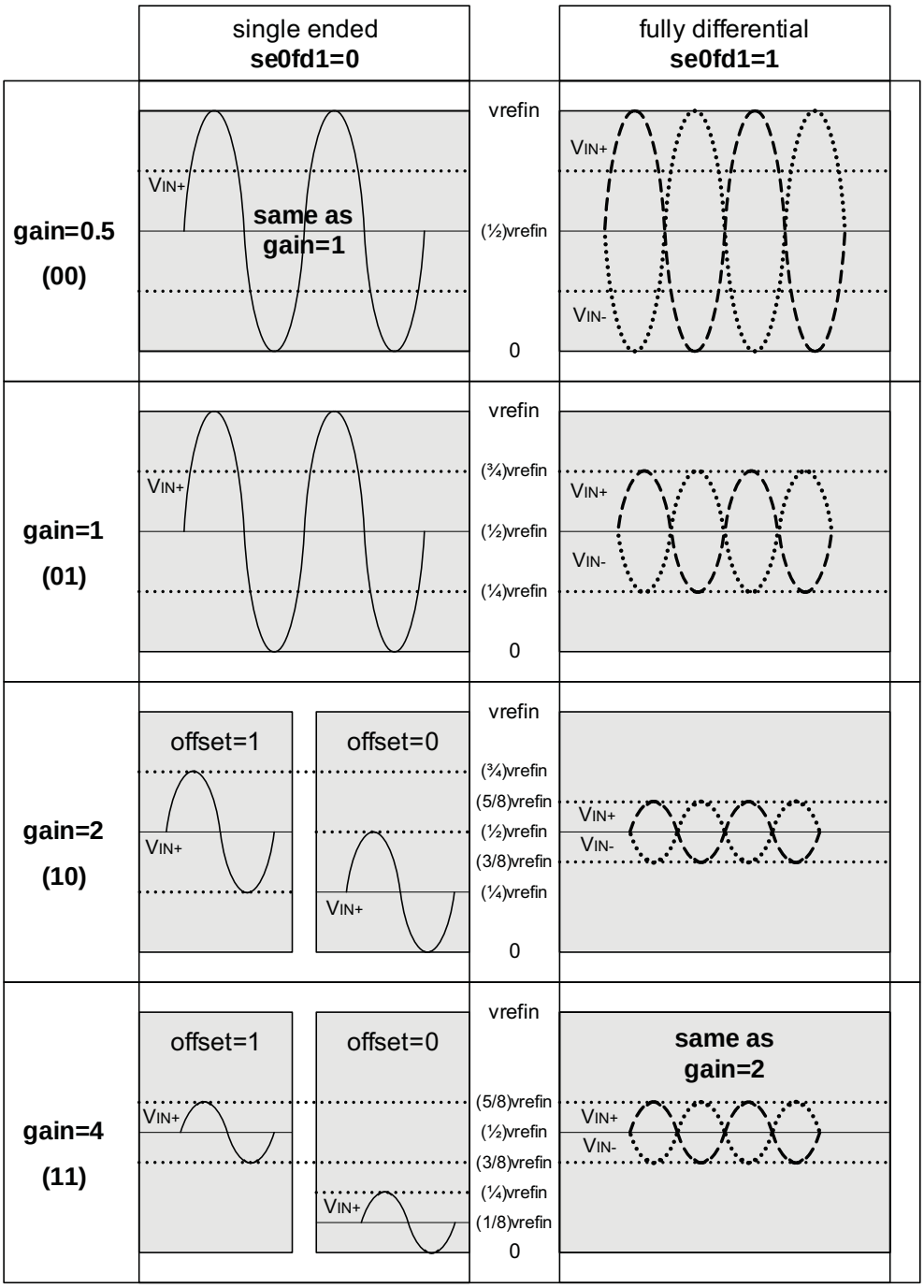
GAIN<0:1>	GAIN (DIFF = 0)	GAIN (DIFF = 1)
00	1	0.5
01	1	1
10	2	2
11	4	2

To allow full range, analog offset of the ADC can be configured by the OFFSET bit of the Channel Offset Register (ADC_COR). The Offset is only available in Single Ended Mode.

Table 49-7. Offset of the Sample and Hold Unit: OFFSET DIFF and Gain (G)

OFFSET Bit	OFFSET (DIFF = 0)	OFFSET (DIFF = 1)
0	0	0
1	(G-1)Vrefin/2	

Figure 49-7. Analog Full Scale Ranges in Single Ended/Differential Applications Versus Gain and Offset



49.6.10 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.

When the gain, offset or differential input parameters of the analog cell change between two channels, the analog cell may need a specific settling time before starting the tracking phase. In that case, the controller automatically waits during the settling time defined in the "ADC Mode Register". Obviously, if the ANACH option is not set, this time is unused.

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.

49.6.11 Automatic Calibration

The ADC features an automatic calibration (AUTOCALIB) mode for gain errors (calibration).

The automatic calibration sequence can be started at any time writing to '1' the AUTOCAL bit of the ADC Control Register. The automatic calibration sequence requires a software reset command (SWRST in the ADC_CR register) prior to write AUTOCAL bit. The end of calibration sequence is given by the EOCAL bit in the interrupt status register (ADC_ISR), and an interrupt is generated if EOCAL interrupt has been enabled (ADC_IER register).

The calibration sequence will perform an automatic calibration on all enabled channels. The channels required for conversion do not need to be all enabled during the calibration process if they are programmed with the same gain. Only channels with different gain settings need to be enabled. The gain settings of all enabled channels must be set before starting the AUTOCALIB sequence. If the gain settings (ADC_CGR and ADC_COR registers) for a given channel are changed, the AUTOCALIB sequence must then be started again.

The calibration data (on one or more enabled channels) is stored in the internal ADC memory.

Then, when a new conversion is started (on one or more enabled channels), the converted value (in ADC_LCDR or ADC_CDRx registers) is a calibrated value.

Autocalibration is for settings, not for channels. Therefore, if a specific combination of gain has been already calibrated, and a new channel with the same settings is enabled after the initial calibration, there is no need to restart a calibration. If different enabled channels have different gain settings, the corresponding channels must be enabled before starting the calibration.

If a software reset is performed (SWRST bit in ADC_CR) or after power up (or wake-up from Backup mode), the calibration data in the ADC memory is lost.

Changing the ADC running mode (in ADC_CR register) does not affect the calibration data.

Changing the ADC reference voltage (ADVREF pin) requires a new calibration sequence.

For calibration time, gain error after calibration, refer to the 12-bit ADC electrical characteristics section of the product.

49.7 Touchscreen

49.7.1 Touchscreen Mode

The TSMODE parameter of "ADC Touchscreen Mode Register" is used to enable/disable the Touchscreen functionality, to select the type of screen (4-wire or 5-wire) and, in the case of a 4-wire screen, to activate (or not) the pressure measurement.

In 4-wire mode, channel 0, 1, 2 and 3 must not be used for classic ADC conversions. Likewise, in 5-wire mode, channel 0, 1, 2, 3, and 4 must not be used for classic ADC conversions.

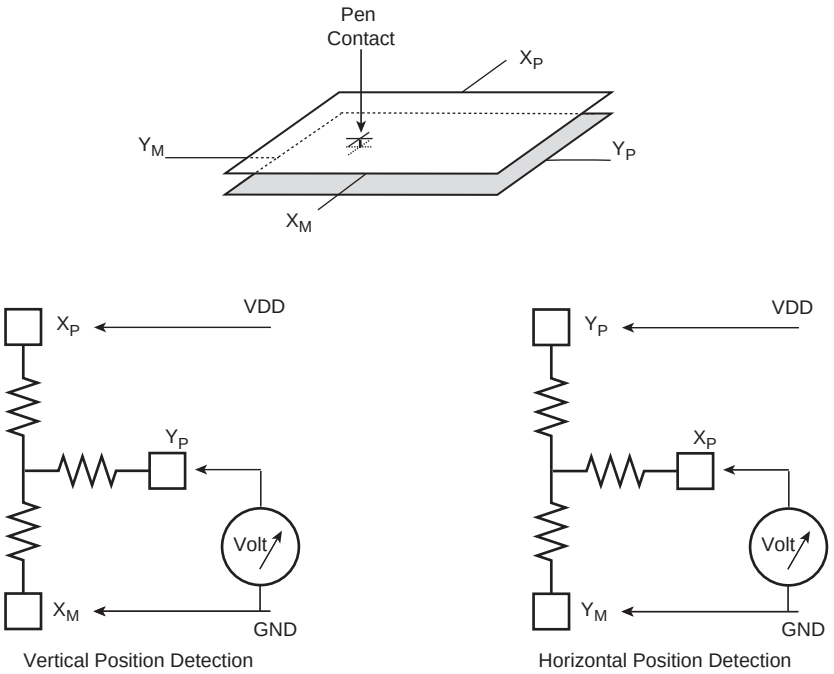
49.7.2 4-wire Resistive Touchscreen Principles

A resistive touchscreen is based on two resistive films, each one being fitted with a pair of electrodes, placed at the top and bottom on one film, and on the right and left on the other. In between, there is a layer acting as an insulator, but also enables contact when you press the screen. This is illustrated in Figure 49-8.

The TSADC controller has the ability to perform without external components:

- Position Measurement
- Pressure Measurement
- Pen Detection

Figure 49-8. Touchscreen Position Measurement



49.7.3 4-wire Position Measurement Method

As shown in Figure 49-8, to detect the position of a contact, a supply is first applied from top to bottom. Due to the linear resistance of the film, there is a voltage gradient from top to bottom. When a contact is performed on the screen, the voltage propagates at the point the two surfaces come into contact with the second film. If the input impedance on the right and left electrodes sense is high enough, the film does not affect this voltage, despite its resistive nature.

For the horizontal direction, the same method is used, but by applying supply from left to right. The range depends on the supply voltage and on the loss in the switches that connect to the top and bottom electrodes.

In an ideal world (linear, with no loss through switches), the horizontal position is equal to:

VY_M / VDD or VY_P / VDD.

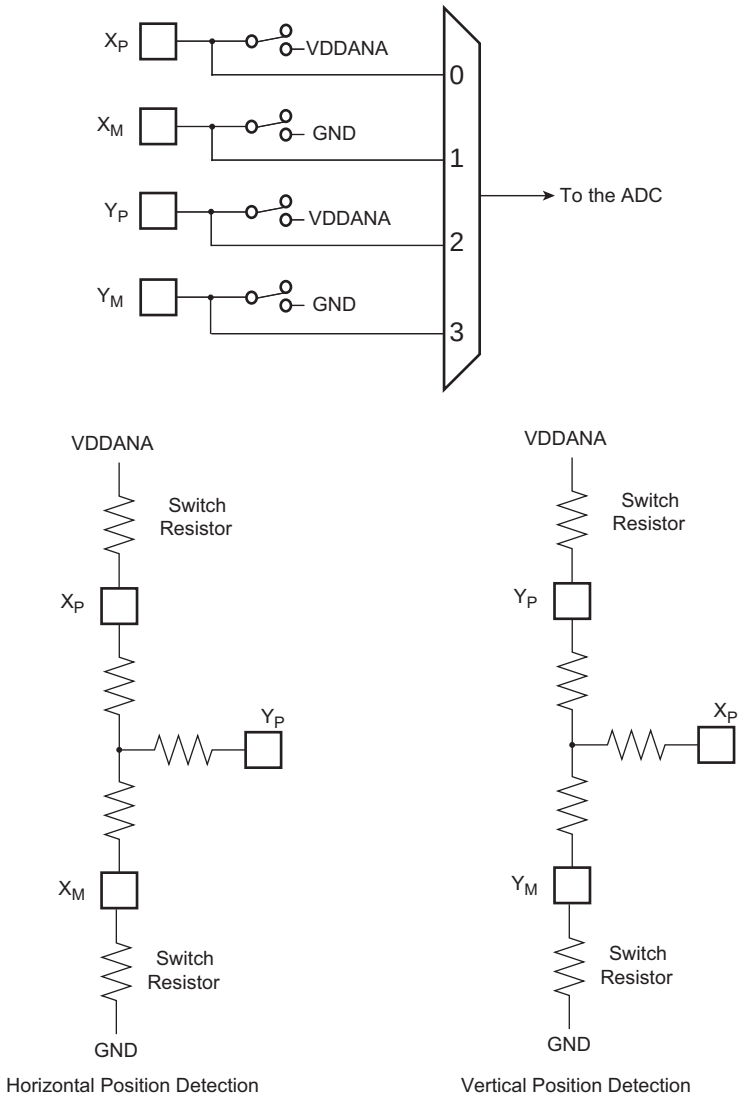
The implementation with on-chip power switches is shown in Figure 49-9. The voltage measurement at the output of the switch compensates for the switches loss.

It is possible to correct for switch loss by performing the operation:

$[VY_P - VX_M] / [VX_P - VX_M]$.

This requires additional measurements, as shown in Figure 49-9.

Figure 49-9. Touchscreen Switches Implementation



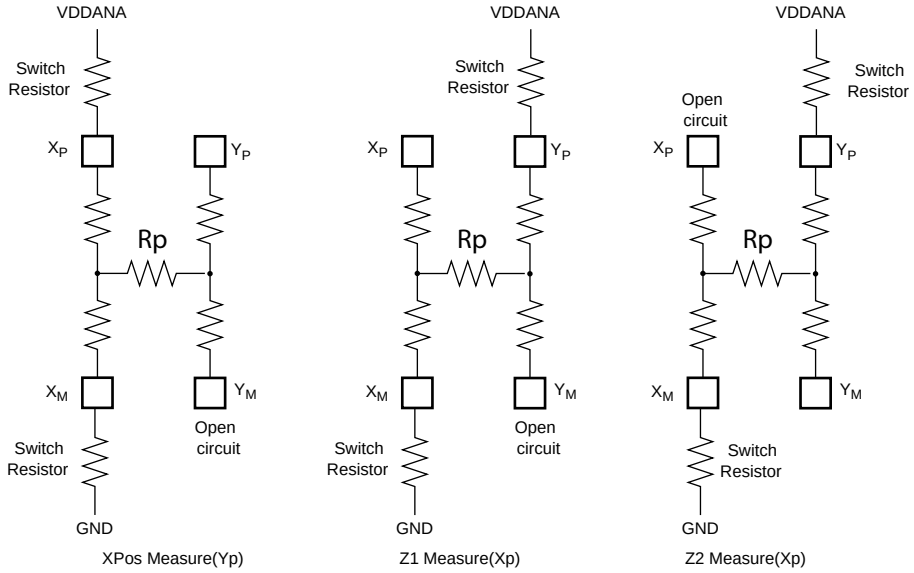
49.7.4 4-wire Pressure Measurement Method

The method to measure the pressure (Rp) applied to the touchscreen is based on the known resistance of the X-Panel resistance (Rxp).

Three conversions (Xpos,Z1,Z2) are necessary to determine the value of Rp (Zaxis resistance).

Rp = Rxp*(Xpos/1024)*[(Z2/Z1)-1]

Figure 49-10.Pressure Measurement



49.7.5 5-wire Resistive Touchscreen Principles

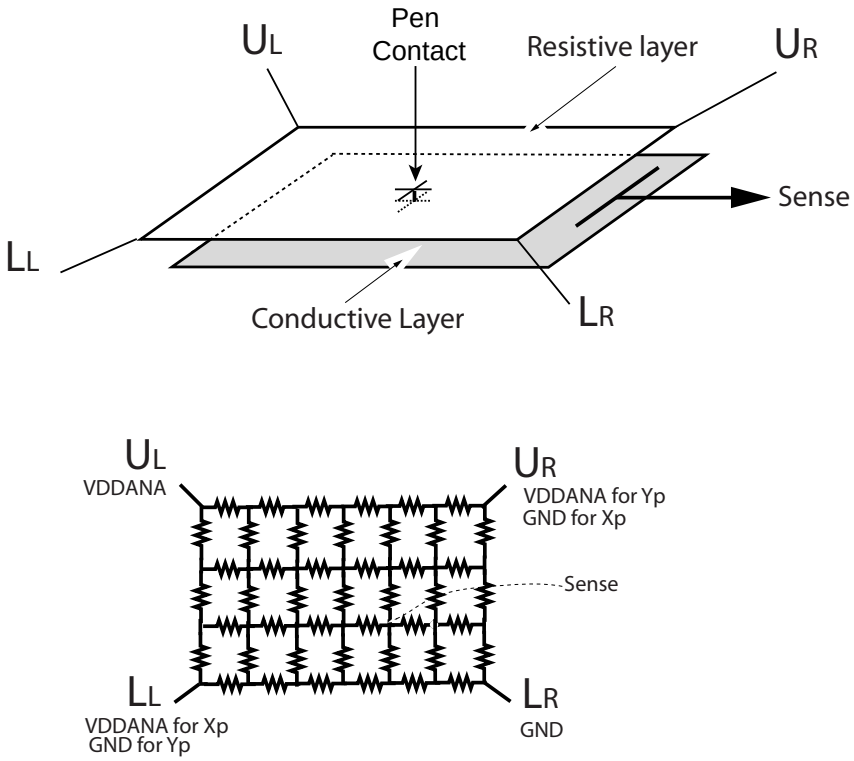
To make a 5-wire touchscreen, a resistive layer with a contact point at each corner and a conductive layer are used.

The 5-wire touchscreen differs from the 4-wire type mainly in that the voltage gradient is applied only to one layer, the resistive layer, while the other layer is the sense layer for both measurements.

The measurement of the X position is obtained by biasing the upper left corner and lower left corner to V_{DDANA} and the upper right corner and lower right to ground.

To measure along the Y axis, bias the upper left corner and upper right corner to V_{DDANA} and bias the lower left corner and lower right corner to ground.

Figure 49-11.5-Wire principle



49.7.6 5-wire Position Measurement Method

In an application only monitoring clicks, 100 points per second is typically needed. For handwriting or motion detection, the number of measurements to consider is approximately 200 points per second. This must take into account that multiple measurements are included (over sampling, filtering) to compute the correct point.

The 5-wire touchscreen panel works by applying a voltage at the corners of the resistive layer and measuring the vertical or horizontal resistive network with the sense input. The ADC converts the voltage measured at the point the panel is touched.

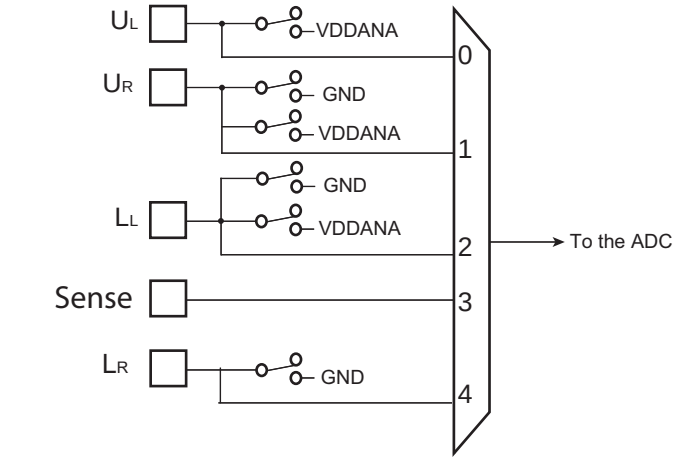
A measurement of the Y position of the pointing device is made by:

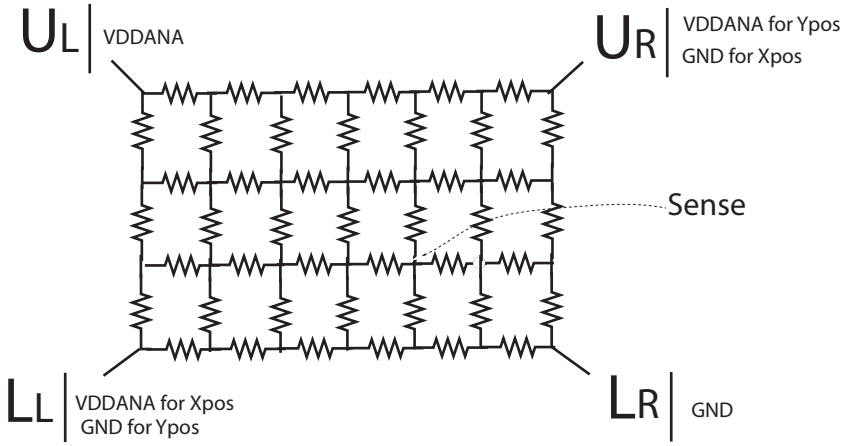
- Connecting Upper left (UL) and upper right (UR) corners to VDDANA
- Connecting Lower left (LL) and lower right (LR) corners to ground.
- The voltage measured is determined by the voltage divider developed at the point of touch (Yposition) and the SENSE input is converted by ADC.

A measurement of the X position of the pointing device is made by:

- Connecting the upper left (UL) and lower left (LL) corners to ground
- Connecting the upper right and lower right corners to VDDANA.
- The voltage measured is determined by the voltage divider developed at the point of touch (Xposition) and the SENSE input is converted by ADC.

Figure 49-12.Touchscreen Switches Implementation



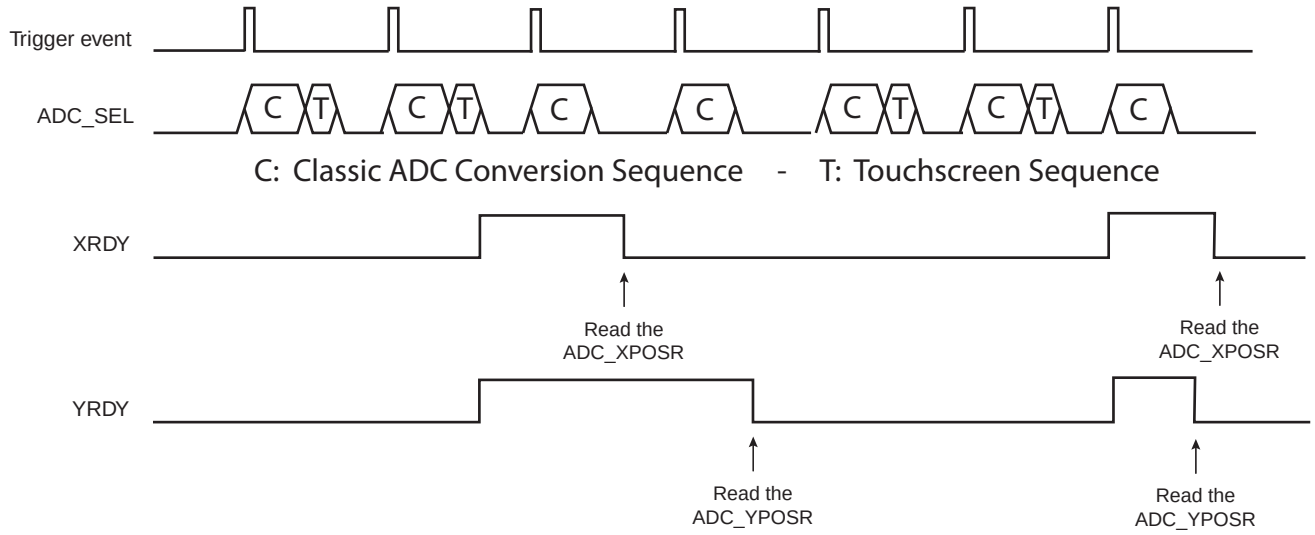


49.7.7 Sequence and Noise Filtering

The ADC Controller can manage ADC conversions and Touchscreen measurement. On each trigger event the sequence of ADC conversions is performed as described in Section 49.6.6 ”Sleep Mode and Conversion Sequencer”. The Touchscreen measure frequency can be specified in number of trigger events by writing the TSFREQ parameter in the “ADC Touchscreen Mode Register”. An internal counter counts triggers up to TSFREQ, and every time it rolls out, a Touchscreen sequence is appended to the classic ADC conversion sequence (see Figure 49-13).

Additionally the user can average multiple Touchscreen measures by writing the TSAV parameter in the “ADC Touchscreen Mode Register”. This can be 1, 2, 4 or 8 measures performed on consecutive triggers as illustrated in Figure 49-13 bellow. Consequently, the TSFREQ parameter must be greater or equal to the TSAV parameter.

Figure 49-13.Insertion of Touchscreen sequences (TSFREQ = 2; TSAV = 1)



49.7.8 Measured Values, Registers and Flags

As soon as the controller finishes the Touchscreen sequence, XRDY, YRDY and PRDY are set and can generate an interrupt. These flags can be read in the “ADC Interrupt Status Register”. They are reset independently by reading in ADC_XPOSR, ADC_YPOSR and ADC_PRESSR. for classic ADC conversions.

The “ADC Touchscreen X Position Register” presents XPOS (V_X - V_{Xmin}) on its LSB and XSCALE (V_{XMAX} - V_{Xmin}) aligned on the 16th bit.

The “ADC Touchscreen Y Position Register” presents YPOS (V_Y - V_{Ymin}) on its LSB and YSCALE (V_{YMAX} - V_{Ymin}) aligned on the 16th bit.

To improve the quality of the measure, the user must calculate: XPOS/XSCALE and YPOS/YSCALE.

V_{XMAX}, V_{Xmin}, V_{YMAX}, and V_{Ymin} are measured at the first start up of the controller. These values can change during use, so it can be necessary to refresh them. Refresh can be done by writing ‘1’ in the CALIB field of the control register (ADC_CR).

The “ADC Touchscreen Pressure Register” presents Z1 on its LSB and Z2 aligned on the 16th bit. See Section 49.7.4 to know how use them.

49.7.9 Pen Detect Method

When there is no contact, it is not necessary to perform a conversion. However, it is important to detect a contact by keeping the power consumption as low as possible.

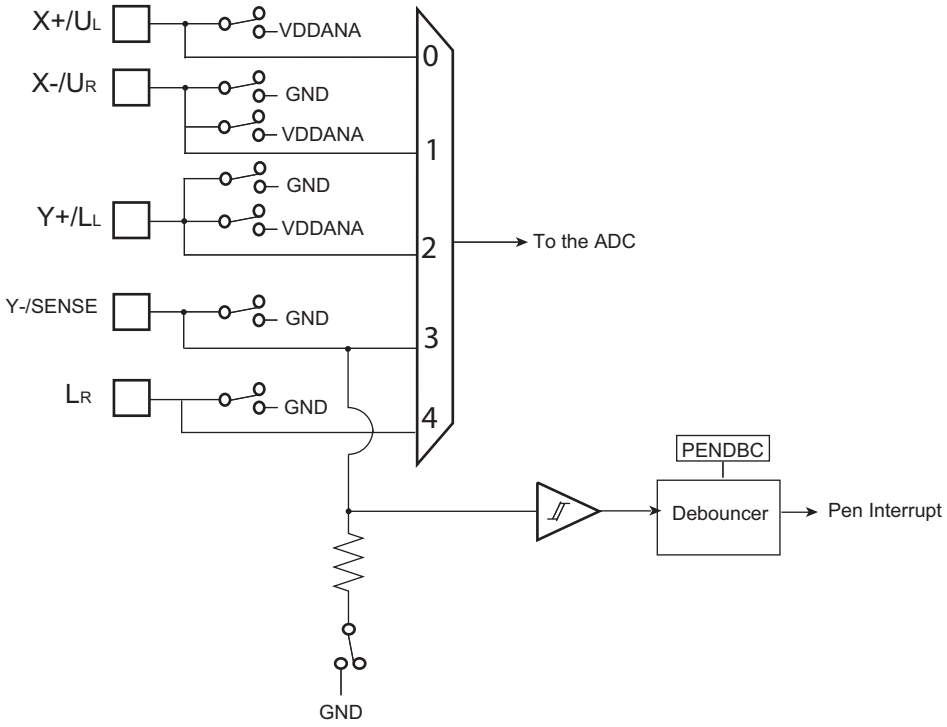
The implementation polarizes one panel by closing the switch on (X_P/U_L) and ties the horizontal panel by an embedded resistor connected to Y_M / Sense. This resistor is enabled by a fifth switch. Since there is no contact, no current is flowing and there is no related power consumption. As soon as a contact occurs, a current is flowing in the Touchscreen and a Schmitt trigger detects the voltage in the resistor.

The Touchscreen Interrupt configuration is entered by programming the PENDET bit in the “ADC Touchscreen Mode Register”. If this bit is written at 1, the controller samples the pen contact state when it is not converting and waiting for a trigger.

To complete the circuit, a programmable debouncer is placed at the output of the Schmitt trigger. This debouncer is programmable up to 2^{15} ADC clock periods. The debouncer length can be selected by programming the field PENDBC in “ADC Touchscreen Mode Register”.

Due to the analog switch’s structure, the debouncer circuitry is only active when no conversion (Touchscreen or classic ADC channels) is in progress. Thus, if the time between the end of a conversion sequence and the arrival of the next trigger event is lower than the debouncing time configured on PENDBC, the debouncer will not detect any contact.

Figure 49-14.Touchscreen Pen Detect



The Touchscreen Pen Detect can be used to generate an ADC interrupt to wake up the system. The Pen Detect generates two types of status, reported in the “ADC Interrupt Status Register”:

- the PEN bit is set as soon as a contact exceeds the debouncing time as defined by PENDBC and remains set until ADC_SR is read.
- the NOPEN bit is set as soon as no current flows for a time over the debouncing time as defined by PENDBC and remains set until ADC_SR is read.

Both bits are automatically cleared as soon as the Status Register (ADC_SR) is read, and can generate an interrupt by writing the “ADC Interrupt Enable Register”.

Moreover, the rising of either one of them clears the other, they cannot be set at the same time.

The PENS bit of the ADC_SR indicates the current status of the pen contact.

49.7.10 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, ADC_TSMR) the structure differs. Each data read 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.

As soon as touchscreen conversions are required, the pen detection function may help the post-processing of the buffer. To get more details refer to Section 49.7.10.4 ”Pen Detection Status”.

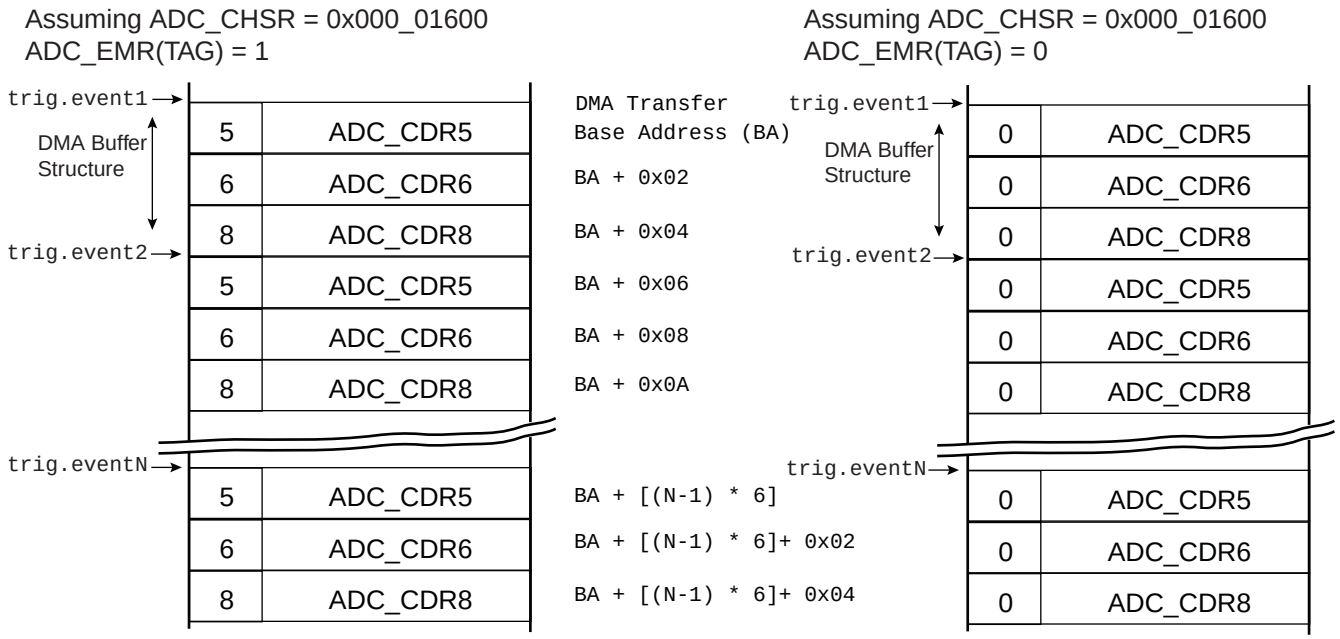
49.7.10.1 Classical ADC Channels Only

When no touchscreen conversion is required (i.e. TSMODE = 0 in ADC_TSMR register), the structure of data within the buffer is defined by the ADC_MR, ADC_CHSR, ADC_SEQR1, ADC_SEQR2 registers.

If the user sequence is not used (i.e. USEQ is cleared in ADC_MR register) then only the value of ADC_CHSR register defines the data structure. For each trigger event, enabled channels will be consecutively stored in ADC_LCDR register and automatically read to the buffer.

When the user sequence is configured (i.e. USEQ is set in ADC_MR register) not only does ADC_CHSR register modify the data structure of the buffer, but ADC_SEQR1, ADC_SEQR2 registers may modify the data structure of the buffer as well.

Figure 49-15. Buffer Structure when TSMODE = 0



49.7.10.2 Touchscreen Channels Only

When only touchscreen conversions are required (i.e. TSMODE differs from 0 in ADC_TSMR register and ADC_CHSR equals 0), the structure of data within the buffer is defined by the ADC_TSMR register.

When TSMODE = 1 or 3, each trigger event adds 2 half-words in the buffer (assuming TSAV = 0), first half-word being XPOS of ADC_XPOSR register then YPOS of ADC_YPOSR register. If TSAV/TSFREQ differs from 0, the data structure remains unchanged. Not all trigger events add data to the buffer.

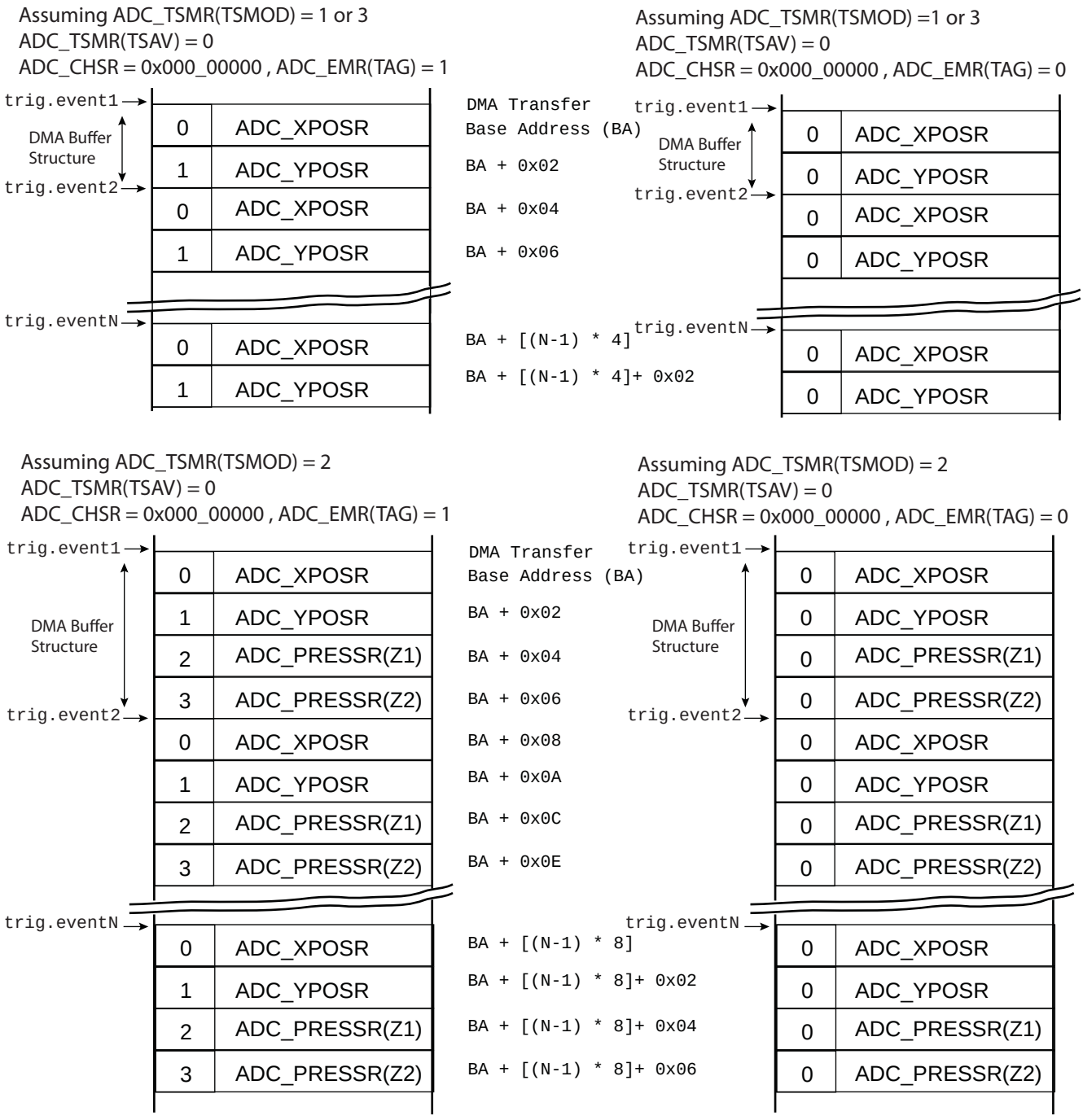
When TSMODE = 2, each trigger event adds 4 half-words to the buffer (assuming TSAV=0), first half-word being XPOS of ADC_XPOSR register followed by YPOS of ADC_YPOSR register and finally Z1 followed by Z2, both located in ADC_PRESSR register.

When TAG is set (ADC_EMR), the CHNB field (4 most significant bit of the ADC_LCDR) register is set to 0 when XPOS is transmitted and set to 1 when YPOS is transmitted, allowing an easier post-processing of the buffer or better checking buffer integrity. In case 4-wire with pressure mode is selected, Z1 value is transmitted to the buffer along with tag set to 2 and Z2 is tagged with value 3.

XSCALE and YSCALE (calibration values) are not transmitted to the buffer because they are supposed to be constant and moreover only measured at the very first start up of the controller or upon user request.

There is no change in buffer structure whatever the value of PENDET bit configuration in ADC_TSMR register but it is recommended to use the pen detection function for buffer post-processing (refer to “Pen Detection Status” on page 1641).

Figure 49-16. Buffer Structure when only touchscreen channels are enabled



49.7.10.3 Interleaved Channels

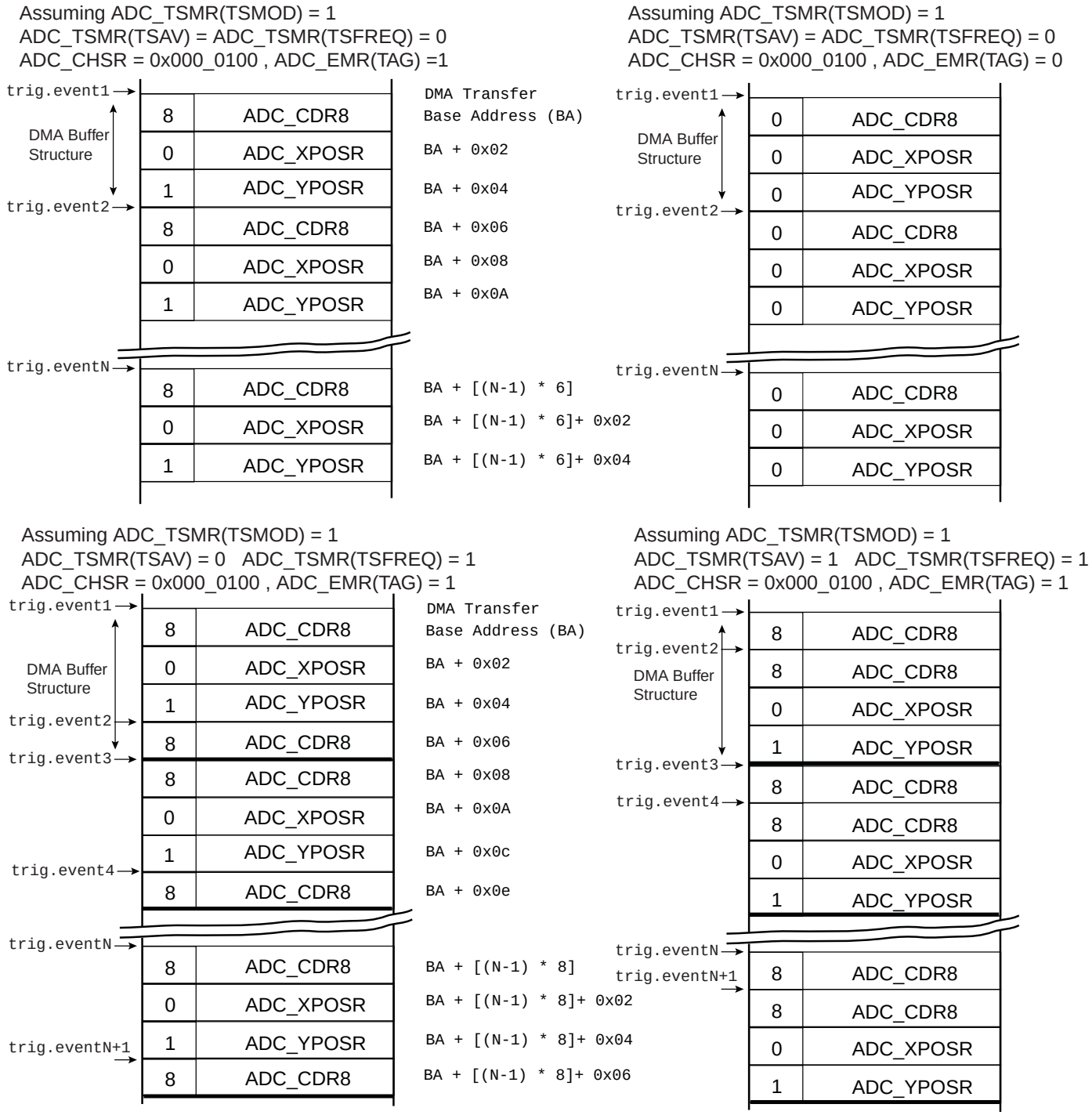
When both classic ADC channels (CH4/CH5 up to CH12 are set in ADC_CHSR) and touchscreen conversions are required (TSMODE differs from 0 in ADC_TSMR register) the structure of the buffer differs according to TSAV and TSFREQ values.

If TSFREQ differs from 0, not all events generate touchscreen conversions, therefore buffer structure is based on 2^{TSFREQ} trigger events. Given a TSFREQ value, the location of touchscreen conversion results depends on TSAV value.

When TSFREQ = 0, TSAV must equal 0.

There is no change in buffer structure whatever the value of PENDET bit configuration in ADC_TSMR register but it is recommended to use the pen detection function for buffer post-processing (refer to “Pen Detection Status” on page 1641).

Figure 49-17. Buffer Structure when classic ADC and touchscreen channels are interleaved



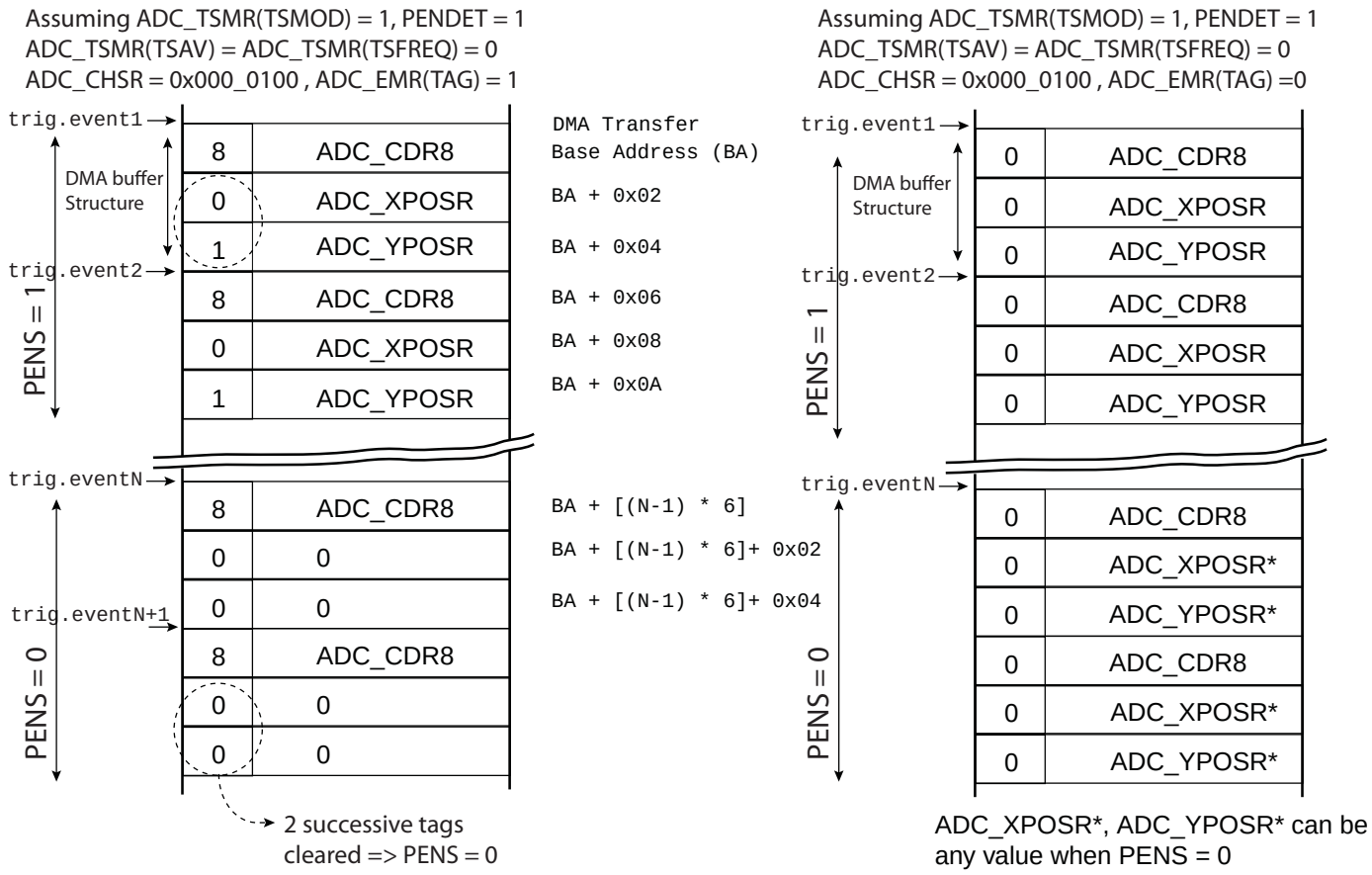
49.7.10.4 Pen Detection Status

If the pen detection measure is enabled (PENDET is set in ADC_TSMR register), the XPOS, YPOS, Z1, Z2 values transmitted to the buffer through ADC_LCDR register are cleared (including the CHNB field), if the PENS flag of ADC_ISR register is 0. When the PENS flag is set, XPOS, YPOS, Z1, Z2 are normally transmitted.

Therefore, using pen detection together with tag function eases the post-processing of the buffer, especially to determine which touchscreen converted values correspond to a period of time when the pen was in contact with the screen.

When the pen detection is disabled or the tag function is disabled, XPOS, YPOS, Z1, Z2 are normally transmitted without tag and no relationship can be found with pen status, thus post-processing may not be easy.

Figure 49-18. Buffer Structure with and without pen detection enabled



49.7.11 Fault Output

The ADC Controller internal fault output is directly connected to PWM fault input. Fault output may be asserted according to the configuration of ADC_EMR (Extended Mode Register) and ADC_CWR (Compare Window Register) and converted values. When the Compare occurs, the ADC fault output generates a pulse of one Master Clock Cycle to the PWM fault input. This fault line can be enabled or disabled within PWM. Should it be activated and asserted by the ADC Controller, the PWM outputs are immediately placed in a safe state (pure combinational path). Note that the ADC fault output connected to the PWM is not the COMPE bit. Thus the Fault Mode (FMOD) within the PWM configuration must be FMOD = 1.

49.7.12 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 1646
- "ADC Channel Sequence 1 Register" on page 1649
- "ADC Channel Sequence 2 Register" on page 1650
- "ADC Channel Enable Register" on page 1651
- "ADC Channel Disable Register" on page 1652
- "ADC Extended Mode Register" on page 1661
- "ADC Compare Window Register" on page 1662
- "ADC Channel Gain Register" on page 1663
- "ADC Channel Offset Register" on page 1664
- "ADC Analog Control Register" on page 1666
- "ADC Touchscreen Mode Register" on page 1667
- "ADC Trigger Register" on page 1672

49.8 Analog-to-Digital Converter (ADC) User Interface

Table 49-8. 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
0x48	Channel Gain Register	ADC_CGR	Read-write	0x00000000
0x4C	Channel Offset Register	ADC_COR	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	–	–	–
0x94	Analog Control Register	ADC_ACR	Read-write	0x00000100
0x98 - 0xAC	Reserved	–	–	–
0xB0	Touchscreen Mode Register	ADC_TSMR	Read-write	0x00000000
0xB4	Touchscreen X Position Register	ADC_XPOSR	Read-only	0x00000000
0xB8	Touchscreen Y Position Register	ADC_YPOSR	Read-only	0x00000000
0xBC	Touchscreen Pressure Register	ADC_PRESSR	Read-only	0x00000000
0xC0	Trigger Register	ADC_TRGR	Read-write	0x00000000
0xC4 - 0xE0	Reserved	–	–	–
0xE4	Write Protect Mode Register	ADC_WPMR	Read-write	0x00000000

Table 49-8. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0xE8	Write Protect Status Register	ADC_WPSR	Read-only	0x00000000
0xEC - 0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

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

49.8.1 ADC Control Register

Name: ADC_CR

Address: 0xF8018000

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
–	–	–	–	AUTOCAL	TSCALIB	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.

- **TSCALIB: Touchscreen Calibration**

0: No effect.

1: Programs screen calibration (VDD/GND measurement)

The calibration sequence is performed during the next sequence when command is launched during an already started conversion sequence, or at the start of the second conversion sequence located after the TSCALIB command, if it is launched when no conversion is in progress (sleep mode, waiting a trigger event).

TSCALIB measurement sequence does not affect the last data converted register (ADC_LDCR).

- **AUTOCAL: Automatic Calibration of ADC**

0: No effect.

1: Launches an automatic calibration of the ADC cell on the next sequence.

49.8.2 ADC Mode Register

Name: ADC_MR

Address: 0xF8018004

Access: Read-write

31	30	29	28	27	26	25	24
USEQ	–	TRANSFER		TRACKTIM			
23	22	21	20	19	18	17	16
ANACH	–	SETTLING		STARTUP			
15	14	13	12	11	10	9	8
PRESCAL							
7	6	5	4	3	2	1	0
–	FWUP	SLEEP	–	TRGSEL			–

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 1673.

- **TRGSEL: Trigger Selection**

Value	Name	Description
0	ADC_TRIG0	ADTRG
1	ADC_TRIG1	TIOA0
2	ADC_TRIG2	TIOA1
3	ADC_TRIG3	TIOA2
4	ADC_TRIG4	PWM event line 0
5	ADC_TRIG5	PWM_even line 1
6	ADC_TRIG6	Reserved
7	–	Reserved

Note: The trigger selection can be performed only if TRGMOD=1,2 or 3 in “ADC Trigger Register”.

- **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 wake-up time can be modified by programming FWUP bit

- **FWUP: Fast Wake Up**

Value	Name	Description
0	OFF	If SLEEP is 1 then both ADC Core and reference voltage circuitry are OFF between conversions
1	ON	If SLEEP is 1 then 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
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

- **SETTLING: Analog Settling Time**

Value	Name	Description
0	AST3	3 periods of ADCClock
1	AST5	5 periods of ADCClock
2	AST9	9 periods of ADCClock
3	AST17	17 periods of ADCClock

- **ANACH: Analog Change**

Value	Name	Description
0	NONE	No analog change on channel switching: DIFF0, GAIN0 and OFF0 are used for all channels
1	ALLOWED	Allows different analog settings for each channel. See ADC_CGR and ADC_COR Registers

- **TRACKTIM: Tracking Time**

  Tracking Time = (TRACKTIM + 1) * ADCClock periods.

- **TRANSFER: Transfer Period**

  This field must be programmed with value 2.

- **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.

49.8.3 ADC Channel Sequence 1 Register

Name: ADC_SEQR1

Address: 0xF8018008

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” .

- **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.

49.8.4 ADC Channel Sequence 2 Register

Name: ADC_SEQR2

Address: 0xF801800C

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
–				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” .

- **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.

49.8.5 ADC Channel Enable Register

Name: ADC_CHER

Address: 0xF8018010

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” .

- **CHx: Channel x Enable**

0: No effect.

1: Enables the corresponding channel.

Note: If USEQ = 1 in the ADC_MR register, CHx corresponds to the xth channel of the sequence described in ADC_SEQR1 and ADC_SEQR2.

49.8.6 ADC Channel Disable Register

Name: ADC_CHDR

Address: 0xF8018014

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” .

- **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.

49.8.7 ADC Channel Status Register

Name: ADC_CHSR

Address: 0xF8018018

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: The corresponding channel is disabled.

1: The corresponding channel is enabled.

49.8.8 ADC Last Converted Data Register

Name: ADC_LCDR

Address: 0xF8018020

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.

49.8.9 ADC Interrupt Enable Register

Name: ADC_IER

Address: 0xF8018024

Access: Write-only

31	30	29	28	27	26	25	24
–	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
EOCAL	PRDY	YRDY	XRDY	–	–	–	–
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**
- **XRDY: Touchscreen Measure XPOS Ready Interrupt Enable**
- **YRDY: Touchscreen Measure YPOS Ready Interrupt Enable**
- **PRDY: Touchscreen Measure Pressure Ready Interrupt Enable**
- **EOCAL: End of Calibration Sequence**
- **DRDY: Data Ready Interrupt Enable**
- **GOVRE: General Overrun Error Interrupt Enable**
- **COMPE: Comparison Event Interrupt Enable**
- **PEN: Pen Contact Interrupt Enable**
- **NOPEN: No Pen Contact Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

49.8.10 ADC Interrupt Disable Register

Name: ADC_IDR

Address: 0xF8018028

Access: Write-only

31	30	29	28	27	26	25	24
–	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
EOCAL	PRDY	YRDY	XRDY	–	–	–	–
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**
- **XRDY: Touchscreen Measure XPOS Ready Interrupt Disable**
- **YRDY: Touchscreen Measure YPOS Ready Interrupt Disable**
- **PRDY: Touchscreen Measure Pressure Ready Interrupt Disable**
- **EOCAL: End of Calibration Sequence**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **COMPE: Comparison Event Interrupt Disable**
- **PEN: Pen Contact Interrupt Disable**
- **NOPEN: No Pen Contact Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

49.8.11 ADC Interrupt Mask Register

Name: ADC_IMR

Address: 0xF801802C

Access: Read-only

31	30	29	28	27	26	25	24
–	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
EOCAL	PRDY	YRDY	XRDY	–	–	–	–
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**
- **XRDY: Touchscreen Measure XPOS Ready Interrupt Mask**
- **YRDY: Touchscreen Measure YPOS Ready Interrupt Mask**
- **PRDY: Touchscreen Measure Pressure Ready Interrupt Mask**
- **EOCAL: End of Calibration Sequence**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **COMPE: Comparison Event Interrupt Mask**
- **PEN: Pen Contact Interrupt Mask**
- **NOPEN: No Pen Contact Interrupt Mask**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

49.8.12 ADC Interrupt Status Register

Name: ADC_ISR

Address: 0xF8018030

Access: Read-only

31	30	29	28	27	26	25	24
PENS	NOPEN	PEN	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
EOCAL	PRDY	YRDY	XRDY	–	–	–	–
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: The corresponding analog channel is disabled, or the conversion is not finished. This flag is cleared when reading the corresponding ADC_CDRx registers.

1: The corresponding analog channel is enabled and conversion is complete.

- **XRDY: Touchscreen XPOS Measure Ready**

0: No Measure has been performed since the last read of ADC_XPOSR.

1: At least one Measure has been performed since the last read of ADC_ISR.

- **YRDY: Touchscreen YPOS Measure Ready**

0: No Measure has been performed since the last read of ADC_YPOSR.

1: At least one Measure has been performed since the last read of ADC_ISR.

- **PRDY: Touchscreen Pressure Measure Ready**

0: No Measure has been performed since the last read of ADC_PRESSR.

1: At least one Measure has been performed since the last read of ADC_ISR.

- **EOCAL: End of Calibration Sequence**

0: Calibration sequence is on going, or no calibration sequence has been requested.

1: Calibration sequence 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.

- **PEN: Pen contact**

0: No pen contact since the last read of ADC_ISR.

1: At least one pen contact since the last read of ADC_ISR.

- **NOPEN: No Pen contact**

0: No loss of pen contact since the last read of ADC_ISR.

1: At least one loss of pen contact since the last read of ADC_ISR.

- **PENS: Pen detect Status**

0: The pen does not press the screen.

1: The pen presses the screen.

Note: PENS is not a source of interruption.

49.8.13 ADC Overrun Status Register

Name: ADC_OVER

Address: 0xF801803C

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.

49.8.14 ADC Extended Mode Register

Name: ADC_EMR

Address: 0xF8018040

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” .

- **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.

49.8.15 ADC Compare Window Register

Name: ADC_CWR

Address: 0xF8018044

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” .

- **LOWTHRES: Low Threshold**

Low threshold associated to compare settings of the ADC_EMR register.

- **HIGHTHRES: High Threshold**

High threshold associated to compare settings of the ADC_EMR register.

49.8.16 ADC Channel Gain Register

Name: ADC_CGR

Address: 0xF8018048

Access: Read-write

31	30	29	28	27	26	25	24
–		–		–		–	
23	22	21	20	19	18	17	16
GAIN11		GAIN10		GAIN9		GAIN8	
15	14	13	12	11	10	9	8
GAIN7		GAIN6		GAIN5		GAIN4	
7	6	5	4	3	2	1	0
GAIN3		GAIN2		GAIN1		GAIN0	

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

- **GAINx: Gain for Channel x**

Gain applied on input of analog-to-digital converter.

GAINx		Gain applied when DIFFx = 0	Gain applied when DIFFx = 1
0	0	1	0.5
0	1	1	1
1	0	2	2
1	1	4	2

The DIFFx mentioned in this table is described in “ADC Channel Offset Register” .

49.8.17 ADC Channel Offset Register

Name: ADC_COR

Address: 0xF801804C

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	DIFF11	DIFF10	DIFF9	DIFF8
23	22	21	20	19	18	17	16
DIFF7	DIFF6	DIFF5	DIFF4	DIFF3	DIFF2	DIFF1	DIFF0
15	14	13	12	11	10	9	8
–	–	–	–	OFF11	OFF10	OFF9	OFF8
7	6	5	4	3	2	1	0
OFF7	OFF6	OFF5	OFF4	OFF3	OFF2	OFF1	OFF0

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

- **OFFx: Offset for channel x**

0: No Offset.

1: Centers the analog signal on Vrefin/2 before the gain scaling. The Offset applied is: (G-1)Vrefin/2

where G is the gain applied (see the description of “ADC Channel Gain Register”).

- **DIFFx: Differential inputs for channel x**

0: Single Ended Mode.

1: Fully Differential Mode.

49.8.18 ADC Channel Data Register

Name: ADC_CDRx [x=0..11]

Address: 0xF8018050

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.

49.8.19 ADC Analog Control Register

Name: ADC_ACR

Address: 0xF8018094

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
–	–	–	–	–	–	PENDETSENS	

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

- **PENDETSENS: Pen Detection Sensitivity**

Allows to modify the pen detection input pull-up resistor value. (See the product electrical characteristics for further details).

49.8.20 ADC Touchscreen Mode Register

Name: ADC_TSMR

Address: 0xF80180B0

Access: Read-write

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

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

- **TSMODE: Touchscreen Mode**

Value	Name	Description
0	NONE	No Touchscreen
1	4_WIRE_NO_PM	4-wire Touchscreen without pressure measurement
2	4_WIRE	4-wire Touchscreen with pressure measurement
3	5_WIRE	5-wire Touchscreen

When TSMOD equals 01 or 10 (i.e. 4-wire mode), channel 0, 1, 2 and 3 must not be used for classic ADC conversions. When TSMOD equals 11 (i.e. 5-wire mode), channel 0, 1, 2, 3, and 4 must not be used.

- **TSAV: Touchscreen Average**

Value	Name	Description
0	NO_FILTER	No Filtering. Only one ADC conversion per measure
1	AVG2CONV	Averages 2 ADC conversions
2	AVG4CONV	Averages 4 ADC conversions
3	AVG8CONV	Averages 8 ADC conversions

- **TSFREQ: Touchscreen Frequency**

Defines the Touchscreen Frequency compared to the Trigger Frequency.

TSFREQ must be greater or equal to TSAV.

The Touchscreen Frequency is:

Touchscreen Frequency = Trigger Frequency / 2^{TSFREQ}

- **TSSCTIM: Touchscreen Switches Closure Time**

Defines closure time of analog switches necessary to establish the measurement conditions.

The Closure Time is:

Switch Closure Time = (TSSCTIM * 4) ADCClock periods.

- **PENDET: Pen Contact Detection Enable**

0: Pen contact detection disable.

1: Pen contact detection enable.

When PENDET = 1, XPOS, YPOS, Z1, Z2 values of ADC_XPOSR, ADC_YPOSR, ADC_PRESSR registers are automatically cleared when PENS = 0 in ADC_ISR.

- **NOTSDMA: No TouchScreen DMA**

0: XPOS, YPOS, Z1, Z2 are transmitted in ADC_LCDR.

1: XPOS, YPOS, Z1, Z2 are never transmitted in ADC_LCDR, therefore the buffer does not contains touchscreen values.

- **PENDBC: Pen Detect Debouncing Period**

Debouncing period = 2^{PENDBC} ADCClock periods.

49.8.21 ADC Touchscreen X Position Register

Name: ADC_XPOSR

Address: 0xF80180B4

Access: Read-only

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

- **XPOS: X Position**

The Position measured is stored here. if XPOS = 0 or XPOS = XSIZE, the pen is on the border.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR register), XPOS is tied to 0 while there is no detection of contact on the touchscreen (i.e. when PENS bitfield is cleared in ADC_ISR register).

- **XSCALE: Scale of XPOS**

Indicates the max value that XPOS can reach. This value should be close to 2^{12}.

49.8.22 ADC Touchscreen Y Position Register

Name: ADC_YPOSR

Address: 0xF80180B8

Access: Read-only

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

- **YPOS: Y Position**

The Position measured is stored here. if YPOS = 0 or YPOS = YSIZE, the pen is on the border.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR register), YPOS is tied to 0 while there is no detection of contact on the touchscreen (i.e. when PENS bitfield is cleared in ADC_ISR register).

- **YSCALE: Scale of YPOS**

Indicates the max value that YPOS can reach. This value should be close to 2^{12}

49.8.23 ADC Touchscreen Pressure Register

Name: ADC_PRESSR

Address: 0xF80180BC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	Z2			
23	22	21	20	19	18	17	16
Z2							
15	14	13	12	11	10	9	8
–	–	–	–	Z1			
7	6	5	4	3	2	1	0
Z1							

- **Z1: Data of Z1 Measurement**

Data Z1 necessary to calculate pen pressure.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR register), Z1 is tied to 0 while there is no detection of contact on the touchscreen (i.e. when PENS bitfield is cleared in ADC_ISR register).

- **Z2: Data of Z2 Measurement**

Data Z2 necessary to calculate pen pressure.

When pen detection is enabled (PENDET set to ‘1’ in ADC_TSMR register), Z2 is tied to 0 while there is no detection of contact on the touchscreen (i.e. when PENS bitfield is cleared in ADC_ISR register).

Note: These two values are unavailable if TSMODE is not set to 2 in ADC_TSMR register.

49.8.24 ADC Trigger Register

Name: ADC_TRGR

Address: 0xF80180C0

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	PEN_TRIG	Pen Detect Trigger (shall be selected only if PENDET is set and TSAMOD = Touchscreen only mode)
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, ADC_TSMR.

When TRGMOD is set to pen detect trigger (i.e. 100) and averaging is used (i.e. bitfield TSAV differs from 0 in ADC_TSMR register) only 1 measure is performed. Thus, XRDY, YRDY, PRDY, DRDY will not rise on pen contact trigger. To achieve measurement, several triggers must be provided either by software or by setting the TRGMOD on continuous trigger (i.e. 110) until flags rise.

49.8.25 ADC Write Protect Mode Register

Name: ADC_WPMR

Address: 0xF80180E4

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 1646
- “ADC Channel Sequence 1 Register” on page 1649
- “ADC Channel Sequence 2 Register” on page 1650
- “ADC Channel Enable Register” on page 1651
- “ADC Channel Disable Register” on page 1652
- “ADC Extended Mode Register” on page 1661
- “ADC Compare Window Register” on page 1662
- “ADC Channel Gain Register” on page 1663
- “ADC Channel Offset Register” on page 1664
- “ADC Analog Control Register” on page 1666
- “ADC Touchscreen Mode Register” on page 1667
- “ADC Trigger Register” on page 1672

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x414443	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

49.8.26 ADC Write Protect Status Register

Name: ADC_WPSR

Address: 0xF80180E8

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.

50. True Random Number Generator (TRNG)

50.1 Description

The True Random Number Generator (TRNG) passes the American *NIST Special Publication 800-22 and Diehard Random Tests Suites.*

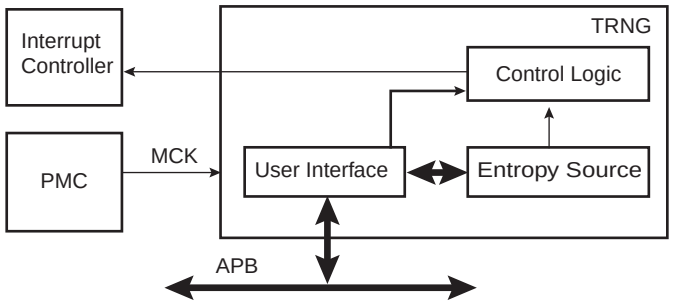
The TRNG may be used as an entropy source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3.

50.2 Embedded Characteristics

- Passed NIST Special Publication 800-22 Tests Suite
- Passed Diehard Random Tests Suite
- May be used as Entropy Source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3
- Provides a 32-bit Random Number Every 84 Clock Cycles

50.3 Block Diagram

Figure 50-1. TRNG Block Diagram



50.4 Product Dependencies

50.4.1 Power Management

The TRNG interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TRNG user interface clock. The user interface clock is independent from any clock that may be used in the entropy source logic circuitry. The source of entropy can be enabled before enabling the user interface clock.

50.4.2 Interrupt

The TRNG interface has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TRNG.

Table 50-1. Peripheral IDs

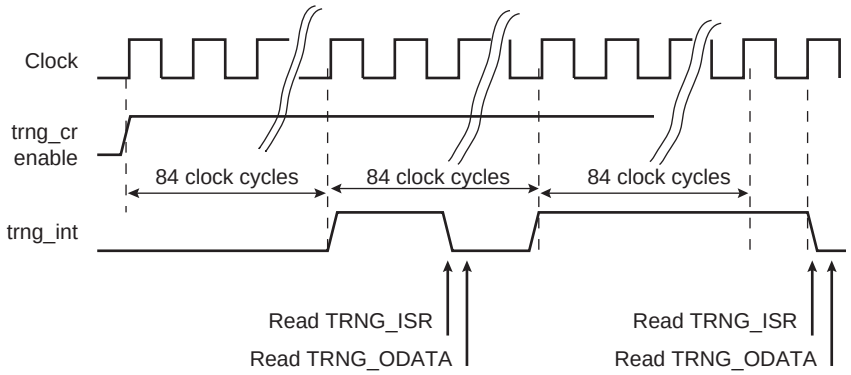
Instance	ID
TRNG	45

50.5 Functional Description

As soon as the TRNG is enabled in the control register (TRNG_CR), the generator provides one 32-bit value every 84 clock cycles. Interrupt trng_int can be enabled in the TRNG_IER (respectively disabled in the TRNG_IDR). This interrupt is set when a new random value is available and is cleared when the status register (TRNG_ISR) is read. The flag DATRDY of the (TRNG_ISR) is set when the random data is ready to be read out on the 32-bit output data register (TRNG_ODATA).

The normal mode of operation checks that the status register flag equals 1 before reading the output data register when a 32-bit random value is required by the software application.

Figure 50-2. TRNG Data Generation Sequence



50.6 True Random Number Generator (TRNG) User Interface

Table 50-2. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TRNG_CR	Write-only	–
0x10	Interrupt Enable Register	TRNG_IER	Write-only	–
0x14	Interrupt Disable Register	TRNG_IDR	Write-only	–
0x18	Interrupt Mask Register	TRNG_IMR	Read-only	0x0000_0000
0x1C	Interrupt Status Register	TRNG_ISR	Read-only	0x0000_0000
0x50	Output Data Register	TRNG_ODATA	Read-only	0x0000_0000
0xFC	Reserved	–	–	–

50.6.1 TRNG Control Register

Name: TRNG_CR

Address: 0xF8040000

Access: Write-only

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

- **ENABLE: Enables the TRNG to provide random values**

0: Disables the TRNG.

1: Enables the TRNG if 0x524E47 (“RNG” in ASCII) is written in KEY field at the same time.

- **KEY: Security Key.**

Value	Name	Description
0x524E47	PASSWD	Writing any other value in this field aborts the write operation.

50.6.2 TRNG Interrupt Enable Register

Name: TRNG_IER

Address: 0xF8040010

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
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

50.6.3 TRNG Interrupt Disable Register

Name: TRNG_IDR

Address: 0xF8040014

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
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

50.6.4 TRNG Interrupt Mask Register

Name: TRNG_IMR

Address: 0xF8040018

Reset: 0x0000_0000

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
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

50.6.5 TRNG Interrupt Status Register

Name: TRNG_ISR

Address: 0xF804001C

Reset: 0x0000_0000

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
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready**

0: Output data is not valid or TRNG is disabled.

1: New Random value is completed.

DATRDY is cleared when this register is read.

50.6.6 TRNG Output Data Register

Name: TRNG_ODATA

Address: 0xF8040050

Reset: 0x0000_0000

Access: Read-only

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

- **ODATA: Output Data**

The 32-bit Output Data register contains the 32-bit random data.

51. Advanced Encryption Standard (AES)

51.1 Description

The Advanced Encryption Standard (AES) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 197* specification.

The AES supports all five confidentiality modes of operation for symmetrical key block cipher algorithms (ECB, CBC, OFB, CFB and CTR), as specified in the *NIST Special Publication 800-38A Recommendation.* It is compatible with all these modes via Peripheral DMA Controller channels, minimizing processor intervention for large buffer transfers.

The 128-bit/192-bit/256-bit key is stored in four/six/eight 32-bit registers (AES_KEYWRx) which are all write-only.

The 128-bit input data and initialization vector (for some modes) are each stored in four 32-bit registers (AES_IDATARx and AES_IVRx) which are all write-only.

As soon as the initialization vector, the input data and the key are configured, the encryption/decryption process may be started. Then the encrypted/decrypted data are ready to be read out on the four 32-bit output data registers (AES_ODATARx) or through the DMA channels.

51.2 Embedded Characteristics

- Compliant with *FIPS Publication 197, Advanced Encryption Standard (AES)*
- 128-bit/192-bit/256-bit Cryptographic Key
- 12/14/16 Clock Cycles Encryption/Decryption Processing Time with a 128-bit/192-bit/256-bit Cryptographic Key
- Double Input Buffer Optimizes Runtime
- Support of the Five Standard Modes of Operation Specified in the *NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation - Methods and Techniques:*
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC) including CBC-MAC
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)
- 8-, 16-, 32-, 64- and 128-bit Data Sizes Possible in CFB Mode
- Last Output Data Mode Allows Optimized Message Authentication Code (MAC) Generation
- Connection to DMA Optimizes Data Transfers for all Operating Modes

51.3 Product Dependencies

51.3.1 Power Management

The AES may be clocked through the Power Management Controller (PMC), so the programmer must first to configure the PMC to enable the AES clock.

51.3.2 Interrupt

The AES interface has an interrupt line connected to the Interrupt Controller.

Handling the AES interrupt requires programming the Interrupt Controller before configuring the AES.

Table 51-1. Peripheral IDs

Instance	ID
AES	43

51.4 Functional Description

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information.

Encryption converts data to an unintelligible form called ciphertext. Decrypting the ciphertext converts the data back into its original form, called plaintext. The CIPHER bit in the AES Mode Register (AES_MR) allows selection between the encryption and the decryption processes.

The AES is capable of using cryptographic keys of 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. This 128-bit/192-bit/256-bit key is defined in the Key Registers (AES_KEYWRx).

The input to the encryption processes of the CBC, CFB, and OFB modes includes, in addition to the plaintext, a 128-bit data block called the initialization vector (IV), which must be set in the Initialization Vector Registers (AES_IVRx). The initialization vector is used in an initial step in the encryption of a message and in the corresponding decryption of the message. The Initialization Vector Registers are also used by the CTR mode to set the counter value.

51.4.1 Operation Modes

The AES supports the following modes of operation:

- ECB: Electronic Code Book
- CBC: Cipher Block Chaining
- OFB: Output Feedback
- CFB: Cipher Feedback
  - CFB8 (CFB where the length of the data segment is 8 bits)
  - CFB16 (CFB where the length of the data segment is 16 bits)
  - CFB32 (CFB where the length of the data segment is 32 bits)
  - CFB64 (CFB where the length of the data segment is 64 bits)
  - CFB128 (CFB where the length of the data segment is 128 bits)
- CTR: Counter

The data pre-processing, post-processing and data chaining for the concerned modes are automatically performed. Refer to the *NIST Special Publication 800-38A Recommendation* for more complete information.

These modes are selected by setting the OPMOD field in the AES_MR.

In CFB mode, five data sizes are possible (8, 16, 32, 64 or 128 bits), configurable by means of the CFBS field in the AES_MR (Section 51.6.2 “AES Mode Register” on page 1693).

In CTR mode, the size of the block counter embedded in the module is 16 bits. Therefore, there is a rollover after processing 1 megabyte of data. If the file to be processed is greater than 1 megabyte, this file must be split into fragments of 1 megabyte. Prior to loading the first fragment into AES_IDATARx, AES_IVRx must be cleared. For any fragment, after the transfer is completed and prior to transferring the next fragment, AES_IVR0 must be programmed so that the fragment number (0 for the first fragment, 1 for the second one, and so on) is written in the 16 MSB of AES_IVR0.

51.4.2 Double Input Buffer

The input data register can be double-buffered to reduce the runtime of large files.

This mode allows writing a new message block when the previous message block is being processed. This is only possible when DMA accesses are performed (SMOD = 0x2).

The DUALBUFF bit in AES_MR must be set to 1 to access the double buffer.

51.4.3 Start Modes

The SMOD field in the AES_MR allows selection of the encryption (or decryption) start mode.

51.4.3.1 Manual Mode

The sequence order is as follows:

- Write the AES_MR with all required fields, including but not limited to SMOD and OPMOD.
- Write the 128-bit/192-bit/256-bit key in the Key Registers (AES_KEYWRx).
- Write the initialization vector (or counter) in the Initialization Vector Registers (AES_IVRx).

Note: The Initialization Vector Registers concern all modes except ECB.

- Set the bit DATRDY (Data Ready) in the AES Interrupt Enable register (AES_IER), depending on whether an interrupt is required or not at the end of processing.
- Write the data to be encrypted/decrypted in the authorized Input Data Registers (See Table 51-2).

Table 51-2. Authorized Input Data Registers

Operation Mode	Input Data Registers to Write
ECB	All
CBC	All
OFB	All
128-bit CFB	All
64-bit CFB	AES_IDATAR0 and AES_IDATAR1
32-bit CFB	AES_IDATAR0
16-bit CFB	AES_IDATAR0
8-bit CFB	AES_IDATAR0
CTR	All

Note: In 64-bit CFB mode, writing to AES_IDATAR2 and AES_IDATAR3 registers is not allowed and may lead to errors in processing.

Note: In 32-, 16- and 8-bit CFB modes, writing to AES_IDATAR1, AES_IDATAR2 and AES_IDATAR3 registers is not allowed and may lead to errors in processing.

- Set the START bit in the AES Control register AES_CR to begin the encryption or the decryption process.
- When processing completes, the DATRDY bit in the AES Interrupt Status Register (AES_ISR) raises. If an interrupt has been enabled by setting the DATRDY bit in AES_IER, the interrupt line of the AES is activated.
- When the software reads one of the Output Data Registers (AES_ODATARx), the DATRDY bit is automatically cleared.

51.4.3.2 Auto Mode

The Auto Mode is similar to the manual one, except that in this mode, as soon as the correct number of Input Data registers is written, processing is automatically started without any action in the Control Register.

51.4.4 DMA Mode

The DMA Controller can be used in association with the AES to perform an encryption/decryption of a buffer without any action by the software during processing.

The SMOD field of the AES_MR must be set to 0x2 and the DMA must be configured with non incremental addresses.

The start address of any transfer descriptor must be set to AES_IDATAR0.

The DMA chunk size configuration depends on the AES mode of operation and is listed in Table 51-3 "DMA Data Transfer Type for the Different Operation Modes".

When writing data to AES with a first DMA channel, data are first fetched from a memory buffer (source data). It is recommended to configure the size of source data to “words” even for CFB modes. On the contrary, the destination data

size depends on the mode of operation. When reading data from the AES with the second DMA channel, the source data is the data read from AES and data destination is the memory buffer. In this case, source data size depends on the AES mode of operation and is listed in Table 51-3.

Table 51-3. DMA Data Transfer Type for the Different Operation Modes

Operation Mode	Chunk Size	Destination/Source Data Transfer Type
ECB	4	Word
CBC	4	Word
OFB	4	Word
CFB 128-bit	4	Word
CFB 64-bit	1	Word
CFB 32-bit	1	Word
CFB 16-bit	1	Half-word
CFB 8-bit	1	Byte
CTR	4	Word

51.4.5 Last Output Data Mode

This mode is used to generate cryptographic checksums on data (MAC) by means of cipher block chaining encryption algorithm (CBC-MAC algorithm for example).

After each end of encryption/decryption, the output data are available either on the output data registers for Manual and Auto mode or at the address specified in the receive buffer pointer for DMA mode (See Table 51-4 "Last Output Data Mode Behavior versus Start Modes").

The Last Output Data bit (LOD) in the AES Mode Register (AES_MR) allows retrieval of only the last data of several encryption/decryption processes.

Therefore, there is no need to define a read buffer in DMA mode.

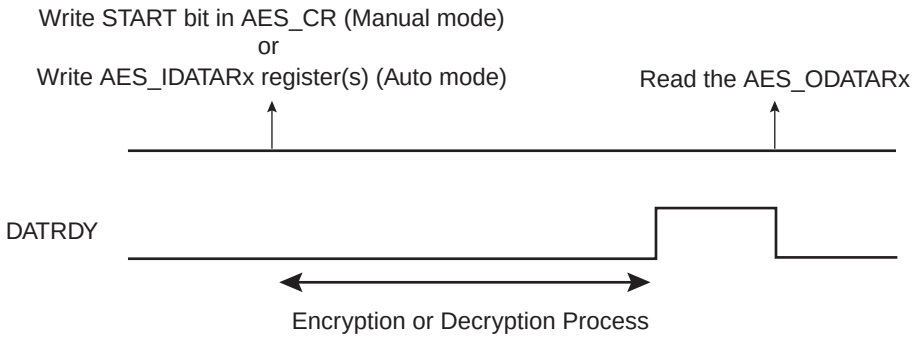
This data are only available on the Output Data Registers (AES_ODATARx).

51.4.6 Manual and Auto Modes

51.4.6.1 If LOD = 0

The DATRDY flag is cleared when at least one of the Output Data Registers is read (See Figure 51-1).

Figure 51-1. Manual and Auto Modes with LOD = 0

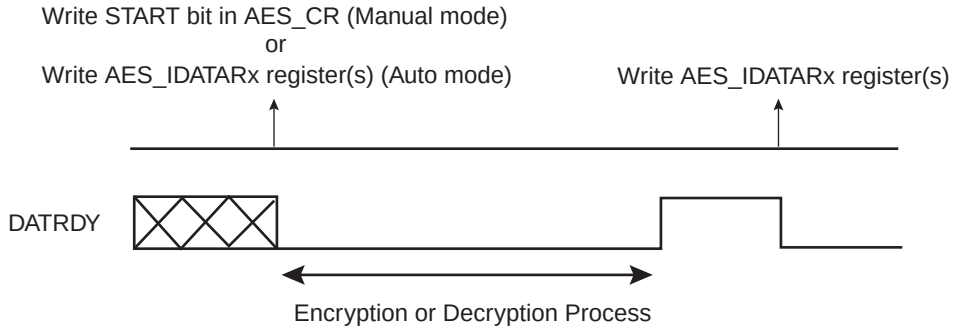


If the user does not want to read the output data registers between each encryption/decryption, the DATRDY flag will not be cleared. If the DATRDY flag is not cleared, the user cannot know the end of the following encryptions/decryptions.

51.4.6.2 If LOD = 1

The DATRDY flag is cleared when at least one Input Data Register is written, so before the start of a new transfer (See Figure 51-2). No more Output Data Register reads are necessary between consecutive encryptions/decryptions.

Figure 51-2. Manual and Auto Modes with LOD = 1



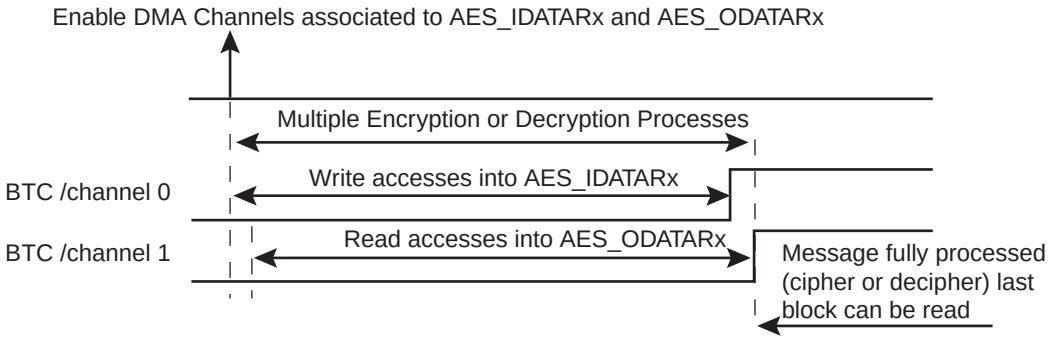
51.4.7 DMA Mode

51.4.7.1 If LOD = 0

This mode may be used for all AES operating modes except CBC-MAC where LOD = 1 mode is recommended.

The end of the encryption/decryption is notified by the end of DMA transfer associated to AES_ODATARx (see Figure 51-3). Two DMA channels are required: one for writing message blocks to AES_IDATARx and one to obtain the result from AES_ODATARx.

Figure 51-3. DMA transfer with LOD = 0



51.4.7.2 If LOD = 1

This mode is recommended to process AES CBC-MAC operating mode.

The user must first wait for the DMA flag (BTC = Buffer Transfer Complete) to rise, then for DATRDY to ensure that the encryption/decryption is completed (see Figure 51-4).

In this case, no receive buffers are required.

The output data are only available on the Output Data Registers (AES_ODATARx).

Figure 51-4. DMA transfer with LOD = 1

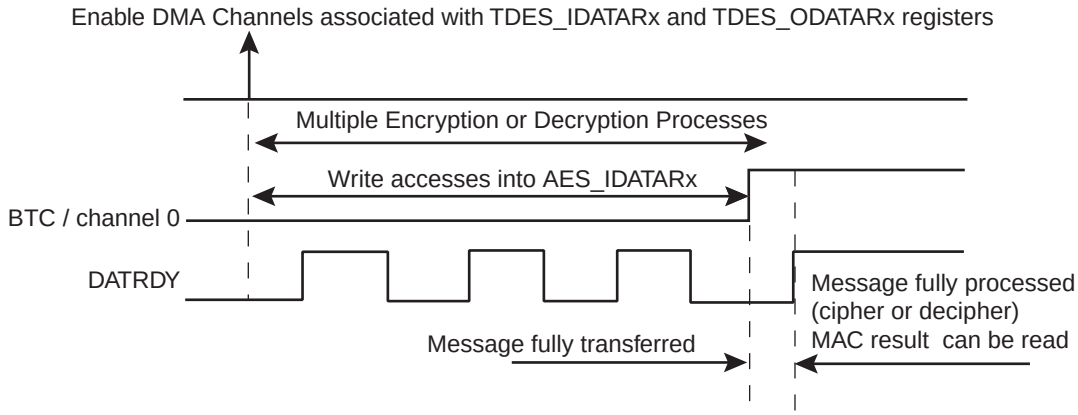


Table 51-4 summarizes the different cases.

Table 51-4. Last Output Data Mode Behavior versus Start Modes

Sequence	Manual and Auto Modes		DMA Transfer	
	LOD = 0	LOD = 1	LOD = 0	LOD = 1
DATRDY Flag Clearing Condition(1)	At least one Output Data Register must be read	At least one Input Data Register must be written	Not used	Managed by the DMA
End of Encryption/Decryption Notification	DATRDY	DATRDY	2 DMA flags (BTC[n/m])	DMA flag (BTC[n]) then AES DATRDY
Encrypted/Decrypted Data Result Location	In the Output Data Registers	In the Output Data Registers	At the address specified in the Channel Buffer Transfer Descriptor	In the Output Data Registers

Note: 1. Depending on the mode, there are other ways of clearing the DATRDY flag. See Section 51.6.6 "AES Interrupt Status Register".

Warning: In DMA mode, reading to the Output Data registers before the last data transfer may lead to unpredictable results.

51.5 Security Features

51.5.1 Unspecified Register Access Detection

When an unspecified register access occurs, the URAD bit in the Interrupt Status Register (AES_ISR) raises. Its source is then reported in the Unspecified Register Access Type field (URAT). Only the last unspecified register access is available through the URAT field.

Several kinds of unspecified register accesses can occur:

- Input Data Register written during the data processing when SMOD = IDATAR0_START
- Output Data Register read during data processing
- Mode Register written during data processing
- Output Data Register read during sub-keys generation
- Mode Register written during sub-keys generation
- Write-only register read access

The URAD bit and the URAT field can only be reset by the SWRST bit in the AES_CR.

51.6 Advanced Encryption Standard (AES) User Interface

Table 51-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	AES_CR	Write-only	–
0x04	Mode Register	AES_MR	Read-write	0x0
0x08–0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	AES_IER	Write-only	–
0x14	Interrupt Disable Register	AES_IDR	Write-only	–
0x18	Interrupt Mask Register	AES_IMR	Read-only	0x0
0x1C	Interrupt Status Register	AES_ISR	Read-only	0x0
0x20	Key Word Register 0	AES_KEYWR0	Write-only	–
0x24	Key Word Register 1	AES_KEYWR1	Write-only	–
0x28	Key Word Register 2	AES_KEYWR2	Write-only	–
0x2C	Key Word Register 3	AES_KEYWR3	Write-only	–
0x30	Key Word Register 4	AES_KEYWR4	Write-only	–
0x34	Key Word Register 5	AES_KEYWR5	Write-only	–
0x38	Key Word Register 6	AES_KEYWR6	Write-only	–
0x3C	Key Word Register 7	AES_KEYWR7	Write-only	–
0x40	Input Data Register 0	AES_IDATAR0	Write-only	–
0x44	Input Data Register 1	AES_IDATAR1	Write-only	–
0x48	Input Data Register 2	AES_IDATAR2	Write-only	–
0x4C	Input Data Register 3	AES_IDATAR3	Write-only	–
0x50	Output Data Register 0	AES_ODATAR0	Read-only	0x0
0x54	Output Data Register 1	AES_ODATAR1	Read-only	0x0
0x58	Output Data Register 2	AES_ODATAR2	Read-only	0x0
0x5C	Output Data Register 3	AES_ODATAR3	Read-only	0x0
0x60	Initialization Vector Register 0	AES_IVR0	Write-only	–
0x64	Initialization Vector Register 1	AES_IVR1	Write-only	–
0x68	Initialization Vector Register 2	AES_IVR2	Write-only	–
0x6C	Initialization Vector Register 3	AES_IVR3	Write-only	–
0x70–0xFC	Reserved	–	–	–

51.6.1 AES Control Register

Name: AES_CR

Address: 0xF8038000

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
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	START

- **START: Start Processing**

0: No effect

1: Starts manual encryption/decryption process.

- **SWRST: Software Reset**

0: No effect.

1: Resets the AES. A software triggered hardware reset of the AES interface is performed.

51.6.2 AES Mode Register

Name: AES_MR

Address: 0xF8038004

Access: Read-write

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

23	22	21	20	19	18	17	16
CKEY				–	CFBS		

15	14	13	12	11	10	9	8
LOD	OPMOD			KEYSIZE		SMOD	

7	6	5	4	3	2	1	0
PROCDLY				DUALBUFF	–	–	CIPHER

- **CIPHER: Processing Mode**

0: Decrypts data.

1: Encrypts data.

- **DUALBUFF: Dual Input Buffer**

Value	Name	Description
0x0	INACTIVE	AES_IDATARx cannot be written during processing of previous block.
0x1	ACTIVE	AES_IDATARx can be written during processing of previous block when SMOD = 0x2. It speeds up the overall runtime of large files.

- **PROCDLY: Processing Delay**

$$\text{Processing Time} = 12 \times (PROCDLY + 1)$$

The Processing Time represents the number of clock cycles that the AES needs in order to perform one encryption/decryption.

Note: The best performance is achieved with PROCDLY equal to 0.

- **SMOD: Start Mode**

Value	Name	Description
0x0	MANUAL_START	Manual Mode
0x1	AUTO_START	Auto Mode
0x2	IDATAR0_START	AES_IDATAR0 access only Auto Mode

Values which are not listed in the table must be considered as “reserved”.

If a DMA transfer is used, 0x2 must be configured. Refer to Section 51.4.4 ”DMA Mode” for more details.

- **KEYSIZE: Key Size**

Value	Name	Description
0x0	AES128	AES Key Size is 128 bits
0x1	AES192	AES Key Size is 192 bits
0x2	AES256	AES Key Size is 256 bits

Values which are not listed in the table must be considered as “reserved”.

- **OPMOD: Operation Mode**

Value	Name	Description
0x0	ECB	ECB: Electronic Code Book mode
0x1	CBC	CBC: Cipher Block Chaining mode
0x2	OFB	OFB: Output Feedback mode
0x3	CFB	CFB: Cipher Feedback mode
0x4	CTR	CTR: Counter mode (16-bit internal counter)

Values which are not listed in the table must be considered as “reserved”.

For CBC-MAC operating mode, please set OPMOD to CBC and LOD to 1.

- **LOD: Last Output Data Mode**

0: No effect.

After each end of encryption/decryption, the output data are available either on the output data registers (Manual and Auto modes) or at the address specified in the Channel Buffer Transfer Descriptor for DMA mode.

In Manual and Auto modes, the DATRDY flag is cleared when at least one of the Output Data registers is read.

1: The DATRDY flag is cleared when at least one of the Input Data Registers is written.

No more Output Data Register reads is necessary between consecutive encryptions/decryptions (see “Last Output Data Mode” on page 1687).

Warning: In DMA mode, reading to the Output Data registers before the last data encryption/decryption process may lead to unpredictable results.

- **CFBS: Cipher Feedback Data Size**

Value	Name	Description
0x0	SIZE_128BIT	128-bit
0x1	SIZE_64BIT	64-bit
0x2	SIZE_32BIT	32-bit
0x3	SIZE_16BIT	16-bit
0x4	SIZE_8BIT	8-bit

Values which are not listed in table must be considered as “reserved”.

- **CKEY: Key**

Value	Name	Description
0xE	PASSWD	This field must be written with 0xE the first time that AES_MR is programmed. For subsequent programming of the AES_MR, any value can be written, including that of 0xE. Always reads as 0.

51.6.3 AES Interrupt Enable Register

Name: AES_IER

Address: 0xF8038010

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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **DATRDY: Data Ready Interrupt Enable**
- **URAD: Unspecified Register Access Detection Interrupt Enable**

51.6.4 AES Interrupt Disable Register

Name: AES_IDR

Address: 0xF8038014

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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **DATRDY: Data Ready Interrupt Disable**
- **URAD: Unspecified Register Access Detection Interrupt Disable**

51.6.5 AES Interrupt Mask Register

Name: AES_IMR

Address: 0xF8038018

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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **DATRDY: Data Ready Interrupt Mask**
- **URAD: Unspecified Register Access Detection Interrupt Mask**

51.6.6 AES Interrupt Status Register

Name: AES_ISR

Address: 0xF803801C

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
URAT				–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready**

0: Output data not valid.

1: Encryption or decryption process is completed.

DATRDY is cleared when a Manual encryption/decryption occurs (START bit in AES_CR) or when a software triggered hardware reset of the AES interface is performed (SWRST bit in AES_CR).

LOD = 0 (AES_MR):

In Manual and Auto mode, the DATRDY flag can also be cleared when at least one of the Output Data Registers is read.

In DMA mode, DATRDY is set and cleared automatically.

LOD = 1 (AES_MR):

In Manual and Auto mode, the DATRDY flag can also be cleared when at least one of the Input Data Registers is written.

In DMA mode, DATRDY is set and cleared automatically.

- **URAD: Unspecified Register Access Detection Status**

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

URAD bit is reset only by the SWRST bit in the AES_CR.

- **URAT: Unspecified Register Access:**

Value	Name	Description
0x0	IDR_WR_PROCESSING	Input Data Register written during the data processing when SMOD = 0x2 mode.
0x1	ODR_RD_PROCESSING	Output Data Register read during the data processing.
0x2	MR_WR_PROCESSING	Mode Register written during the data processing.
0x3	ODR_RD_SUBKGEN	Output Data Register read during the sub-keys generation.
0x4	MR_WR_SUBKGEN	Mode Register written during the sub-keys generation.
0x5	WOR_RD_ACCESS	Write-only register read access.

Only the last Unspecified Register Access Type is available through the URAT field.

URAT field is reset only by the SWRST bit in the AES_CR.

51.6.7 AES Key Word Register x

Name: AES_KEYWRx

Address: 0xF8038020

Access: Write-only

31	30	29	28	27	26	25	24
KEYW							

23	22	21	20	19	18	17	16
KEYW							

15	14	13	12	11	10	9	8
KEYW							

7	6	5	4	3	2	1	0
KEYW							

- **KEYW: Key Word**

The four/six/eight 32-bit Key Word registers set the 128-bit/192-bit/256-bit cryptographic key used for encryption/decryption.

AES_KEYWR0 corresponds to the first word of the key and respectively AES_KEYWR3/AES_KEYWR5/AES_KEYWR7 to the last one.

These registers are write-only to prevent the key from being read by another application.

51.6.8 AES Input Data Register x

Name: AES_IDATARx

Address: 0xF8038040

Access: Write-only

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

- **IDATA: Input Data Word**

The four 32-bit Input Data registers set the 128-bit data block used for encryption/decryption.

AES_IDATAR0 corresponds to the first word of the data to be encrypted/decrypted, and AES_IDATAR3 to the last one.

These registers are write-only to prevent the input data from being read by another application.

51.6.9 AES Output Data Register x

Name: AES_ODATARx

Address: 0xF8038050

Access: Read-only

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

- **ODATA: Output Data**

The four 32-bit Output Data registers contain the 128-bit data block that has been encrypted/decrypted.

AES_ODATAR0 corresponds to the first word, AES_ODATAR3 to the last one.

51.6.10 AES Initialization Vector Register x

Name: AES_IVRx

Address: 0xF8038060

Access: Write-only

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

- **IV: Initialization Vector**

The four 32-bit Initialization Vector registers set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input.

AES_IVR0 corresponds to the first word of the Initialization Vector, AES_IVR3 to the last one.

These registers are write-only to prevent the Initialization Vector from being read by another application.

For CBC, OFB and CFB modes, the IV input value corresponds to the initialization vector.

For CTR mode, the IV input value corresponds to the initial counter value.

Note: These registers are not used in ECB mode and must not be written.

52. Triple Data Encryption Standard (Triple DES)

52.1 Description

The Triple Data Encryption Standard (TDES) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 46-3* specification.

The TDES supports the four different confidentiality modes of operation (ECB, CBC, OFB and CFB), specified in the *FIPS (Federal Information Processing Standard) Publication 81* and is compatible with the Peripheral Data Controller channels for all of these modes, minimizing processor intervention for large buffer transfers.

The 64-bit long keys and input data (and initialization vector for some modes) are each stored in two 32-bit registers (TDES_KEYxWRx, TDES_IDATARx and TDES_IVRx) which are both write-only.

As soon as the initialization vector, the input data and the key are configured, the encryption/decryption process may be started. Then the encrypted/decrypted data is ready to be read out on the two 32-bit output data registers (TDES_ODATARx) or through the DMA channels.

52.2 Embedded Characteristics

- Supports Single Data Encryption Standard (DES) and Triple Data Encryption Algorithm (TDEA or TDES)
- Compliant with *FIPS Publication 46-3, Data Encryption Standard (DES)*
- 64-bit Cryptographic Key for TDES
- Two-key or Three-key Algorithms for TDES
- 18-clock Cycles Encryption/Decryption Processing Time for DES
- 50-clock Cycles Encryption/Decryption Processing Time for TDES
- Support the Four Standard Modes of Operation specified in the *FIPS Publication 81, DES Modes of Operation*
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
- 8-bit, 16-bit, 32-bit and 64-bit Data Sizes Possible in CFB Mode
- Last Output Data Mode Allowing Optimized Message (Data) Authentication Code (MAC) Generation
- Connection to DMA Optimizes Data Transfers for all Operating Modes

52.3 Product Dependencies

52.3.1 Power Management

The TDES may be clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the TDES clock.

52.3.2 Interrupt

The TDES interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the TDES interrupt requires programming the AIC before configuring the TDES.

Peripheral IDs

Instance	ID
TDES	44

52.4 Functional Description

The Data Encryption Standard (DES) and the Triple Data Encryption Algorithm (TDES) specify FIPS-approved cryptographic algorithms that can be used to protect electronic data. The TDES bit in the TDES Mode Register (TDES_MR) is used to select either the single DES or the Triple DES mode.

Encryption (enciphering) converts data to an unintelligible form called ciphertext. Decrypting (deciphering) the ciphertext converts the data back into its original form, called plaintext. The CIPHER bit in the TDES Mode Register is used to choose between encryption and decryption.

A DES is capable of using cryptographic keys of 64 bits to encrypt and decrypt data in blocks of 64 bits. This 64-bit key is defined in the Key 1 Word Registers (TDES_KEY1WRx).

A TDES key consists of three DES keys, which is also referred to as a key bundle. These three 64-bit keys are defined, respectively, in the Key 1, 2 and 3 Word Registers (TDES_KEY1WRx, TDES_KEY2WRx and TDES_KEY3WRx). In Triple DES mode (TDESMOD set to 1), the KEYMOD bit in the TDES Mode Register is used to choose between a two- and a three-key algorithm:

- In three-key encryption mode, the data is first encrypted with Key 1, then decrypted using Key 2 and then encrypted with Key 3.
- In three-key decryption mode, the data is decrypted with Key 3, then encrypted with Key 2 and then decrypted using Key 1.
- In two-key encryption mode, the data is first encrypted with Key 1, then decrypted using Key 2 and then encrypted with Key 1.
- In two-key decryption mode, the data is decrypted with Key 1, then encrypted with Key 2 and then decrypted using Key 1.

The input to the encryption processes of the CBC, CFB, and OFB modes includes, in addition to the plaintext, a 64-bit data block called the initialization vector (IV), which must be set in the Initialization Vector Registers (TDES_IVRx). The initialization vector is used in an initial step in the encryption of a message and in the corresponding decryption of the message.

52.4.1 Operation Modes

The TDES supports the following modes of operation:

- ECB: Electronic Code Book
- CBC: Cipher Block Chaining
- OFB: Output Feedback
- CFB: Cipher Feedback
  - CFB8 (CFB where the length of the data segment is 8 bits)
  - CFB16 (CFB where the length of the data segment is 16 bits)
  - CFB32 (CFB where the length of the data segment is 32 bits)
  - CFB64 (CFB where the length of the data segment is 64 bits)

The data pre-processing, post-processing and data chaining for each mode are automatically performed. Refer to the *FIPS Publication 81* for more complete information.

These modes are selected by setting the OPMOD field in the TDES Mode Register (TDES_MR).

In CFB mode, four data sizes are possible (8, 16, 32 and 64 bits), configurable by means of the CFBS field in the mode register. (See “TDES Mode Register” on page 1712.).

The OFB and CFB modes of operation are only available if 2-key mode is selected (KEYMOD=1 in TDES_MR register).

52.4.2 Start Modes

The SMOD field in the TDES Mode Register (TDES_MR) allows selection of encryption (or decryption) start mode.

52.4.2.1 Manual Mode

The sequence order is as follows:

- Write the Mode Register (TDES_MR) with all required fields, including but not limited to SMOD and OPMOD.
- Write the 64-bit key(s) in the different Key Word Registers (TDES_KEYxWRx), depending on whether one, two or three keys are required.
- Write the initialization vector (or counter) in the Initialization Vector Registers (TDES_IVRx).

Note: The Initialization Vector Registers concern all modes except ECB.

- Set the bit DATRDY (Data Ready) in the TDES Interrupt Enable register (TDES_IER), depending on whether an interrupt is required or not at the end of processing.
- Write the data to be encrypted/decrypted in the authorized Input Data Registers (See Table 52-1).

Table 52-1. Authorized Input Data Registers

Operation Mode	Input Data Registers to Write
ECB	All
CBC	All
OFB	All
CFB 64-bit	All
CFB 32-bit	TDES_IDATAR0
CFB 16-bit	TDES_IDATAR0
CFB 8-bit	TDES_IDATAR0

Note: In 32-bit, 16-bit and 8-bit CFB mode, writing to TDES_IDATAR1 register is not allowed and may lead to errors in processing.

- Set the START bit in the TDES Control register TDES_CR to begin the encryption or the decryption process.
- When the processing completes, the bit DATRDY in the TDES Interrupt Status Register (TDES_ISR) raises. If an interrupt has been enabled by setting the bit DATRDY in TDES_IER, the interrupt line of the TDES is activated.

- When the software reads one of the Output Data Registers (TDES_ODATARx), the DATRDY bit is automatically cleared.

52.4.2.2 Auto Mode

The Auto Mode is similar to Manual Mode, except that, as soon as the correct number of Input Data registers is written, processing is automatically started without any action in the control register.

52.4.2.3 DMA Mode

The DMA Controller can be used in association with the TDES to perform an encryption/decryption of a buffer without any action by the software during processing.

The SMOD field of TDES_MR must be set to 0x2 and the DMA must be configured with non-incremental addresses.

The start address of any transfer descriptor must be set in TDES_IDATAR0 register.

The DMA chunk size configuration depends on the TDES mode of operation and is listed in Table 52-2 "DMA Data Transfer Type for the Different Operation Modes".

When writing data to TDES with the first DMA channel, data will be fetched from a memory buffer (source data). It is recommended to configure the size of source data to “words” even for CFB modes. On the contrary, the destination data size depends on the mode of operation. When reading data from the TDES with the second DMA channel, the source data is the data read from TDES and data destination is the memory buffer. In this case, source data size depends on the TDES mode of operation and is listed in Table 52-2.

Table 52-2. DMA Data Transfer Type for the Different Operation Modes

Operation Mode	Chunk Size	Destination/Source Data Transfer Type
ECB	1	Word
CBC	1	Word
OFB	1	Word
CFB 64-bit	1	Word
CFB 32-bit	1	Word
CFB 16-bit	1	Half-word
CFB 8-bit	1	Byte

52.4.3 Last Output Data Mode

This mode is used to generate cryptographic checksums on data (MAC) using a CBC-MAC or a CFB encryption algorithm (See *FIPS Publication 81 Appendix F*).

After each end of encryption/decryption, the output data is available either on the output data registers for Manual and Auto mode or at the address specified in the receive buffer pointer for DMA mode (See Table 52-3 "Last Output Data Mode Behavior versus Start Modes").

The Last Output Data bit (LOD) in the TDES Mode Register (TDES_MR) allows retrieval of only the last data of several encryption/decryption processes.

This data is only available on the Output Data Registers (TDES_ODATARx).

No more Output Data Register reads are necessary between consecutive encryptions/decryptions.

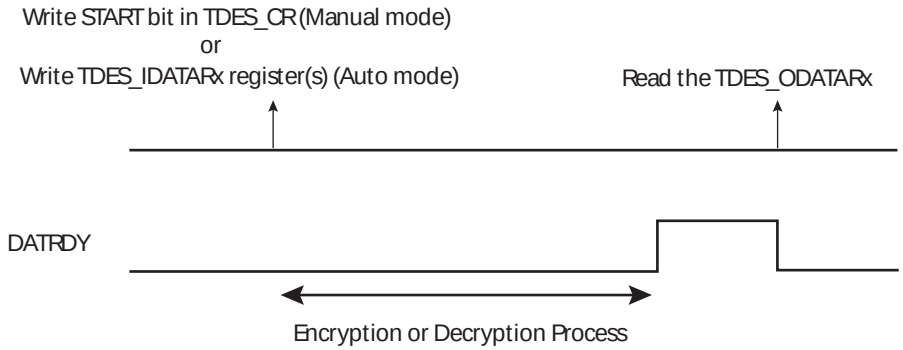
Therefore, there is no need to define a read buffer in DMA mode.

52.4.3.1 Manual and Auto Modes

If LOD = 0:

The DATRDY flag is cleared when at least one of the Output Data Registers is read. See Figure 52-1.

Figure 52-1. Manual and Auto Modes with LOD = 0

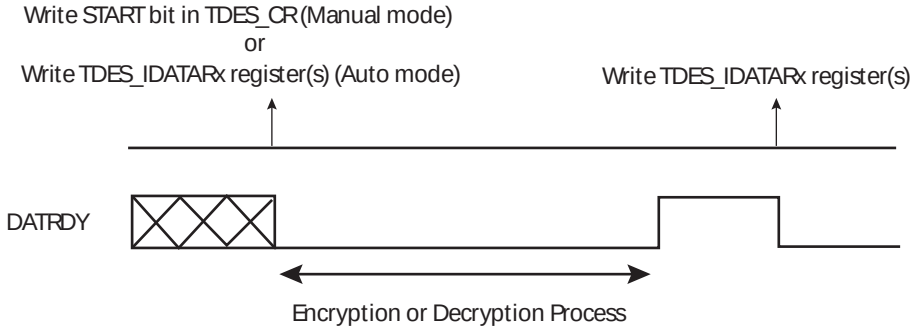


If the user does not want to read the output data registers between each encryption/decryption, the DATRDY flag will not be cleared. If the DATRDY flag is not cleared, the user will not be informed of the end of the encryptions/decryptions that follow.

If LOD = 1:

The DATRDY flag is cleared when at least one Input Data Register is written, before the start of a new transfer. See Figure 52-2. No further Output Data Register reads are necessary between consecutive encryptions/decryptions.

Figure 52-2. Manual and Auto Modes with LOD = 1



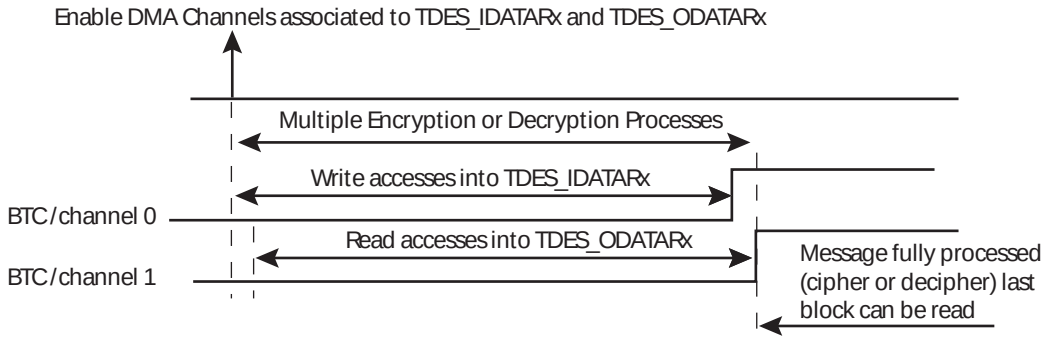
52.4.3.2 DMA Mode

If LOD = 0

This mode may be used for all TDES operating modes except CBC-MAC where LOD = 1 mode is recommended.

The end of the encryption/decryption is notified by the end of DMA transfer associated to TDES_ODATARx registers (see Figure 52-3). Two DMA channels are required, one for writing message blocks to TDES_IDATARx registers and the other one to get back the processed from TDES_ODATARx registers.

Figure 52-3. DMA transfer with LOD = 0



If LOD = 1

This mode is recommended to process TDES CBC-MAC operating mode.

The user must first wait for the DMA flag (BTC = Buffer Transfer Complete) to rise, then for DATRDY to ensure that the encryption/decryption is completed (see Figure 52-4).

In this case, no receive buffers are required.

The output data is only available on the Output Data Registers (TDES_ODATARx).

Figure 52-4. DMA transfer with LOD = 1

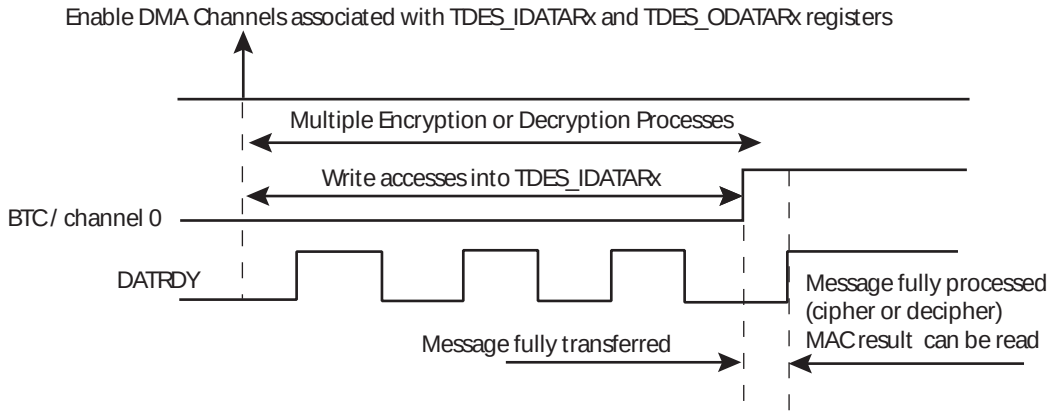


Table 52-3 summarizes the different cases.

Table 52-3. Last Output Data Mode Behavior versus Start Modes

	Manual and Auto Modes		DMA Transfer	
	LOD = 0	**LOD = 1**	**LOD = 0**	**LOD = 1**
DATRDY Flag Clearing Condition(1)	At least one Output Data Register must be read	At least one Input Data Register must be written	Not used	Managed by the DMA
Encrypted/Decrypted Data Result Location	In the Output Data Registers	In the Output Data Registers	Not available	In the Output Data Registers
End of Encryption/Decryption	DATRDY	DATRDY	2 DMA flags (BTC[n/m])	DMA flag (BTC[n]) then TDES DATRDY

Note: Depending on the mode, there are other ways of clearing the DATRDY flag. See: Section 52.5.6 "TDES Interrupt Status Register".

Warning: In DMA mode, reading to the Output Data registers before the last data transfer may lead to unpredictable results.

52.4.4 Security Features

52.4.4.1 Unspecified Register Access Detection

When an unspecified register access occurs, the URAD bit in the Interrupt Status Register (TDES_ISR) raises. Its source is then reported in the Unspecified Register Access Type field (URAT). Only the last unspecified register access is available through the URAT field.

Several kinds of unspecified register accesses can occur:

- Input Data Register written during the data processing in DMA mode
- Output Data Register read during the data processing
- Mode Register written during the data processing
- Write-only register read access

The URAD bit and the URAT field can only be reset by the SWRST bit in the TDES_CR control register.

52.5 Triple Data Encryption Standard (TDES) User Interface

Table 52-4. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TDES_CR	Write-only	–
0x04	Mode Register	TDES_MR	Read-write	0x2
0x08-0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	TDES_IER	Write-only	–
0x14	Interrupt Disable Register	TDES_IDR	Write-only	–
0x18	Interrupt Mask Register	TDES_IMR	Read-only	0x0
0x1C	Interrupt Status Register	TDES_ISR	Read-only	0x0000001E
0x20	Key 1 Word Register 0	TDES_KEY1WR0	Write-only	–
0x24	Key 1 Word Register 1	TDES_KEY1WR1	Write-only	–
0x28	Key 2 Word Register 0	TDES_KEY2WR0	Write-only	–
0x2C	Key 2 Word Register 1	TDES_KEY2WR1	Write-only	–
0x30	Key 3 Word Register 0	TDES_KEY3WR0	Write-only	–
0x34	Key 3 Word Register 1	TDES_KEY3WR1	Write-only	–
0x38-0x3C	Reserved	–	–	–
0x40	Input Data Register 0	TDES_IDATAR0	Write-only	–
0x44	Input Data Register 1	TDES_IDATAR1	Write-only	–
0x48-0x4C	Reserved	–	–	–
0x50	Output Data Register 0	TDES_ODATAR0	Read-only	0x0
0x54	Output Data Register 1	TDES_ODATAR1	Read-only	0x0
0x58-0x5C	Reserved	–	–	–
0x60	Initialization Vector Register 0	TDES_IVR0	Write-only	–
0x64	Initialization Vector Register 1	TDES_IVR1	Write-only	–
0x68 - 0xFC	Reserved	–	–	–

52.5.1 TDES Control Register

Name: TDES_CR

Address: 0xF803C000

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
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	START

- **START: Start Processing**

0: No effect

1: Starts Manual encryption/decryption process.

- **SWRST: Software Reset**

0: No effect.

1: Resets the TDES. A software triggered hardware reset of the TDES interface is performed.

52.5.2 TDES Mode Register

Name: TDES_MR

Address: 0xF803C004

Access: Read-write

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

- **CIPHER: Processing Mode**

0 (DECRYPT): Decrypts data.

1 (ENCRYPT): Encrypts data.

- **TDESMOD: ALGORITHM mode**

0 (SINGLE_DES): Single DES processing using TDES_KEY1WRx registers.

1 (TRIPLE_DES): Triple DES processing using registers TDES_KEY1WRx, TDES_KEY2WRx and TDES_KEY3WRx if KEYMOD is set.

- **KEYMOD: Key Mode**

0: Three-key algorithm is selected.

1: Two-key algorithm is selected. There is no need to write TDES_KEY3WRx registers.

- **SMOD: Start Mode**

Value	Name	Description
0x0	MANUAL_START	Manual Mode
0x1	AUTO_START	Auto Mode
0x2	IDATAR0_START	TDES_IDATAR0 access only Auto Mode

Values which are not listed in the table must be considered as “reserved”.

If a DMA transfer is used, 0x2 must be configured. Refer to Section 52.4.3.2 ”DMA Mode” for more details.

- **OPMOD: Operation Mode**

Value	Name	Description
0x0	ECB	ECB: Electronic Code Book mode
0x1	CBC	CBC: Cipher Block Chaining mode
0x2	OFB	OFB: Output Feedback mode
0x3	CFB	CFB: Cipher Feedback mode

For CBC-MAC operating mode, please set OPMOD to CBC and LOD to 1.

The OFB and CFB modes of operation are only available if 2-key mode is selected (KEYMOD=1).

- **LOD: Last Output Data Mode**

0: No effect.

After each end of encryption/decryption, the output data is available either on the output data registers (Manual and Auto modes).

In Manual and Auto modes, the DATRDY flag is cleared when at least one of the Output Data registers is read.

1: The DATRDY flag is cleared when at least one of the Input Data Registers is written.

No more Output Data Register reads are necessary between consecutive encryptions/decryptions (See “Last Output Data Mode” on page 1706.).

Warning: In DMA mode, reading to the Output Data registers before the last data encryption/decryption process may lead to unpredictable result.

- **CFBS: Cipher Feedback Data Size**

Value	Name	Description
0x0	SIZE_64BIT	64-bit
0x1	SIZE_32BIT	32-bit
0x2	SIZE_16BIT	16-bit
0x3	SIZE_8BIT	8-bit

52.5.3 TDES Interrupt Enable Register

Name: TDES_IER

Address: 0xF803C010

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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Enable**
- **URAD: Unspecified Register Access Detection Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

52.5.4 TDES Interrupt Disable Register

Name: TDES_IDR

Address: 0xF803C014

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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Disable**
- **URAD: Unspecified Register Access Detection Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

52.5.5 TDES Interrupt Mask Register

Name: TDES_IMR

Address: 0xF803C018

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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Mask**
- **URAD: Unspecified Register Access Detection Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

52.5.6 TDES Interrupt Status Register

Name: TDES_ISR

Address: 0xF803C01C

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
		URAT		–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready**

0: Output data is not valid.

1: Encryption or decryption process is completed.

DATRDY is cleared when a Manual encryption/decryption occurs (START bit in TDES_CR) or when a software triggered hardware reset of the TDES interface is performed (SWRST bit in TDES_CR).

LOD = 0 (TDES_MR):

In Manual and Auto mode, the DATRDY flag can also be cleared when at least one of the Output Data Registers is read.

LOD = 1 (TDES_MR):

In Manual and Auto mode, the DATRDY flag can also be cleared when at least one of the Input Data Registers is written.

- **URAD: Unspecified Register Access Detection Status**

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

URAD bit is reset only by the SWRST bit in the TDES_CR control register.

- **URAT: Unspecified Register Access**

Value	Name	Description
0x0	IDR_WR_PROCESSING	Input Data Register written during the data processing when SMOD=0x2 mode.
0x1	ODR_RD_PROCESSING	Output Data Register read during the data processing.
0x2	MR_WR_PROCESSING	Mode Register written during the data processing.
0x3	WOR_RD_ACCESS	Write-only register read access.

Only the last Unspecified Register Access Type is available through the URAT field.

URAT field is reset only by the SWRST bit in the TDES_CR control register.

52.5.7 TDES Key 1 Word Register x

Name: TDES_KEY1WRx

Address: 0xF803C020

Access: Write-only

31	30	29	28	27	26	25	24
KEY1W							

23	22	21	20	19	18	17	16
KEY1W							

15	14	13	12	11	10	9	8
KEY1W							

7	6	5	4	3	2	1	0
KEY1W							

- **KEY1W: Key 1 Word**

The two 32-bit Key 1 Word Registers allow to set the 64-bit cryptographic key used for encryption/decryption.

KEY1W0 corresponds to the first word of the key and KEY1W1 to the last one.

These registers are write-only to prevent the key from being read by another application.

52.5.8 TDES Key 2 Word Register x

Name: TDES_KEY2WRx

Address: 0xF803C028

Access: Write-only

31	30	29	28	27	26	25	24
KEY2W							

23	22	21	20	19	18	17	16
KEY2W							

15	14	13	12	11	10	9	8
KEY2W							

7	6	5	4	3	2	1	0
KEY2W							

- **KEY2W: Key 2 Word**

The two 32-bit Key 2 Word Registers allow to set the 64-bit cryptographic key used for encryption/decryption.

KEY2W0 corresponds to the first word of the key and KEY2W1 to the last one.

These registers are write-only to prevent the key from being read by another application.

Note: KEY2WRx registers are not used in DES mode.

.

52.5.9 TDES Key 3 Word Register x

Name: TDES_KEY3WRx

Address: 0xF803C030

Access: Write-only

31	30	29	28	27	26	25	24
KEY3W							

23	22	21	20	19	18	17	16
KEY3W							

15	14	13	12	11	10	9	8
KEY3W							

7	6	5	4	3	2	1	0
KEY3W							

- **KEY3W: Key 3 Word**

The two 32-bit Key 3 Word Registers allow to set the 64-bit cryptographic key used for encryption/decryption.

KEY3W0 corresponds to the first word of the key and KEY3W1 to the last one.

These registers are write-only to prevent the key from being read by another application.

Note: KEY3WRx registers are not used in DES mode, TDES with two-key algorithm selected.

52.5.10 TDES Input Data Register x

Name: TDES_IDATARx

Address: 0xF803C040

Access: Write-only

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

- **IDATA: Input Data**

The two 32-bit Input Data registers allow to set the 64-bit data block used for encryption/decryption.

IDATA0 corresponds to the first word of the data to be encrypted/decrypted, and IDATA1 to the last one.

These registers are write-only to prevent the input data from being read by another application.

52.5.11 TDES Output Data Register x

Name: TDES_ODATARx

Address: 0xF803C050

Access: Read-only

31	30	29	28	27	26	25	24
ODATA							

23	22	21	20	19	18	17	16
ODATA							

15	14	13	12	11	10	9	8
ODATA							

7	6	5	4	3	2	1	0
ODATA							

- **ODATA: Output Data**

The two 32-bit Output Data registers contain the 64-bit data block which has been encrypted/decrypted.

ODATA1 corresponds to the first word, ODATA2 to the last one.

52.5.12 TDES Initialization Vector Register x

Name: TDES_IVRx

Address: 0xF803C060

Access: Write-only

31	30	29	28	27	26	25	24
IV							

23	22	21	20	19	18	17	16
IV							

15	14	13	12	11	10	9	8
IV							

7	6	5	4	3	2	1	0
IV							

- **IV: Initialization Vector**

The two 32-bit Initialization Vector registers are used to set the 64-bit initialization vector data block, which is used by some modes of operation as an additional initial input.

IV1 corresponds to the first word of the Initialization Vector, IV2 to the last one.

These registers are write-only to prevent the Initialization Vector from being read by another application.

Note: These registers are not used for ECB mode and must not be written.

53. Secure Hash Algorithm (SHA)

53.1 Description

The Secure Hash Algorithm (SHA) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 180-2* specification.

The 512/1024-bit block of message is respectively stored in 16/32 x 32-bit registers (SHA_IDATARx/SHA_IODATARx) which are all write-only.

As soon as the input data is written, the hash processing may be started. The registers comprising the block of a padded message must be entered consecutively. Then the message digest is ready to be read out on the 5 up to 8/16 x 32-bit output data registers (SHA_IODATARx) or through the DMA channels.

53.2 Embedded Characteristics

- Supports Secure Hash Algorithm (SHA1, SHA224, SHA256, SHA384, SHA512)
- Compliant with *FIPS Publication 180-2*
- Configurable Processing Period:
  - 85 Clock Cycles to get a fast SHA1 runtime, 88 clock cycles for SHA384,SHA512 or 209 Clock Cycles for Maximizing Bandwidth of Other Applications
  - 72 Clock Cycles to get a fast SHA224, SHA256 runtime or 194 Clock Cycles for Maximizing Bandwidth of Other Applications
- Connection to DMA Channel Capabilities Optimizes Data Transfers
- Double Input Buffer Optimizes Runtime

53.3 Product Dependencies

53.3.1 Power Management

The SHA may be clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the SHA clock.

53.3.2 Interrupt

The SHA interface has an interrupt line connected to the Interrupt Controller.

Handling the SHA interrupt requires programming the interrupt controller before configuring the SHA.

Table 53-1. Peripheral IDs

Instance	ID
SHA	42

53.4 Functional Description

The Secure Hash Algorithm (SHA) module requires a padded message according to FIPS180-2 specification. The first block of the message must be indicated to the module by a specific command. The SHA module produces a N-bit message digest each time a block is written and processing period ends. N is 160 for SHA1, 224 for SHA224, 256 for SHA256, 384 for SHA384, 512 for SHA512.

53.4.1 SHA Algorithm

The module can process SHA1, SHA224, SHA256, SHA384, SHA512 by means of a configuration field in the SHA_MR register.

53.4.2 Processing Period

The processing period can be configured.

The short processing period allows to allocate bandwidth to the SHA module whereas the long processing period allocates more bandwidth on the system bus to other applications (example: DMA channels not associated with SHA).

In SHA1 mode, the shortest processing period is 85 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA384, SHA512 mode, the shortest processing period is 88 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA256 and SHA224 mode, the shortest processing period is 72 clock cycles + 2 clock cycles for start command synchronization. The longest period is 194 clock cycles + 2 clock cycles.

53.4.3 Double Input Buffer

The input data register can be double-buffered to reduce the runtime of large files.

This mode allows to write a new message block while the previous message block is being processed. This is only possible when DMA accesses are performed (SMOD=0x2).

The DUALBUFF field in SHA_MR register must be set to 1 to get access to double buffer.

53.4.4 Start Modes

The SMOD field in the SHA Mode Register (SHA_MR) is used to select the hash processing start mode.

53.4.4.1 Manual Mode

The sequence is as follows:

- Set the bit DATRDY (Data Ready) in the SHA Interrupt Enable Register (SHA_IER), depending on whether an interrupt is required or not at the end of processing.
- For the first block of a message, the FIRST command must be set by writing a 1 into the corresponding bit of the Control Register (SHA_CR). For the other blocks, there is nothing to write in this Control Register.
- Write the block to be processed in the Input Data Registers.
- Set the START bit in the SHA Control Register SHA_CR to begin the processing.
- When the processing completes, the bit DATRDY in the SHA Interrupt Status Register (SHA_ISR) raises. If an interrupt has been enabled by setting the bit DATRDY in SHA_IER, the interrupt line of the SHA is activated.
- Repeat the write procedure for each block, start procedure and wait for the interrupt procedure up to the last block of the entire message. Each time the start procedure is complete, the DATRDY flag is cleared.
- After the last block is processed (DATRDY flag is set, if an interrupt has been enabled by setting the bit DATRDY in SHA_IER, the interrupt line of the SHA is activated), read the message digest in the Output Data Registers. The DATRDY flag is automatically cleared when reading the SHA_IODATARx registers.

53.4.4.2 Auto Mode

Auto Mode is similar to Manual Mode, except that in this mode, as soon as the correct number of Input Data Registers is written, processing is automatically started without any action in the control register.

53.4.4.3 DMA Mode

The DMA can be used in association with the SHA to perform the algorithm on a complete message without any action by the software during processing.

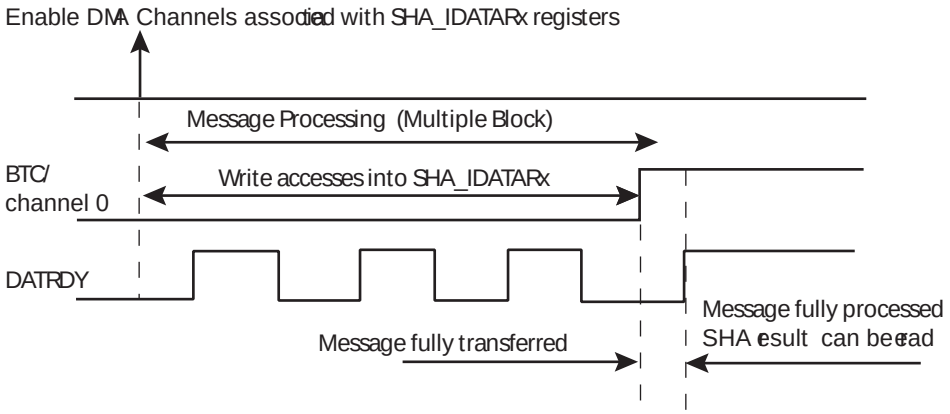
The SMOD field of the SHA_MR must be set to 0x2.

The DMA must be configured with non incremental addresses.

The start address of any transfer descriptor must be set to point to the SHA_IDATAR0 register.

The DMA chunk size must be set to transfer, for each trigger request, 16 words of 32 bits when processing SHA1/SHA256 algorithms or 32 words of 32 bits when SHA384/SHA512 are being used.

Figure 53-1. Enable DMA Channels



53.4.4.4 SHA Register Endianism

In ARM processor based products, the AHB bus and processors manipulate data in Little Endian form. However, following the protocol of FIPS 180-2 specification, data is collected, processed and stored by the SHA module in a Big Endian form. The data presented to the SHA module (written to SHA_IDATAxR) must be in Little Endian form. The data read from the SHA module (read from SHA_IODATAxR) will be in Little Endian form.

The SHA interface automatically converts into Big Endian format words that are presented into Little Endian. Likewise, the SHA interface returns hash results into Little Endian format even if the internal processing is Big Endian.

Managing how data is presented to the SHA registers should be managed by software.

As a clarification of this process consider the following example.

If the first 64 bits of a message (according to FIPS180-2, i.e. Big Endian format) to be processed is 0xcafe_dede_0123_4567 then the SHA_IDATA0R and SHA_IDATA1R registers should be written with the following pattern:

- SHA_IDATA0R = 0xefac
- SHA_IDATA1R = 0xeded

In a Little Endian system, the message starting with pattern 0xcafe_dede_0123_4567 will be stored into memories as follows:

- 0xca stored at initial offset (for example 0x00),
- then 0xfe stored at initial offset + 1 (i.e. 0x01),
- 0xde stored at initial offset + 2 (i.e. 0x02),
- 0xde stored at initial offset +3 (i.e. 0x03).

Let's assume the message is received through a serial to parallel communication channel, the first received character is 0xca and stored at first memory location (initial offset), second octet being 0xfe is stored at initial offset + 1.

When reading on a 32-bit Little Endian system bus, the first word read back from system memory is 0x_dede_feca.

When the SHA_IODATAxR registers are read, the hash result is organized in Little Endian format, allowing system memory storage in the same format as the message.

Taking an example from the FIPS 180-2 specification Appendix B.1, the endianism conversion can be observed.

For this example, the 512-bit message is:

0x6162638000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000018

and the expected SHA-256 result is:

0xba7816bf_8f01cfea_414140de_5dae2223_b00361a3_96177a9c_b410ff61_f20015ad

If the message has not already been stored in the system memory, the first step is to convert the input message to Little Endian before writing to the SHA_IDATAxR registers. This would result in a write of:

SHA_IDATA0R = 0x80636261...... SHA_IDATA15R = 0x18000000

The data in the output message digest registers, SHA_IODATAxR, contain SHA_IODATAxR = 0xbf1678ba... SHA_IODATA7R = 0xad1500f2 which is the Little Endian format of 0xba7816bf,..., 0xf20015ad.

Reading SHA_IODATA0R to SHA_IODATA1R and storing into a Little Endian memory system forces hash results to be stored in the same format as the message.

When the output message is read, the user can convert back to Big Endian for a resulting message value of:

0xba7816bf_8f01cfea_414140de_5dae2223_b00361a3_96177a9c_b410ff61_f20015ad

53.4.5 Security Features

When an unspecified register access occurs, the URAD bit in the Interrupt Status Register (SHA_ISR) raises. Its source is then reported in the Unspecified Register Access Type field (URAT). Only the last unspecified register access is available through the URAT field.

Several kinds of unspecified register accesses can occur:

- Input Data Register written during the data processing in DMA mode
- Input/Output Data Register read during the data processing
- Mode Register written during the data processing
- Write-only register read access

The URAD bit and the URAT field can only be reset by the SWRST bit in the SHA_CR control register.

53.5 Secure Hash Algorithm (SHA) User Interface

Table 53-2. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	SHA_CR	Write-only	–
0x04	Mode Register	SHA_MR	Read-write	0x0000100
0x08-0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	SHA_IER	Write-only	–
0x14	Interrupt Disable Register	SHA_IDR	Write-only	–
0x18	Interrupt Mask Register	SHA_IMR	Read-only	0x0
0x1C	Interrupt Status Register	SHA_ISR	Read-only	0x0
0x20-0x3C	Reserved			–
0x40	Input Data 0 Register	SHA_IDATAR0	Write-only	–
...	...	...	...	...
0x7C	Input Data 15 Register	SHA_IDATAR15	Write-only	–
0x80	Input/Output Data 0 Register	SHA_IODATAR0	Read-write	0x0
...	...	...	...	...
0x9C	Input/Output Data 7 Register	SHA_IODATAR7	Read-write	0x0
0xA0	Input/Output Data 8 Register	SHA_IODATAR8	Read-write	0x0
...	...	...	...	...
0xBC	Input/Output Data 15 Register	SHA_IODATAR15	Read-write	0x0
0x94-0xFC	Reserved	–	–	–

53.5.1 SHA Control Register

Name: SHA_CR

Address: 0xF8034000

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
7	6	5	4	3	2	1	0
–	–	–	FIRST	–	–	–	START

- **START: Start Processing**

0: No effect

1: Starts Manual hash algorithm process

- **FIRST: First Block of a Message**

0: No effect

1: Indicates that the next block to process is the first one of a message.

- **SWRST: Software Reset**

0: No effect.

1: Resets the SHA. A software triggered hardware reset of the SHA interface is performed.

53.5.2 SHA Mode Register

Name: SHA_MR

Address: 0xF8034004

Access: Read-write

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

- **SMOD: Start Mode**

Value	Name	Description
0x0	MANUAL_START	Manual Mode
0x1	AUTO_START	Auto Mode
0x2	IDATAR0_START	SHA_IDATAR0 access only Auto Mode

Values which are not listed in table must be considered as “reserved”.

If a DMA transfer is used, either 0x1 or 0x2 must be configured. Refer to “DMA Mode” on page 1726 for more details.

- **PROCDLY: Processing Delay**

Value	Name	Description
0x0	SHORTEST	SHA processing runtime is the shortest one
0x1	LONGEST	SHA processing runtime is the longest one

When SHA1 algorithm is processed, runtime period is either 85 or 209 clock cycles.

When SHA256 or SHA224 algorithm is processed, runtime period is either 72 or 194 clock cycles.

When SHA384 or SHA512 algorithm is processed, runtime period is either 88 or 209 clock cycles.

- **ALGO: SHA Algorithm.**

Value	Name	Description
0x0	SHA1	SHA1 algorithm processed
0x1	SHA256	SHA256 algorithm processed
0x2	SHA384	SHA384 algorithm processed
0x3	SHA512	SHA512 algorithm processed
0x4	SHA224	SHA224 algorithm processed

Values which are not listed in table must be considered as “reserved”.

- **DUALBUFF: Dual Input BUFFer**

Value	Name	Description
0x0	INACTIVE	SHA_IDATARx and SHA_IODATARx cannot be written during processing of previous block.
0x1	ACTIVE	SHA_IDATARx and SHA_IODATARx can be written during processing of previous block when SMOD=0x2. It speeds up the overall runtime of large files.

53.5.3 SHA Interrupt Enable Register

Name: SHA_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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Enable**
- **URAD: Unspecified Register Access Detection Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

53.5.4 SHA Interrupt Disable Register

Name: SHA_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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Disable**
- **URAD: Unspecified Register Access Detection Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

53.5.5 SHA Interrupt Mask Register

Name: SHA_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
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Mask**

- **URAD: Unspecified Register Access Detection Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

53.5.6 SHA Interrupt Status Register

Name: SHA_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
URAT				–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready**

0: Output data is not valid.

1: 512-bit block process is completed.

DATRDY is cleared when a Manual process occurs (START bit in SHA_CR) or when a software triggered hardware reset of the SHA interface is performed (SWRST bit in SHA_CR).

- **URAD: Unspecified Register Access Detection Status**

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

URAD bit is reset only by the SWRST bit in the SHA_CR control register.

URAT field indicates the unspecified access type.

- **URAT: Unspecified Register Access Type**

Value	Description
0x0	Input Data Register 0 to 15 written during the data processing in DMA mode. (URAD=0x1 and URAT=0x0 can occur only if DUALBUFF is cleared in SHA_MR)
0x1	Output Data Register read during the data processing.
0x2	Mode Register written during the data processing.
0x3	Write-only register read access.

Only the last Unspecified Register Access Type is available through the URAT field.

URAT field is reset only by the SWRST bit in the SHA_CR control register.

53.5.7 SHA Input Data x Register

Name: SHA_IDATARx [x=0.15]

Address: 0xF8034040

Access: Write-only

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

- **IDATA: Input Data**

The 32-bit Input Data registers allow to load the data block used for hash processing.

These registers are write-only to prevent the input data from being read by another application.

SHA_IDATAR0 corresponds to the first word of the block, SHA_IDATAR15 to the last word of the last block in case SHA algorithm is set to SHA1,SHA224,SHA256 or SHA_IODATA15R to the last word of the block if SHA algorithm is SHA384 or SHA512 (please refer to “SHA Input/Output Data x Register” on page 1737).

53.5.8 SHA Input/Output Data x Register

Name: SHA_IODATARx [x=0.15]

Address: 0xF8034080

Access: Read-write

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

- **IODATA: Input/Output Data**

These registers can be used to read the resulting message digest and to write the second part of the message block when the SHA algorithm is SHA-384 or SHA-512.

SHA_IODATA0R to SHA_IODATA15R registers can be written or read but reading these offsets does not return the content of corresponding parts (words) of the message block. Only results from SHA calculation can be read through these registers.

When SHA processing is in progress, these registers return 0x0000.

SHA_IODATA0R corresponds to the first word of message digest; SHA_IODATA5R to the last one in SHA1 mode, SHA_ODATA6R in SHA224, SHA_IODATA7R in SHA256, SHA_IODATA11R in SHA384 or SHA_IODATA15R in SHA512.

When SHA224 is selected, the content of SHA_ODATA7R must not be taken into account.

When SHA384 is selected, the content of SHA_IODATA12R to SHA_IODATA15R must not be taken into account.

54. Electrical Characteristics

54.1 Absolute Maximum Ratings

Table 54-1. Absolute Maximum Ratings*

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).....1.5V
(VDDIODDR).....2.0V
(VDDIOM, VDDIOPx, VDDUTMII, VDDOSC, VDDANA and VDDBU).....4.0V
(VDDFUSE).....3.3V
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.

54.2 DC Characteristics

The following characteristics are applicable to the operating temperature range T_A = -40°C to +105°C, unless otherwise specified.

Table 54-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_A	Operating Temperature	—	-40		+105	°C
$V_{VDDCORE}$	Core DC Supply Voltage	—	1.08	1.2	1.32	V
$V_{VDDUTMIC}$	UDPHS and UHPHS UTMI+ Core DC Supply Voltage	—	1.08	1.2	1.32	V
$V_{VDDUTMII}$	UDPHS and UHPHS UTMI+ Interface DC Supply Voltage	—	3.0	3.3	3.6	V
V_{VDDBU}	Backup DC Supply Voltage	—	1.65	—	3.6	V
$V_{VDDPLLA}$	PLLA DC Supply Voltage	—	1.08	1.2	1.32	V
V_{VDDOSC}	Oscillator and PLL UTMI DC Supply Voltage	If PLL UTMI is used the V_{VDDIO} range is to be 3.0V to 3.6V	1.65	—	3.6	V
V_{VDDIOM}	EBI I/Os DC Supply Voltage	—	1.65/3.0	1.8/3.3	1.95/3.6	V
$V_{VDDIODDR}$	SDRAM I/Os DC Supply Voltage	DDR2 or LP-DDR usage	1.7	1.8	1.95	V
		LP-DDR2 usage	1.14	1.2	1.30	
$V_{VDDIOP0}$	Peripheral I/Os DC Supply Voltage	—	1.65	—	3.6	V
$V_{VDDIOP1}$	Peripheral I/Os DC Supply Voltage	—	1.65	—	3.6	V
$V_{VDDFUSE}$	FUSE Box DC Supply Voltage	For FUSE programming only	2.25	2.5	2.75	V
$I_{VDDFUSE}$	VDDFUSE current	During FUSE programming	—	—	40	mA
V_{VDDANA}	Analog DC Supply Voltage	—	3.0	3.3	3.6	V
V_{IL}	Input Low-level Voltage	V_{VDDIO} in 3.3V range	-0.3	—	0.8	V
		V_{VDDIO} in 1.8V range	-0.3	—	0.3 x V_{VDDIO}	
V_{IH}	Input High-level Voltage	V_{VDDIO} in 3.3V range	2	—	V_{VDDIO} + 0.3	V
		V_{VDDIO} in 1.8V range	0.7 x V_{VDDIO}	—	V_{VDDIO} + 0.3	
V_{HYS}	Schmitt trigger Hysteresis	All PIO lines V_{VDDIOx} in 3.3V range	0.34	—	—	V
		All PIO lines V_{VDDIOx} in 1.8V range	0.21	—	—	
V_{OL}	Output Low-level Voltage	I_O Max	—	—	0.4	V
V_{OH}	Output High-level Voltage	I_O Max	V_{VDDIO} - 0.4	—	—	V
R_{PULL}	Pull-up Resistance	All PIO lines V_{VDDIOx} in 1.8V range	100	160	310	kΩ
	Pull-down Resistance	All PIO lines NTRST and NRST V_{VDDIOx} in 3.3V range	45	70	130	

Table 54-2. DC Characteristics (Continued)

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
I_O	Output Current	TDI, TCK, NTRST, BMS, TMS, TDO, NRST, NANDRDY, PA0-PA27, PA29-PA31, PB28-PB31, PC0-PC14, PC16-PC23, PC25-PC31, PD0-PD8, PD10-PD11, PD13-PD31 $V_{VDDIOPx}$ in 1.8V range		5 (LO_DRIVE)	16 (ME_DRIVE)	22 (HI_DRIVE)	mA
		TDI, TCK, NTRST, BMS, TMS, TDO, NRST, NANDRDY, PA0-PA27, PA29-PA31, PB28-PB31, PC0-PC14, PC16-PC23, PC25-PC31, PD0-PD8, PD10-PD11, PD13-PD31 $V_{VDDIOPx}$ in 3.3V range		8 (LO_DRIVE)	26 (ME_DRIVE)	38 (HI_DRIVE)	
		NCS3, NRD, NWE, PA28, PB0-PB27, PC15, PC24, PD9, PD12, PE0-PE31, D0-D15		3 (LO_DRIVE)	6 (ME_DRIVE)	7 (HI_DRIVE)	
R_{SERIAL}	Serial Resistor	TMS, TDO, NRST, NANDRDY, PA0-PA27, PA29-PA31, PB28-PB31, PC0-PC14, PC16-PC23, PC25-PC31, PD0-PD8, PD10-PD11, PD13-PD19		—	30	—	Ω
		NCS3, NRD, NWE, PA28, PB0-PB27, PC15, PC24, PD9, PD12, PE0-PE31, D0-D15		—	13	—	
I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.1V, MCK = 0 Hz, excluding POR All inputs driven TMS, TDI, TCK, NRST = 1	T_A = 25°C	—	0.4	—	mA
			T_A = 85°C	—	—	4.3	
			T_A = 105°C	—	—	8.2	
		On V_{VDDBU} = 3.3V, Logic cells consumption, excluding POR All inputs driven WKUP = 0	T_A = 25°C	—	1.2	—	µA
			T_A = 85°C	—	1.4	—	
			T_A = 105°C	—	1.8	2.6	

54.3 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
- Software used for power consumption measurements: DHRYSTONE
- The Instruction and Data Caches are enabled. The Memory Management Unit (MMU) is enabled.

54.3.1 Static current

Static current is measured when the system has no activity at all. It is also called leakage current and is very sensitive to process and operating temperature. This means the value may fluctuate drastically between two different devices, and vary according to the operating temperature. Figures are given in Table 54-2.

This variation explains most of the difference between two measures done on different devices or with different temperatures.

54.3.2 Active Mode

Active Mode is the normal running mode with the core clock running from a PLL. The power management controller can be used to adapt the frequency and to disable the peripheral clocks.

Note: Active power consumption may vary for ±5% depending on compilator used and process.

54.3.3 Low-power Modes

The various low-power modes are described below in the following subsections.

54.3.3.1 Backup Mode

The purpose of Backup Mode is to achieve the lowest power consumption possible in a system which is performing periodic wake-ups to perform tasks but not requiring fast startup time.

The Zero-power Power-on Reset, RTC, Backup registers and 32 kHz oscillator (RC or Crystal Oscillator selected by software in the Supply Controller) are running. The core supply is off.

The system can be awakened from this mode through the WKUP0 pin or an RTC wake-up event.

Backup mode is entered with the help of the Shutdown Controller that asserts the SHDN output pin. The SHDN pin is to be connected to the enable pin of the VDDCORE regulator.

Exit from Backup mode happens if one of the following enable wake-up events occurs:

- WKUP0 pin (level transition, configurable debouncing)
- RTC alarm

The system will restart as for a reset event.

54.3.3.2 Idle Mode

The purpose of Idle Mode is to optimize power consumption of the device versus response time. In this mode, only the core clock is stopped. The peripheral clocks, including the DDR Controller clock, can be enabled. The current consumption in this mode is application dependent.

This mode is entered via the Wait for Interrupt (WFI) instruction and PCK disabling.

The processor can be awakened from an interrupt. The system will resume where it was before entering in WFI mode.

Note: The power consumption includes the static current.

54.3.3.3 Ultra Low-power Mode

The purpose of Ultra Low-power Mode is to reduce the power consumption of the device to the minimum without disconnecting VDDCORE power supply. It is a combination of very low frequency operations and Idle Mode.

This mode is entered via the following steps:

1. Set the DDR in Self Refresh Mode
2. Reduce the system clock (PCK and MCK) to the minimum with the help of the PMC:
   - PCK and MCK configuration is to be defined regarding the expected power consumption and wake-up time. Please refer to Table 54-3 for details
   - PLLs are disabled. CKGR_PLLAR (eventually CKGR_PLLBR) is set to 0x3F00. CKGR_UCKR is set to 0.
   - Main oscillator is disabled. MOSCXTEN is set to 0 in CKGR_MOR.
   - Eventually 12 MHz RC oscillator is disabled. MOSCRCEN is set to 0 in CKGR_MOR.
3. Enter in Wait for Interrupt (WFI) mode and disable the PCK clock.

The processor can be awakened from an interrupt.

Once revived, the system must reprogram the system clocks (OSC, PLL, PCK, MCK, DDRCK) to recover the previous state. Data is maintained in the external memory.

54.3.3.4 Low-power Mode Summary Table

The modes detailed above are the main low-power modes. Each part can be set to on or off separately and wake-up sources can be individually configured. Table 54-3 below shows a summary of the configurations of the low-power modes.

Table 54-3. Low-power Mode Configuration Summary

Mode	32K RC, 12 MHz RC, 32 kHz Osc, RTC, Backup Registers, POR (Backup Region)	VDDCORE External Regulator	Core Memory Peripherals	Mode Entry	Potential Wake-Up Sources	Core at Wake Up	PIO State while in Low-power Mode	PIO State at Wake Up	Consumption(2)	Wake-Up Time(1)
Backup	ON	OFF	OFF (not powered)	Shutdown Controller	WKUP0 pin RTC alarm	Reset	Reset	Inputs with pull-ups	1.2 µA typ(3)	Start-up time
Idle	ON	ON	Powered (not clocked)	WFI	Any interrupt	Clocked back at full speed	Previous state saved	Unchanged	See Table 54-7 10 mA for each PLL(4)	285 ns @ 132 MHz
Ultra Low-power	ON	ON	Powered (not clocked)	DDR in Self Refresh PMC WFI	Any interrupt	Clocked back at previous one	Previous state saved	Unchanged	See Table 54-8	See Table 54-8

Notes:
1. When considering wake-up time, the time required to start the PLL is not taken into account. Once started, the device works with the Main Oscillator. The user has to add the PLL start-up time if it is needed in the system. The wake-up time is defined as the time taken for wake up until the first instruction is fetched.
2. The external loads on PIOs are not taken into account in the calculation.
3. Total current consumption
4. Depends on PLL frequency

54.3.4 Power Consumption versus Modes

The values in Table 54-4, Table 54-5, Table 54-6, Table 54-7, and Table 54-8 are measured values of the power consumption under the following operating conditions:

- Parts are from typical process
- V_{DDIOM} = 1.8 V
- $V_{DDIOP0 \text{ and } 1}$ = 3.3 V
- V_{DDPLLA} = 1.2 V
- V_{DDCORE} = 1.2 V unless specified otherwise in the table
- V_{DDBU} = 3.3 V
- T_A = as specified in the table
- There is no consumption on the I/Os of the device
- All the peripheral clocks are disabled

Figure 54-1. Measures Schematics

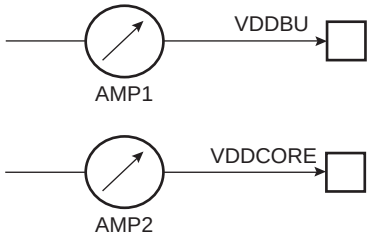


Table 54-4. VDDCORE Power Consumption in Active Mode (AMP2)

Mode	Conditions	VDDCORE	Consumption			Unit
			T_A 25°C	T_A 85°C	T_A 105°C	
Active	ARM Core clock is 396 MHz. MCK is 132 MHz.	1.2 V	90	95	101	mA
Active	ARM Core clock is 498 MHz. MCK is 166 MHz.	1.26 V	117	124	129	mA
Active	ARM Core clock is 528 MHz. MCK is 132 MHz.	1.26 V	117	124	129	mA

Table 54-5. Power Consumption by Peripheral in Active Mode (T_A 25°C, T_A 85°C and T_A 105°C)

Peripheral	Max Frequency (MHz)(1)	Consumption µA/MHz(2)	Conditions
PIO Controller	133	4.5	—
USART	66	3.0	—
UHPHS	133	12	No activity
		98	Connect tested EK (as USB Host) to another A5 EK (as USB Device), writing a file into USB key @ 10 MB/s, CPU usage 24%. This includes DDR2 controller power consumption on VDDCORE. Power consumption = 9.6 × transfer speed (MB/s) + 12

Table 54-5. Power Consumption by Peripheral in Active Mode (T_A 25°C, T_A 85°C and T_A 105°C) (Continued)

Peripheral	Max Frequency (MHz)[1]	Consumption µA/MHz[2]	Conditions
UDPHS	133	31	No activity
		88	Connect EK to PC. Read/write a file into DDR2 memory @ 14 MB/s, CPU usage = 17%. Power consumption = 4.1 × transfer speed (MB/s) + 31
ADC	66	4.0	—
TWI	66	1.3	—
SPI	133	2.5	—
PWM	133	6.5	—
HSMCI	133	14.5	—
SSC	66	3.5	—
Timer Counter Channels	133	3.0	—
DMA	133	50	No activity
		66	Data transfer from IF0 to IF1 with 64-byte full FIFO. Bandwidth usage is about 51%. Power consumption = 32.2 × bandwidth usage + 50.
SMD	24	5.5	—
ISI	133	30	—
EMAC	133	7	No activity
		174	EK is acting as a server. CPU usage is 16%. Power consumption = 4.2 × transfer speed (Mb/s) + 7. This includes DDR2 controller power consumption on VDDCORE.
GMAC	133	36	No activity
		160	EK is acting as a server. CPU usage is 24%. Power consumption = 1.2 × transfer speed (Mb/s) + 36. This includes DDR2 controller power consumption on VDDCORE.
LCDC	133	13.0	No activity
		160	Frame rate is 43 fps, 5 layers (Base 800px*600px, Ovr1 640px*480px, Ovr2 640px*480px, HEO 32px*48px, Cur 128px*128px), 24-bit data path, pixel clock is 44 MHz. This includes DDR2 controller power consumption on VDDCORE.
		230	Frame rate is maximum (85 fps), 5 layers (Base 800px*600px, Ovr1 640px*480px, Ovr2 640px*480px, HEO 32px*48px, Cur 128px*128px), 24-bit data path, pixel clock is 44 MHz. This includes DDR2 controller power consumption on VDDCORE.
CAN	66	4.5	—
SHA	133	4.0	—
AES	133	2.0	—
TDES	133	1.0	—
TRNG	133	0.2	—

Notes:
1. In order to reduce power consumption, each Peripheral Clock has been timed to an optimum value. The peripheral frequency is programmable with the help of a divider in the PMC_PCR. Programming a divider giving higher frequency than required will lead to an unpredictable behavior.
2. Reference frequency is the peripheral frequency. It can be a division (1, 2, 4, 8) of MCK. Please refer to the PMC section for more details.

Table 54-6. VDDBU Power Consumption in Backup mode (AMP1)

Mode	Conditions	VDDBU	Consumption			Unit
			T_A 25°C	T_A 85°C	T_A 105°C	
Backup	Backup Mode	3.3V	1.2 TYP 1.4 MAX	1.4 TYP 1.8 MAX	1.9 TYP 2.6 MAX	µA

Table 54-7. VDDCORE Power Consumption in Idle Mode (AMP2)

Mode	Conditions	VDDCORE	Consumption (mA)			Wakeup Time (ns)
			T_A 25°C	T_A 85°C	T_A 105°C	
Idle	ARM Core clock is disabled. MCK is 132 MHz.	1.2V	24	29	33	285

Table 54-8. VDDCORE Power Consumption in Ultra Low-Power Mode (AMP2)

Mode	Conditions	VDDCORE	Consumption (mA)			Wakeup Time (µs)
			T_A 25°C	T_A 85°C	T_A 105°C	
ULP 12MHz	ARM Core clock is disabled. MCK is 12 MHz.	1.2V	1.68	5.5	9.6	3.80
ULP 750kHz	ARM Core clock is disabled. MCK is 750 kHz.	1.2V	0.5	4.3	8.4	60.8
ULP 187MHz	ARM Core clock is disabled. MCK is 187.5 kHz.	1.2V	0.43	4.3	8.3	246
ULP 32kHz	ARM Core clock is disabled. MCK is 32 kHz.	1.2V	0.40	4.2	8.2	1390
ULP 512Hz	ARM Core clock is disabled. MCK is 512 Hz.	1.2V	0.40	4.1	8.2	88400

54.4 Clock Characteristics

54.4.1 Processor Clock Characteristics

Table 54-9. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	VDDCORE[1.08V, 1.32V], T_A = 85°C	250[1]	400	MHz
		VDDCORE[1.2V, 1.32V], T_A = 85°C	250[1]	536	

Note: 1. Limitation for DDR2 usage only. There are no limitations to LP-DDR and LP-DDR2. Ratio between PCK and MCK must be DIV2, DIV3 or DIV4.

54.4.2 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 54-10. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	Master Clock Frequency	VDDCORE[1.08V, 1.32V], T_A = 85°C	125(1)	134	MHz
		VDDCORE[1.2V, 1.32V], VDDIODDR[1.75V, 1.9V], DDR2 mode only, T_A = 85°C	125(1)	166	

Note: 1. Limitation for DDR2 usage only. There are no limitations to LP-DDR and LP-DDR2.

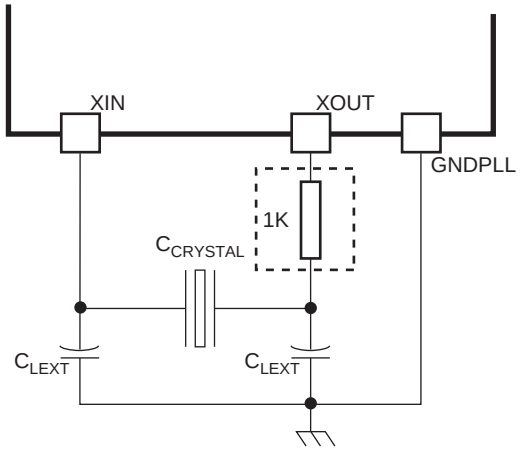
54.5 Main Oscillator Characteristics

Table 54-11. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Freq	Operating Frequency	FREQ = 00, 01[2] FREQ = 10 FREQ = 11	8 20 44	—	16 24 48	MHz
C_{LOAD}[1]	Load Capacitance	FREQ = 00, 01 FREQ = 10 FREQ = 11	12.5 4 4	—	17.5 8 6	pF
C_{LEXT}[1]	External Load Capacitance	—	—	—		pF
C_{PARA}	Parasitic Load Capacitance	—	0.6	0.7	0.8	pF
—	Duty Cycle	—	40	—	60	%
t_{ST}	Startup Time	—	—	—	2	ms
I_{DDST}	Standby Current Consumption	Standby mode	—	—	0.1	µA
P_{ON}	Drive Level	FREQ = 00, 01 FREQ = 10 FREQ = 11	—	—	150 300 400	µW
$I_{DD\ ON}$	Current Dissipation	@ 12 MHz	—	0.5	1	mA

Note:
1. The external capacitors value can be determined by using the following formula: $C_{LEXT} = (2*C_{LOAD}) - C_{BOARD} - C_{ROUTING} - C_{PACKAGE} - (C_{PARA}*2)$, where C_{LEXT}: external capacitor value which must be soldered from XIN to GND and XOUT to GND, C_{LOAD}: crystal targeted load. Please refer to C_{LOAD} parameter in the electrical specification, C_{BOARD}: external calculated (or measured) parasitic value due to board, $C_{ROUTING}$: parasitic capacitance due to internal chip routing, typically 1.5 pF, $C_{PACKAGE}$: parasitic capacitance due to package and bonding, typically 0.75 pF, C_{PARA}: internal parasitic load due to internal structure.
2. FREQ field defines the input frequency for the UTMI and the Main Oscillator. It is defined in the UTMI Clock Trimming Register (SFR_UTMICKTRIM) located in the SFR section. Take care to select the correct FREQ value—this has a direct influence on the USB frequency.

Figure 54-2. Main Oscillator Schematics



54.5.1 Crystal Oscillator Characteristics

The following characteristics are applicable to the operating temperature range T_A = -40°C to +105°C and worst case of power supply, unless otherwise specified.

Table 54-12. Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistance	@16 MHz, FREQ = 00, 01 @24 MHz, FREQ = 10 @48 MHz, FREQ = 11	—	—	80 150 100	Ω
C_M	Motional Capacitance	FREQ = 00, 01(1) FREQ = 10 with ESRmax = 150 Ω FREQ = 11 with ESRmax = 100 Ω	5 1.3 2	—	9 2 3.2	fF
C_S	Shunt Capacitance	FREQ = 00, 01 FREQ = 10 with ESRmax = 150 Ω FREQ = 11 with ESRmax = 100 Ω	—	—	7 1 1.3	pF

Note: 1. FREQ field defines the input frequency for the UTMI and the Main Oscillator. It is defined in UTMI Clock Trimming Register, located in SFR section. Take care to select the correct FREQ value, this has a direct influence on USB frequency.

54.5.2 XIN Clock Characteristics

Table 54-13. 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 x t_{CPXIN}	0.6 x t_{CPXIN}	ns
t_{CLXIN}	XIN Clock Low Half-period	—	0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	ns
C_{IN}	XIN Input Capacitance	(1)	—	25	pF
R_{IN}	XIN Pulldown Resistor	(1)	—	500	kΩ
V_{IN}	XIN Voltage	(1)	VDDOSC	VDDOSC	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.

54.6 12 MHz RC Oscillator Characteristics

Table 54-14. 12 MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F0	Nominal Frequency	—	11.4	12	12.6	MHz
Duty	Duty Cycle	—	45	50	55	%
$I_{DD\ ON}$	Power Consumption Oscillation	—	—	220	—	µA
t_{ON}	Startup time	—	—	—	10	µs
$I_{DD\ STDBY}$	Standby consumption	—	—	—	22	µA

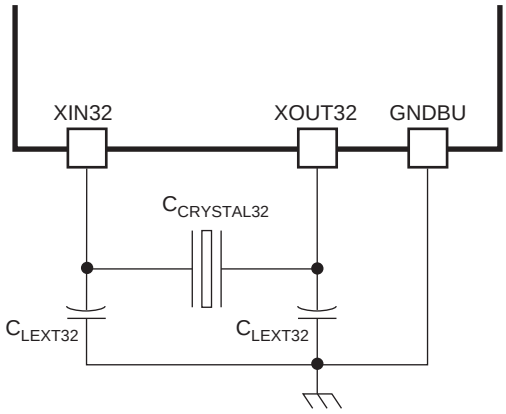
54.7 32 kHz Oscillator Characteristics

Table 54-15. 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
P_{ON}	Drive Level	—		—	—	0.2	µW

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 54-3. 32 kHz Oscillator Schematics



54.7.1 32 kHz Crystal Characteristics

Table 54-16. 32 kHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Crystal @ 32768 kHz	—	50	100	kΩ
C_M	Motional Capacitance	Crystal @ 32768 kHz	0.6	—	3	fF
C_S	Shunt Capacitance	Crystal @ 32768 kHz	0.6	—	2	pF
$I_{DD\ ON}$	Current dissipation	R_S = 50 kΩ(1) $C_{CRYSTAL32}$ = 6 pF	—	0.3	0.65	µA
		R_S = 50 kΩ(1) $C_{CRYSTAL32}$ = 12.5pF	—	0.45	0.7	µA
		R_S = 100 kΩ(1) $C_{CRYSTAL32}$ = 6 pF	—	0.45	0.75	µA
		R_S = 100 kΩ(1) $C_{CRYSTAL32}$ = 12.5 pF	—	0.45	0.65	µA
$I_{DD\ STDBY}$	Standby consumption	—	—	5	10	nA

54.7.2 XIN32 Clock Characteristics

Table 54-17. 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)	VDDBU	VDDBU	V
V_{INIL32}	XIN32 Input Low Level Voltage	(1)	-0.3	0.3 x V_{VDDBU}	V
V_{INIH32}	XIN32 Input High Level Voltage	(1)	0.7 x V_{VDDBU}	V_{VDDBU} + 0.3	V

Notes: 1. These characteristics apply only when the 32768 kHz oscillator is in bypass mode (i.e., when RCEN = 0, OSC32EN = 0, OSCSEL = 1 and OSC32BYP = 1) in the SCKC_CR. See section “Slow Clock Controller Configuration Register” in the Slow Clock Controller (SCKC) section of this datasheet.

54.8 32 kHz RC Oscillator Characteristics

Table 54-18. 32 kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPRCz})$	Crystal Oscillator Frequency	—	30.4	32	33.6	kHz
—	Duty Cycle	—	45	—	55	%
t_{ST}	Startup Time	—	—	290	500	µs
$I_{DD\ ON}$	Power Consumption Oscillation	After startup time	—	1.1	2.1	µA
$I_{DD\ STDBY}$	Standby consumption	—	—	—	0.2	µA

54.9 PLL Characteristics

Table 54-19. PLLA Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output Frequency	VDDPLLA[1.08V, 1.32V] ICPLLA = 0 OUTA = 0	400	—	800	MHz
		VDDPLLA[1.2V, 1.32V] ICPLLA = 0 OUTA = 0	400	—	1000	
f_{IN}	Input Frequency	—	8	—	50	MHz
I_{PLL}	Current Consumption	Active mode @ 800 MHz	—	10	14	mA
		Standby mode	—	1	200	µA
t	Startup Time	—	—	25	100	µs

54.9.1 UTMI PLL Characteristics

Table 54-20. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{IN}	Input Frequency	—	—	12	—	MHz
f_{OUT}	Output Frequency	—	—	480	—	MHz
I_{PLL}	Current Consumption	Active mode	—	5	8	mA
		Standby mode	—	—	1.5	µA
t	Startup Time	—	—	—	50	µs

54.10 USB HS Characteristics

54.10.1 Electrical Characteristics

Table 54-21. Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
R_{PUI}	Bus Pull-up Resistor on Upstream Port (idle bus)	In LS or FS Mode	—	1.5	—	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

54.10.2 Static Power Consumption

Table 54-22. 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.

54.10.3 Dynamic Power Consumption

Table 54-23. 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.

54.11 12-Bit ADC Characteristics

Table 54-24. Analog Power Supply Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{VDDANA}	ADC Analog Supply	12-bit	2.4	3.0	3.6	V
	Max. Voltage Ripple	RMS value, 10 kHz to 20 MHz	—	—	20	mV
I_{VDDANA}	Current Consumption	Sleep Mode	—	4	8	µA
		Fast Wake Up Mode	—	2.2	3.4	mA
		Normal Mode (IBCTL= 00)	—	3.3	4.8	mA
		Normal Mode (IBCTL= 01)	—	4.4	6.4	mA

Note: Use IBCTL = 00 for Sampling Frequency below 500 kHz and IBCTL = 01 between 500 kHz and 1 MHz.

Table 54-25. Channel Conversion Time and ADC Clock

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{ADC}	ADC Clock Frequency	—	1	—	20	MHz
t_{CP_ADC}	ADC Clock Period	—	50	—	1000	ns
f_S	Sampling Frequency	—	—	—	1	MHz
$t_{START-UP}$	ADC Startup time	From OFF Mode to Normal Mode: - Voltage Reference OFF - Analog Circuitry OFF	20	30	40	µs
		From Standby Mode to Normal Mode: - Voltage Reference ON - Analog Circuitry OFF	4	8	12	
$t_{TRACKTIM}$	Track and Hold Time	See Section 54.11.1.1 “Track and Hold Time versus Source Output Impedance” for more details	160	—	—	ns
t_{CONV}	Conversion Time	—	—	20	—	t_{CP_ADC}
t_{SETTLE}	Settling Time	Settling time to change offset and gain	200	—	—	ns

Table 54-26. External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Unit
ADVREF Input Voltage Range, 12-bit	$2.4V < V_{VDDIN} < 3.6V$	2.4	—	VDDANA	V
ADVREF Current	—	—	—	600	µA
ADVREF Input DC impedance	—	6	8	10	kΩ

54.11.1 Static performance characteristics

In the tables that follow, the LSB is relative to the analog scale:

- Single Ended (ex: ADVREF = 3.0 V)
  - Gain = 1, LSB = (3.0 V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5 V / 4096) = 366 µV
  - Gain = 4, LSB = (750 mV / 4096) = 183 µV
- Differential (ex: ADVREF = 3.0 V)
  - Gain = 0.5, LSB = (6.0 V / 4096) = 1465 µV
  - Gain = 1, LSB = (3.0 V / 4096) = 732 µV
  - Gain = 4, LSB = (750 mV / 4096) = 366 µV

Table 54-27. INL, DNL, 12-bit mode, VDDANA 2.4V to 3.6V supply voltage conditions

Parameter	Conditions	Min	Typ	Max	Unit
Resolution	—	—	12	—	Bit
Integral Non-linearity (INL)	All mode, all gain Typ = 3.0V supply, 27°C	-4	±1	+4	LSB
Differential Non-linearity (DNL)	All mode, all gain Typ = 3.0V supply, 27°C	-2	±0.5	+2	LSB

Table 54-28. Gain Error, 12-bit Mode, VDDANA 2.4V to 3.6V supply voltage conditions

Parameter	Conditions	Min	Typ	Max	Unit
Gain Error, Differential mode, uncalibrated	Any gain and offset values	-45	—	+45	LSB
Gain Error, Differential mode, after calibration	Gain = 0.5	-8	—	+8	
	Gain = 1	-16	—	+16	
	Gain = 2	-24	—	+24	
Gain Error, Single Ended, uncalibrated	Any gain and offset values	-45	—	+45	
Gain Error, Single Ended, after calibration	Gain = 1	-8	—	+8	
	Gain = 2	-16	—	+16	
	Gain = 4	-24	—	+24	

Table 54-29. Error offset with or without calibration, 12-bit Mode, VDDANA 2.4V to 3.6V supply voltage conditions

Parameter	Conditions	Min	Typ	Max	Unit
Offset Error	Single Ended, Gain = 1 / Differential mode Gain = 0.5	-8	—	+8	LSB
	Single Ended, Gain = 2 / Differential mode Gain = 1	-16	—	+16	
	Single Ended, Gain = 4 / Differential mode Gain = 2	-32	—	+32	

Table 54-30. Dynamic Performance Characteristics in Single ended and 12 bits mode (1)

Parameter	Conditions	Min	Typ	Max	Unit
Signal to Noise Ratio - SNR	—	54	61	—	dB
Total Harmonic Distortion - THD	—	—	-80	-68	dB
Signal to Noise and Distortion - SINAD	—	54	61	—	dB
Effective Number of Bits - ENOB	(1)	8.7	9.8	—	bits

Note: 1. ADC Clock (f_{ADC}) = 20 MHz, f_S = 1 MHz, f_{IN} = 127 kHz, IBCTL = 01, FFT using 1024 points or more, frequency band = [1 kHz, 500 kHz] – Nyquist conditions fulfilled.

Table 54-31. Dynamic Performance Characteristics in Differential and 12 bits mode(1)

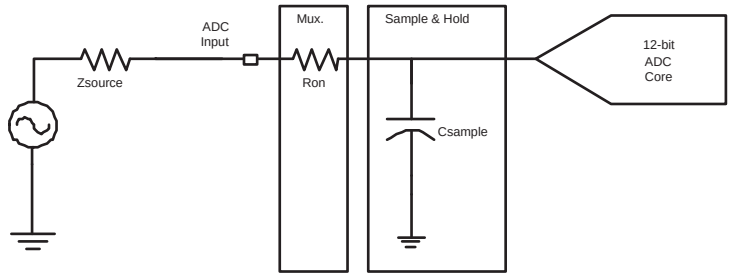
Parameter	Conditions	Min	Typ	Max	Unit
Signal to Noise Ratio - SNR	—	58	63	—	dB
Total Harmonic Distortion - THD	—	—	-80	-72	dB
Signal to Noise and Distortion - SINAD	—	58	63	—	dB
Effective Number of Bits - ENOB	(1)	9.3	10.2	—	bits

Note: 1. ADC Clock (f_{ADC}) = 20 MHz, f_S = 1 MHz, f_{IN} = 127 kHz, IBCTL = 01, FFT using 1024 points or more, frequency band = [1 kHz, 500 kHz] – Nyquist conditions fulfilled.

54.11.1.1 Track and Hold Time versus Source Output Impedance

The following figure gives a simplified acquisition path.

Figure 54-4. Simplified Acquisition Path



During the tracking phase the ADC needs to track the input signal during the tracking time shown below:

- 10-bit mode: $t_{TRACK} = 0.042 \times Z_{SOURCE} + 160$
- 12-bit mode: $t_{TRACK} = 0.054 \times Z_{SOURCE} + 205$

With t_{TRACK} expressed in ns and Z_{SOURCE} expressed in Ohms.

Two cases must be considered:

1. The calculated tracking time (t_{TRACK}) is lower than 15 t_{CP_ADC}.

Set TRANSFER = 1 and TRACTIM = 0.

In this case, the allowed Z_{SOURCE} can be computed versus the ADC frequency with the hypothesis of $t_{TRACK} = 15 \times t_{CP\_ADC}$:

Where $t_{CP\_ADC} = 1/f_{ADC}$. See Table 54-32 on page 1756.

Table 54-32. Source Impedance Values

f_{ADC} = ADC clock (MHz)	Z_{SOURCE} (kΩ) for 12 bits	Z_{SOURCE} (kΩ) for 10 bits
20.00	10	14
16.00	14	19
10.67	22	30
8.00	31	41
6.40	40	52
5.33	48	63
4.57	57	74
4.00	66	85
3.56	74	97
3.20	83	108
2.91	92	119
2.67	100	130
2.46	109	141
2.29	118	152
2.13	126	164
2.00	135	175
1.00	274	353

2. The calculated tracking time (t_{TRACK}) is higher than 15 t_{CP_ADC}.

Set TRANSFER = 1 and TRACTIM = 0.

In this case, a timer will trigger the ADC in order to set the correct sampling rate according to the Track time.

The maximum possible sampling frequency will be defined by t_{TRACK} in nanoseconds, computed by the previous formula but with minus 15 × t_{CP_ADC} and plus TRANSFER time.

- 10 bit mode: $1/f_S = t_{TRACK} - 15 \times t_{CP\_ADC} + 5\ t_{CP\_ADC}$
- 12 bit mode: $1/f_S = t_{TRACK} - 15 \times t_{CP\_ADC} + 5\ t_{CP\_ADC}$

Note: C_{SAMPLE} and R_{ON} are taken into account in the formulas.

Table 54-33. Analog Inputs

Parameter	Min	Typ	Max	Unit
Input Voltage Range	0	—	V_{ADVREF}	V
Input Leakage Current	—	—	±0.5	µA
Input Capacitance	—	—	8	pF

Note: Input Voltage range can be up to VDDANA without destruction or over-consumption.
If VDDIO < V_{ADVREF} max input voltage is VDDIO.

54.11.1.2 ADC Application Information

For more information on data converter terminology, please refer to the application note:

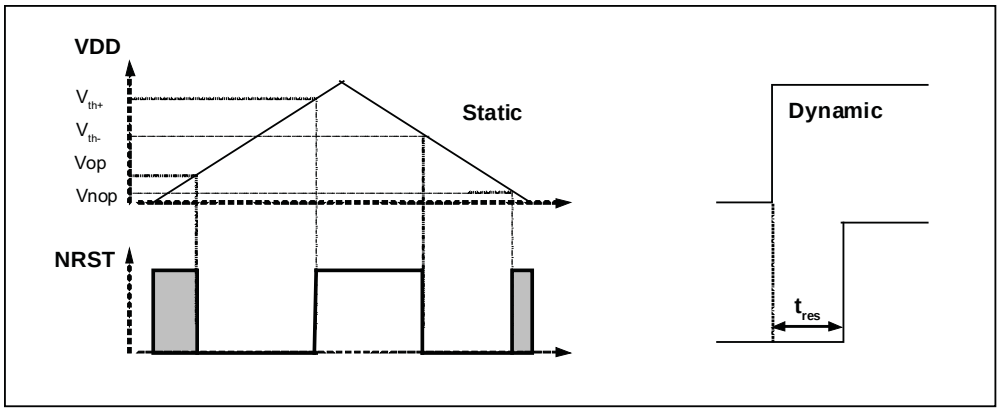
Data Converter Terminology, Atmel literature No. 6022

http://www.atmel.com/dyn/resources/prod_documents/doc6022.pdf

54.12 POR Characteristics

General presentation of Power-On-Reset (POR) characteristics is provided in Figure 54-5.

Figure 54-5. 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 VDD raises upper 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.

54.12.1 Core Power Supply POR Characteristics

Table 54-34. Core Power Supply POR Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{th+}	Threshold Voltage Rising	Minimum Slope of +2.3 V/ms	0.85	1.02	1.08	V
V_{th-}	Threshold Voltage Falling	—	0.8	0.96	1.06	V
t_{RES}	Reset Time	—	100	260	600	µs
I_{DD}	Current Consumption	After t_{RES}	—	250	500	nA

54.12.2 Backup Power Supply POR Characteristics

Table 54-35. Backup Power Supply POR Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{th+}	Threshold Voltage Rising	Minimum Slope of +1.9 V/ms	1.50	1.54	1.60	V
V_{th-}	Threshold Voltage Falling	—	1.39	1.46	1.53	V
t_{RES}	Reset Time	—	80	180	600	µs
I_{DD}	Current Consumption	After t_{RES}	—	300	500	nA

54.13 HSMC Timings

54.13.1 Timing conditions

HSMC Timings are given in MAX corners.

Timings assuming a capacitance load on data, control and address pads are given in Table 54-36.

Table 54-36. Capacitance Load

Supply	Corner	
	Max	Min
3.3V	50 pF	50 pF
1.8V	30 pF	30 pF

In tables that follow, t_{CPMCK} is MCK period.

54.13.2 Timing extraction

54.13.2.1 Read Timings

Table 54-37. SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	Parameter	Min		Max		Unit
	VDDIOM supply	1.8V	3.3V	1.8V	3.3V	
NO HOLD SETTINGS (nrd hold = 0)						
SMC_1	Data Setup before NRD High	9.6	6.8	—	—	ns
SMC_2	Data Hold after NRD High	0	0	—	—	ns
HOLD SETTINGS (nrd hold ≠ 0)						
SMC_3	Data Setup before NRD High	9.4	6.6	—	—	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} + 2.4	(nrd setup + nrd pulse)* t_{CPMCK} + 2.1	—	—	ns
SMC_6	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} + 2	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} + 1.9	—	—	ns
SMC_7	NRD Pulse Width	nrd pulse * t_{CPMCK} + 1.8	nrd pulse * t_{CPMCK} +1.2	—	—	ns

Table 54-38. SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	Parameter	Min		Max		Unit
	VDDIOM supply	1.8V	3.3V	1.8V	3.3V	
NO HOLD SETTINGS (ncs rd hold = 0)						
SMC_8	Data Setup before NCS High	18.5	16.2	—	—	ns
SMC_9	Data Hold after NCS High	0	0	—	—	ns
HOLD SETTINGS (ncs rd hold ≠ 0)						
SMC_{10}	Data Setup before NCS High	17	14.7	—	—	ns
SMC_{11}	Data Hold after NCS High	0	0	—	—	ns
HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)						
SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS High	(ncs rd setup + ncs rd pulse)* t_{CPMCK} + 19.1	(ncs rd setup + ncs rd pulse)* t_{CPMCK} + 18.8	—	—	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} + 16.8	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} +16.1	—	—	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length * t_{CPMCK} + 2	ncs rd pulse length * t_{CPMCK} + 1.5	—	—	ns

54.13.2.2 Write Timings

Table 54-39. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	Min		Max		Unit
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
HOLD or NO HOLD SETTINGS (nwe hold ≠ 0, nwe hold = 0)						
SMC_{15}	Data Out Valid before NWE High	nwe pulse * t_{CPMCK} + 2.4	nwe pulse * t_{CPMCK} + 1.6	—	—	ns
SMC_{16}	NWE Pulse Width	nwe pulse * t_{CPMCK} + 1.9	nwe pulse * t_{CPMCK} + 1.3	—	—	ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NWE low	nwe setup * t_{CPMCK} + 2.6	nwe pulse * t_{CPMCK} + 2	—	—	ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} + 2.2	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} + 1.9	—	—	ns
HOLD SETTINGS (nwe hold ≠ 0)						
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 change	nwe hold * t_{CPMCK} + 2.3	nwe hold * t_{CPMCK} +1.1	—	—	ns
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold)* t_{CPMCK} + 1.9	(nwe hold - ncs wr hold)* t_{CPMCK} +1	—	—	ns
NO HOLD SETTINGS (nwe hold = 0)						
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25, NCS change(1)	2	1.6	—	—	ns

Notes: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “NWE hold length”.

Table 54-40. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min		Max		Unit
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse * t_{CPMCK} + 9.8	ncs wr pulse * t_{CPMCK} + 9.1	—	—	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse * t_{CPMCK} + 2	ncs wr pulse * t_{CPMCK} + 1.5	—	—	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS low	ncs wr setup * t_{CPMCK} + 4.3	ncs wr setup * t_{CPMCK} + 3.7	—	—	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} + 2.4	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} + 2.1	—	—	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25, change	ncs wr hold * t_{CPMCK} + 2.6	ncs wr hold * t_{CPMCK} + 1.7	—	—	ns

Table 54-40. SMC Write NCS Controlled (WRITE_MODE = 0) (Continued)

Symbol	Parameter	Min		Max		Unit
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold)* t_{CPMCK} + 5.5	(ncs wr hold - nwe hold)* t_{CPMCK} + 5.6	—	—	ns

Figure 54-6. SMC Timings - NCS Controlled Read and Write

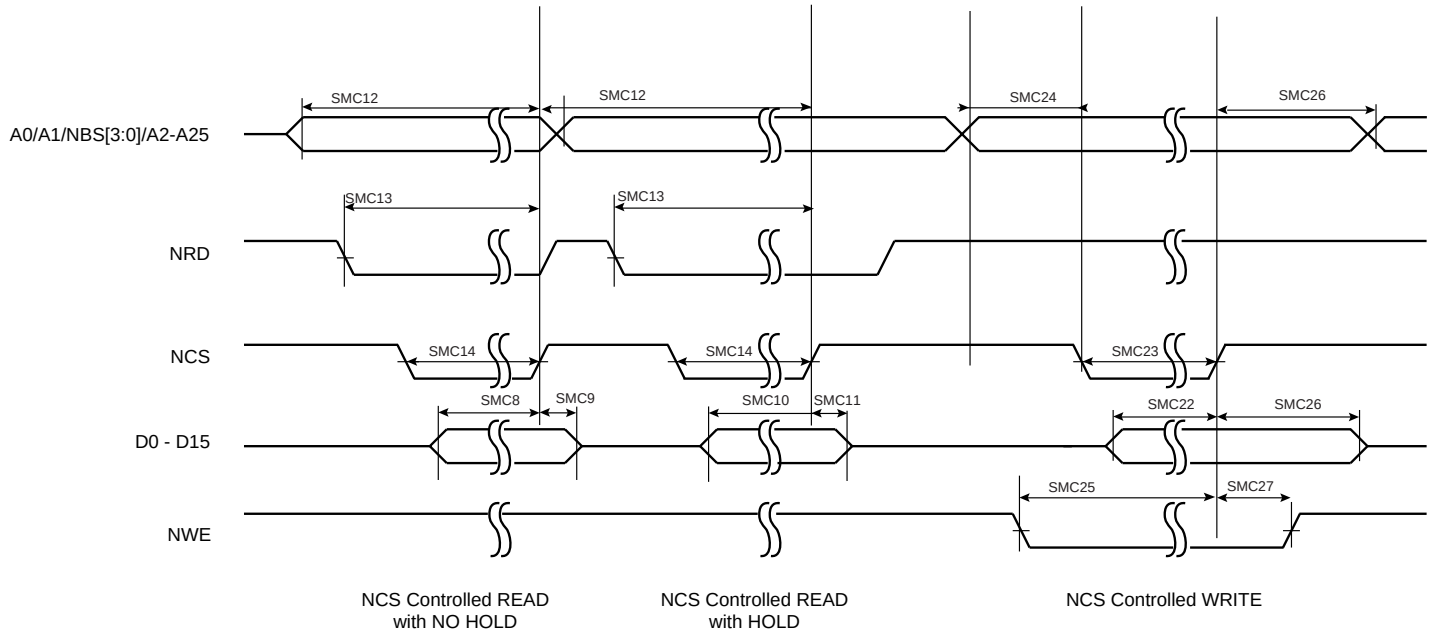
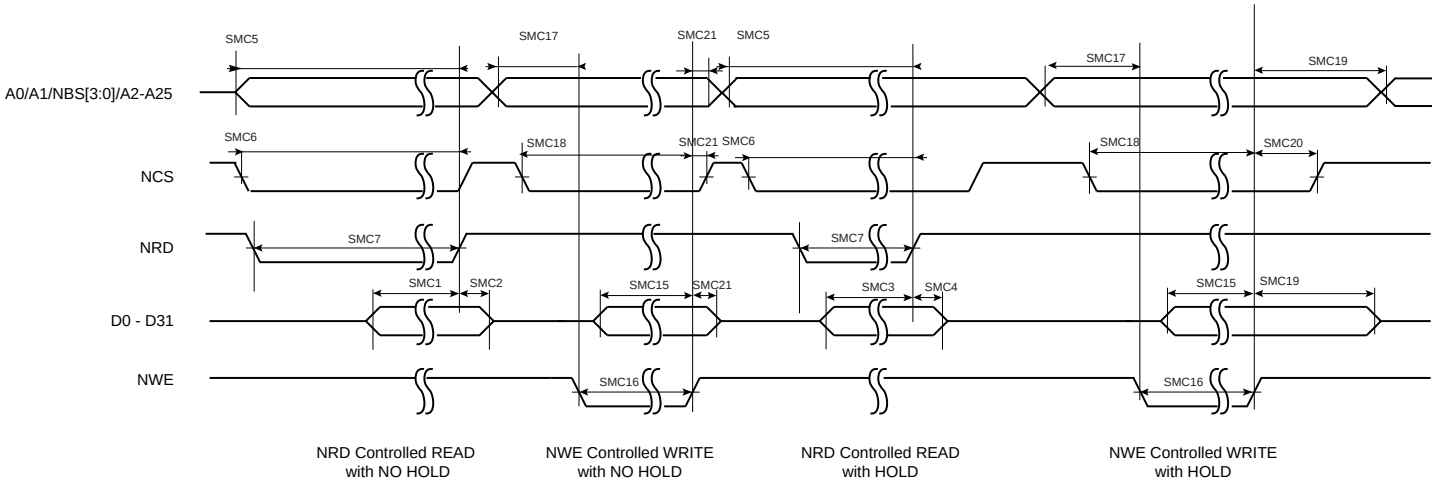


Figure 54-7. SMC Timings - NRD Controlled Read and NWE Controlled Write



54.13.2.3 FPGA Timings

SMC and PCK2 can be used to interface a FPGA. PCK2 is to be programmed to output MCK. READ_MODE and WRITE_MODE are to be configured to 0.

Figure 54-8. FPGA Timings

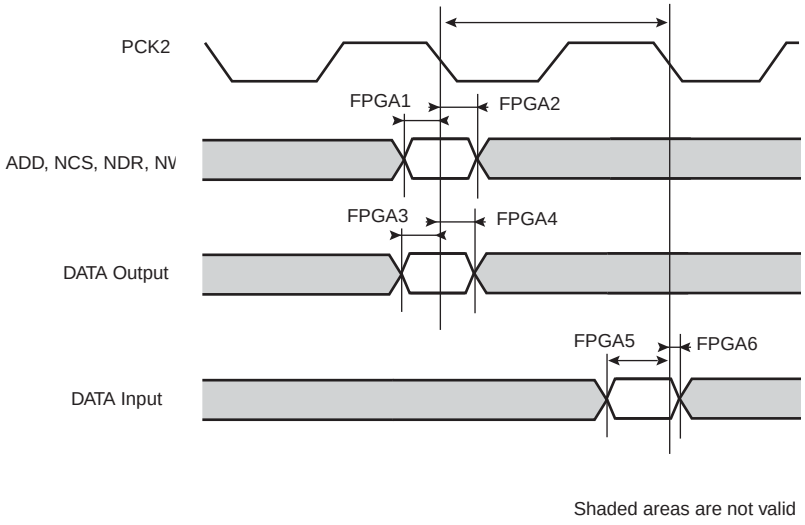


Table 54-41. FPGA Timings vs PCK2

Symbol	VDDIOM Supply / VDDCORE Supply / Parameter	1.65–1.95V / 1.08–1.32V / Min	3.0–3.6V / 1.08–1.32V / Min	3.0–3.6V / 1.2–1.32V / Min	1.65–1.95V / 1.08–1.32V / Max	3.0–3.6V / 1.08–1.32V / Max	3.0–3.6V / 1.2–1.32V / Max	Unit
$FPGA_1$	Address, NCS, NRD, NWE Setup before PCK2 falling edge	1.78	2.07	1.78	—	—	—	ns
$FPGA_2$	Address, NCS, NRD, NWE Hold after PCK2 falling edge	0.29	0.35	1.22	—	—	—	ns
$FPGA_3$	Data Out Setup before PCK2 falling edge	1.46	1.77	1.59	—	—	—	ns
$FPGA_4$	Data Out Hold after PCK2 falling edge	0	0.27	1.10	—	—	—	ns
$FPGA_5$	Data In Setup before PCK2 falling edge	—	—	—	14.15	12.08	10.39	ns
$FPGA_6$	Data In Hold after PCK2 falling edge	—	—	—	0	0	0	ns

54.14 SPI Timings

54.14.1 Timing Conditions

Timings assuming a capacitance load on MISO, SPCK and MOSI are given in Table 54-42.

Table 54-42. Capacitance Load for MISO, SPCK and MOSI (Product Dependent)

Supply	Corner	
	Max	Min
3.3V	40 pF	40 pF
1.8V	20 pF	40 pF

These values may be product dependent and should be confirmed by the specification.

54.14.2 Timing Extraction

Figure 54-9. SPI Master mode 1 and 2

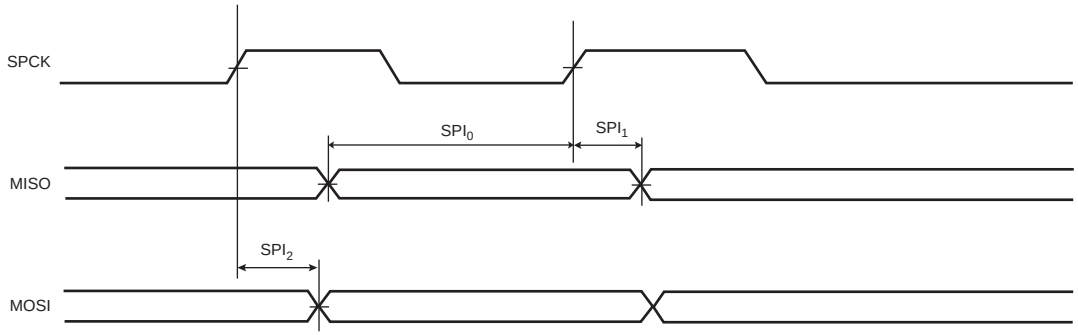


Figure 54-10.SPI Master mode 0 and 3

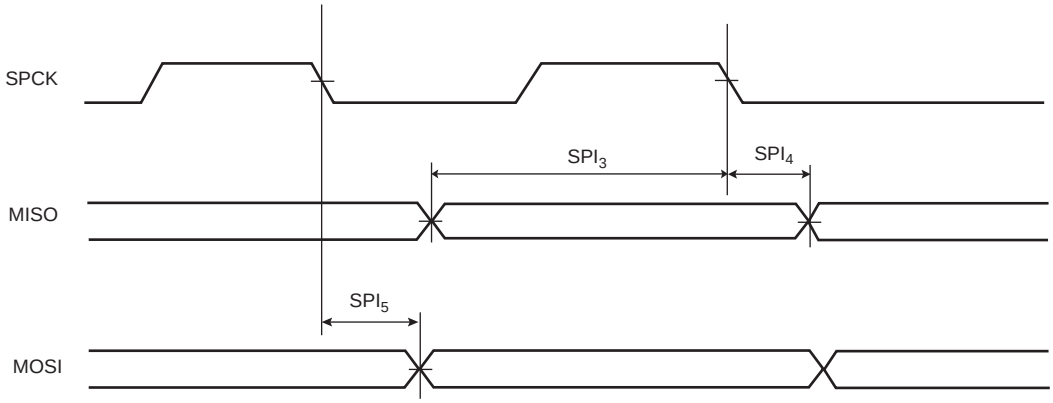


Figure 54-11.SPI Slave Mode 0 and 3

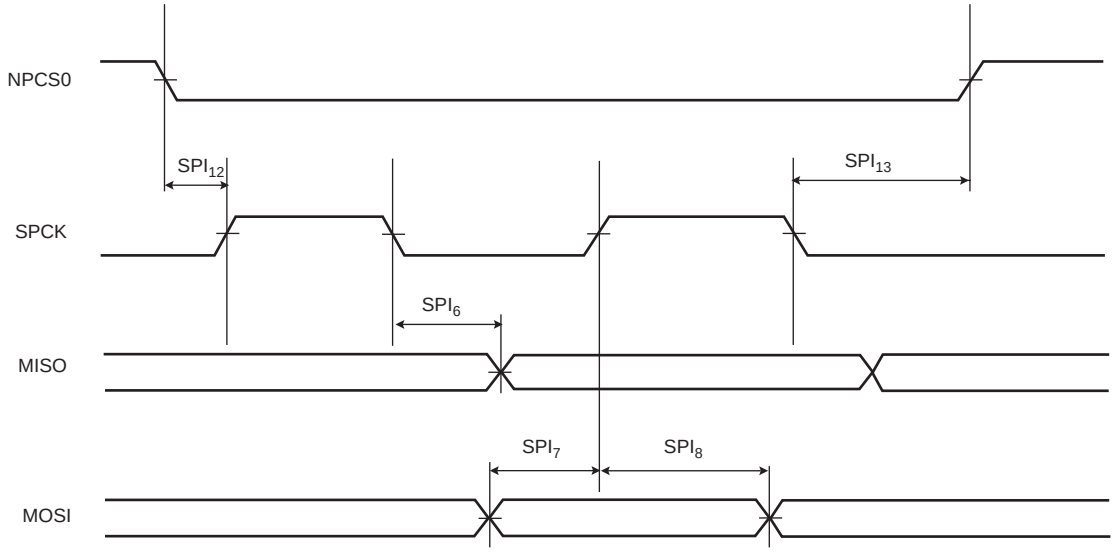


Figure 54-12.SPI Slave Mode 1 and 2

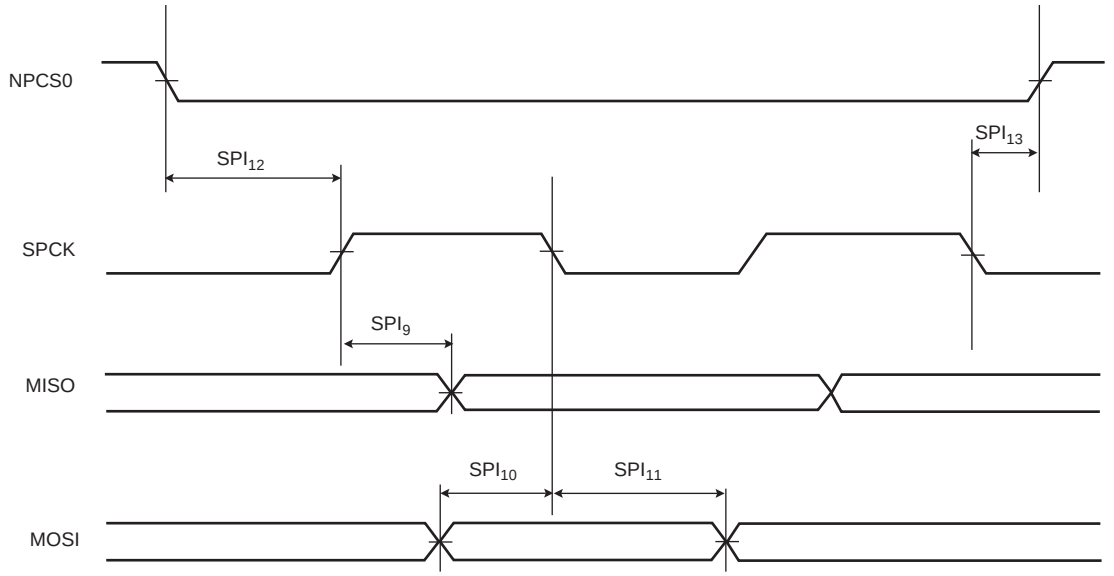


Figure 54-13.SPI Slave Mode - NPCS Timings

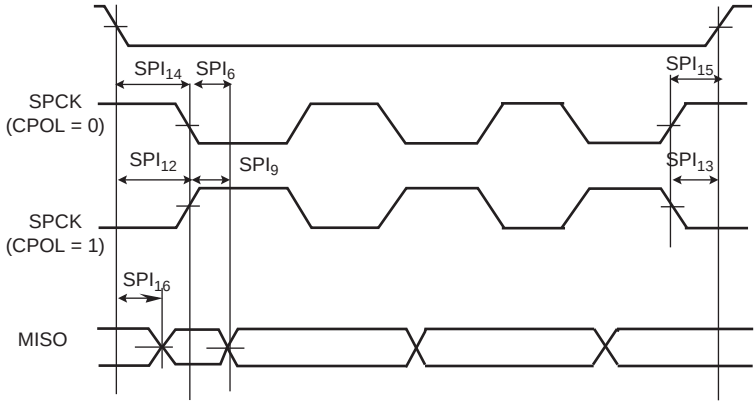


Table 54-43. SPI Timings with 3v3 peripheral supply

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	MISO Setup time before SPCK rises	—	8.7	—	ns
SPI_1	MISO Hold time after SPCK rises	—	5.9	—	ns
SPI_2	SPCK rising to MOSI	—	0.8(1)	3.7(1)	ns
SPI_3	MISO Setup time before SPCK falls	—	8.7	—	ns
SPI_4	MISO Hold time after SPCK falls	—	5.8	—	ns
SPI_5	SPCK falling to MOSI	—	0.9(1)	3.8(1)	ns
Slave Mode					
SPI_6	SPCK falling to MISO	—	5.8(1)	9.4(1)	ns
SPI_7	MOSI Setup time before SPCK rises	—	2	—	ns
SPI_8	MOSI Hold time after SPCK rises	—	0.3	—	ns
SPI_9	SPCK rising to MISO	—	5.8(1)	9.4(1)	ns
SPI_{10}	MOSI Setup time before SPCK falls	—	2	—	ns
SPI_{11}	MOSI Hold time after SPCK falls	—	0.3	—	ns
SPI_{12}	NPCS0 setup to SPCK rising	—	3.5	—	ns
SPI_{13}	NPCS0 hold after SPCK falling	—	19.5	—	ns
SPI_{14}	NPCS0 setup to SPCK falling	—	1.9	—	ns
SPI_{15}	NPCS0 hold after SPCK rising	—	19.8	—	ns
SPI_{16}	NPCS0 falling to MISO valid	—	—	8.8	ns

Notes: 1. For output signals, Min and Max access time must be extracted. The Min access time is the time between the SPCK rising or falling edge and the signal change. The Max access timing is the time between the SPCK rising or falling edge and the signal stabilizes. Figure 54-13 illustrates Min and Max accesses for SPI2. The same applies for SPI5, SPI6, SPI9.

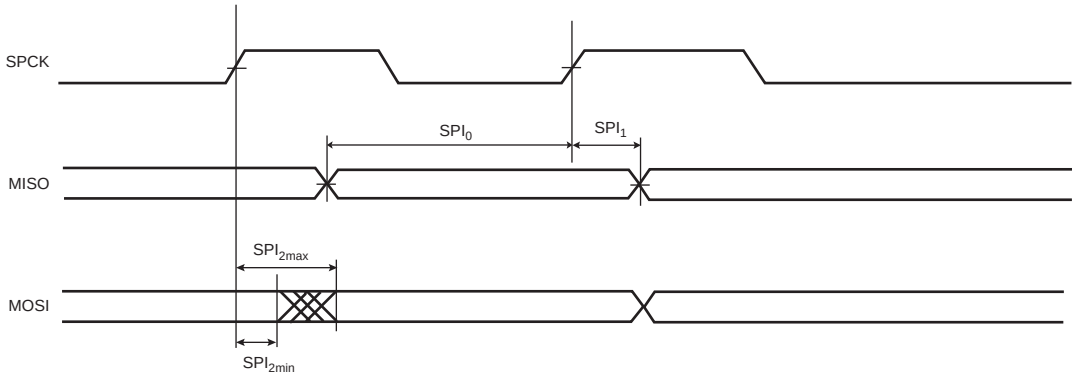
Table 54-44. SPI Timings with 1v8 Peripheral Supply

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	MISO Setup time before SPCK rises	—	10.9	—	ns
SPI_1	MISO Hold time after SPCK rises	—	0	—	ns
SPI_2	SPCK rising to MOSI	—	0.8(1)	3.5(1)	ns
SPI_3	MISO Setup time before SPCK falls	—	10.4	—	ns
SPI_4	MISO Hold time after SPCK falls	—	0	—	ns
SPI_5	SPCK falling to MOSI	—	1.2(1)	3.2(1)	ns
Slave Mode					
SPI_6	SPCK falling to MISO	—	3.2(1)	9.4(1)	ns
SPI_7	MOSI Setup time before SPCK rises	—	1.7	—	ns

Table 54-44. SPI Timings with 1v8 Peripheral Supply (Continued)

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_8	MOSI Hold time after SPCK rises	—	1	—	ns
SPI_9	SPCK rising to MISO	—	2.9(1)	10.9(1)	ns
SPI_{10}	MOSI Setup time before SPCK falls	—	1.7	—	ns
SPI_{11}	MOSI Hold time after SPCK falls	—	1	—	ns
SPI_{12}	NPCS0 setup to SPCK rising	—	2.1	—	ns
SPI_{13}	NPCS0 hold after SPCK falling	—	19	—	ns
SPI_{14}	NPCS0 setup to SPCK falling	—	3.2	—	ns
SPI_{15}	NPCS0 hold after SPCK rising	—	19.6	—	ns
SPI_{16}	NPCS0 falling to MISO valid	—	—	10.6	ns

Figure 54-14.Min and Max access time for SPI output signal



54.15 MPDDRC Timings

54.15.1 Board Design Constraints

As SAMA5D3 series embeds impedance calibrated pads, there are no capacitive constraints on DDR signals. However, a board must be designed and equipped in order to respect DQS setup times, including propagation time and intrinsic delay in SDRAM device. In all cases, line length to memory device must not exceed 5 cm.

54.15.2 DDR2-SDRAM

Table 54-45. System Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
t_{DDRCK}	DDRCK Cycle time	VDDCORE[1.08V,1.32V] T_A = 85°C	7.5	8.0	ns
		VDDCORE[1.2V,1.32V], VDDIODDR[1.75V,1.9V], T_A = 85°C	6.0	8.0	ns
t_{SDQS}	DQS setup time	VDDCORE[1.08V,1.32V] T_A = 85°C	—	0.6	ns
		VDDCORE[1.2V,1.32V], VDDIODDR[1.75V,1.9V], T_A = 85°C	—	0.7	ns

54.15.3 LPDDR1-SDRAM

Table 54-46. System Clock Waveform Parameters

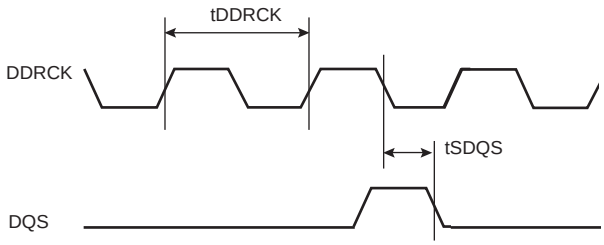
Symbol	Parameter	Conditions	Min	Max	Unit
t_{DDRCK}	DDRCK Cycle time	VDDCORE[1.08V,1.32V] T_A = 85°C	7.5	—	ns
t_{SDQS}	DQS setup time	VDDCORE[1.08V,1.32V] T_A = 85°C	—	7.3	ns

54.15.4 LPDDR2-SDRAM

Table 54-47. System Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
t_{DDRCK}	DDRCK Cycle time	VDDCORE[1.08V,1.32V] T_A = 85°C	7.5	—	ns
t_{SDQS}	DQS setup time	VDDCORE[1.08V,1.32V] T_A = 85°C	—	6.2	ns

Figure 54-15.DQS Timings



54.16 SSC Timings

54.16.1 Timing conditions

Timings assuming a capacitance load are given in Table 54-48.

Table 54-48. Capacitance Load

Supply	Corner	
	Max	Min
3.3V	30 pF	30 pF
1.8V	20 pF	20 pF

These values may be product dependant and should be confirmed by the specification.

54.16.2 Timing Extraction

Figure 54-16.SSC Transmitter, TK and TF in Output

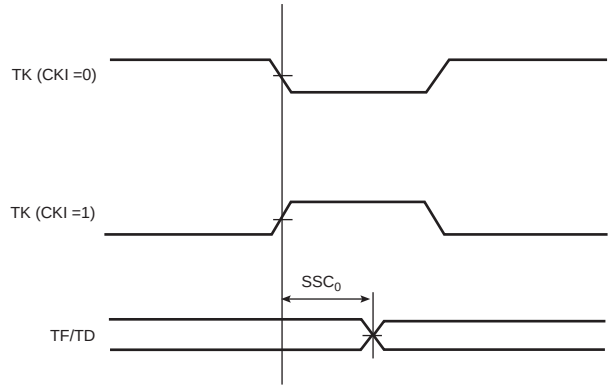


Figure 54-17.SSC Transmitter, TK in Input and TF in Output

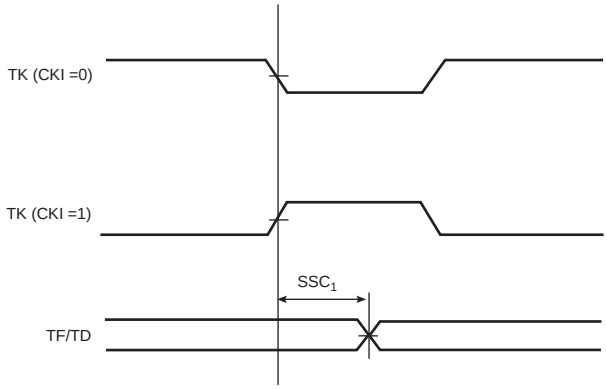


Figure 54-18.SSC Transmitter, TK in Output and TF in Input

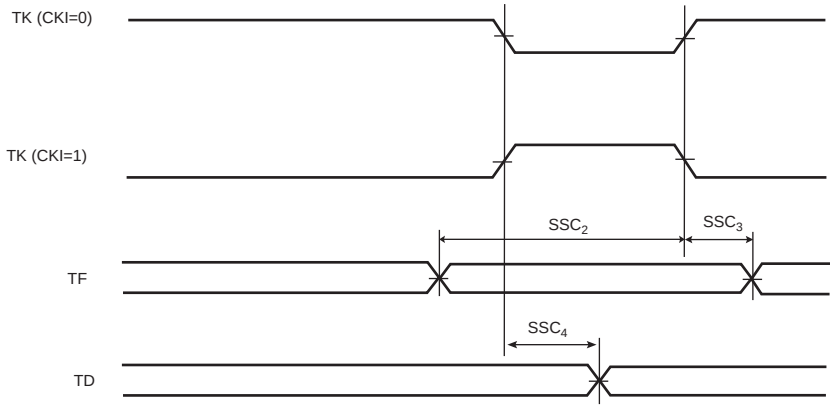


Figure 54-19.SSC Transmitter, TK and TF in Input

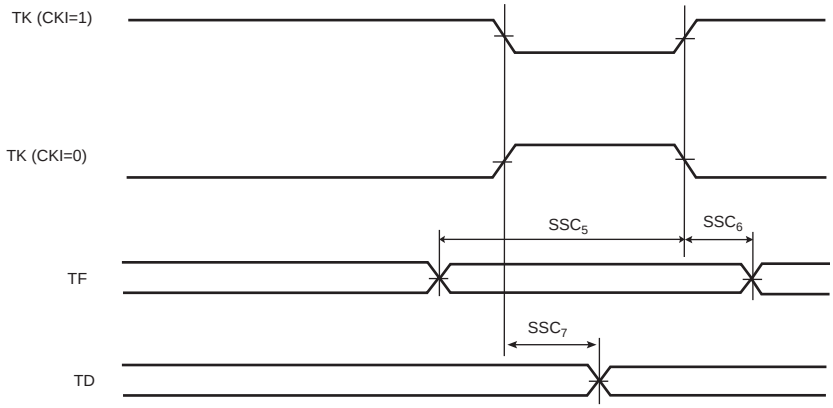


Figure 54-20.SSC Receiver RK and RF in Input

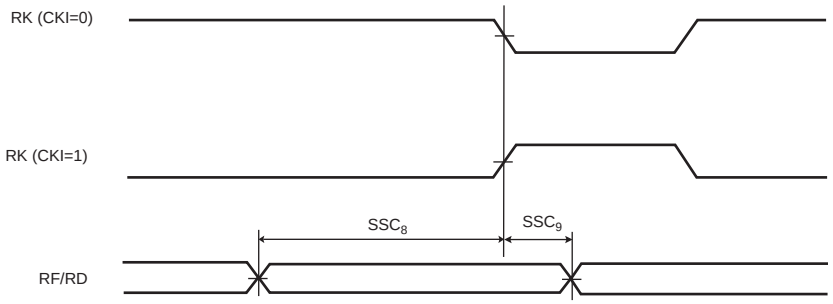


Figure 54-21.SSC Receiver, RK in Input and RF in Output

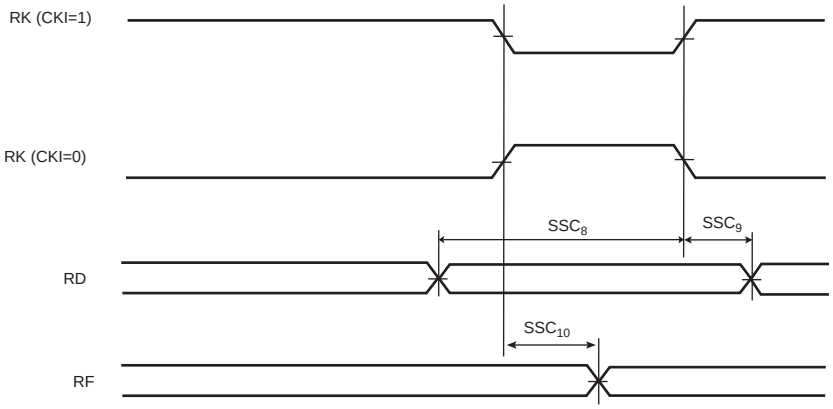


Figure 54-22.SSC Receiver, RK and RF in Output

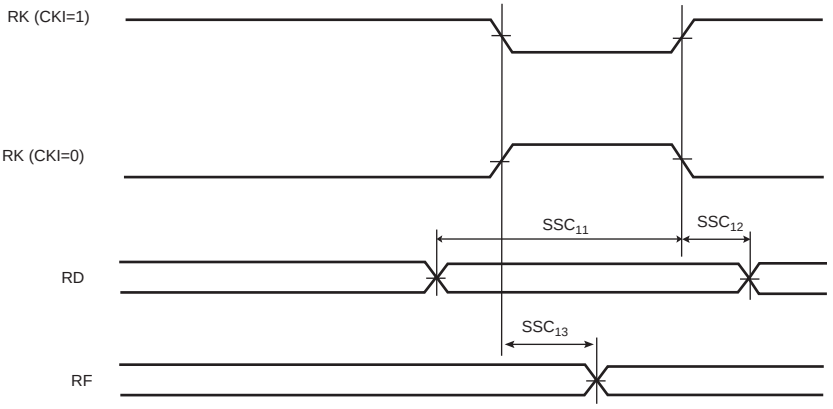


Figure 54-23.SSC Receiver, RK in Output and RF in Input

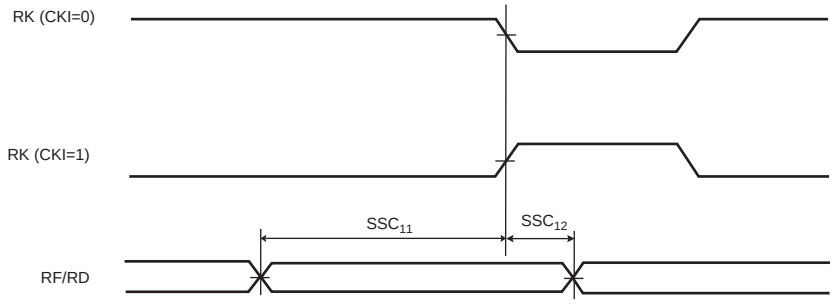


Table 54-49. SSC Timings with 3.3V Peripheral supply

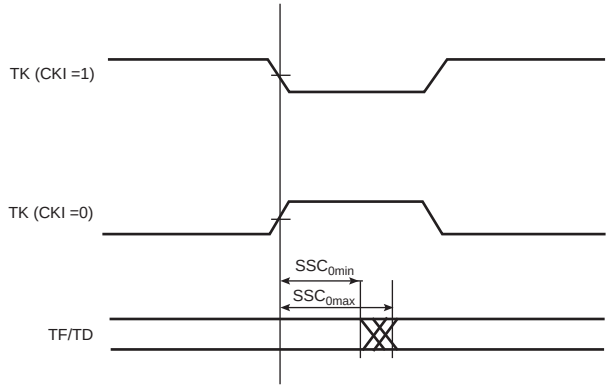
Symbol	Parameter	Conditions	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)	—	1.2(2)	3.8 (2)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	—	2.3 (2)	10.6 (2)	ns
SSC_2	TF setup time before TK edge (TK output)	—	7.7	—	ns
SSC_3	TF hold time after TK edge (TK output)	—	7.3	—	ns
SSC_4(1)	TK edge to TF/TD (TK output, TF input)	—	1.2 (+2*t_{CPMCK})(1)(2)	2.1(+2*t_{CPMCK})(1)(2)	ns
SSC_5	TF setup time before TK edge (TK input)	—	0	—	ns
SSC_6	TF hold time after TK edge (TK input)	—	t_{CPMCK}	—	ns
SSC_7(1)	TK edge to TF/TD (TK input, TF input)	—	2.4 (+3*t_{CPMCK})(1)(2)	8.8 (+3*t_{CPMCK})(1)(2)	ns
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)	—	0	—	ns
SSC_9	RF/RD hold time after RK edge (RK input)	—	t_{CPMCK}	—	ns
SSC_{10}	RK edge to RF (RK input)	—	2.5(2)	8.8(2)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	—	7.6 - t_{CPMCK}	—	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	—	t_{CPMCK} - 2.2	—	ns
SSC_{13}	RK edge to RF (RK output)	—	0.7(2)	2.2(2)	ns

Notes:
1. Timings SSC4 and SSC7 depend on the start condition. When STTDLY = 0 (Receive start delay) and START = 4, or 5 or 7 (Receive Start Selection), two Periods of the MCK must be added to timings.
2. For output signals (TF, TD, RF), Min and Max access times are defined. The Min access time is the time between the TK (or RK) edge and the signal change. The Max access timing is the time between the TK edge and the signal stabilization. Figure 54-24 illustrates Min and Max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

Table 54-50. SSC Timing with 1V8 Peripheral supply

Symbol	Parameter	Conditions	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)	—	1.2(2)	3.8 (2)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	—	2.3 (2)	10.6 (2)	ns
SSC_2	TF setup time before TK edge (TK output)	—	7.7	—	ns
SSC_3	TF hold time after TK edge (TK output)	—	7.3	—	ns
SSC_4(1)	TK edge to TF/TD (TK output, TF input)	—	1.2 (+2*t_{CPMCK})(1)(2)	2.1(+2*t_{CPMCK})(1)(2)	ns
SSC_5	TF setup time before TK edge (TK input)	—	0	—	ns
SSC_6	TF hold time after TK edge (TK input)	—	t_{CPMCK}	—	ns
SSC_7(1)	TK edge to TF/TD (TK input, TF input)	—	2.4 (+3*t_{CPMCK})(1)(2)	8.8 (+3*t_{CPMCK})(1)(2)	ns
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)	—	0	—	ns
SSC_9	RF/RD hold time after RK edge (RK input)	—	t_{CPMCK}	—	ns
SSC_{10}	RK edge to RF (RK input)	—	2.5(2)	8.8(2)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	—	7.6 - t_{CPMCK}	—	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	—	t_{CPMCK} - 2.2	—	ns
SSC_{13}	RK edge to RF (RK output)	—	0.7(2)	2.2(2)	ns

Figure 54-24.Min and Max Access Time of Output Signals



54.17 ISI Timings

54.17.1 Timing conditions

Timings assuming capacitance loads are given in Table 54-51.

Table 54-51. Capacitance Load

Supply	Corner	
	Max	Min
3.3V	30 pF	30 pF
1.8V	20 pF	20 pF

These values may be product dependant and should be confirmed by the specification.

54.17.2 Timing Extraction

Figure 54-25.ISI Timing Diagram

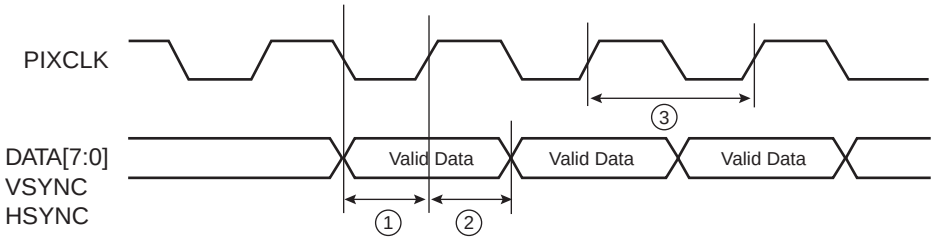


Table 54-52. ISI Timings with Peripheral Supply 3.3V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	3.1	—	ns
ISI_2	DATA/VSYNC/HSYNC hold time	1	—	ns
ISI_3	PIXCLK frequency	—	80	MHz

Table 54-53. ISI Timings with Peripheral Supply 1.8V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	3.4	—	ns
ISI_2	DATA/VSYNC/HSYNC hold time	1.2	—	ns
ISI_3	PIXCLK frequency	—	80	MHz

54.18 MCI Timings

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.

54.19 EMAC/GMAC Timings

54.19.1 Timing conditions

Timings assuming a capacitance load on data and clock are given in Table 54-54.

Table 54-54. Capacitance Load on Data, Clock Pads

Supply	Corner	
	Max	Min
3.3V	20 pF	0 pF
1.8V	20 pF	0 pF

These values may be product dependant and should be confirmed by the specification.

54.19.2 Timing constraints

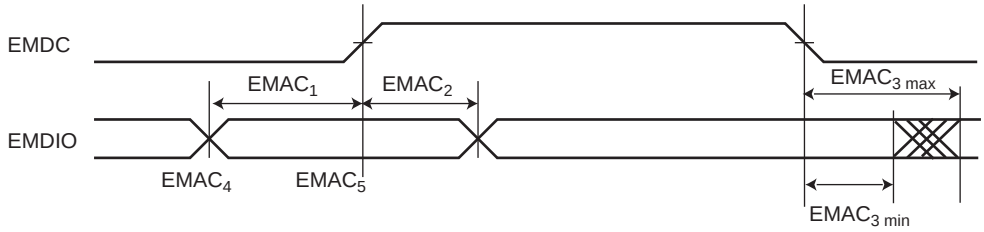
To satisfy the timings of standard given in Table 54-55 and Table 54-56, the Ethernet controller (EMAC) must be constrained in MAX corners.

Table 54-55. EMAC Signals Relative to EMDC

Symbol	Parameter	Min	Max	Unit
$EMAC_1$	Setup for EMDIO from EMDC rising	10	—	ns
$EMAC_2$	Hold for EMDIO from EMDC rising	10	—	ns
$EMAC_3$	EMDIO toggling from EMDC rising	0(1)	300(1)	ns

Notes: 1. For EMAC output signals, Min and Max access time are defined. The Min access time is the time between the EDMC rising edge and the signal change. The Max access timing is the time between the EDMC rising edge and the signal stabilizes. Figure 54-26 illustrates Min and Max accesses for EMAC3.

Figure 54-26.Min and Max access time of EMAC output signals



54.19.2.1 MII Mode

Table 54-56. EMAC MII Specific Signals

Symbol	Parameter	Min	Max	Unit
$EMAC_4$	Setup for ECOL from ETXCK rising	10	—	ns
$EMAC_5$	Hold for ECOL from ETXCK rising	10	—	ns
$EMAC_6$	Setup for ECRS from ETXCK rising	10	—	ns
$EMAC_7$	Hold for ECRS from ETXCK rising	10	—	ns
$EMAC_8$	ETXER toggling from ETXCK rising	10(1)	25(1)	ns
$EMAC_9$	ETXEN toggling from ETXCK rising	10(1)	25(1)	ns
$EMAC_{10}$	ETX toggling from ETXCK rising	10(1)	25(1)	ns
$EMAC_{11}$	Setup for ERX from ERXCK	10	—	ns
$EMAC_{12}$	Hold for ERX from ERXCK	10	—	ns
$EMAC_{13}$	Setup for ERXER from ERXCK	10	—	ns
$EMAC_{14}$	Hold for ERXER from ERXCK	10	—	ns
$EMAC_{15}$	Setup for ERXDV from ERXCK	10	—	ns
$EMAC_{16}$	Hold for ERXDV from ERXCK	10	—	ns

Notes: 1. See Note (1) of Table 54-55.

Figure 54-27.EMAC MII Mode

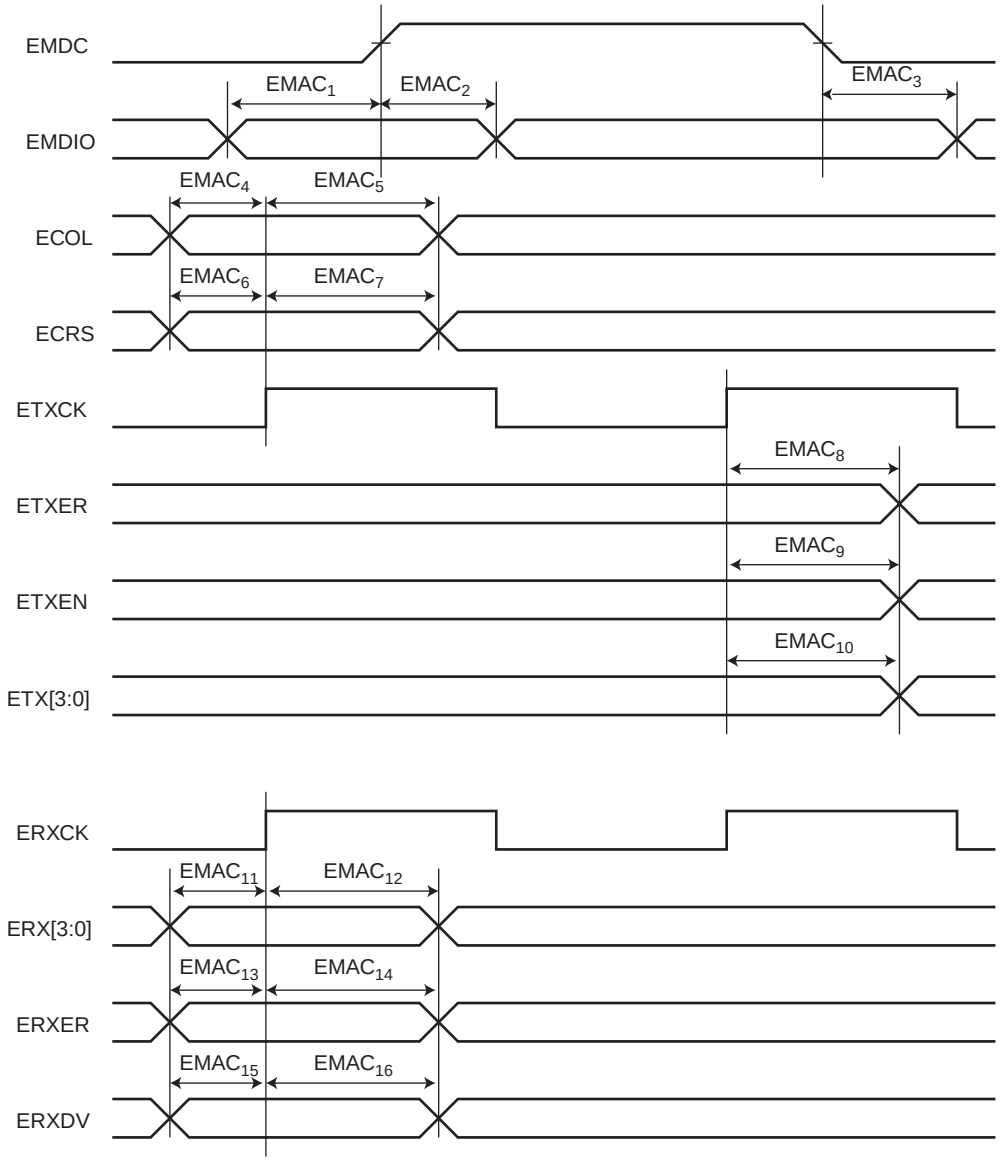
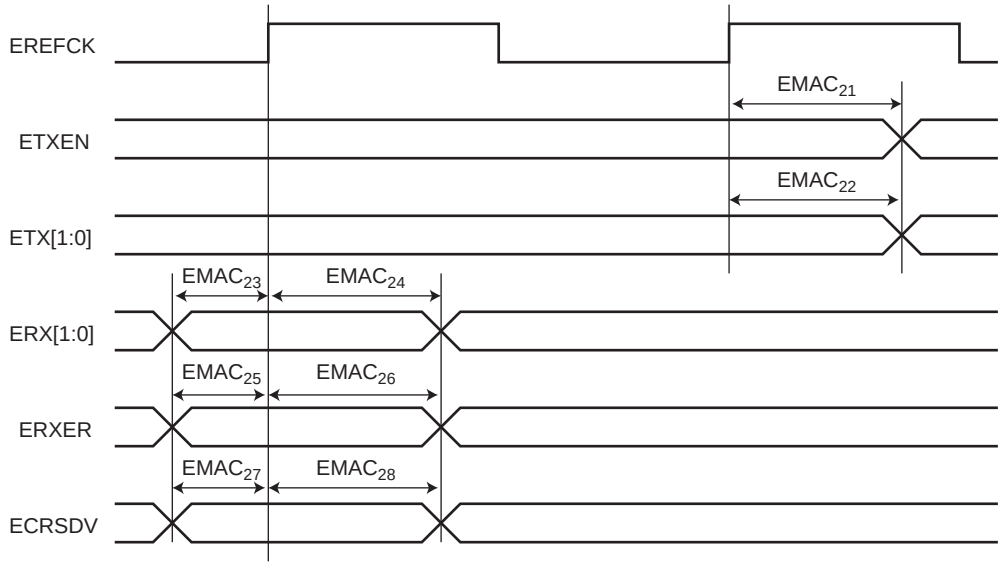


Table 54-57. RMII Mode

Symbol	Parameter	Min	Max	Unit
$EMAC_{21}$	ETXEN toggling from EREFCK rising	2[1]	16[1]	ns
$EMAC_{22}$	ETX toggling from EREFCK rising	2[1]	16[1]	ns
$EMAC_{23}$	Setup for ERX from EREFCK rising	4	—	ns
$EMAC_{24}$	Hold for ERX from EREFCK rising	2	—	ns
$EMAC_{25}$	Setup for ERXER from EREFCK rising	4	—	ns
$EMAC_{26}$	Hold for ERXER from EREFCK rising	2	—	ns
$EMAC_{27}$	Setup for ECRSDV from EREFCK rising	4	—	ns
$EMAC_{28}$	Hold for ECRSDV from EREFCK rising	2	—	ns

Notes: 1. See Note (1) of Table 54-55.

Figure 54-28.EMAC RMII Timings



54.20 UART in SPI Mode Timings

54.20.1 Timing conditions

Timings assuming a capacitance load on MISO, SPCK and MOSI are given in Table 54-58.

Table 54-58. Capacitance Load for MISO, SPCK and MOSI (product dependent)

Supply	Corner	
	Max	Min
3.3V	40 pF	40 pF
1.8V	20 pF	40 pF

These values may be product dependant and should be confirmed by the specification.

54.20.2 Timing Extraction

Figure 54-29.UART SPI Master mode

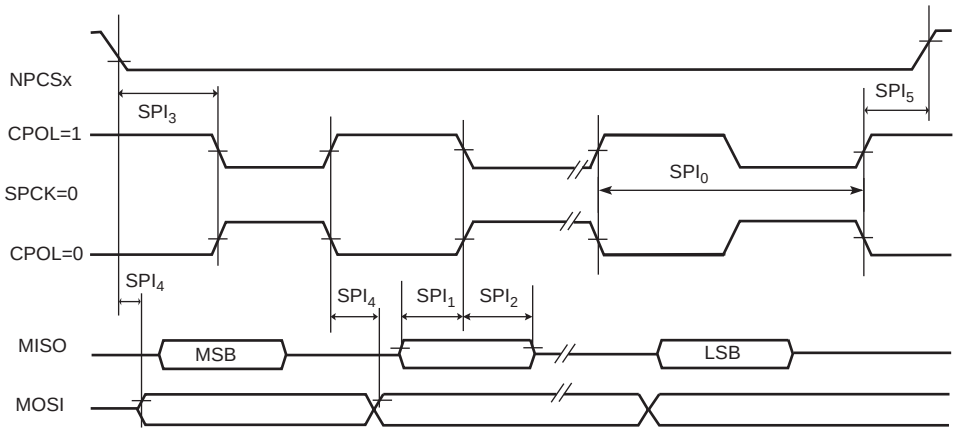


Figure 54-30.UART SPI Slave mode

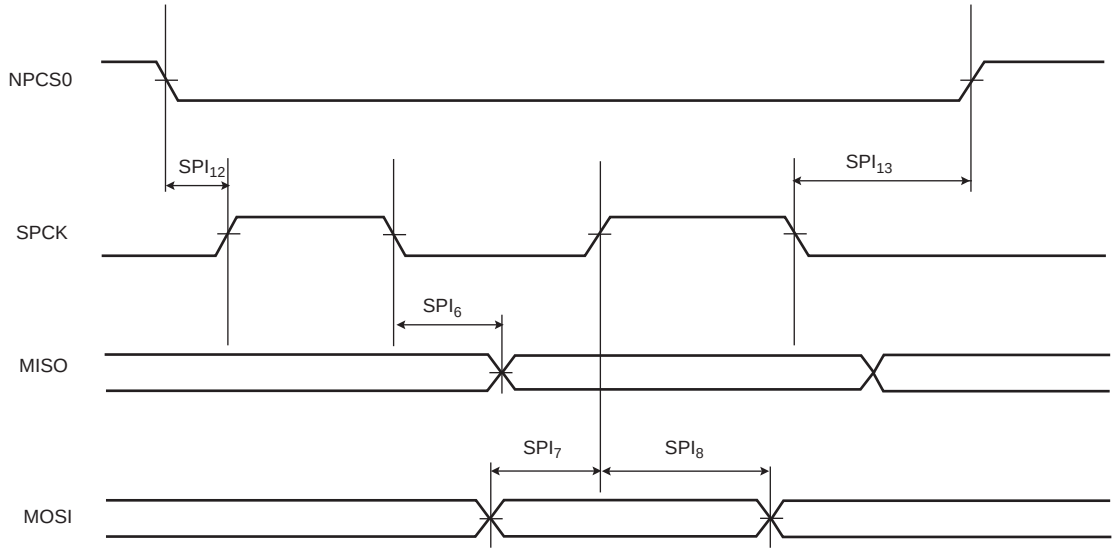


Figure 54-31.SPI Slave mode - NPCS timings

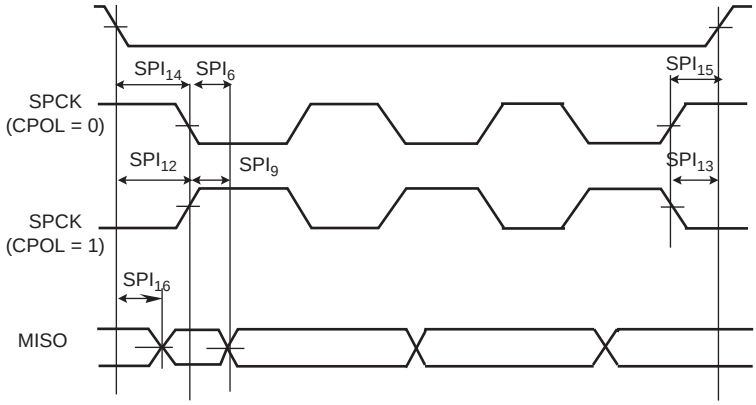


Table 54-59. UART SPI Timings 3.3V Peripheral Supply

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	SPCK Period	—	MCK/6	—	ns
SPI_1	Input Data Setup Time	—	2.4	—	ns
SPI_2	Input Data Hold Time	—	0.4	—	ns
SPI_3	Chip Select Active to Serial Clock	—	—	1.1	ns
SPI_4	Output Data Setup Time	—	—	6.5	ns
SPI_5	Serial Clock to Chip Select Inactive	—	—	0.6	ns
Slave Mode					
SPI_6	SPCK falling to MISO	—	2.2(1)	8.7(1)	ns
SPI_7	MOSI Setup time before SPCK rises	—	—	—	ns
SPI_8	MOSI Hold time after SPCK rises	—	0.1	—	ns
SPI_9	SPCK rising to MISO	—	5.6(1)	8(1)	ns
SPI_{10}	MOSI Setup time before SPCK falls	—	2	—	ns
SPI_{11}	MOSI Hold time after SPCK falls	—	0.5	—	ns
SPI_{12}	NPCS0 setup to SPCK rising	—	1	—	ns
SPI_{13}	NPCS0 hold after SPCK falling	—	0.9	—	ns
SPI_{14}	NPCS0 setup to SPCK falling	—	0.8	—	ns
SPI_{15}	NPCS0 hold after SPCK rising	—	0.4	—	ns
SPI_{16}	NPCS0 falling to MISO valid	—	—	7.6	ns

Notes: 1. For output signals, Min and Max access time must be extracted. The Min access time is the time between the SPCK rising or falling edge and the signal change. The Max access timing is the time between the SPCK rising or falling edge and the signal stabilizes. Figure 54-13 illustrates Min and Max accesses for SPI2. The same applies for SPI5, SPI6, SPI9.

54.21 Two-wire Interface Characteristics

Table 54-60 describes the requirements for devices connected to the Two-wire Serial Bus.

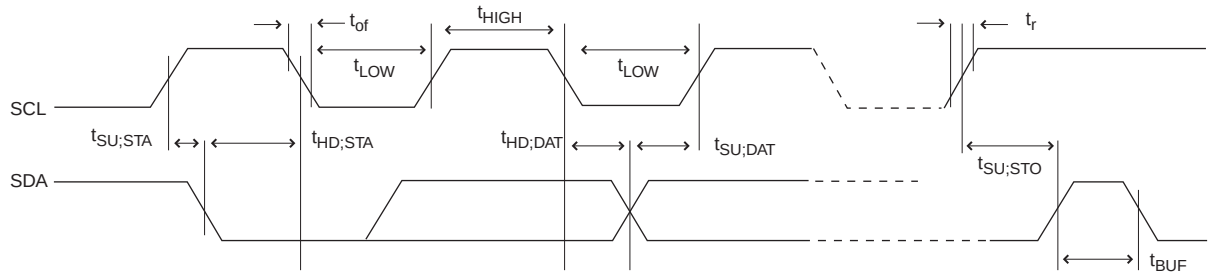
For timing symbols, please refer to Figure 54-32.

Table 54-60. Two-wire Serial Bus Requirements

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IL}	Input Low-voltage	—	-0.3	0.3 V_{VDDIO}	V
V_{IH}	Input High-voltage	—	0.7xV_{VDDIO}	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 54-32	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} ≤ 100 kHz	$\frac{V_{VDDIO} - 0,4V}{3mA}$	$\frac{1000ns}{C_b}$	Ω
		f_{TWCK} > 100 kHz	$\frac{V_{VDDIO} - 0,4V}{3mA}$	$\frac{300ns}{C_b}$	Ω
t_{LOW}	Low Period of the TWCK Clock	f_{TWCK} ≤ 100 kHz	(3)	—	µs
		f_{TWCK} > 100 kHz	(3)	—	µs
t_{HIGH}	High Period of the TWCK Clock	f_{TWCK} ≤ 100 kHz	(4)	—	µs
		f_{TWCK} > 100 kHz	(4)	—	µs
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{TWCK} ≤ 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} ≤ 100 kHz	t_{HIGH}	—	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	—	µs
$t_{HD;DAT}$	Data Hold Time	f_{TWCK} ≤ 100 kHz	0	3 x t_{CP_MCK} (5)	µs
		f_{TWCK} > 100 kHz	0	3 x t_{CP_MCK} (5)	µs
$t_{SU;DAT}$	Data Setup Time	f_{TWCK} ≤ 100 kHz	t_{LOW} - 3 x t_{CP_MCK} (5)	—	ns
		f_{TWCK} > 100 kHz	t_{LOW} - 3 x t_{CP_MCK} (5)	—	ns
$t_{SU;STO}$	Setup Time for STOP Condition	f_{TWCK} ≤ 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} ≤ 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: $t_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 4) \times t_{MCK}$
4. The TWCK high period is defined as follows: $t_{high} = ((\text{CHDIV} \times 2^{\text{CKDIV}}) + 4) \times t_{MCK}$
5. t_{CP_MCK} = MCK bus period.

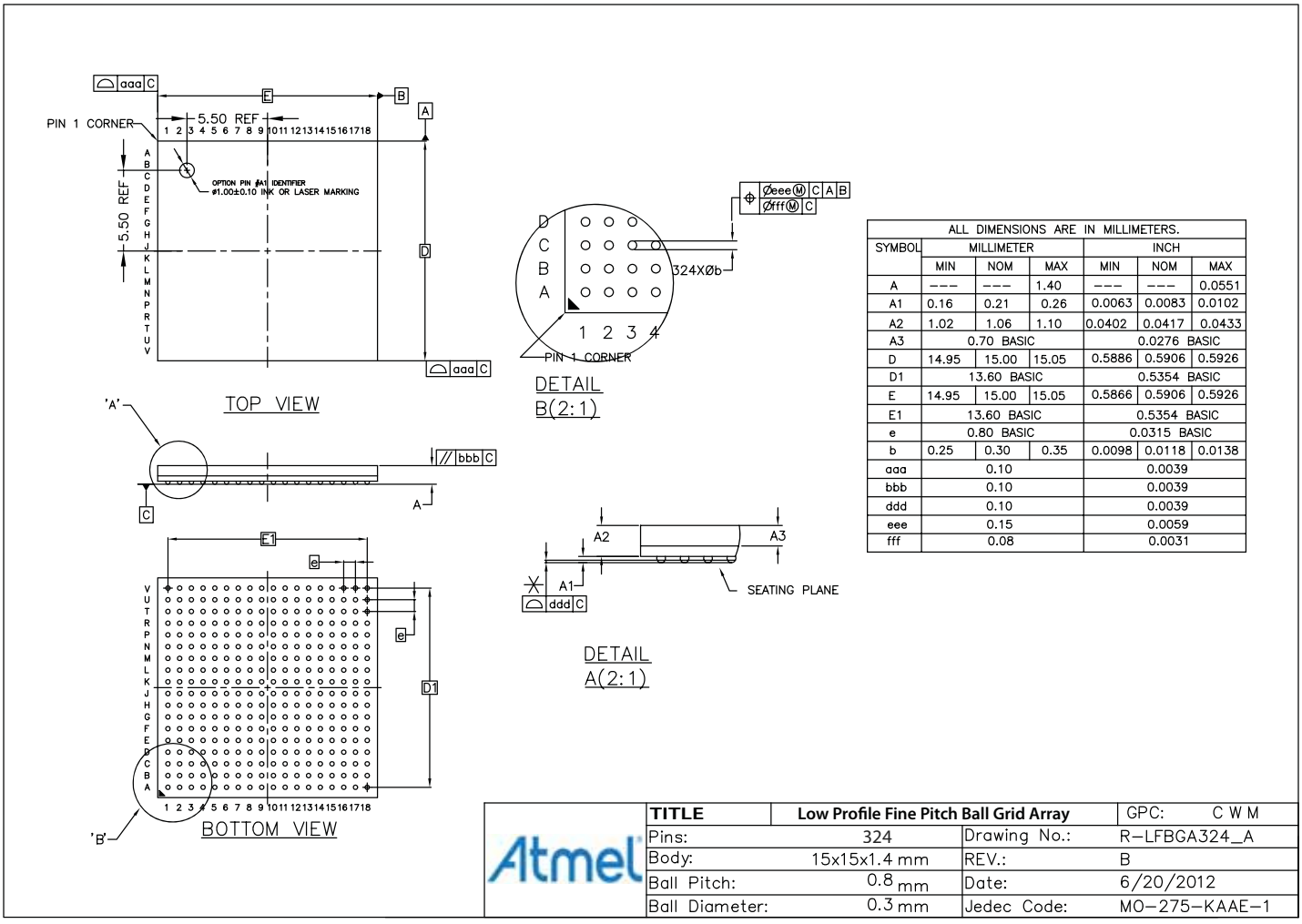
Figure 54-32.Two-wire Serial Bus Timing



55. Mechanical Characteristics

55.1 324-ball LFBGA Mechanical Characteristics

Figure 55-1. 324-ball LFBGA Package Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.						
SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.40	---	---	0.0551
A1	0.16	0.21	0.26	0.0063	0.0083	0.0102
A2	1.02	1.06	1.10	0.0402	0.0417	0.0433
A3	0.70 BASIC			0.0276 BASIC		
D	14.95	15.00	15.05	0.5886	0.5906	0.5926
D1	13.60 BASIC			0.5354 BASIC		
E	14.95	15.00	15.05	0.5866	0.5906	0.5926
E1	13.60 BASIC			0.5354 BASIC		
e	0.80 BASIC			0.0315 BASIC		
b	0.25	0.30	0.35	0.0098	0.0118	0.0138
aaa	0.10			0.0039		
bbb	0.10			0.0039		
ddd	0.10			0.0039		
eee	0.15			0.0059		
fff	0.08			0.0031		

TITLE	Low Profile Fine Pitch Ball Grid Array	GPC: C W M
Pins:	324	Drawing No.: R-LFBGA324_A
Body:	15x15x1.4 mm	REV.: B
Ball Pitch:	0.8 mm	Date: 6/20/2012
Ball Diameter:	0.3 mm	Jedec Code: MO-275-KAAE-1

Table 55-1. 324-ball LFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 55-2. Device and 324-ball LFBGA Package Maximum Weight

400	mg

Table 55-3. Package Reference

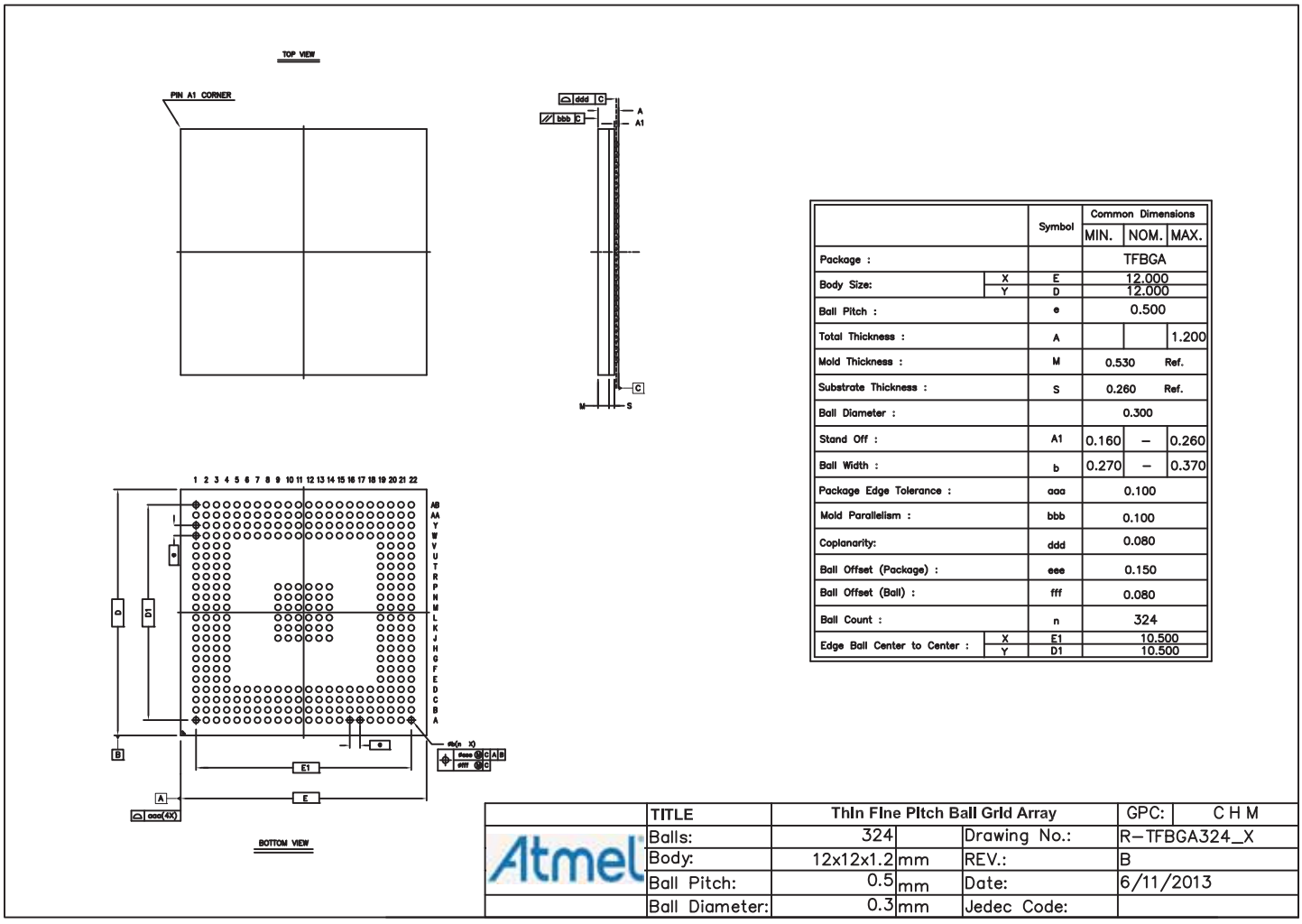
JEDEC Drawing Reference	MO-275-KAAE-1
JESD97 Classification	e8

Table 55-4. Package Information

Ball Land	0.350 mm ± 0.05
Nominal Ball Diameter	0.30 mm
Solder Mask Opening	0.275 mm ± 0.05
Solder Mask Definition	SMD
Solder	SAC105

55.2 324-ball TFBGA Mechanical Characteristics

Figure 55-2. 324-ball TFBGA Package Drawing



		Symbol	Common Dimensions		
			MIN.	NOM.	MAX.
Package :			TFBGA		
Body Size:	X	E		12.000	
	Y	D		12.000	
Ball Pitch :		e		0.500	
Total Thickness :		A			1.200
Mold Thickness :		M		0.530 Ref.	
Substrate Thickness :		S		0.260 Ref.	
Ball Diameter :				0.300	
Stand Off :		A1	0.160	–	0.260
Ball Width :		b	0.270	–	0.370
Package Edge Tolerance :		aaa		0.100	
Mold Parallelism :		bbb		0.100	
Coplanarity:		ddd		0.080	
Ball Offset (Package) :		eee		0.150	
Ball Offset (Ball) :		fff		0.080	
Ball Count :		n		324	
Edge Ball Center to Center :	X	E1		10.500	
	Y	D1		10.500	

	TITLE	Thin Fine Pitch Ball Grid Array			GPC: CHM
Atmel	Balls:	324		Drawing No.:	R–TFBGA324_X
	Body:	12x12x1.2	mm	REV.:	B
	Ball Pitch:	0.5	mm	Date:	6/11/2013
	Ball Diameter:	0.3	mm	Jedec Code:	

Table 55-5. 324-ball TFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 55-6. Device and 324-ball TFBGA Package Maximum Weight

280	mg

Table 55-7. Package Reference

JEDEC Drawing Reference	—
JESD97 Classification	—

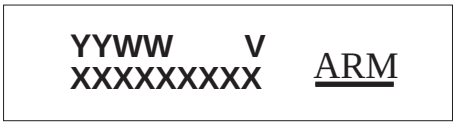
Table 55-8. Package Information

Ball Land	0.350 mm ± 0.05
Nominal Ball Diameter	0.30 mm
Solder Mask Opening	0.275 mm ± 0.05
Solder Mask Definition	SMD
Solder	LF35

55.3 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

56. SAMA5D3 Ordering Information

Table 56-1. SAMA5D3 Ordering Information

Ordering Code	Carrier Type	Packing	Package Type	Operating Temperature Range
ATSAMA5D31A-CU	LFBGA324	Tray	Green	Industrial -40°C to 85°C
ATSAMA5D31A-CUR		Tape and Reel		
ATSAMA5D31A-CFU	TFBGA324	Tray		
ATSAMA5D31A-CFUR		Tape and Reel		
ATSAMA5D33A-CU	LFBGA324	Tray		
ATSAMA5D33A-CUR		Tape and Reel		
ATSAMA5D34A-CU		Tray		
ATSAMA5D34A-CUR		Tape and Reel		
ATSAMA5D35A-CU		Tray		
ATSAMA5D35A-CUR		Tape and Reel		
ATSAMA5D36A-CU		Tray		
ATSAMA5D36A-CUR		Tape and Reel		
ATSAMA5D35A-CN		Tray		Industrial -40°C to 105°C
ATSAMA5D35A-CNR		Tape and Reel		
ATSAMA5D36A-CN		Tray		
ATSAMA5D36A-CNR		Tape and Reel		

57. SAMA5D3 Series Errata

The present errata provides information on Rev. A parts.

57.1 Standard Boot Strategies

57.1.1 Boot ROM: Boot on MCI1 is Not Working

Boot on MCI1 is not working.

Problem Fix/Workaround

Use a different boot media.

57.1.2 Boot ROM: NAND Flash Detection using ONFI Parameters does Not Work

During Nandflash initialization, the ONFI parameter detection may not work correctly. This can lead to incorrect configuration of ECC settings and to reading wrong data from the Nandflash memory, thus making it impossible to boot from this memory.

Problem Fix/Workaround

When programming the bootable program in the Nandflash, always use the header method, with any Nandflash memory, ONFI compliant or not.

57.2 Static Memory Controller (HSMC)

57.2.1 HSMC: ECC Value after ERASE is Not Correct

When ERASE page command is issued, the NAND Flash page resets to "ALL_ONE" pattern. The ECC value of ALL_ONE is a fixed but non-zero pattern leading to error detection after page erase.

Problem Fix/Workaround

None.

57.3 LCD Controller (LCDC)

57.3.1 LCDC: LCDC PWM Is Not Usable with DIV_1

When PWMPS field is set to DIV_1 value, the pwm ouput is stuck.

It works fine for every other divider (DIV_2 up to DIV_64).

Problem Fix/Workaround

None.

57.4 PWM Controller (PWM)

57.4.1 PWM: Wrong Counter Event.

A synchronous channel (but not channel 0) can generate a wrong counter event, which does not match the alignment configuration of the channel 0.

The issue occurs if the alignment configuration of the channel is different from that of channel 0.

Problem Fix/Workaround

None.

57.5 Gigabit Ethernet MAC (GMAC)

57.5.1 GMAC: TX Packet Buffer DMA Lockup when Reading Used Status

In the following conditions:

- Frame N-1 has completed the transmission on GMII and its post-transmit status has been passed from the MAC to DMA.
- The descriptor write back for frame N-1 has been requested on AHB and is awaiting completion.
- Frame N has completed the transmission on GMII and its post-transmit status has been passed from the MAC to DMA.
- A “used” bit has been read from the buffer descriptor queue after the descriptors associated with frame N. This means a buffer descriptor with bit [31] set in word 1 of the descriptor pair.
- Status associated with this used bit is passed from MAC to DMA on the exact AHB clock cycle that the writeback request for frame N-1 completes (either; BREADY high in the case of HREADY or high in the data phase in the case of AHB).

When this precise sequence occurs, the AHB write back request for frame N never actually occurs, but the TX DMA state machine waits in the DMA_MANWR state, awaiting the AHB response for the write. This response never happens since the request was never sent.

Problem Fix/Workaround

1. Avoid the issue by sending only one frame at a time.

To send only a single frame at a time, each frame queued in the descriptor buffers must be followed by a buffer descriptor with the used bit set. GEM will send the frame and then halt and update status. Once frame status indicates that the frame has completed, then a new frame may be queued and transmit DMA restarted.

2. Avoid issue by changing used bit status to exhausted mid-frame status.

For this workaround the buffer descriptors with used bit set, that mark the end of the queues are replaced with two zero length descriptors with particular bits set or clear. The effect is an exhausted mid-frame status instead of the usual used bit read status.

This requires changes to SW for adding new frames to queues.

For this discussion, assume that the last descriptor associated with the last valid frame queued is descriptor N, which is at address offset X from the queue base pointer.

Currently the end of the queue is marked with descriptor N+1 with its used bit set as follows:

- descriptor N:

  X+0x00: FC800000

  X+0x04: 0000803C - Last frame queued
- descriptor N+1:

  X+0x08: xxxxxxxx

  +0x0C: 8xxxxxxx - Used bit set (‘x’ is a don’t care value)

For the workaround buffer descriptor N+1 and also N+2 should be replaced with the following:

- descriptor N:

  X+0x00: FC800000

  X+0x04: 0000803C - Last frame queued
- descriptor N+1:

  X+0x08: 00000000

  X+0x0C: 00000000
- descriptor N+2:

  X+0x10: 00000000

X+0x14: 80008000 - last buffer and used bit set

57.5.2 GMAC: TX Packet Buffer DMA Writeback Status Overflow

The issue occurs due to an overflow of the pipeline for the post-transmit status that is returned from the read side of the packet buffer memory back to the write side.

Problem Fix/Workaround

As there are four pipeline stages, having only 3 outstanding frames at any one time will avoid the issue.

57.6 Watchdog Timer (WDT)

57.6.1 WDT: WDRPROC = 1 leads to Unpredictible Behavior

A Processor Reset not generated in conjunction with PERRST leads to Unpredictible Behavior. That is the reason why the Watchdog must not activate the processor reset only. WDRPROC bit must not be set in WDT_MR.

Problem Fix/Workaround

Use only WDRPROC = 0, leading to activate all resets on Watchdog fault if WDRRSTEN is set.

Revision History

In the tables that follow, the most recent version of the document appears first.

“rfo” indicates changes requested during document review and approval loop.

Doc. Rev. 11121C	Comments	Change Request Ref.
	“Description”	
	- minor editorial changes	rfo
	- added SAMA5D36 device reference (“four SAMA5D3 devices” --> “five SAMA5D3 devices”)	8822
	Section 1. “Features”	
	Minor editorial and formatting changes	rfo
	Bullet “Memories”:	
	- replaced “Boot on NAND Flash” with “Boot on 8-bit NAND Flash” in first subbullet	rfo
	- replaced “DDR2/LPDDR/LPDDR2” with “DDR2/LPDDR/LPDDR2 with datapath scrambling” in third subbullet	rfo
	- replaced “Controller with SLC/MLC NAND Support” with “Controller with datapath scrambling and SLC/MLC NAND Support” in fourth subbullet	rfo
	Bullet “System running up to 166 MHz”: deleted “Power-on Reset Cells” from first subbullet	rfo
	Bullet “Peripherals: deleted subbullet “Write Protected Registers”	rfo
	Inserted new bullet “Safety” (includes six subfeatures)	rfo
	Bullet “Security”: deleted subbullet “SHA: Supports Secure Hash Algorithm (SHA1, SHA224, SHA256, SHA384, SHA512)”	rfo
	Introduced 324-ball TFBGA package	9035
	Table 1-1 “SAMA5D3 Device Differences”: changed table title (was “SAMA5D3 Devices”); reorganized table content; added SAMA5D36 device	rfo 8822
	Section 2. “Block Diagram”: added a note on peripheral bridges APB0/APB1 and updated Figure 2-1 “SAMA5D3 Block Diagram”	8956
	Section 3. “Signal Description”	
	Table 3-1 “Signal Description List”:	rfo
	- removed empty columns “Frequency” and “Comments”	
	- in DDR2/LPDDR Controller signals, replaced instances of “I/O/-PD” and “I/O- PD” with “I/O”	
	- minor punctuation and formatting changes	
	Section 4. “Package and Pinout”	
	Editorial changes and formatting changes; inserted dashes in all empty table cells	rfo
	Added Section 4.3 “324-ball TFBGA Package (12 x 12 x 1.2 mm, pitch 0.5 mm)”	9035
	Added Section 4.4 “324-ball TFBGA Package Pinout”	9035
	Table 4-1 “SAMA5D3 Pinout for 324-ball LFBGA Package”: inserted “Reference voltage” as “I/O Type” for pin C13	rfo
	Table 4-3 “SAMA5 I/O Types Description”: added column headers	rfo
	Table 4-4 “SAMA5 I/O Type Assignation and Frequency”: changed column header “I/O Frequency (MHz)” to read “Max I/O Frequency (MHz)”	rfo

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 5. “Power Considerations”	
	Minor punctuation and formatting changes	rfo
	Table 5-1 “SAMA5D3 Power supplies”: updated “Powers” information for VDDOSC	8818/rfo
	Section 6. “Memories”	
	Minor editorial changes	rfo
	Figure 6-1 “Memory Mapping”: corrected register name (was SCKCR; is SCKC_CR)	rfo
	Section 7. “Real-time Event Management”: minor formatting changes	rfo
	Section 8. “System Controller”	
	Minor punctuation and formatting changes	rfo
	Figure 8-1 “SAMA5D3 System Controller Block Diagram”: corrected register name (was SCKCR; is SCKC_CR)	rfo
	Table 8-1 “Chip Identification of SAMA5D3 Devices”: added SAMA5D36 device	8822
	Section 9. “Peripherals”	
	Minor editorial and formatting changes	rfo
	Section 9.3 “Peripheral Signal Multiplexing on I/O Lines”: updated content to reference additional multiplexing table for TFBGA324	rfo
	Section 10. “ARM Cortex-A5 Processor”	
	Minor formatting and editorial changes throughout	
	Amended Section 10.1.1 “Power Management” with bullet list of main levels of power management	8978
	Table 10-4 “CP15 Registers”: for register No. 12, replaced empty row with row “Interrupts management / Read/Write”	rfo
	Section 10.5.2 “Memory Management System”: deleted content “When the processor is put into dormant mode ... in the Cortex-A5 Technical Reference Manual, DDI0433” from end of section	rfo
	Section 11. “SAMA5D3 Series Debug and Test”	
	Removed RTCK references in Figure 11-1 “Debug and Test Block Diagram”, Table 11-1 “Debug and Test Pin List”, and Section 11.6.3 “JTAG Signal Description” (removed the last paragraph)	rfo
	Section 12. “Standard Boot Strategies”	
	Minor editorial and formatting changes	rfo
	Updated note “JTAG access is disabled...”	rfo
	Table 12-1 Values of the Boot Sequence Configuration Register: replaced column header “NAND Flash” with “8-bit NAND Flash”	rfo
	Section 12.3.4.1 “NAND Flash Boot: NAND Flash Detection”: added note “Booting on 16-bit NAND Flash is not possible; only 8-bit NAND Flash memories are supported” below bullet “Through the ONFI parameters...” and below “NAND Flash Specific Header Detection” bitmap	rfo
	Section 13. “Boot Sequence Controller (BSC)”	
	Minor editorial and formatting changes throughout	
	Section 13.4.1 “Boot Sequence Configuration Register”:	9006
	- in description of field BOOT, replaced instance of BOOTKEY with WPKEY	
	- changed name of field BOOTKEY to WPKEY and updated field description	
	Section 14. “AXI Bus Matrix (AXIMX)”: No changes	
	Section 15. “Bus Matrix (MATRIX)”: No changes	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 16. “Special Function Registers (SFR)”	
	Minor punctuation changes	
	Table 16-1 "Register Mapping": inserted reserved row for offsets 0x20–0x24	8539
	Section 17. “Advanced Interrupt Controller (AIC)”	
	Minor formatting changes throughout	
	Section 17.8.4.4 “Fast Interrupt Handlers”: replaced 6-step sequence with 5-step sequence (step 3 updated)	rfo
	Section 17.9.23 “AIC Write Protect Mode Register”: updated KEY field description	rfo
	Section 18. “Watchdog Timer (WDT)”	
	Section 18.1 “Description”: replaced “(slow clock at 32768 kHz)” with “(slow clock around 32 kHz)”	rfo
	Section 18.2 “Embedded Characteristics”: added “Watchdog Clock is independent from Processor Clock”	rfo
	Section 18.4 “Functional Description”: replaced “(with a typical Slow Clock of 32768 kHz)” with “(with a typical Slow Clock of 32.768 kHz)”	rfo
	Section 18.5.1 “Watchdog Timer Control Register”: updated KEY field description	8844
	Section 18.5.2 “Watchdog Timer Mode Register”:	rfo
	- added note “The first write access prevents any further modification of the value of this register, read accesses remain possible”	
	- added 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.”	
	Section 18.5.3 “Watchdog Timer Status Register”: deleted note “The WDD and WDV values ... of period earlier than expected.”	rfo
	Section 19. “Reset Controller (RSTC)”	
	Minor editorial and formatting changes throughout	rfo
	Section 19.4.4.1 “General Reset”: replaced “remain valid for Y cycles” with “remain valid for 3 cycles”	rfo
	Section 19.4.4.4 “Software Reset”: removed phrase “Except for Debug purposes,” from “PERRST” bullet	9017
	Section 19.5.1 “Reset Controller Control Register”: updated KEY field description	9043
	Section 19.5.2 “Reset Controller Status Register”, in ‘RSTTYP’ field configuration table:	
	- replaced column headers “RSTTYP Reset/Type/Comments” with “Value/Name/Description” and replaced binary configuration values with decimal values’	9043
	- updated reset type names in ‘RSTTYP’ field description:	rfo
	-- was “General Reset”; is “GENERAL_RST”	
	-- was “Wake Up Reset”; is “WKUP_RST”	
	-- was “Watchdog Reset”; is “WDT_RST”	
	-- was “Software Reset”; is “SOFT_RST”	
	-- was “User Reset”; is “USER_RST”	
	Section 19.5.3 “Reset Controller Mode Register”: updated KEY field description	9043
	Section 20. “Shutdown Controller (SHDWC)”	
	Table 20-2 Register Mapping: added “0x0000_0003” as a reset value of SHDW_MR	8828
	Section 20.7.1 “Shutdown Control Register”: updated KEY field description	8869
	Section 20.7.2 “Shutdown Mode Register”: updated WKMODE0 and CPTWK0 field descriptions	8432
	Section 20.7.3 “Shutdown Status Register”: updated WAKEUP0 field description	8432
	Section 21. “General-Purpose Backup Registers (GPBR)”: No changes	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 22. “Periodic Interval Timer (PIT)”: No changes	
	Section 23. “Real-time Clock (RTC)”	
	Section 23.2 “Embedded Characteristics”: added new bullet "Safety/security features” (includes “Valid Time and Date Programmation Check” as subbullet)	8544
	Updated Section 23.5.3 “Alarm”	8900/9027
	Added note in Section 23.6.5 “RTC Time Alarm Register” and Section 23.6.6 “RTC Calendar Alarm Register”	9027
	Section 24. “Slow Clock Controller (SCKC)”	
	Minor editorial changes throughout	rfo
	Replaced all instances of “Read-write” with “Read/Write”	rfo
	Section 24.3 “Block Diagram”: corrected referenced register name (was “Slow Clock Configuration Register”; is “Slow Clock Controller Configuration Register”)	rfo
	Table 24-1 “Register Mapping”: corrected register name (was “Slow Clock Configuration Register”; is “Slow Clock Controller Configuration Register”)	rfo
	Section 24.4.1 “Slow Clock Controller Configuration Register”: corrected register name (was “Slow Clock Configuration Register”)	rfo
	Section 25. “Fuse Controller (FUSE)” (relocated—was previously section 51)	
	Minor editorial changes throughout	rfo
	Reformatted Table 25-1 “FUSE Bit Mapping”	rfo
	Section 25.4.2 “Fuse Programming”:	rfo
	- in step 1, replaced “using the SWEL field” with “using the WSEL field”	
	- in step 5, replaced “Check the WS bit of FUSE_SRx” with “Check the WS bit of FUSE_IR”	
	Reformatted register description sections	rfo
	Section 25.5.1 “Fuse Control Register”: updated field descriptions	rfo
	“Clock Generator“ section is now a subsection of Section 26. “Power Management Controller (PMC)”	rfo

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 26. “Power Management Controller (PMC)”	
	Minor editorial and formatting changes throughout (including replacing instances of “Read-write” with “Read/Write”)	rfo
	Section 26.1.4 “Main Clock Selection”: removed “SAMA5D3” reference in first sentence	8983
	Updated Section 26.1.5.2 “12 MHz Fast RC Oscillator Clock Frequency Adjustment”	8852
	Updated Section 26.1.5.4 “Main Clock Oscillator Selection”	rfo
	Added Section 26.1.5.5 “Switching Main Clock between the Main RC Oscillator and Fast Crystal Oscillator”	9425
	Section 26.1.5.6 “Software Sequence to Detect the Presence of Fast Crystal”:	
	- updated register names (PLL_MCKR –> PMC_MCKR, PLL_SR –> PMC_SR)	rfo
	- added “during this sequence the Main RC oscillator must be kept enabled (MOSCRCEN = 1)” to sequence introduction	9245
	Updated Figure 26-8 “General Clock Block Diagram”	9022
	Updated Section 26.2.7 “DDR2/LPDDR/LPDDR2 Clock”	rfo
	Section 26.2.12 “Programming Sequence”: revised content (changed total number of steps—was 6; is 10)	rfo
	Added Section 26.2.14 “Register Write Protection”	rfo
	Table 26-3 “Register Mapping”:	rfo
	- changed PMC_PLLICPR access from “Write-only” to “Read/Write”	
	- changed reset value (was 0x0100_0100; is 0x0000_0000)	
	Section 26.2.15.8 “PMC Clock Generator Main Oscillator Register”:	
	- corrected order of field descriptions	rfo
	- updated KEY field description	8837
	Added note in Section 26.2.15.9 “PMC Clock Generator Main Clock Frequency Register”	8852
	Section 26.2.15.14 “PMC Programmable Clock Register”: appended name PMC_PCKx with “[x = 0..2]”	9071
	Section 26.2.15.17 “PMC Status Register”: replaced “of the main on-chip RC oscillator clock” with “of the fast crystal oscillator clock” in the description of fields CFDEV and CFDS	rfo
	Section 26.2.15.20 “PLL Charge Pump Current Register”:	rfo
	- changed access from “Write-only” to “Read/Write”	
	- replaced “Should be written to 0” with “Should be written to 3” in description of field IPLL_PLLA	
	Section 26.2.15.21 “PMC Write Protection Mode Register”:	
	- modified register name (was PMC Write Protect Mode Register)	rfo
	- updated WPKEY field description	8838
	- replaced list of protectable registers with cross-reference to section “Register Write Protection”	rfo
	Section 26.2.15.22 “PMC Write Protection Status Register”: modified register name (was PMC Write Protect Status Register) and updated description of field WPVSRC	rfo
	Section 26.2.15.26 “PMC Peripheral Control Register”:	9085
	- changed access from “Write-only” to “Read-Write”	
	- updated PID field description	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 27. “Parallel Input/Output Controller (PIO)”	
	Figure 27-3 “I/O Line Control Logic”: added pull-down resistor and registers	rfo
	Section 27.5.14 “Register Write Protection”: Changed section title and revised content	rfo
	Section 27.7.29 “PIO Slow Clock Divider Debouncing Register”: below bitmap, replaced name “DIVx” with “DIV”	rfo
	Section 27.7.46 “PIO Write Protection Mode Register”: modified register name and aligned bit descriptions; replaced list of protectable registers with cross-reference to Section 27.5.14 “Register Write Protection”	8522/rfo rfo
	Section 27.7.47 “PIO Write Protection Status Register”: modified register name and aligned bit descriptions; removed note	rfo
	Section 27.7.48 “PIO Schmitt Trigger Register”: updated SCHMITTx field description	rfo
	Section 28. “External Memories”: No changes	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 29. “Multi-port DDR-SDRAM Controller (MPDDRC)”	
	Formatting and editorial changes throughout	
	Updated Section 29.1 “Description”	rfo
	Section 29.2 “Embedded Characteristics”:	rfo
	- replaced “CAS Latency of 2, 3, 4, 5, 6 Supported” with “CAS Latency of 2, 3 Supported”	
	- replaced “OCD (Off-chip Driver) Mode Not Supported” with “OCD (Off-chip Driver) mode, ODT (On-die Termination) are not supported”	
	- added new bullet “Dynamic Scrambling with user key (no impact on bandwidth)”	9223
	Section 29.5.1 “DDR-SDRAM Controller Write Cycle”: replaced “fixed to 2/3/4/5 in case of DDR2-SDRAM” with “fixed to 2 in case of DDR2-SDRAM”	rfo
	Section 29.5.2 “DDR-SDRAM Controller Read Cycle”: updated latency range (was 2 to 6; is 2 to 3) and corresponding number of anticipated read accesses	rfo
	Section 29.5.4 “Multi-port Functionality”: replaced “set at 2 or 6 cycles (CAS latency)” with “set at 2 or 3 cycles (CAS latency)”	rfo
	Added new Section 29.6 “Scrambling/Unscrambling Function”	9223
	Section 29.8 “AHB Multi-port DDR-SDRAM Controller (MPDDRC) User Interface”: changed all instances of “Read-write” to “Read/Write”	rfo
	Table 29-25 “Register Mapping”: corrected the offset values of MPDDRC DLL Status MASTERx registers	8884
	Section 29.8.1 “MPDDRC Mode Register”: updated format of field description tables and replaced some descriptions with tables	8883
	Updated Section 29.8.3 “MPDDRC Configuration Register”	8883
	Section 29.8.4 “MPDDRC Timing Parameter 0 Register”: added a note on SDCK	rfo
	Section 29.8.5 “MPDDRC Timing Parameter 1 Register”:	
	- added a note on SDCK	rfo
	- updated size of field TRFC in bitmap (TRFC now spans bits 4:0)	9058
	- updated TRFC field description	rfo
	Section 29.8.6 “MPDDRC Timing Parameter 2 Register”: added a note on SDCK	rfo
	Section 29.8.7 “MPDDRC Low-power Register”: updated format of field description tables and replaced some descriptions with tables	8883
	Section 29.8.10 “MPDDRC Low-power DDR2 Timing Calibration Register”: added a note on SDCK	rfo
	Section 29.8.11 “MPDDRC I/O Calibration Register”:	
	- added a note on SDCK	rfo
	- updated RDIV field description	8883
	Section 29.8.15 “MPDDRC Memory Device Register”: updated format of field description tables and replaced some descriptions with tables	8883
	Section 29.8.17 “MPDDRC Write Protect Mode Register”:	8883
	- updated the register name from “MPDDRC Write Protect Control Register” (MPDDRC_WPCR) to “MPDDRC Write Protect Mode Register” (MPDDRC_WPMR)	
	- updated WPEN and WPKEY field descriptions	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 30. “Static Memory Controller (SMC)”	
	Editorial and formatting changes throughout	rfo
	Replaced all instances of “HSMC” with “SMC” (except in register names)	rfo
	Replaced instances of “Read-write” with “Read/Write”	rfo
	Section 30.1 “Description”: deleted first paragraph “The External Bus Interface is designed ... a Static Memory Controller (HSMC).”	rfo
	Section 30.2 “Embedded Characteristics”:	
	- removed following bullets:	9352
	-- Programmable Flash Data Width 8 bits or 16 bits	
	-- Supports NAND Flash and SmartMedia™ Devices with 8- or 16-bit Data Path	
	-- Supports NAND Flash with Page Sizes of 528, 1056, 2112 and 4224 Bytes, Specified by Software	
	-- Supports 1-bit Correction for a Page of 512, 1024, 2048 and 4096 Bytes with 8- or 16-bit Data Path	
	-- Supports 1-bit Correction per 512 Bytes of Data for a Page Size of 512, 2048 and 4096 Bytes with 8-bit Data Path	
	-- Supports 1-bit Correction per 256 Bytes of Data for a Page Size of 512, 2048 and 4096 Bytes with 8-bit Data Path	
	-- Programmable Spare Area Size	
	- added bullet “Supports NAND Flash Devices with 8-bit Data Path”	
	Added Section 30.16 “Register Write Protection”	9089
	Section 30.17.4 “NFC SRAM”: replaced single mapping table with five mapping tables	rfo
	Section 30.17.4.1 “NFC SRAM Mapping”: updated second paragraph	rfo
	Figure 30-40 “NAND Write Operation with Spare Encoding”: reformatted to unhide caption “Write NAND operation with SPAREEN set to one” in upper portion of figure	rfo
	Figure 30-41 “NAND Write Operation”: reformatted to unhide caption “Write NAND operation with SPAREEN set to zero” in upper portion of figure	rfo
	Table 30-19 “Register Mapping”:	
	- changed names of following four registers:	9089
	-- HSMC_OCMS (was “HSMC OCMS Register”; is “HSMC Off Chip Memory Scrambling Register”)	
	-- HSMC_KEY1 (was “HSMC OCMS KEY1 Register”; is “HSMC Off Chip Memory Scrambling Key1 Register”)	
	-- HSMC_KEY2 (was “HSMC OCMS KEY2 Register”; is “HSMC Off Chip Memory Scrambling Key2 Register”)	
	-- “HSMC Write Protection Control Register (HSMC_WPCR)” changed to “HSMC Write Protection Mode Register (HSMC_WPMR)”	
	- changed access of HSMC_WPMR (was write-only; is read-write)	9089
	- updated access for registers HSMC_KEY1 and HSMC_KEY2: (was Write-only; is Write-once)	rfo
	Section 30.20.1 “HSMC NFC Configuration Register”: updated PAGESIZE field description	8986
	Section 30.20.8 “HSMC NFC Bank Register”: updated BANK bit description	rfo
	Section 30.20.32 “HSMC Setup Register”—added following sentence below bitmap: “This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Control Register”	9089
	Section 30.20.33 “HSMC Pulse Register”—added following sentence below bitmap: “This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Control Register”	9089
	Section 30.20.34 “HSMC Cycle Register”—added following sentence below bitmap: “This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Control Register”	9089
	Section 30.20.35 “HSMC Timings Register”—added following sentence below bitmap: “This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Control Register”	9089

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 30. “Static Memory Controller (SMC)” (continued)	
	Section 30.20.36 “HSMC Mode Register”:	9089
	- added following sentence below bitmap: “This register can only be written if the WPEN bit is cleared in the HSMC Write Protection Control Register”	
	- updated READ_MODE and WRITE_MODE field descriptions	
	Changed name of Section 30.20.37 “HSMC Off Chip Memory Scrambling Register” (was “HSMC OCMS Register”)	rfo
	Changed name of Section 30.20.38 “HSMC Off Chip Memory Scrambling Key1 Register” (was “HSMC OCMS KEY1 Register”))	rfo
	Changed name of Section 30.20.39 “HSMC Off Chip Memory Scrambling Key2 Register” (was “HSMC OCMS KEY2 Register”)	rfo
	Section 30.20.40 “HSMC Write Protection Mode Register”: changed section name (was “HSMC Write Protection Control Register”); changed access from write-only to read-write; updated field names (were “WP_EN” and “WP_KEY”; are “WPEN” and “WPKEY”) and updated descriptions of same	rfo
	Section 30.20.41 “HSMC Write Protection Status Register”: updated field names (were “WP_VS” and “WP_VSRC”; are “WPVS” and “WPVSRC”); corrected size of field WPVS (was 4 bits; is 1 bit)	rfo
	Section 31. “DMA Controller (DMAC)”	
	Updated introduction text in Section 31.3.1 “DMA Controller 0” and Section 31.3.2 “DMA Controller 1”	8955/rfo
	Section 31.8.1 “DMAC Global Configuration Register”: updated ARB_CFG field description	8835
	Section 31.8.15 “DMAC Channel x [x = 0..7] Descriptor Address Register”: added DMA Master Interface references to DSCR_IF field description	rfo
	Section 31.8.17 “DMAC Channel x [x = 0..7] Control B Register”: added DMA Master Interface references to DIF and SIF field descriptions	rfo
	Section 31.8.21 “DMAC Write Protect Mode Register”: updated WPKEY field description	8834
	Section 32. “AHB LCD Controller (LCDC)”	
	Section 32.2 “Embedded Characteristics”: updated “Display Size” (was “up to 2048x2048”; is “up to 2048x2048, or up to 720p in video format”	8808
	In Section 32.6.7.1 “Line Striding” and in Section 32.6.7.2 “Pixel Striding”: inserted formula for pixel address calculation	9019
	Section 32.7 “LCD Controller (LCDC) User Interface”: changed “name” on Overlay registers in the corresponding register sections (“LCDC_OVR1CHER” --> “LCDC_OVRCHER1”, etc.)	8812
	Updated Table 32-54 ”Register Mapping”:	
	- changed “name” on Overlay registers (“LCDC_OVR1CHER” --> “LCDC_OVRCHER1”, etc.)	8812
	- added new register LCDC_ATTR	9024
	Section 32.7.10 “LCD Controller Status Register”: updated SIPSTS bit description	9008
	Added Section 32.7.15 “LCD Controller Attribute Register”	9024
	Section 32.7.96 “High End Overlay Layer Configuration 1 Register”: added detail on rotation angle to YUV422ROT bit description	9019

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 33. “Image Sensor Interface (ISI)”	
	Formatting and editorial changes throughout document (including harmonization of register names)	
	Section 33.2 “Embedded Characteristics”: updated “Preview Path” bullet to specify “Up to 2048*2048 in Grayscale mode” and “Up to 640*480 in RGB mode”	9078
	Section 33.5.2 “ISI Configuration 2 Register”: updated field descriptions	8988
	Section 33.5.11 “ISI Status Register”: updated field descriptions	8988
	Section 33.5.12 “ISI Interrupt Enable Register”: added missing bit configuration values	rfo
	Section 33.5.13 “ISI Interrupt Disable Register”: added missing bit configuration values	rfo
	Section 33.5.16 “DMA Channel Disable Register”: updated field descriptions	8988
	Section 33.5.17 “DMA Channel Status Register”: updated field descriptions	8988
	Section 33.5.19 “DMA Preview Control Register”: updated field descriptions	8988
	Section 33.5.22 “DMA Codec Control Register”: updated field descriptions	8988
	Section 33.5.24 “ISI Write Protection Control Register”: updated field names in bitmap and updated field descriptions	8886
	Section 34. “USB High Speed Device Port (UDPHS)”	
	Improved readabiltiy of graphics	9004
	Minor formatting and editorial changes throughout	rfo
	Figure 34-2 “Board Schematic”: replaced resistor 6K8 with 5K62	rfo
	Section 34.5.1 “Power Management”: corrected names of referenced registers	rfo
	Figure 34-3 “USB Transfer Events”:	rfo
	- added column headers	
	- in CONTROL row, replaced two diameter symbols with arrow symbols	
	Section 35. “USB Host High Speed Port (UHPHS)”	
	Minor formatting and editorial changes throughout	rfo
	Figure 35-2 “Board Schematic to Interface UHP High-speed Host Controller”: replaced resistor 6K8 with 5K62	rfo
	Section 35.5.2 “Power Management”: corrected names of referenced registers and bits	rfo
	Section 36. “Gigabit Ethernet MAC (GMAC)”	
	Minor editorial and formatting changes throughout	
	Section 36.6.1.6 “Interrupts”: in first paragraph, deleted content “Depending on the overall system ... CPU enters the interrupt handler” and “Note that in the default ... be write-one-to-clear if desired”	rfo
	Section 36.6.2 “Statistics Registers”: deleted sentence “In order to reduce overall design area, the Statistics Registers may be optionally removed in the configuration file if they are deemed unnecessary for a particular design.”	8865
	Section 36.7.1 “Network Control Register”: removed bit RDS (“Read Snapshot” function not supported)	rfo
	Section 36.7.35 “Stacked VLAN Register”: added missing description to field ESVLAN	rfo
	Section 37. “Ethernet MAC 10/100 (EMAC)”: No changes	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 38. “High Speed MultiMedia Card Interface (HSMCI)”	
	Minor formatting and editorial changes throughout	
	Harmonized register access type naming throughout (“Read-write” change to “Read/Write”, “Read” changed to “Read-only”, “Write” changed to “Write-only”)	rfo
	Figure 38-1 “Block Diagram (8-bit configuration)”: added “(8-bit configuration)” to title	
	Figure 38-2 “Block Diagram (4-bit configuration)”: added “(4-bit configuration)” to title; added missing note below figure	
	Section 38.8.1 “Command - Response Operation”: reorganized table content in ALL_SEND_CID command example	rfo
	Figure 38-9 “Read Functional Flow Diagram”: corrected HSMCI_MR to HSMCI_BLKR when referring to Block Length field that is not available in HSMCI_MR; deleted note “This field is also accessible in the HSMCI Block Register (HSMCI_BLKR)”	9012
	Figure 38-10 “Write Functional Flow Diagram”: corrected HSMCI_MR to HSMCI_BLKR when referring to Block Length field that is not available in HSMCI_MR; deleted note “This field is also accessible in the HSMCI Block Register (HSMCI_BLKR)”	9012
	Figure 38-11 “Read Multiple Block and Write Multiple Block”: corrected HSMCI_MR to HSMCI_BLKR when referring to Block Length field that is not available in HSMCI_MR	9012
	Section 38.13 “Register Write Protection”: changed title (was “Write Protection Registers”); revised content	rfo
	Section 38.14.7 “HSMCI Block Register”: deleted sentence “This field is also accessible in the HSMCI Mode Register (HSMCI_MR)” from BLKLEN field description	rfo
	Section 38.14.18 “HSMCI Write Protection Mode Register”:	
	- modified register name (was HSMCI Write Protect Mode Register)	rfo
	- updated field names in bitmap and updated field descriptions	8827/8868
	- replaced list of protectable registers with cross-reference to section “Register Write Protection”	rfo
	Section 38.14.19 “HSMCI Write Protection Status Register”: modified register name (was HSMCI Write Protect Status Register) and updated register description	rfo
	Section 39. “Serial Peripheral Interface (SPI)”	
	Minor editorial and formatting changes	
	Section 39.7.4 “SPI Slave Mode”: in paragraph beginning with “Then, a new data is loaded...”, replaced “the Shift Register is not modified and the last received character is retransmitted” with “the SPI_TDR is retransmitted”	8792
	Table 39-5 ”Register Mapping”:	8840
	- replaced offset range 0x4C–0xE0 with 0x40–0xE0	
	- replaced offset range 0x00E8–0x00F8 with 0x00EC–0x00F8	
	Section 39.8.10 “SPI Write Protection Mode Register”: updated WPKEY field description	8840

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 40. “Two-wire Interface (TWI)”	
	Formatting and minor editorial changes throughout	rfo
	Replaced “N” with “NA” as acronym for “Not Acknowledge” where needed	
	Section 40.6.2 “Power Management”: deleted rogue bullet “Enable the peripheral clock” below section heading	rfo
	Added missing “yes” and “no” in following figures:	8944/rfo
	- Figure 40-18 “TWI Write Operation with Multiple Data Bytes with or without Internal Address”	
	- Figure 40-20 “TWI Read Operation with Single Data Byte and Internal Address”	
	- Figure 40-21 “TWI Read Operation with Multiple Data Bytes with or without Internal Address”	
	Section 40.10 “Slave Mode”: added Section 40.10.6 “Using the DMA Controller”	9055
	Changed register acronym in Section 40.11 “Write Protection System” and renamed referenced field and flag names	8845
	Table 40-7 “Register Mapping”: inserted reserved offset range 0x38–0xE0	8944
	Section 40.12.6 “TWI Status Register”: updated NACK (used in Master mode) bit description	9145
	Section 40.12.11 “TWI Transmit Holding Register”: corrected access (was Read-write; is Write-only)	9050
	Changed register acronym in Section 40.12.12 “TWI Write Protection Mode Register” and updated filed names and field descriptions	8845
	Changed register acronym in Section 40.12.13 “TWI Write Protection Status Register” and updated filed names and field descriptions	8845
	Section 41. “Synchronous Serial Controller (SSC)”	
	Minor editorial and formatting changes throughout	
	Section 41.9.17 “SSC Write Protect Mode Register”: updated WPKEY field description	8841
	Section 42. “Debug Unit (DBGU)”: No changes	
	Section 43. “Universal Asynchronous Receiver Transmitter (UART)”	
	Minor editorial and formatting changes throughout	
	Changed register bit configuration value notations from ‘0 =’ to ‘0:’ and from ‘1 =’ to ‘1:’	rfo
	Table 43-3 “Register Mapping”:	
	- inserted offset range 0x0040 - 0x00E8 (reserved)	8537
	- changed access from Read-write to Read/Write where applicable	rfo
	Section 44. “Universal Synchronous Asynchronous Receiver Transceiver (USART)”	
	Minor formatting and editorial changes throughout	
	Corrected Figure 44-22 “Parity Error” for stop bit value	8943
	Table 44-10 “Maximum Timeguard Length Depending on Baud Rate”: replaced baud rate 33400 bit/s with 38400 bit/s	8943
	Table 44-11 “Maximum Time-out Period”: replaced baud rate 33400 bit/s with 38400 bit/s	8943
	Table 44-13 “IrDA Baud Rate Error”: added missing units of measure to column headers (“Bit/s” for Baud Rate and “µs” for Pulse Time)	rfo
	Section 44.7.5.3 “IrDA Demodulator”: replaced instances of “T” with “t” when used to express time	rfo
	Table 44-15 “Register Mapping”:	8943
	- inserted offset range 0x0054–0x005C (reserved)	
	- inserted offset range 0x0060–0x00E0 (reserved)	
	- replaced range 0x5C–0xFC with range 0x00EC–0x00FC (reserved)	
	Section 44.8.22 “USART Write Protect Mode Register”: updated WPKEY field description	8791

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 45. “Controller Area Network (CAN)”	
	Minor editorial formatting changes throughout	
	Section 45.6.3 “Interrupt”: replaced “AIC” occurrences with “interrupt controller”	8594
	Table 45-4 “Receive Mailbox Objects”: added missing title	rfo
	Table 45-5 “Transmit Mailbox Objects”: added missing title	rfo
	Section 45.7.4.1 “CAN Bit Timing Configuration”: moved three bullets describing the phase segments to precede the bullet “TIME QUANTAM”	rfo
	Section 45.8.1 “CAN Controller Initialization”: replaced “AIC” occurrences with “interrupt controller”	8594
	Figure 45-10 “Possible Initialization Procedure”: replaced instance of “AIC” with “Interrupt Controller”	8594
	WPMR register reformat using WPKEY table in Section 45.9.12 “CAN Write Protection Mode Register”	8867
	Section 46. “Software Modem Device (SMD)”	
	Section 46.1 “Description”: replaced acronym HLSD with LSD	9014
	Added Section 46.4 “Software Modem Device (SMD) User Interface” (includes Section 46.4.1 “SMD Drive Register”)	8950
	Section 47. “Timer Counter (TC)”	
	Editorial and formatting changes throughout	
	Section 47.1 “Description”: replaced “and TIOA1 inputs” with “and TIOB1 inputs”	8885
	Figure 47-5 “Example of Transfer with DMAC” replaced “Internal PDC trigger” with “Peripheral trigger”	9325
	Section 47.6.15.3 “Direction Status and Change Detection”: rewrote sixth paragraph for clarity	rfo
	Section 47.6.15.4 “Position and Rotation Measurement”: rewrote first paragraph for clarity	rfo
	Section 47.6.15.5 “Speed Measurement”: replaced sentence “The speed can be read on TC_RA0 register in TC_CMR0” with “The speed can be read on field RA in register TC_RA0”	rfo
	Section 47.6.17 “Register Write Protection”: changed title (was “Write Protection System”); revised content	rfo
	Section 47.7.2 “TC Channel Mode Register: Capture Mode”: updated TCCLKS field description (values 0 to 4)	9107
	Section 47.7.3 “TC Channel Mode Register: Waveform Mode”:	
	- updated ENETRG bit description	8885
	- updated TCCLKS field description (values 0 to 4)	9107
	Section 47.7.15 “TC Block Mode Register”:	
	- updated FILTER bit description	rfo
	- corrected TC2XC2S field configuration values: value 2 is TIOA0 (was TIOA1); value 3 is TIOA1 (was TIOA2)	9353
	Section 47.7.20 “TC Write Protection Mode Register”: modified register name (was “TC Write Protect Mode Register”); updated field descriptions	rfo/8842

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 48. “Pulse Width Modulation Controller (PWM)”	
	Editorial and formatting changes throughout; replaced instances of “Read-write” with “Read/Write”	rfo
	Section 48.6.2.2 “Comparator”: corrected PWM waveform period formulas	9123
	Table 6-1 “Fault Inputs”: in “Polarity Level” column, replaced all instances of “1” with “To be configured to 1”	rfo
	Section 48.6.5.4 “Changing the Synchronous Channels Update Period”: removed cross-reference to section “Method 3: Automatic write of duty-cycle values and automatic trigger of the update” from first paragraph	rfo
	Section 48.6.1 “PWM Clock Generator”: replaced instances of “F” with “f” when used to represent frequency	rfo
	Section 48.6.5.7 “Register Write Protection”: at end of section, replaced sentence “The WPVS and PWM_WPSR fields are automatically reset after reading the PWM_WPSR register” with “The WPVS and WPVSRC fields are automatically cleared after reading the PWM_WPSR”	rfo
	Section 48.7.9 “PWM Sync Channels Mode Register”: removed table row for value 3 “reserved” in UPDM field description	rfo
	Section 48.7.30 “PWM Write Protection Control Register”: updated WPKEY and WPCMD field descriptions	8849
	Section 49. “Analog-to-Digital Converter (ADC)”	
	Editorial and formatting changes throughout	
	Added title to Figure 49-6 “Hardware Trigger Delay”	8997
	Section 49.5.8 “Conversion Performances”: updated wording (‘see the product DC Characteristics section’ --> ‘see the product electrical characteristics’)	9184
	Section 49.8.25 “ADC Write Protect Mode Register”: updated WPKEY field description	8856
	Section 50. “True Random Number Generator (TRNG)”	
	Figure 50-1 “TRNG Block Diagram”: reformatted graphics to improve readability	9001
	Section 50.4.1 “Power Management”: minor editorial changes	rfo
	Section 50.5 “Functional Description”: minor editorial changes	rfo
	Section 50.6.1 “TRNG Control Register”: updated KEY field description	8843
	Section 51. “Advanced Encryption Standard (AES)”	
	Editorial and minor formatting changes throughout	
	Section 51.2 “Embedded Characteristics”: added bullet “Double Input Buffer Optimizes Runtime”	9076
	Added Section 51.4.2 “Double Input Buffer”	9076
	Table 51-4 “Last Output Data Mode Behavior versus Start Modes”	9103
	- reorganized order of rows	
	- replaced “Not available” with “At the address specified in the Channel Buffer Transfer Descriptor” in row Encrypted/Decrypted Data Result Location at DMA Transfer LOD = 0	
	Section 51.6.2 “AES Mode Register”:	
	- updated CKEY field description	8859
	- updated LOD field description	9103
	- added field DUALBUFF	9076
	Section 51.6.10 “AES Initialization Vector Register x”: updated IV field description	8892
	Section 52. “Triple Data Encryption Standard (Triple DES)”: No changes	
	Section 53. “Secure Hash Algorithm (SHA)”: No changes	

Doc. Rev. 11121C	Comments	Change Request Ref.
	Section 54. “Electrical Characteristics”	
	Minor editorial and formatting changes throughout	rfo
	Table 54-1 “Absolute Maximum Ratings*”:	rfo
	- deleted “Operating Temperature (Industrial).....-40° C to +85°C”	
	- added (VDDFUSE)...3.3V	
	Section 54.2 “DC Characteristics”: in first sentence, replaced “T_A = -40°C to +85°C” with “T_A = -40°C to +105°C”	rfo
	Table 54-2 “DC Characteristics”:	rfo
	- deleted parameters $V_{VDDCORErip}$ / VDDCORE ripple, $V_{VDDBUrip}$ / VDDBU ripple, $V_{VDDPLLArip}$ / VDDPLLA ripple, and $V_{VDDOSCrip}$ / VDDOSC ripple	
	- added parameters $V_{VDDFUSE}$ / DC Supply FUSE box and $I_{VDDFUSE}$ / VDDFUSE current	
	- updated parameter I_{SC} / Static Current	
	- updated parameter name and conditions of V_{VDDOSC}	
	Updated Section 54.3 “Power Consumption”	rfo
	Table 54-9 “Processor Clock Waveform Parameters”: updated conditions, minimum values and footnote	rfo
	Table 54-10 “Master Clock Waveform Parameters”: updated conditions and footnote	rfo
	Section 54.5.1 “Crystal Oscillator Characteristics”: in first sentence, replaced “T_A = -40°C to +85°C” with “T_A = -40°C to +105°C”	rfo
	Table 54-15 “32 kHz Oscillator Characteristics”: corrected referenced register name and section in note	rfo
	Table 54-19 “PLLA Characteristics”: updated output frequency conditions	9031
	Table 54-45 “System Clock Waveform Parameters”: replaced “T = 85°C” with “T_A = 85°C” in conditions	rfo
	Table 54-46 “System Clock Waveform Parameters”: replaced “T = 85°C” with “T_A = 85°C” in conditions	rfo
	Table 54-47 “System Clock Waveform Parameters”: replaced “T = 85°C” with “T_A = 85°C” in conditions	rfo
	Section 54.18 “MCI Timings”: updated content; removed “MCI Timings” figure and table	rfo
	Section 54.21 “Two-wire Interface Characteristics”: updated Table 54-60 “Two-wire Serial Bus Requirements” and the corresponding notes	8819
	Table 54-56 “EMAC MII Specific Signals”: in footnote, corrected link to note 2 of Table 54-55 “EMAC Signals Relative to EMDC” (was incorrectly linked to note of Table 54-49 “SSC Timings with 3.3V Peripheral supply”)	rfo
	Table 54-57 “RMII Mode”: in footnote, corrected link to note 2 of Table 54-55 “EMAC Signals Relative to EMDC” (was incorrectly linked to note of Table 54-49 “SSC Timings with 3.3V Peripheral supply”)	rfo
	Section 55. “Mechanical Characteristics”	
	Added Section 55.2 “324-ball TFBGA Mechanical Characteristics” (introduced new package)	9035
	Section 56. “SAMA5D3 Ordering Information”	
	Updated Table 56-1 “SAMA5D3 Ordering Information” (includes adding SAMA5D36 device references and introduction of 324-ball TFBGA package)	8822 9035
	Section 57. “SAMA5D3 Series Errata”	
	Updated Section 57.3 “LCD Controller (LCDC)”	8971
	Added Section 57.6 “Watchdog Timer (WDT)”	8969

Doc. Rev. 11121B	Comments	Change Request Ref.
	Introduction:	
	Section 8.1 “Chip Identification”, updated Chip ID: “0x8A5C07C1” --> “0x8A5C07C2”.	rfo
	“Description” , added a cross-reference to Table 1-1 “SAMA5D3 Device Differences” in the last paragraph.	rfo
	Replaced “Cortex™” references with “Cortex®” in “Description” and further on in the entire document.	rfo
	Removed “AT91SAM” from the document title.	
	Standard Boot Strategies:	
	Section 12.3.4.1 “NAND Flash Boot: NAND Flash Detection”, added the eccBitReq field description in “NAND Flash Specific Header Detection” .	8796
	SFR:	
	Added a row for SFR_UTMICKTRIM register (offset value “0x30”) in Table 16-1 "Register Mapping" and the corresponding Table 16.3.5 "UTMI Clock Trimming Register".	8683
	External Memories:	
	Section 28.1.5 “Product Dependencies”, updated LPDDR2 Mode data in Table 28-2 “DDR2 I/O Lines Usage vs Operating Modes” (DDR_WE, DDR_RAS - DDR_CAS, and DDR_A[13..0]).	rfo
	Section 28.1.6.2 “2x16-bit LPDDR2”, added references on CAx LP-DDR2 signals and Table 28-3 “CAx LP-DDR2 Signal Connection”.	
	USART:	
	Added a paragraph on IRDA_FILTER programming criteria in Section 44.7.5.3 “IrDA Demodulator” and in the corresponding field description in Section 44.8.20 “USART IrDA FILTER Register”.	8508
	Section 44.8.18 “USART FI DI RATIO Register”, expanded FI_DI_RATIO field to 16 bits in the register table.	8643
	FUSE:	
	Section 50.2 “Embedded Characteristics”, added references on FUSE bits.	8785
	Added Section 50.2.1 “FUSE Bit Mapping” and Section 50.2.2 “Special Functions”.	
	Electrical Characteristics:	
	Section 54.6 “12 MHz RC Oscillator Characteristics”, updated the I_{DDON} value in Table 54-12.	rfo
	Section 54.7 “32 kHz Oscillator Characteristics”, added a row on P_{ON} in Table 54-13.	
	Section 54.11 “12-Bit ADC Characteristics”, updated data in:	
	- Table 54-22 “Analog Power Supply Characteristics”	
	- Table 54-23 “Channel Conversion Time and ADC Clock”	
	- Table 54-24 “External Voltage Reference Input”	
	- Table 54-25 “INL, DNL, 12-bit mode, VDDANA 2.4V to 3.6V supply voltage conditions”	
	- Table 54-26 “Gain Error, 12-bit Mode, VDDANA 2.4V to 3.6V supply voltage conditions”	
	- Table 54-27 “Error offset with or without calibration, 12-bit Mode, VDDANA 2.4V to 3.6V supply voltage conditions”	
	- Table 54-28 “Dynamic Performance Characteristics in Single ended and 12 bits mode (1)”; and	
	- Table 54-29 “Dynamic Performance Characteristics in Differential and 12 bits mode(1)”	
	Added Section 54.15 “MPDDRC Timings”	rfo
	Section 54.16 “SSC Timings”, updated PIXCLK frequency maximum value and fixed transmitter parameter data for SSC_7 in Table 54-47 “SSC Timings with 3.3V Peripheral supply” and Table 54-48 “SSC Timing with 1V8 Peripheral supply”	rfo

Doc. Rev. 11121B	Comments	Change Request Ref.
	Mechanical Characteristics: Added “Nominal Ball Diameter” and “Solder” rows in Table 55-4 “Package Information”.	rfo
	Errata: Added the introduction paragraph in Section 57. “SAMA5D3 Series Errata”. Added Section 57.1.2 “Boot ROM: NAND Flash Detection using ONFI Parameters does Not Work”.	rfo 8785

Doc. Rev. 11121A	Comments	Change Request Ref.
	First Issue	

Table of Contents

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