Features

- Incorporates the ARM7TDMI® ARM® Thumb® Processor
  - High-performance 32-bit RISC Architecture
  - High-density 16-bit Instruction Set
  - Leader in MIPS/Watt
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- Internal High-speed Flash
  - 512 Kbytes, Organized in Two Contiguous Banks of 1024 Pages of 256 Bytes Dual Plane (SAM7SE512)
  - 256 Kbytes (SAM7SE256) Organized in One Bank of 1024 Pages of 256 Bytes Single Plane (SAM7SE256)
  - 32 Kbytes (SAM7SE32) Organized in One Bank of 256 Pages of 128 Bytes Single Plane (SAM7SE32)
  - Single Cycle Access at Up to 30 MHz in Worst Case Conditions
  - Prefetch Buffer Optimizing Thumb Instruction Execution at Maximum Speed
  - Page Programming Time: 6 ms, Including Page Auto-erase, Full Erase Time: 15 ms
  - 10,000 Erase Cycles, 10-year Data Retention Capability, Sector Lock Capabilities, Flash Security Bit
  - Fast Flash Programming Interface for High Volume Production
- 32 Kbytes (SAM7SE512/256) or 8 Kbytes (SAM7SE32) of Internal High-speed SRAM, Single-cycle Access at Maximum Speed
- One External Bus Interface (EBI)
  - Supports SDRAM, Static Memory, Glueless Connection to CompactFlash® and ECC-enabled NAND Flash
- Memory Controller (MC)
  - Embedded Flash Controller
  - Memory Protection Unit
  - Abort Status and Misalignment Detection
- Reset Controller (RSTC)
  - Based on Power-on Reset Cells and Low-power Factory-calibrated Brownout Detector
  - Provides External Reset Signal Shaping and Reset Source Status
- Clock Generator (CKGR)
  - Low-power RC Oscillator, 3 to 20 MHz On-chip Oscillator and One PLL
- Power Management Controller (PMC)
  - Power Optimization Capabilities, Including Slow Clock Mode (Down to 500 Hz) and Idle Mode
  - Three Programmable External Clock Signals
- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Two External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - Two-wire UART and Support for Debug Communication Channel interrupt, Programmable ICE Access Prevention
  - Mode for General Purpose Two-wire UART Serial Communication
- Periodic Interval Timer (PIT)
  - 20-bit Programmable Counter plus 12-bit Interval Counter
- Windowed Watchdog (WDT)
  - 12-bit key-protected Programmable Counter

AT91SAM ARM-based Flash MCU

SAM7SE512 SAM7SE256 SAM7SE32

6222H–ATARM–25-Jan-12

  - Provides Reset or Interrupt Signals to the System
  - Counter May Be Stopped While the Processor is in Debug State or in Idle Mode
- Real-time Timer (RTT)
  - 32-bit Free-running Counter with Alarm
  - Runs Off the Internal RC Oscillator
- Three Parallel Input/Output Controllers (PIO)
  - Eighty-eight Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line
  - Individually Programmable Open-drain, Pull-up Resistor and Synchronous Output
  - Schmitt Trigger on All inputs
- Eleven Peripheral DMA Controller (PDC) Channels
- One USB 2.0 Full Speed (12 Mbits per second) Device Port
  - On-chip Transceiver, Eight Endpoints, 2688-byte Configurable Integrated FIFOs
- One Synchronous Serial Controller (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I²S Analog Interface Support, Time Division Multiplex Support
  - High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer
- Two Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - Individual Baud Rate Generator, IrDA® Infrared Modulation/Demodulation
  - Support for ISO7816 T0/T1 Smart Card, Hardware Handshaking, RS485 Support
  - Full Modem Line Support on USART1
- One Master/Slave Serial Peripheral Interfaces (SPI)
  - 8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects
- One Three-channel 16-bit Timer/Counter (TC)
  - Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- One Four-channel 16-bit PWM Controller (PWMC)
- One Two-wire Interface (TWI)
  - Master, Multi-Master and Slave Mode Support, All Two-wire Atmel EEPROMs Supported
  - General Call Supported in Slave Mode
- One 8-channel 10-bit Analog-to-Digital Converter, Four Channels Multiplexed with Digital I/Os
- SAM-BA®
  - Default Boot program
  - Interface with SAM-BA Graphic User Interface
- IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins
- Four High-current Drive I/O lines, Up to 16 mA Each
- Power Supplies
  - Embedded 1.8V Regulator, Drawing up to 100 mA for the Core and External Components
  - 1.8V or 3,3V VDDIO I/O Lines Power Supply, Independent 3.3V VDDFLASH Flash Power Supply
  - 1.8V VDDCORE Core Power Supply with Brownout Detector
- Fully Static Operation:
  - Up to 55 MHz at 1.8V and 85° C Worst Case Conditions
  - Up to 48 MHz at 1.65V and 85° C Worst Case Conditions
- Available in a 128-lead LQFP Green Package, or a 144-ball LFBGA RoHS-compliant Package

1. Description

Atmel's SAM7SE Series is a member of its Smart ARM Microcontroller family based on the 32-bit ARM7™ RISC processor and high-speed Flash memory.

- SAM7SE512 features a 512-Kbyte high-speed Flash and a 32 Kbyte SRAM.
- SAM7SE256 features a 256-Kbyte high-speed Flash and a 32 Kbyte SRAM.
- SAM7SE32 features a 32-Kbyte high-speed Flash and an 8 Kbyte SRAM.

It also embeds a large set of peripherals, including a USB 2.0 device, an External Bus Interface (EBI), and a complete set of system functions minimizing the number of external components.

The EBI incorporates controllers for synchronous DRAM (SDRAM) and Static memories and features specific circuitry facilitating the interface for NAND Flash, SmartMedia and CompactFlash.

The device is an ideal migration path for 8/16-bit microcontroller users looking for additional performance, extended memory and higher levels of system integration.

The embedded Flash memory can be programmed in-system via the JTAG-ICE interface or via a parallel interface on a production programmer prior to mounting. Built-in lock bits and a security bit protect the firmware from accidental overwrite and preserve its confidentiality.

The SAM7SE Series system controller includes a reset controller capable of managing the power-on sequence of the microcontroller and the complete system. Correct device operation can be monitored by a built-in brownout detector and a watchdog running off an integrated RC oscillator.

By combining the ARM7TDMI processor with on-chip Flash and SRAM, and a wide range of peripheral functions, including USART, SPI, External Bus Interface, Timer Counter, RTT and Analog-to-Digital Converters on a monolithic chip, the SAM7SE512/256/32 is a powerful device that provides a flexible, cost-effective solution to many embedded control applications.

1.1 Configuration Summary of the SAM7SE512, SAM7SE256 and SAM7SE32

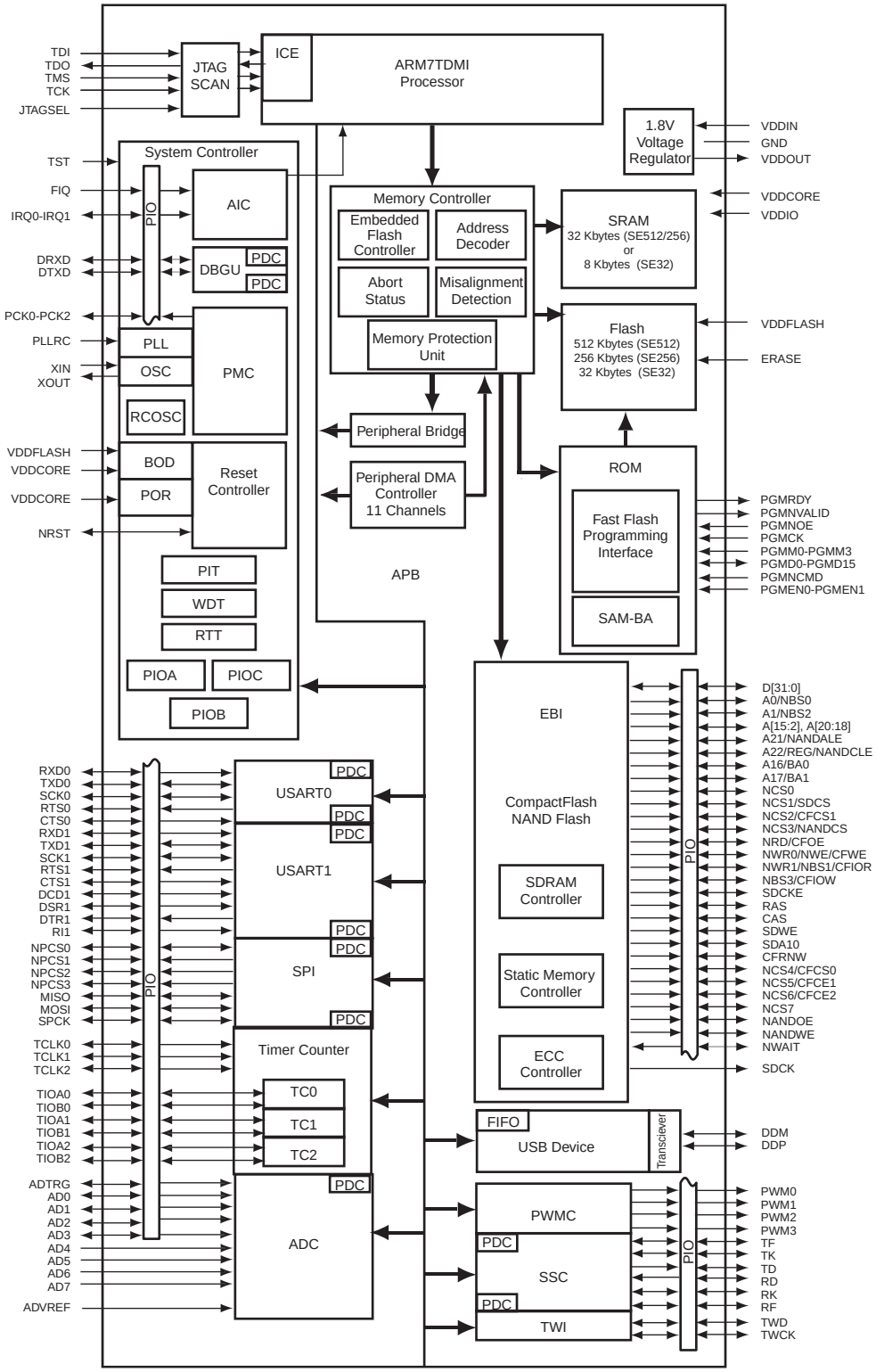
The SAM7SE512, SAM7SE256 and SAM7SE32 differ in memory sizes and organization. Table 1-1 below summarizes the configurations for the three devices.

Table 1-1. Configuration Summary

Device	Flash Size	Flash Organization	RAM Size
SAM7SE512	512K bytes	dual plane	32K bytes
SAM7SE256	256K bytes	single plane	32K bytes
SAM7SE32	32K bytes	single plane	8K bytes

2. Block Diagram

Figure 2-1. SAM7SE512/256/32 Block Diagram Signal Description



3. Signal Description

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDIN	Voltage Regulator and ADC Power Supply Input	Power		3V to 3.6V
VDDOUT	Voltage Regulator Output	Power		1.85V
VDDFLASH	Flash and USB Power Supply	Power		3V to 3.6V
VDDIO	I/O Lines Power Supply	Power		3V to 3.6V or 1.65V to 1.95V
VDDCORE	Core Power Supply	Power		1.65V to 1.95V
VDDPLL	PLL	Power		1.65V to 1.95V
GND	Ground	Ground		
Clocks, Oscillators and PLLs				
XIN	Main Oscillator Input	Input		
XOUT	Main Oscillator Output	Output		
PLLRC	PLL Filter	Input		
PCK0 - PCK2	Programmable Clock Output	Output		
ICE and JTAG				
TCK	Test Clock	Input		No pull-up resistor
TDI	Test Data In	Input		No pull-up resistor
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		No pull-up resistor.
JTAGSEL	JTAG Selection	Input		Pull-down resistor (1)
Flash Memory				
ERASE	Flash and NVM Configuration Bits Erase Command	Input	High	Pull-down resistor (1)
Reset/Test				
NRST	Microcontroller Reset	I/O	Low	Open drain with pull-up resistor (1)
TST	Test Mode Select	Input	High	Pull-down resistor (1)
Debug Unit				
DRXD	Debug Receive Data	Input		
DTXD	Debug Transmit Data	Output		
AIC				
IRQ0 - IRQ1	External Interrupt Inputs	Input		
FIQ	Fast Interrupt Input	Input		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
PIO				
PA0 - PA31	Parallel IO Controller A	I/O		Pulled-up input at reset
PB0 - PB31	Parallel IO Controller B	I/O		Pulled-up input at reset
PC0 - PC23	Parallel IO Controller C	I/O		Pulled-up input at reset
USB Device Port				
DDM	USB Device Port Data -	Analog		
DDP	USB Device Port Data +	Analog		
USART				
SCK0 - SCK1	Serial Clock	I/O		
TXD0 - TXD1	Transmit Data	I/O		
RXD0 - RXD1	Receive Data	Input		
RTS0 - RTS1	Request To Send	Output		
CTS0 - CTS1	Clear To Send	Input		
DCD1	Data Carrier Detect	Input		
DTR1	Data Terminal Ready	Output		
DSR1	Data Set Ready	Input		
RI1	Ring Indicator	Input		
Synchronous Serial Controller				
TD	Transmit Data	Output		
RD	Receive Data	Input		
TK	Transmit Clock	I/O		
RK	Receive Clock	I/O		
TF	Transmit Frame Sync	I/O		
RF	Receive Frame Sync	I/O		
Timer/Counter				
TCLK0 - TCLK2	External Clock Inputs	Input		
TIOA0 - TIOA2	Timer Counter I/O Line A	I/O		
TIOB0 - TIOB2	Timer Counter I/O Line B	I/O		
PWM Controller				
PWM0 - PWM3	PWM Channels	Output		
Serial Peripheral Interface				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
SPCK	SPI Serial Clock	I/O		
NPCS0	SPI Peripheral Chip Select 0	I/O	Low	
NPCS1-NPCS3	SPI Peripheral Chip Select 1 to 3	Output	Low	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
Two-Wire Interface				
TWD	Two-wire Serial Data	I/O		
TWCK	Two-wire Serial Clock	I/O		
Analog-to-Digital Converter				
AD0-AD3	Analog Inputs	Analog		Digital pulled-up inputs at reset
AD4-AD7	Analog Inputs	Analog		Analog Inputs
ADTRG	ADC Trigger	Input		
ADVREF	ADC Reference	Analog		
Fast Flash Programming Interface				
PGMEN0-PGMEN2	Programming Enabling	Input		
PGMM0-PGMM3	Programming Mode	Input		
PGMD0-PGMD15	Programming Data	I/O		
PGMRDY	Programming Ready	Output	High	
PGMNVALID	Data Direction	Output	Low	
PGMNOE	Programming Read	Input	Low	
PGMCK	Programming Clock	Input		
PGMNCMD	Programming Command	Input	Low	
External Bus Interface				
D[31:0]	Data Bus	I/O		
A[22:0]	Address Bus	Output		
NWAIT	External Wait Signal	Input	Low	
Static Memory Controller				
NCS[7:0]	Chip Select Lines	Output	Low	
NWR[1:0]	Write Signals	Output	Low	
NRD	Read Signal	Output	Low	
NWE	Write Enable	Output	Low	
NUB	NUB: Upper Byte Select	Output	Low	
NLB	NLB: Lower Byte Select	Output	Low	
EBI for CompactFlash Support				
CFCE[2:1]	CompactFlash Chip Enable	Output	Low	
CFOE	CompactFlash Output Enable	Output	Low	
CFWE	CompactFlash Write Enable	Output	Low	
CFIOR	CompactFlash I/O Read Signal	Output	Low	
CFIOW	CompactFlash I/O Write Signal	Output	Low	
CFRNW	CompactFlash Read Not Write Signal	Output		
CFCS[1:0]	CompactFlash Chip Select Lines	Output	Low	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
EBI for NAND Flash Support				
NANDCS	NAND Flash Chip Select Line	Output	Low	
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
NANDCLE	NAND Flash Command Line Enable	Output	Low	
NANDALE	NAND Flash Address Line Enable	Output	Low	
SDRAM Controller				
SDCK	SDRAM Clock	Output		Tied low after reset
SDCKE	SDRAM Clock Enable	Output	High	
SDCS	SDRAM Controller Chip Select Line	Output	Low	
BA[1:0]	Bank Select	Output		
SDWE	SDRAM Write Enable	Output	Low	
RAS - CAS	Row and Column Signal	Output	Low	
NBS[3:0]	Byte Mask Signals	Output	Low	
SDA10	SDRAM Address 10 Line	Output		

Note: 1. Refer to Section 6. "I/O Lines Considerations" .

4. Package

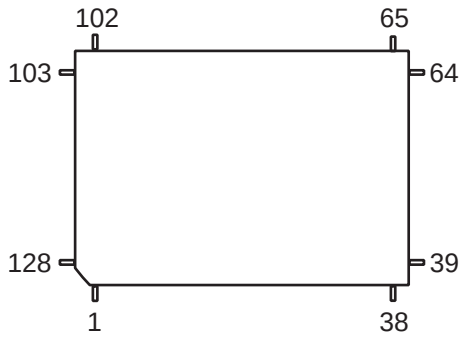
The SAM7SE512/256/32 is available in:

- 20 x 14 mm 128-lead LQFP package with a 0.5 mm lead pitch.
- 10x 10 x 1.4 mm 144-ball LFBGA package with a 0.8 mm lead pitch

4.1 128-lead LQFP Package Outline

Figure 4-1 shows the orientation of the 128-lead LQFP package and a detailed mechanical description is given in the Mechanical Characteristics section of the full datasheet.

Figure 4-1. 128-lead LQFP Package Outline (Top View)



4.2 128-lead LQFP Pinout

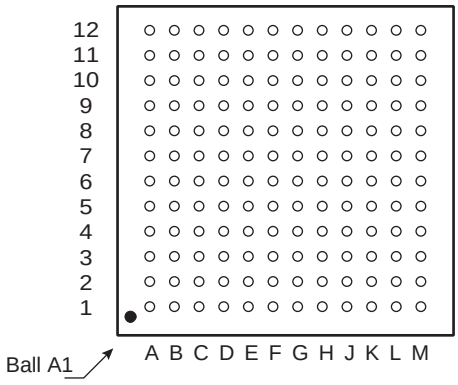
Table 4-1. Pinout in 128-lead LQFP Package

1	ADVREF	33	PB31	65	TDI	97	SDCK
2	GND	34	PB30	66	TDO	98	PC8
3	AD7	35	PB29	67	PB2	99	PC7
4	AD6	36	PB28	68	PB1	100	PC6
5	AD5	37	PB27	69	PB0	101	PC5
6	AD4	38	PB26	70	GND	102	PC4
7	VDDOUT	39	PB25	71	VDDIO	103	PC3
8	VDDIN	40	PB24	72	VDDCORE	104	PC2
9	PA20/PGMD8/AD3	41	PB23	73	NRST	105	PC1
10	PA19/PGMD7/AD2	42	PB22	74	TST	106	PC0
11	PA18/PGMD6/AD1	43	PB21	75	ERASE	107	PA31
12	PA17/PGMD5/AD0	44	PB20	76	TCK	108	PA30
13	PA16/PGMD4	45	GND	77	TMS	109	PA29
14	PA15/PGMD3	46	VDDIO	78	JTAGSEL	110	PA28
15	PA14/PGMD2	47	VDDCORE	79	PC23	111	PA27/PGMD15
16	PA13/PGMD1	48	PB19	80	PC22	112	PA26/PGMD14
17	PA12/PGMD0	49	PB18	81	PC21	113	PA25/PGMD13
18	PA11/PGMM3	50	PB17	82	PC20	114	PA24/PGMD12
19	PA10/PGMM2	51	PB16	83	PC19	115	PA23/PGMD11
20	PA9/PGMM1	52	PB15	84	PC18	116	PA22/PGMD10
21	VDDIO	53	PB14	85	PC17	117	PA21/PGMD9
22	GND	54	PB13	86	PC16	118	VDDCORE
23	VDDCORE	55	PB12	87	PC15	119	GND
24	PA8/PGMM0	56	PB11	88	PC14	120	VDDIO
25	PA7/PGMNVALID	57	PB10	89	PC13	121	DM
26	PA6/PGMNOE	58	PB9	90	PC12	122	DP
27	PA5/PGMRDY	59	PB8	91	PC11	123	VDDFLASH
28	PA4/PGMNCMD	60	PB7	92	PC10	124	GND
29	PA3	61	PB6	93	PC9	125	XIN/PGMCK
30	PA2/PGMEN2	62	PB5	94	GND	126	XOUT
31	PA1/PGMEN1	63	PB4	95	VDDIO	127	PLLRC
32	PA0/PGMEN0	64	PB3	96	VDDCORE	128	VDDPLL

4.3 144-ball LFBGA Package Outline

Figure 4-2 shows the orientation of the 144-ball LFBGA package and a detailed mechanical description is given in the Mechanical Characteristics section.

Figure 4-2. 144-ball LFBGA Package Outline (Top View)



4.4 144-ball LFBGA Pinout

Table 4-2. SAM7SE512/256/32 Pinout for 144-ball LFBGA Package

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	PB7	D1	VDDCORE	G1	PC18	K1	PC11
A2	PB8	D2	VDDCORE	G2	PC16	K2	PC6
A3	PB9	D3	PB2	G3	PC17	K3	PC2
A4	PB12	D4	TDO	G4	PC9	K4	PC0
A5	PB13	D5	TDI	G5	VDDIO	K5	PA27/PGMD15
A6	PB16	D6	PB17	G6	GND	K6	PA26/PGMD14
A7	PB22	D7	PB26	G7	GND	K7	GND
A8	PB23	D8	PA14/PGMD2	G8	GND	K8	VDDCORE
A9	PB25	D9	PA12/PGMD0	G9	GND	K9	VDDFLASH
A10	PB29	D10	PA11/PGMM3	G10	AD4	K10	VDDIO
A11	PB30	D11	PA8/PGMM0	G11	VDDIN	K11	VDDIO
A12	PB31	D12	PA7/PGMNVALID	G12	VDDOUT	K12	PA18/PGMD6/AD1
B1	PB6	E1	PC22	H1	PC15	L1	SDCK
B2	PB3	E2	PC23	H2	PC14	L2	PC7
B3	PB4	E3	NRST	H3	PC13	L3	PC4
B4	PB10	E4	TCK	H4	VDDCORE	L4	PC1
B5	PB14	E5	ERASE	H5	VDDCORE	L5	PA29
B6	PB18	E6	TEST	H6	GND	L6	PA24/PGMD12
B7	PB20	E7	VDDCORE	H7	GND	L7	PA21/PGMD9
B8	PB24	E8	VDDCORE	H8	GND	L8	ADVREF
B9	PB28	E9	GND	H9	GND	L9	VDDFLASH
B10	PA4/PGMNCMD	E10	PA9/PGMM1	H10	PA19/PGMD7/AD2	L10	VDDFLASH
B11	PA0/PGMEN0	E11	PA10/PGMM2	H11	PA20/PGMD8/AD3	L11	PA17/PGMD5/AD0
B12	PA1/PGMEN1	E12	PA13/PGMD1	H12	VDDIO	L12	GND
C1	PB0	F1	PC21	J1	PC12	M1	PC8
C2	PB1	F2	PC20	J2	PC10	M2	PC5
C3	PB5	F3	PC19	J3	PA30	M3	PC3
C4	PB11	F4	JTAGSEL	J4	PA28	M4	PA31
C5	PB15	F5	TMS	J5	PA23/PGMD11	M5	PA25/PGMD13
C6	PB19	F6	VDDIO	J6	PA22/PGMD10	M6	DM
C7	PB21	F7	GND	J7	AD6	M7	DP
C8	PB27	F8	GND	J8	AD7	M8	GND
C9	PA6/PGMNOE	F9	GND	J9	VDDCORE	M9	XIN/PGMCK
C10	PA5/PGMRDY	F10	AD5	J10	VDDCORE	M10	XOUT
C11	PA2/PGMEN2	F11	PA15/PGMD3	J11	VDDCORE	M11	PLLRC
C12	PA3	F12	PA16/PGMD4	J12	VDDIO	M12	VDDPLL

5. Power Considerations

5.1 Power Supplies

The SAM7SE512/256/32 has six types of power supply pins and integrates a voltage regulator, allowing the device to be supplied with only one voltage. The six power supply pin types are:

- VDDIN pin. It powers the voltage regulator and the ADC; voltage ranges from 3.0V to 3.6V, 3.3V nominal.
- VDDOUT pin. It is the output of the 1.8V voltage regulator.
- VDDIO pin. It powers the I/O lines; two voltage ranges are supported:
  - from 3.0V to 3.6V, 3.3V nominal
  - or from 1.65V to 1.95V, 1.8V nominal.
- VDDFLASH pin. It powers the USB transceivers and a part of the Flash. It is required for the Flash to operate correctly; voltage ranges from 3.0V to 3.6V, 3.3V nominal.
- VDDCORE pins. They power the logic of the device; voltage ranges from 1.65V to 1.95V, 1.8V typical. It can be connected to the VDDOUT pin with decoupling capacitor. VDDCORE is required for the device, including its embedded Flash, to operate correctly.
- VDDPLL pin. It powers the oscillator and the PLL. It can be connected directly to the VDDOUT pin.

In order to decrease current consumption, if the voltage regulator and the ADC are not used, VDDIN, ADVREF, AD4, AD5, AD6 and AD7 should be connected to GND. In this case VDDOUT should be left unconnected.

No separate ground pins are provided for the different power supplies. Only GND pins are provided and should be connected as shortly as possible to the system ground plane.

5.2 Power Consumption

The SAM7SE512/256/32 has a static current of less than 60 µA on VDDCORE at 25°C, including the RC oscillator, the voltage regulator and the power-on reset when the brownout detector is deactivated. Activating the brownout detector adds 20 µA static current.

The dynamic power consumption on VDDCORE is less than 80 mA at full speed when running out of the Flash. Under the same conditions, the power consumption on VDDFLASH does not exceed 10 mA.

5.3 Voltage Regulator

The SAM7SE512/256/32 embeds a voltage regulator that is managed by the System Controller.

In Normal Mode, the voltage regulator consumes less than 100 µA static current and draws 100 mA of output current.

The voltage regulator also has a Low-power Mode. In this mode, it consumes less than 20 µA static current and draws 1 mA of output current.

Adequate output supply decoupling is mandatory for VDDOUT to reduce ripple and avoid oscillations. The best way to achieve this is to use two capacitors in parallel:

- One external 470 pF (or 1 nF) NPO capacitor should be connected between VDDOUT and GND as close to the chip as possible.

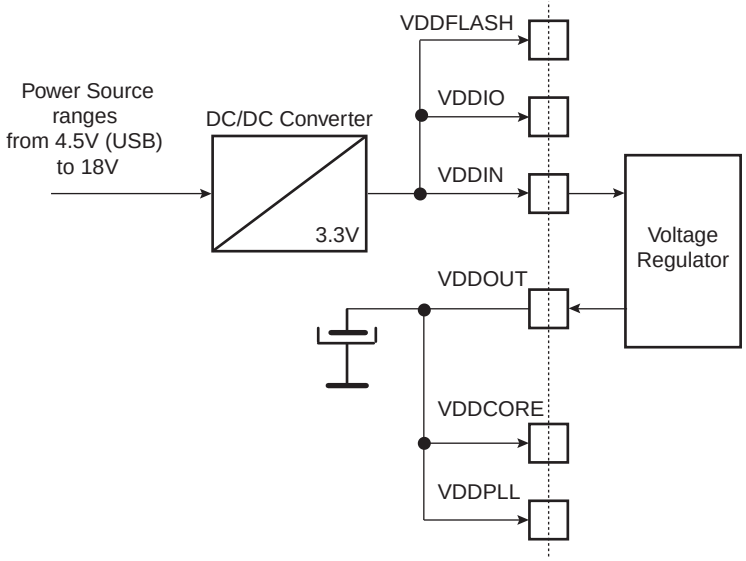
- One external 2.2 µF (or 3.3 µF) X7R capacitor should be connected between VDDOUT and GND.

Adequate input supply decoupling is mandatory for VDDIN in order to improve startup stability and reduce source voltage drop. The input decoupling capacitor should be placed close to the chip. For example, two capacitors can be used in parallel: 100 nF NPO and 4.7 µF X7R.

5.4 Typical Powering Schematics

The SAM7SE512/256/32 supports a 3.3V single supply mode. The internal regulator input connected to the 3.3V source and its output feeds VDDCORE and the VDDPLL. Figure 5-1 shows the power schematics to be used for USB bus-powered systems.

Figure 5-1. 3.3V System Single Power Supply Schematic



6. I/O Lines Considerations

6.1 JTAG Port Pins

TMS, TDI and TCK are Schmitt trigger inputs. TMS, TDI and TCK do not integrate a pull-up resistor.

TDO is an output, driven at up to VDDIO, and has no pull-up resistor.

The JTAGSEL pin is used to select the JTAG boundary scan when asserted at a high level. The JTAGSEL pin integrates a permanent pull-down resistor of about 15 kΩ.

To eliminate any risk of spuriously entering the JTAG boundary scan mode due to noise on JTAGSEL, it should be tied externally to GND if boundary scan is not used, or put in place an external low value resistor (such as 1 kΩ).

6.2 Test Pin

The TST pin is used for manufacturing test or fast programming mode of the SAM7SE512/256/32 when asserted high. The TST pin integrates a permanent pull-down resistor of about 15 kΩ to GND.

To eliminate any risk of entering the test mode due to noise on the TST pin, it should be tied to GND if the FFPI is not used, or put in place an external low value resistor (such as 1 kΩ).

To enter fast programming mode, the TST pin and the PA0 and PA1 pins should be tied high and PA2 tied low.

Driving the TST pin at a high level while PA0 or PA1 is driven at 0 leads to unpredictable results.

6.3 Reset Pin

The NRST pin is bidirectional with an open-drain output buffer. It is handled by the on-chip reset controller and can be driven low to provide a reset signal to the external components or asserted low externally to reset the microcontroller. There is no constraint on the length of the reset pulse, and the reset controller can guarantee a minimum pulse length. This allows connection of a simple push-button on the NRST pin as system user reset, and the use of the NRST signal to reset all the components of the system.

An external power-on reset can drive this pin during the start-up instead of using the internal power-on reset circuit.

The NRST pin integrates a permanent pull-up of about 100 kΩ resistor to VDDIO.

This pin has Schmitt trigger input.

6.4 ERASE Pin

The ERASE pin is used to re-initialize the Flash content and some of its NVM bits. It integrates a permanent pull-down resistor of about 15 kΩ to GND.

To eliminate any risk of erasing the Flash due to noise on the ERASE pin, it should be tied externally to GND, which prevents erasing the Flash from the application, or put in place an external low value resistor (such as 1 kΩ).

This pin is debounced by the RC oscillator to improve the glitch tolerance. When the pin is tied to high during less than 100 ms, ERASE pin is not taken into account. The pin must be tied high during more than 220 ms to perform the re-initialization of the Flash.

6.5 SDCK Pin

The SDCK pin is dedicated to the SDRAM Clock and is an output-only without pull-up. Maximum Output Frequency of this pad is 48 MHz at 3.0V and 25 MHz at 1.65V with a maximum load of 30 pF.

6.6 PIO Controller lines

All the I/O lines PA0 to PA31, PB0 to PB31, PC0 to PC23 integrate a programmable pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the PIO controllers.

Typical pull-up value is 100 kΩ

All the I/O lines have schmitt trigger inputs.

6.7 I/O Lines Current Drawing

The PIO lines PA0 to PA3 are high-drive current capable. Each of these I/O lines can drive up to 16 mA permanently.

The remaining I/O lines can draw only 8 mA.

However, the total current drawn by all the I/O lines cannot exceed 300 mA.

7. Processor and Architecture

7.1 ARM7TDMI Processor

- RISC processor based on ARMv4T Von Neumann architecture
  - Runs at up to 55 MHz, providing 0.9 MIPS/MHz (core supplied with 1.8V)
- Two instruction sets
  - ARM® high-performance 32-bit instruction set
  - Thumb® high code density 16-bit instruction set
- Three-stage pipeline architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

7.2 Debug and Test Features

- EmbeddedICE™ (Integrated embedded in-circuit emulator)
  - Two watchpoint units
  - Test access port accessible through a JTAG protocol
  - Debug communication channel
- Debug Unit
  - Two-pin UART
  - Debug communication channel interrupt handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on all digital pins

7.3 Memory Controller

- Programmable Bus Arbiter
  - Handles requests from the ARM7TDMI and the Peripheral DMA Controller
- Address decoder provides selection signals for
  - Four internal 1 Mbyte memory areas
  - One 256-Mbyte embedded peripheral area
  - Eight external 256-Mbyte memory areas
- Abort Status Registers
  - Source, Type and all parameters of the access leading to an abort are saved
  - Facilitates debug by detection of bad pointers
- Misalignment Detector
  - Alignment checking of all data accesses
  - Abort generation in case of misalignment
- Remap Command
  - Remaps the SRAM in place of the embedded non-volatile memory
  - Allows handling of dynamic exception vectors
- 16-area Memory Protection Unit (Internal Memory and peripheral protection only)

- Individually programmable size between 1K Byte and 1M Byte
- Individually programmable protection against write and/or user access
- Peripheral protection against write and/or user access

- Embedded Flash Controller
  - Embedded Flash interface, up to three programmable wait states
  - Prefetch buffer, buffering and anticipating the 16-bit requests, reducing the required wait states
  - Key-protected program, erase and lock/unlock sequencer
  - Single command for erasing, programming and locking operations
  - Interrupt generation in case of forbidden operation

7.4 External Bus Interface

- Integrates Three External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
  - ECC Controller
- Additional Logic for NAND Flash and CompactFlash® Support
  - NAND Flash support: 8-bit as well as 16-bit devices are supported
  - CompactFlash support: all modes (Attribute Memory, Common Memory, I/O, True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and -ATA SEL (True IDE mode) are not handled.
- Optimized External Bus:
  - 16- or 32-bit Data Bus (32-bit Data Bus for SDRAM only)
  - Up to 23-bit Address Bus, Up to 8-Mbytes Addressable
  - Up to 8 Chip Selects, each reserved to one of the eight Memory Areas
  - Optimized pin multiplexing to reduce latencies on External Memories
- Configurable Chip Select Assignment:
  - Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2, Optional CompactFlash Support
  - Static Memory Controller on NCS3, NCS5 - NCS6, Optional NAND Flash Support
  - Static Memory Controller on NCS4, Optional CompactFlash Support
  - Static Memory Controller on NCS7

7.5 Static Memory Controller

- External memory mapping, 512-Mbyte address space
- 8-, or 16-bit Data Bus
- Up to 8 Chip Select Lines
- Multiple Access Modes supported
  - Byte Write or Byte Select Lines
  - Two different Read Protocols for each Memory Bank

- Multiple device adaptability
  - Compliant with LCD Module
  - Compliant with PSRAM in synchronous operations
  - Programmable Setup Time Read/Write
  - Programmable Hold Time Read/Write
- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time

7.6 SDRAM Controller

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

7.7 Error Corrected Code Controller

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

7.8 Peripheral DMA Controller

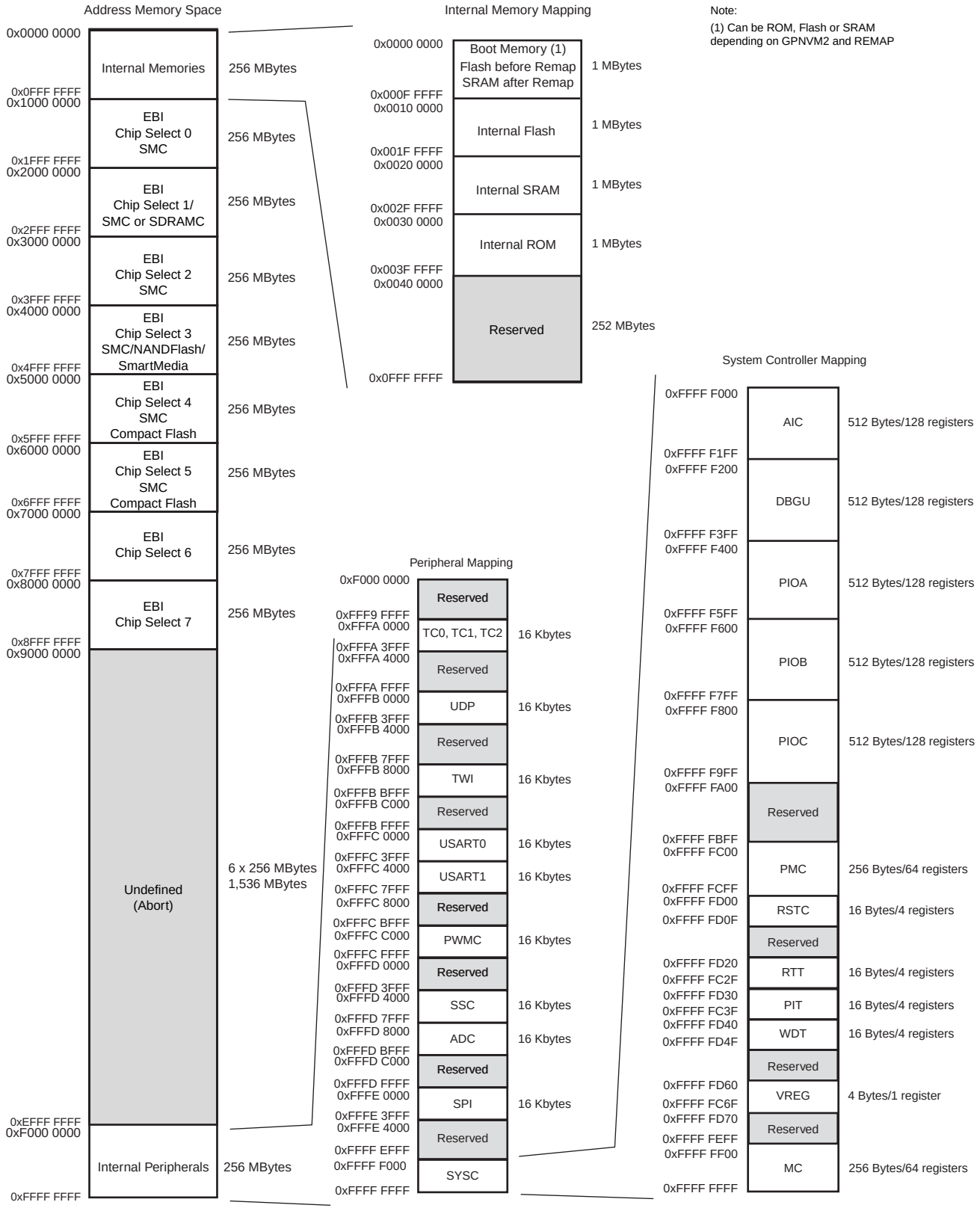
- Handles data transfer between peripherals and memories
- Eleven channels
  - Two for each USART
  - Two for the Debug Unit
  - Two for the Serial Synchronous Controller
  - Two for the Serial Peripheral Interface
  - One for the Analog-to-digital Converter
- Low bus arbitration overhead
  - One Master Clock cycle needed for a transfer from memory to peripheral
  - Two Master Clock cycles needed for a transfer from peripheral to memory
- Next Pointer management for reducing interrupt latency requirements
- Peripheral DMA Controller (PDC) priority is as follows (from the highest priority to the lowest):

Receive	DBGU
Receive	USART0
Receive	USART1
Receive	SSC
Receive	ADC
Receive	SPI
Transmit	DBGU
Transmit	USART0
Transmit	USART1
Transmit	SSC
Transmit	SPI

8. Memories

- 512 Kbytes of Flash Memory (SAM7SE512)
  - dual plane
  - two contiguous banks of 1024 pages of 256 bytes
  - Fast access time, 30 MHz single-cycle access in Worst Case conditions
  - Page programming time: 6 ms, including page auto-erase
  - Page programming without auto-erase: 3 ms
  - Full chip erase time: 15 ms
  - 10,000 write cycles, 10-year data retention capability
  - 32 lock bits, each protecting 32 lock regions of 64 pages
  - Protection Mode to secure contents of the Flash
- 256 Kbytes of Flash Memory (SAM7SE256)
  - single plane
  - one bank of 1024 pages of 256 bytes
  - Fast access time, 30 MHz single-cycle access in Worst Case conditions
  - Page programming time: 6 ms, including page auto-erase
  - Page programming without auto-erase: 3 ms
  - Full chip erase time: 15 ms
  - 10,000 cycles, 10-year data retention capability
  - 16 lock bits, each protecting 16 lock regions of 64 pages
  - Protection Mode to secure contents of the Flash
- 32 Kbytes of Flash Memory (SAM7SE32)
  - single plane
  - one bank of 256 pages of 128 bytes
  - Fast access time, 30 MHz single-cycle access in Worst Case conditions
  - Page programming time: 6 ms, including page auto-erase
  - Page programming without auto-erase: 3 ms
  - Full chip erase time: 15 ms
  - 10,000 cycles, 10-year data retention capability
  - 8 lock bits, each protecting 8 lock regions of 32 pages
  - Protection Mode to secure contents of the Flash
- 32 Kbytes of Fast SRAM (SAM7SE512/256)
  - Single-cycle access at full speed
- 8 Kbytes of Fast SRAM (SAM7SE32)
  - Single-cycle access at full speed

Figure 8-1. SAM7SE Memory Mapping



A first level of address decoding is performed by the Memory Controller, i.e., by the implementation of the Advanced System Bus (ASB) with additional features.

Decoding splits the 4G bytes of address space into 16 areas of 256M bytes. The areas 1 to 8 are directed to the EBI that associates these areas to the external chip selects NC0 to NCS7. The area 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1M byte of internal memory area. The area 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

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

8.1 Embedded Memories

8.1.1 Internal Memories

8.1.1.1 Internal SRAM

The SAM7SE512/256 embeds a high-speed 32-Kbyte SRAM bank. The SAM7SE32 embeds a high-speed 8-Kbyte SRAM bank. After reset and until the Remap Command is performed, the SRAM is only accessible at address 0x0020 0000. After Remap, the SRAM also becomes available at address 0x0.

8.1.1.2 Internal ROM

The SAM7SE512/256/32 embeds an Internal ROM. At any time, the ROM is mapped at address 0x30 0000. The ROM contains the FFPI and the SAM-BA boot program.

8.1.1.3 Internal Flash

- The SAM7SE512 features two banks of 256 Kbytes of Flash.
- The SAM7SE256 features one bank of 256 Kbytes of Flash.
- The SAM7SE32 features one bank of 32 Kbytes of Flash.

At any time, the Flash is mapped to address 0x0010 0000.

A general purpose NVM (GPNVM) bit is used to boot either on the ROM (default) or from the Flash.

This GPNVM bit can be cleared or set respectively through the commands “Clear General-purpose NVM Bit” and “Set General-purpose NVM Bit” of the EFC User Interface.

Setting the GPNVM bit 2 selects the boot from the Flash, clearing it selects the boot from the ROM. Asserting ERASE clears the GPNVM bit 2 and thus selects the boot from the ROM by default.

Figure 8-2. Internal Memory Mapping with GPNVM Bit 2 = 0 (default)

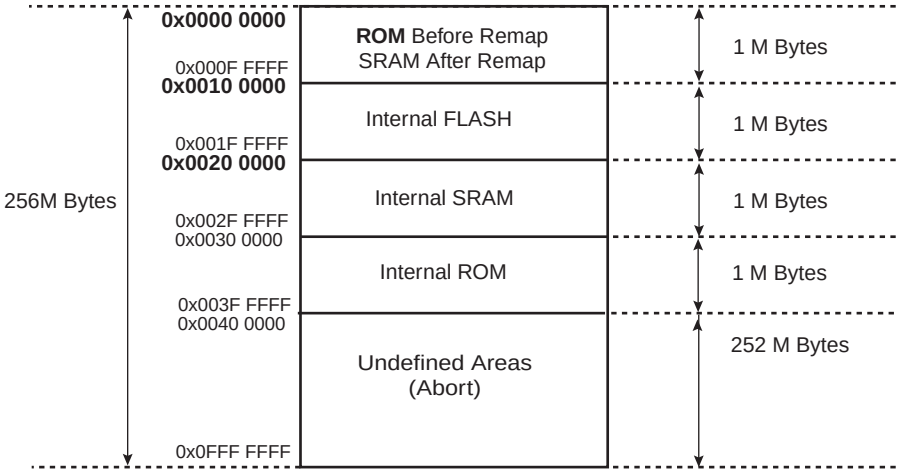
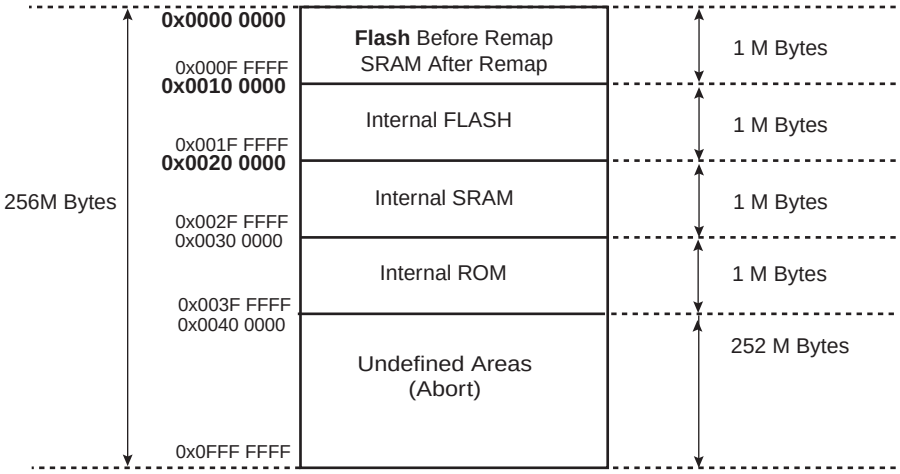


Figure 8-3. Internal Memory Mapping with GPNVM Bit 2 = 1



8.1.2 Embedded Flash

8.1.2.1 Flash Overview

The Flash of the SAM7SE512 is organized in two banks (dual plane) of 1024 pages of 256 bytes. It reads as 131,072 32-bit words.

The Flash of the SAM7SE256 is organized in 1024 pages (single plane) of 256 bytes. It reads as 65,536 32-bit words.

The Flash of the SAM7SE32 is organized in 256 pages (single plane) of 128 bytes. It reads as 8192 32-bit words.

The Flash of the SAM7SE32 contains a 128-byte write buffer, accessible through a 32-bit interface.

The Flash of the SAM7SE512/256 contains a 256-byte write buffer, accessible through a 32-bit interface.

The Flash benefits from the integration of a power reset cell and from the brownout detector. This prevents code corruption during power supply changes, even in the worst conditions.

8.1.2.2 *Embedded Flash Controller*

The Embedded Flash Controller (EFC) manages accesses performed by the masters of the system. It enables reading the Flash and writing the write buffer. It also contains a User Interface, mapped within the Memory Controller on the APB. The User Interface allows:

- programming of the access parameters of the Flash (number of wait states, timings, etc.)
- starting commands such as full erase, page erase, page program, NVM bit set, NVM bit clear, etc.
- getting the end status of the last command
- getting error status
- programming interrupts on the end of the last commands or on errors

The Embedded Flash Controller also provides a dual 32-bit Prefetch Buffer that optimizes 16-bit access to the Flash. This is particularly efficient when the processor is running in Thumb mode.

- Two EFCs (EFC0 and EFC1) are embedded in the SAM7SE512 to control each plane of 256 KBytes. Dual plane organization allows concurrent Read and Program.
- One EFC (EFC0) is embedded in the SAM7SE256 to control the single plane 256 KBytes.
- One EFC (EFC0) is embedded in the SAM7SE32 to control the single plane 32 KBytes.

8.1.2.3 *Lock Regions*

The SAM7SE512 Embedded Flash Controller manages 32 lock bits to protect 32 regions of the flash against inadvertent flash erasing or programming commands. The SAM7SE512 contains 32 lock regions and each lock region contains 64 pages of 256 bytes. Each lock region has a size of 16 Kbytes.

The SAM7SE256 Embedded Flash Controller manages 16 lock bits to protect 16 regions of the flash against inadvertent flash erasing or programming commands. The SAM7SE256 contains 16 lock regions and each lock region contains 64 pages of 256 bytes. Each lock region has a size of 16 Kbytes.

The SAM7SE32 Embedded Flash Controller manages 8 lock bits to protect 8 regions of the flash against inadvertent flash erasing or programming commands. The SAM7SE32 contains 8 lock regions and each lock region contains 32 pages of 128 bytes. Each lock region has a size of 4 Kbytes.

If a locked-region’s erase or program command occurs, the command is aborted and the EFC trigs an interrupt.

The 32 (SAM7SE512), 16 (SAM7SE256) or 8 (SAM7SE32) NVM bits are software programmable through the EFC User Interface. The command “Set Lock Bit” enables the protection. The command “Clear Lock Bit” unlocks the lock region.

Asserting the ERASE pin clears the lock bits, thus unlocking the entire Flash.

8.1.2.4 *Security Bit Feature*

The SAM7SE512/256/32 features a security bit, based on a specific NVM-bit. When the security is enabled, any access to the Flash, either through the ICE interface or through the Fast Flash Programming Interface, is forbidden.

The security bit can only be enabled through the Command “Set Security Bit” of the EFC User Interface. Disabling the security bit can only be achieved by asserting the ERASE pin at 1 and after a full flash erase is performed. When the security bit is deactivated, all accesses to the flash are permitted.

It is important to note that the assertion of the ERASE pin should always be longer than 200 ms.

As the ERASE pin integrates a permanent pull-down, it can be left unconnected during normal operation. However, it is safer to connect it directly to GND for the final application.

8.1.2.5 Non-volatile Brownout Detector Control

Two general purpose NVM (GPNVM) bits are used for controlling the brownout detector (BOD), so that even after a power loss, the brownout detector operations remain in their state.

These two GPNVM bits can be cleared or set respectively through the commands “Clear General-purpose NVM Bit” and “Set General-purpose NVM Bit” of the EFC User Interface.

- GPNVM bit 0 is used as a brownout detector enable bit. Setting the GPNVM bit 0 enables the BOD, clearing it disables the BOD. Asserting ERASE clears the GPNVM bit 0 and thus disables the brownout detector by default.
- GPNVM bit 1 is used as a brownout reset enable signal for the reset controller. Setting the GPNVM bit 1 enables the brownout reset when a brownout is detected, Clearing the GPNVM bit 1 disables the brownout reset. Asserting ERASE disables the brownout reset by default.

8.1.2.6 Calibration Bits

Sixteen NVM bits are used to calibrate the brownout detector and the voltage regulator. These bits are factory configured and cannot be changed by the user. The ERASE pin has no effect on the calibration bits.

8.1.3 Fast Flash Programming Interface

The Fast Flash Programming Interface allows programming the device through either a serial JTAG interface or through a multiplexed fully-handshaked parallel port. It allows gang-programming with market-standard industrial programmers.

The FFPI supports read, page program, page erase, full erase, lock, unlock and protect commands.

The Fast Flash Programming Interface is enabled and the Fast Programming Mode is entered when the TST pin and the PA0 and PA1 pins are all tied high and PA2 tied to low.

- The Flash of the SAM7SE512 is organized in 2048 pages of 256 bytes (dual plane). It reads as 131,072 32-bit words.
- The Flash of the SAM7SE256 is organized in 1024 pages of 256 bytes (single plane). It reads as 65,536 32-bit words.
- The Flash of the SAM7SE32 is organized in 256 pages of 128 bytes (single plane). It reads as 32,768 32-bit words.
- The Flash of the SAM7SE512/256 contains a 256-byte write buffer, accessible through a 32-bit interface.
- The Flash of the SAM7SE32 contains a 128-byte write buffer, accessible through a 32-bit interface.

8.1.4 SAM-BA® Boot

The SAM-BA Boot is a default Boot Program which provides an easy way to program in-situ the on-chip Flash memory.

The SAM-BA Boot Assistant supports serial communication via the DBGU or the USB Device Port.

- Communication via the DBGU supports a wide range of crystals from 3 to 20 MHz via software auto-detection.
- Communication via the USB Device Port is limited to an 18.432 MHz crystal.

The SAM-BA Boot provides an interface with SAM-BA Graphic User Interface (GUI).

The SAM-BA Boot is in ROM and is mapped in Flash at address 0x0 when GPNVM bit 2 is set to 0.

8.2 External Memories

The external memories are accessed through the External Bus Interface.

Refer to the memory map in Figure 8-1 on page 22.

9. System Controller

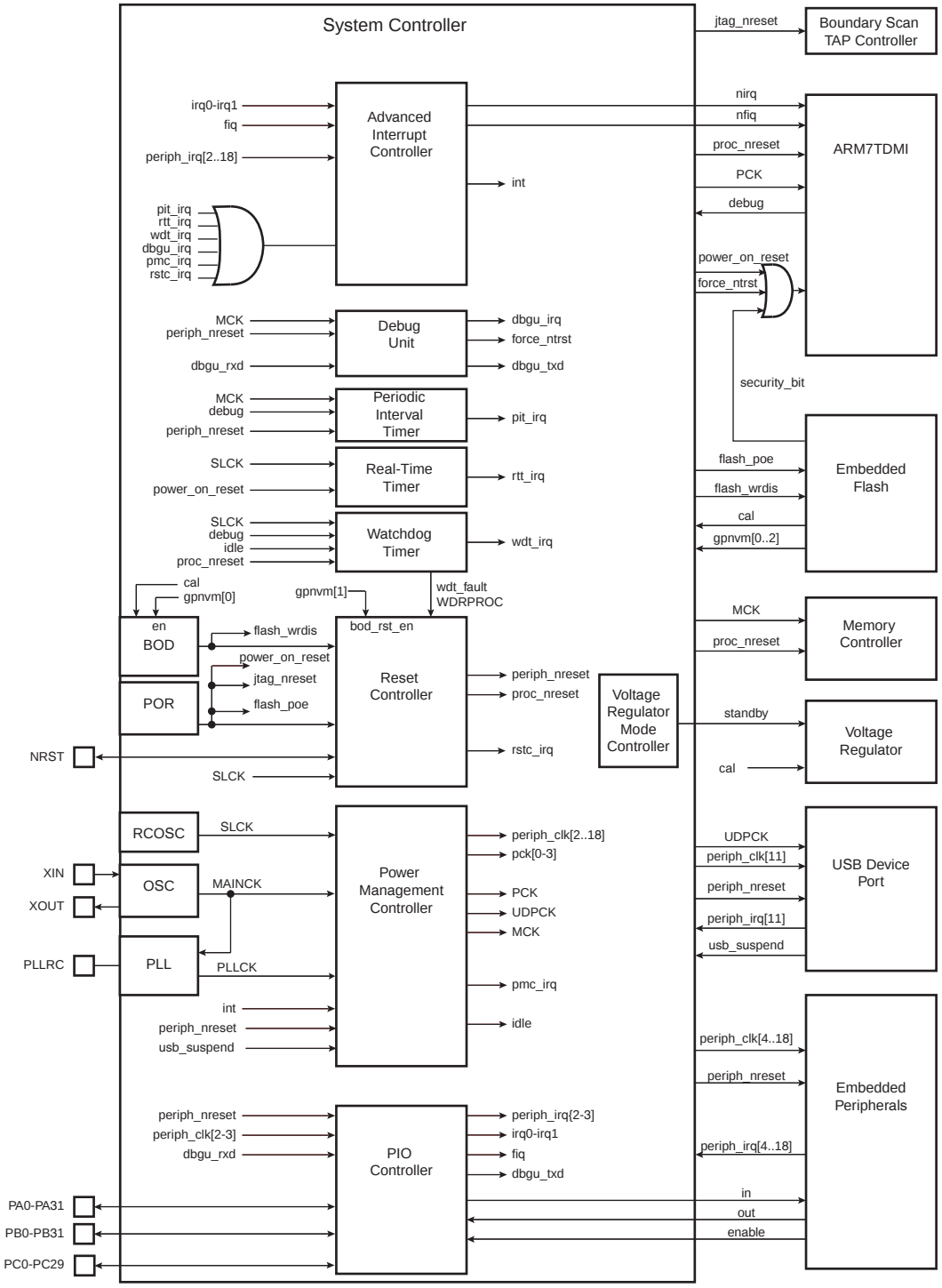
The System Controller manages all vital blocks of the microcontroller: interrupts, clocks, power, time, debug and reset.

The System Controller peripherals are all mapped to the highest 4 Kbytes of address space, between addresses 0xFFFF F000 and 0xFFFF FFFF.

Figure 9-1 on page 29 shows the System Controller Block Diagram.

Figure 8-1 on page 22 shows the mapping of the User Interface of the System Controller peripherals. Note that the Memory Controller configuration user interface is also mapped within this address space.

Figure 9-1. System Controller Block Diagram



9.1 Reset Controller

- Based on one power-on reset cell and a double brownout detector
- Status of the last reset, either Power-up Reset, Software Reset, User Reset, Watchdog Reset, Brownout Reset
- Controls the internal resets and the NRST pin output
- Allows to shape a signal on the NRST line, guaranteeing that the length of the pulse meets any requirement.

9.1.1 Brownout Detector and Power On Reset

The SAM7SE512/256/32 embeds one brownout detection circuit and a power-on reset cell. The power-on reset is supplied with and monitors VDDCORE.

Both signals are provided to the Flash to prevent any code corruption during power-up or power-down sequences or if brownouts occur on the VDDCORE power supply.

The power-on reset cell has a limited-accuracy threshold at around 1.5V. Its output remains low during power-up until VDDCORE goes over this voltage level. This signal goes to the reset controller and allows a full re-initialization of the device.

The brownout detector monitors the VDDCORE and VDDFLASH levels during operation by comparing it to a fixed trigger level. It secures system operations in the most difficult environments and prevents code corruption in case of brownout on the VDDCORE or VDDFLASH.

When the brownout detector is enabled and VDDCORE decreases to a value below the trigger level (Vbot18-, defined as Vbot18 - hyst/2), the brownout output is immediately activated.

When VDDCORE increases above the trigger level (Vbot18+, defined as Vbot18 + hyst/2), the reset is released. The brownout detector only detects a drop if the voltage on VDDCORE stays below the threshold voltage for longer than about 1µs.

The VDDCORE threshold voltage has a hysteresis of about 50 mV, to ensure spike free brownout detection. The typical value of the brownout detector threshold is 1.68V with an accuracy of ± 2% and is factory calibrated.

When the brownout detector is enabled and VDDFLASH decreases to a value below the trigger level (Vbot33-, defined as Vbot33 - hyst/2), the brownout output is immediately activated.

When VDDFLASH increases above the trigger level (Vbot33+, defined as Vbot33 + hyst/2), the reset is released. The brownout detector only detects a drop if the voltage on VDDCORE stays below the threshold voltage for longer than about 1µs.

The VDDFLASH threshold voltage has a hysteresis of about 50 mV, to ensure spike free brownout detection. The typical value of the brownout detector threshold is 2.80V with an accuracy of ± 3.5% and is factory calibrated.

The brownout detector is low-power, as it consumes less than 20 µA static current. However, it can be deactivated to save its static current. In this case, it consumes less than 1µA. The deactivation is configured through the GPNVM bit 0 of the Flash.

9.2 Clock Generator

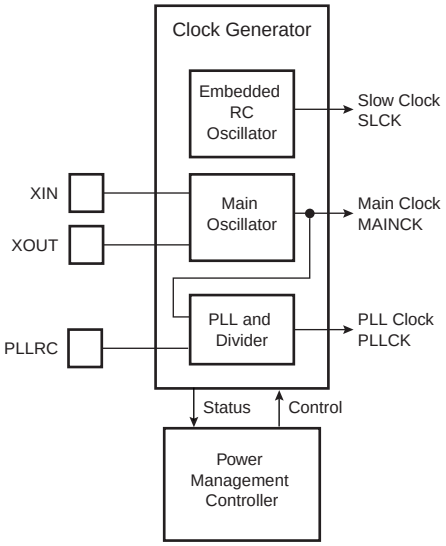
The Clock Generator embeds one low-power RC Oscillator, one Main Oscillator and one PLL with the following characteristics:

- RC Oscillator ranges between 22 KHz and 42 KHz

- Main Oscillator frequency ranges between 3 and 20 MHz
- Main Oscillator can be bypassed
- PLL output ranges between 80 and 220 MHz

It provides SLCK, MAINCK and PLLCK.

Figure 9-2. Clock Generator Block Diagram



9.3 Power Management Controller

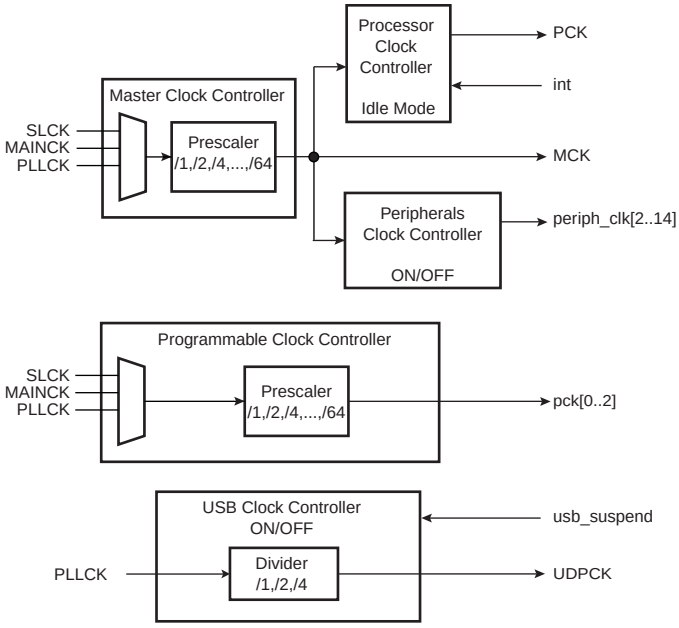
The Power Management Controller uses the Clock Generator outputs to provide:

- the Processor Clock PCK
- the Master Clock MCK
- the USB Clock UDPCK
- all the peripheral clocks, independently controllable
- three programmable clock outputs

The Master Clock (MCK) is programmable from a few hundred Hz to the maximum operating frequency of the device.

The Processor Clock (PCK) switches off when entering processor idle mode, thus allowing reduced power consumption while waiting for an interrupt.

Figure 9-3. Power Management Controller Block Diagram



9.4 Advanced Interrupt Controller

- Controls the interrupt lines (nIRQ and nFIQ) of an ARM Processor
- Individually maskable and vectored interrupt sources
  - Source 0 is reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is reserved for system peripherals (RTT, PIT, EFC, PMC, DBGU, etc.)
  - Other sources control the 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 nIRQ of the processor
  - Handles priority of the interrupt sources
  - Higher priority interrupts can be served during service of lower priority interrupt
- Vectoring
  - Optimizes interrupt service routine branch and execution
  - One 32-bit vector register per interrupt source
  - Interrupt vector register reads the corresponding current interrupt vector
- Protect Mode
  - Easy debugging by preventing automatic operations
- Fast Forcing
  - Permits redirecting any interrupt source on the fast interrupt
- General Interrupt Mask
  - Provides processor synchronization on events without triggering an interrupt

9.5 Debug Unit

- Comprises:
  - One two-pin UART
  - One Interface for the Debug Communication Channel (DCC) support
  - One set of Chip ID Registers
  - One Interface providing ICE Access Prevention
- Two-pin UART
  - USART-compatible User Interface
  - Programmable Baud Rate Generator
  - Parity, Framing and Overrun Error
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
- Debug Communication Channel Support
  - Offers visibility of COMMRX and COMMTX signals from the ARM Processor
- Chip ID Registers
  - Identification of the device revision, sizes of the embedded memories, set of peripherals
  - Chip ID is 0x272A 0A40 (VERSION 0) for SAM7SE512
  - Chip ID is 0x272A 0940 (VERSION 0) for SAM7SE256
  - Chip ID is 0x2728 0340 (VERSION 0) for SAM7SE32

9.6 Periodic Interval Timer

- 20-bit programmable counter plus 12-bit interval counter

9.7 Watchdog Timer

- 12-bit key-protected Programmable Counter running on prescaled SLCK
- Provides reset or interrupt signals to the system
- Counter may be stopped while the processor is in debug state or in idle mode

9.8 Real-time Timer

- 32-bit free-running counter with alarm running on prescaled SLCK
- Programmable 16-bit prescaler for SLCK accuracy compensation

9.9 PIO Controllers

- Three PIO Controllers. PIO A and B each control 32 I/O lines and PIO C controls 24 I/O lines.
- Fully programmable through set/clear registers
- Multiplexing of two peripheral functions per I/O line
- For each I/O line (whether assigned to a peripheral or used as general-purpose I/O)
  - Input change interrupt
  - Half a clock period glitch filter
  - Multi-drive option enables driving in open drain
  - Programmable pull-up on each I/O line
  - Pin data status register, supplies visibility of the level on the pin at any time

- Synchronous output, provides Set and Clear of several I/O lines in a single write

9.10 Voltage Regulator Controller

The purpose of this controller is to select the Power Mode of the Voltage Regulator between Normal Mode (bit 0 is cleared) or Standby Mode (bit 0 is set).

10. Peripherals

10.1 User Interface

The User Peripherals are mapped in the 256 MBytes of the address space between 0xF000 0000 and 0xFFFF EFFF. Each peripheral is allocated 16 Kbytes of address space.

A complete memory map is presented in Figure 8-1 on page 22.

10.2 Peripheral Identifiers

The SAM7SE512/256/32 embeds a wide range of peripherals. Table 10-1 defines the Peripheral Identifiers of the SAM7SE512/256/32. Unique peripheral identifiers are defined for both the Advanced Interrupt Controller and the Power Management Controller.

Table 10-1. Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC[1]		
2	PIOA	Parallel I/O Controller A	
3	PIOB	Parallel I/O Controller B	
4	PIOC	Parallel I/O Controller C	
5	SPI	Serial Peripheral Interface 0	
6	US0	USART 0	
7	US1	USART 1	
8	SSC	Synchronous Serial Controller	
9	TWI	Two-wire Interface	
10	PWMC	PWM Controller	
11	UDP	USB Device Port	
12	TC0	Timer/Counter 0	
13	TC1	Timer/Counter 1	
14	TC2	Timer/Counter 2	
15	ADC[1]	Analog-to Digital Converter	
16-28	reserved		
29	AIC	Advanced Interrupt Controller	IRQ0
30	AIC	Advanced Interrupt Controller	IRQ1

Note: 1. Setting SYSC and ADC bits in the clock set/clear registers of the PMC has no effect. The System Controller is continuously clocked. The ADC clock is automatically started for the first conversion. In Sleep Mode the ADC clock is automatically stopped after each conversion.

10.3 Peripheral Multiplexing on PIO Lines

The SAM7SE512/256/32 features three PIO controllers, PIOA, PIOB and PIOC, that multiplex the I/O lines of the peripheral set.

PIO Controller A and B control 32 lines; PIO Controller C controls 24 lines. Each line can be assigned to one of two peripheral functions, A or B. Some of them can also be multiplexed with the analog inputs of the ADC Controller.

Table 10-2 on page 37 defines how the I/O lines of the peripherals A and B or the analog inputs are multiplexed on the PIO Controller A, B and C. The two columns “Function” and “Comments” have been inserted for the user’s own comments; they may be used to track how pins are defined in an application.

Note that some peripheral functions that are output only may be duplicated in the table.

At reset, all I/O lines are automatically configured as input with the programmable pull-up enabled, so that the device is maintained in a static state as soon as a reset is detected.

10.4 PIO Controller A Multiplexing

Table 10-2. Multiplexing on PIO Controller A

PIO Controller A				Application Usage	
I/O Line	**Peripheral A**	**Peripheral B**	**Comments**	**Function**	**Comments**
PA0	PWM0	A0/NBS0	High-Drive		
PA1	PWM1	A1/NBS2	High-Drive		
PA2	PWM2	A2	High-Drive		
PA3	TWD	A3	High-Drive		
PA4	TWCK	A4			
PA5	RXD0	A5			
PA6	TXD0	A6			
PA7	RTS0	A7			
PA8	CTS0	A8			
PA9	DRXD	A9			
PA10	DTXD	A10			
PA11	NPCS0	A11			
PA12	MISO	A12			
PA13	MOSI	A13			
PA14	SPCK	A14			
PA15	TF	A15			
PA16	TK	A16/BA0			
PA17	TD	A17/BA1	AD0		
PA18	RD	NBS3/CFIOW	AD1		
PA19	RK	NCS4/CFCS0	AD2		
PA20	RF	NCS2/CFCS1	AD3		
PA21	RXD1	NCS6/CFCE2			
PA22	TXD1	NCS5/CFCE1			
PA23	SCK1	NWR1/NBS1/CFIOR			
PA24	RTS1	SDA10			
PA25	CTS1	SDCKE			
PA26	DCD1	NCS1/SDCS			
PA27	DTR1	SDWE			
PA28	DSR1	CAS			
PA29	RI1	RAS			
PA30	IRQ1	D30			
PA31	NPCS1	D31			

10.5 PIO Controller B Multiplexing

Table 10-3. Multiplexing on PIO Controller B

PIO Controller B				Application Usage	
I/O Line	**Peripheral A**	**Peripheral B**	**Comments**	**Function**	**Comments**
PB0	TIOA0	A0/NBS0			
PB1	TIOB0	A1/NBS2			
PB2	SCK0	A2			
PB3	NPCS3	A3			
PB4	TCLK0	A4			
PB5	NPCS3	A5			
PB6	PCK0	A6			
PB7	PWM3	A7			
PB8	ADTRG	A8			
PB9	NPCS1	A9			
PB10	NPCS2	A10			
PB11	PWM0	A11			
PB12	PWM1	A12			
PB13	PWM2	A13			
PB14	PWM3	A14			
PB15	TIOA1	A15			
PB16	TIOB1	A16/BA0			
PB17	PCK1	A17/BA1			
PB18	PCK2	D16			
PB19	FIQ	D17			
PB20	IRQ0	D18			
PB21	PCK1	D19			
PB22	NPCS3	D20			
PB23	PWM0	D21			
PB24	PWM1	D22			
PB25	PWM2	D23			
PB26	TIOA2	D24			
PB27	TIOB2	D25			
PB28	TCLK1	D26			
PB29	TCLK2	D27			
PB30	NPCS2	D28			
PB31	PCK2	D29			

10.6 PIO Controller C Multiplexing

Multiplexing on PIO Controller C

PIO Controller C				Application Usage	
I/O Line	Peripheral A	Peripheral B	Comments	Function	Comments
PC0	D0				
PC1	D1				
PC2	D2				
PC3	D3				
PC4	D4				
PC5	D5				
PC6	D6				
PC7	D7				
PC8	D8	RTS1			
PC9	D9	DTR1			
PC10	D10	PCK0			
PC11	D11	PCK1			
PC12	D12	PCK2			
PC13	D13				
PC14	D14	NPCS1			
PC15	D15	NCS3/NANDCS			
PC16	A18	NWAIT			
PC17	A19	NANDOE			
PC18	A20	NANDWE			
PC19	A21/NANDALE				
PC20	A22/REG/NANDCLE	NCS7			
PC21		NWR0/NWE/CFWE			
PC22		NRD/CFOE			
PC23	CFRNW	NCS0			

10.7 Serial Peripheral Interface

- Supports communication with external serial devices
  - Four chip selects with external decoder allow communication with up to 15 peripherals
  - Serial memories, such as DataFlash® and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External co-processors
- Master or slave serial peripheral bus interface

- 8- to 16-bit programmable data length per chip select
- Programmable phase and polarity per chip select
- Programmable transfer delays per chip select, between consecutive transfers and between clock and data
- Programmable delay between consecutive transfers
- Selectable mode fault detection
- Maximum frequency at up to Master Clock

10.8 Two Wire Interface

- Master, Multi-Master and Slave Mode Operation
- Compatibility with standard two-wire serial memories
- One, two or three bytes for slave address
- Sequential read/write operations
- Bit Rate: Up to 400 Kbit/s
- General Call Supported in Slave Mode

10.9 USART

- Programmable Baud Rate Generator
- 5- to 9-bit full-duplex synchronous or asynchronous serial communications
  - 1, 1.5 or 2 stop bits in Asynchronous Mode
  - 1 or 2 stop bits in Synchronous Mode
  - Parity generation and error detection
  - Framing error detection, overrun error detection
  - MSB or LSB first
  - Optional break generation and detection
  - By 8 or by 16 over-sampling receiver frequency
  - Hardware handshaking RTS - CTS
  - Modem Signals Management DTR-DSR-DCD-RI on USART1
  - Receiver time-out and transmitter timeguard
  - Multi-drop 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
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo

10.10 Serial Synchronous Controller

- Provides serial synchronous communication links used in audio and telecom applications
- Contains an independent receiver and transmitter and a common clock divider

- Offers a configurable frame sync and data length
- Receiver and transmitter can be programmed to start automatically or on detection of different event on the frame sync signal
- Receiver and transmitter include a data signal, a clock signal and a frame synchronization signal

10.11 Timer Counter

- Three 16-bit Timer Counter Channels
  - Two output compare or one input capture per channel
- Wide range of functions including:
  - Frequency measurement
  - Event counting
  - Interval measurement
  - Pulse generation
  - Delay timing
  - Pulse Width Modulation
  - Up/down capabilities
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five internal clock inputs, as defined in Table 10-4

Table 10-4. Timer Counter Clocks Assignment

TC Clock input	Clock
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	MCK/1024

  - Two multi-purpose input/output signals
  - Two global registers that act on all three TC channels

10.12 PWM Controller

- Four channels, one 16-bit counter per channel
- Common clock generator, providing thirteen different clocks
  - One Modulo n counter providing eleven clocks
  - Two independent linear dividers working on modulo n counter outputs
- Independent channel programming
  - Independent enable/disable commands
  - Independent clock selection
  - Independent period and duty cycle, with double buffering
  - Programmable selection of the output waveform polarity
  - Programmable center or left aligned output waveform

10.13 USB Device Port

- USB V2.0 full-speed compliant,12 Mbits per second.
- Embedded USB V2.0 full-speed transceiver
- Embedded 2688-byte dual-port RAM for endpoints
- Eight endpoints
  - Endpoint 0: 64bytes
  - Endpoint 1 and 2: 64 bytes ping-pong
  - Endpoint 3: 64 bytes
  - Endpoint 4 and 5: 512 bytes ping-pong
  - Endpoint 6 and 7: 64 bytes ping-pong
  - Ping-pong Mode (two memory banks) for Isochronous and bulk endpoints
- Suspend/resume logic
- Integrated Pull-up on DDP

10.14 Analog-to-Digital Converter

- 8-channel ADC
- 10-bit 384 Ksamples/sec. or 8-bit 583 Ksamples/sec. Successive Approximation Register ADC
- ±2 LSB Integral Non Linearity, ±1 LSB Differential Non Linearity
- Integrated 8-to-1 multiplexer, offering eight independent 3.3V analog inputs
- External voltage reference for better accuracy on low voltage inputs
- Individual enable and disable of each channel
- Multiple trigger sources
  - Hardware or software trigger
  - External trigger pin
  - Timer Counter 0 to 2 outputs TIOA0 to TIOA2 trigger
- Sleep Mode and conversion sequencer
  - Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels
- Each analog input shared with digital signals

11. ARM7TDMI Processor Overview

11.1 Overview

The ARM7TDMI core executes both the 32-bit ARM and 16-bit Thumb instruction sets, allowing the user to trade off between high performance and high code density.The ARM7TDMI processor implements Von Neuman architecture, using a three-stage pipeline consisting of Fetch, Decode, and Execute stages.

The main features of the ARM7tDMI processor are:

- ARM7TDMI Based on ARMv4T Architecture
- Two Instruction Sets
  - ARM High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
- Three-Stage Pipeline Architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

11.2 ARM7TDMI Processor

For further details on ARM7TDMI, refer to the following ARM documents:

ARM Architecture Reference Manual (DDI 0100E)

ARM7TDMI Technical Reference Manual (DDI 0210B)

11.2.1 Instruction Type

Instructions are either 32 bits long (in ARM state) or 16 bits long (in THUMB state).

11.2.2 Data Type

ARM7TDMI supports byte (8-bit), half-word (16-bit) and word (32-bit) data types. Words must be aligned to four-byte boundaries and half words to two-byte boundaries.

Unaligned data access behavior depends on which instruction is used where.

11.2.3 ARM7TDMI Operating Mode

The ARM7TDMI, based on ARM architecture v4T, supports seven processor modes:

User: The normal ARM program execution state

FIQ: Designed to support high-speed data transfer or channel process

IRQ: Used for general-purpose interrupt handling

Supervisor: Protected mode for the operating system

Abort mode: Implements virtual memory and/or memory protection

System: A privileged user mode for the operating system

Undefined: Supports software emulation of hardware coprocessors

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

11.2.4 ARM7TDMI Registers

The ARM7TDMI processor has a total of 37registers:

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

These registers are not accessible at the same time. The processor state and operating mode determine which registers are available to the programmer.

At any one time 16 registers are visible to the user. The remainder are synonyms used to speed up exception processing.

Register 15 is the Program Counter (PC) and can be used in all instructions to reference data relative to the current instruction.

R14 holds the return address after a subroutine call.

R13 is used (by software convention) as a stack pointer.

Table 11-1. ARM7TDMI ARM Modes and Registers Layout

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

Mode-specific banked registers

Registers R0 to R7 are unbanked registers. This means that each of them refers to the same 32-bit physical register in all processor modes. They are general-purpose registers, with no special uses managed by the architecture, and can be used wherever an instruction allows a general-purpose register to be specified.

Registers R8 to R14 are banked registers. This means that each of them depends on the current mode of the processor.

11.2.4.1 Modes and Exception Handling

All exceptions have banked registers for R14 and R13.

After an exception, R14 holds the return address for exception processing. This address is used to return after the exception is processed, as well as to address the instruction that caused the exception.

R13 is banked across exception modes to provide each exception handler with a private stack pointer.

The fast interrupt mode also banks registers 8 to 12 so that interrupt processing can begin without having to save these registers.

A seventh processing mode, System Mode, does not have any banked registers. It uses the User Mode registers. System Mode runs tasks that require a privileged processor mode and allows them to invoke all classes of exceptions.

11.2.4.2 Status Registers

All other processor states are held in status registers. The current operating processor status is in the Current Program Status Register (CPSR). The CPSR holds:

- four ALU flags (Negative, Zero, Carry, and Overflow)
- two interrupt disable bits (one for each type of interrupt)
- one bit to indicate ARM or Thumb execution
- five bits to encode the current processor mode

All five exception modes also have a Saved Program Status Register (SPSR) that holds the CPSR of the task immediately preceding the exception.

11.2.4.3 Exception Types

The ARM7TDMI supports five types of exception and a privileged processing mode for each type. The types of exceptions are:

- fast interrupt (FIQ)
- normal interrupt (IRQ)
- memory aborts (used to implement memory protection or virtual memory)
- attempted execution of an undefined instruction
- software interrupts (SWIs)

Exceptions are generated by internal and external sources.

More than one exception can occur in the same time.

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

To return after handling the exception, the SPSR is moved to the CPSR, and R14 is moved to the PC. This can be done in two ways:

- by using a data-processing instruction with the S-bit set, and the PC as the destination
- by using the Load Multiple with Restore CPSR instruction (LDM)

11.2.5 ARM Instruction Set Overview

The ARM instruction set is divided into:

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

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

Table 11-2 gives the ARM instruction mnemonic list.

Table 11-2. ARM Instruction Mnemonic List

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

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

11.2.6 Thumb Instruction Set Overview

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

The Thumb instruction set is divided into:

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

In Thumb mode, eight general-purpose registers, R0 to R7, are available that are the same physical registers as R0 to R7 when executing ARM instructions. Some Thumb instructions also

access to the Program Counter (ARM Register 15), the Link Register (ARM Register 14) and the Stack Pointer (ARM Register 13). Further instructions allow limited access to the ARM registers 8 to 15.

Table 11-3 gives the Thumb instruction mnemonic list.

Table 11-3. Thumb Instruction Mnemonic List

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

12. Debug and Test Features

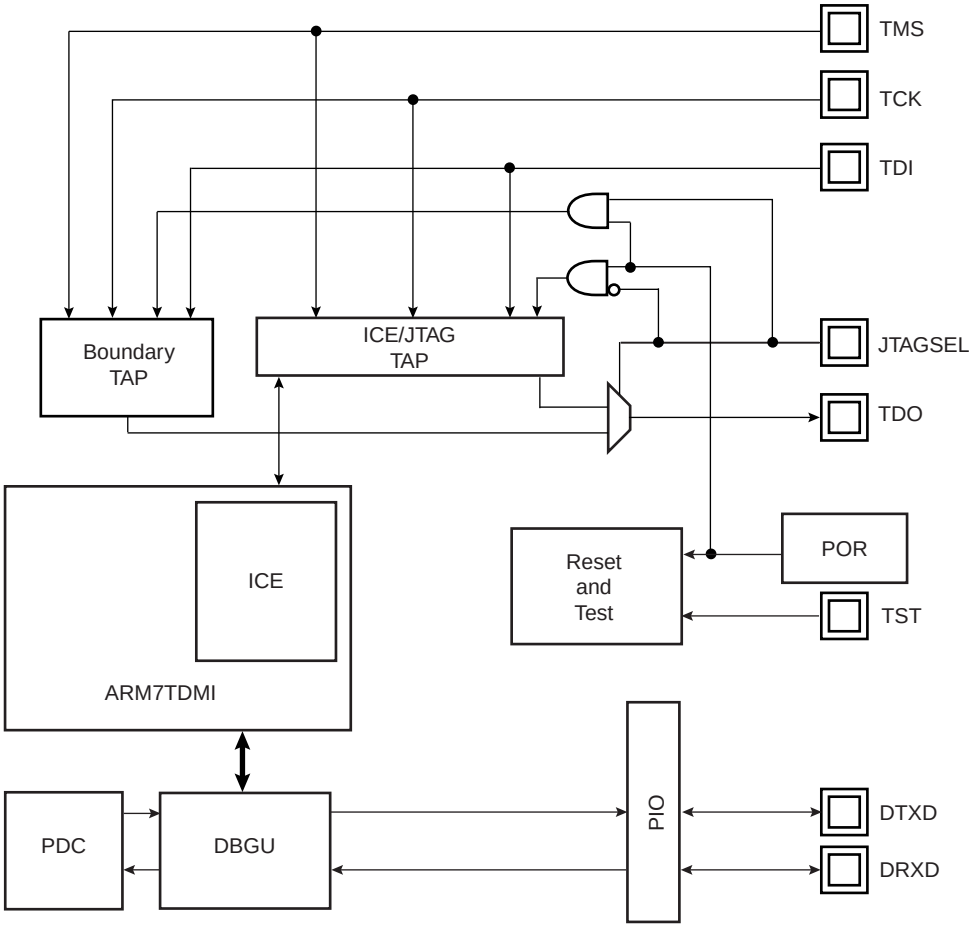
12.1 Overview

The SAM7SE Series Microcontrollers feature a number of complementary debug and test capabilities. A common JTAG/ICE (Embedded ICE) port is used for standard debugging functions, such as downloading code and single-stepping through programs. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

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

12.2 Block Diagram

Figure 12-1. Debug and Test Block Diagram

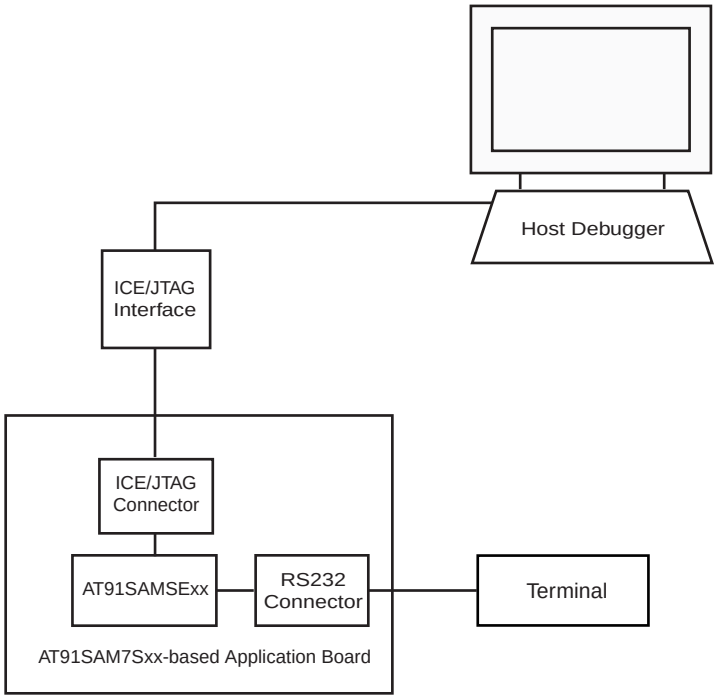


12.3 Application Examples

12.3.1 Debug Environment

Figure 12-2 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program.

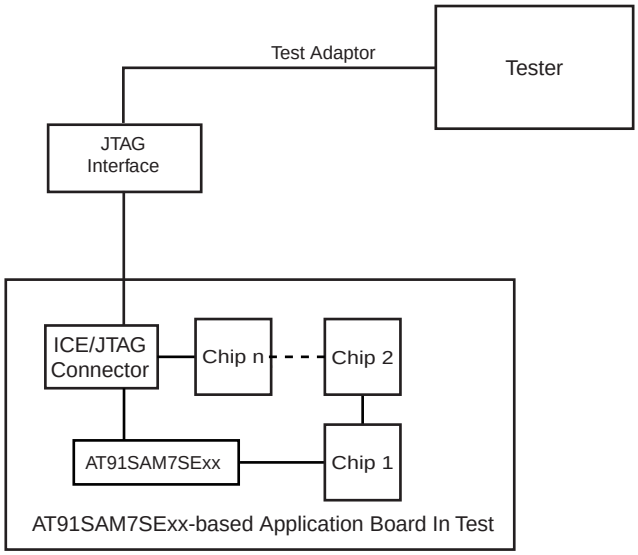
Figure 12-2. Application Debug Environment Example



12.3.2 Test Environment

Figure 12-3 shows a test environment example. Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

Figure 12-3. Application Test Environment Example



12.4 Debug and Test Pin Description

Table 12-1. Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
JTAGSEL	JTAG Selection	Input	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

12.5 Functional Description

12.5.1 Test Pin

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

12.5.2 EmbeddedICE™ (Embedded In-circuit Emulator)

The ARM7TDMI EmbeddedICE is supported via the ICE/JTAG port.The internal state of the ARM7TDMI is examined through an ICE/JTAG port.

The ARM7TDMI processor contains hardware extensions for advanced debugging features:

- In halt mode, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM7TDMI registers. This data can be serially shifted out without affecting the rest of the system.
- In monitor mode, the JTAG interface is used to transfer data between the debugger and a simple monitor program running on the ARM7TDMI processor.

There are three scan chains inside the ARM7TDMI processor that support testing, debugging, and programming of the Embedded ICE. The scan chains are controlled by the ICE/JTAG port.

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

For further details on the Embedded ICE, see the ARM7TDMI (Rev4) Technical Reference Manual (DDI0210B).

12.5.3 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.

Table 12-2. AT91SAM7SExx Chip IDs

Chip Name	Chip ID
AT91SAM7SE32	0x27280340
AT91SAM7SE256	0x272A0940
AT91SAM7SE512	0x272A0A40

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

12.5.4 IEEE 1149.1 JTAG Boundary Scan

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

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

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

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

12.5.4.1 JTAG Boundary-scan Register

The Boundary-scan Register (BSR) contains 353 bits that correspond to active pins and associated control signals.

Each AT91SAM7SExx input/output pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit facilitates the observability of data applied to the pad. The CONTROL bit selects the direction of the pad.

For more information, please refer to BDSL files which are available for the SAM7SE Series.

12.5.5 ID Code Register

Access: Read-only

31	30	29	28	27	26	25	24
VERSION				PART NUMBER			

23	22	21	20	19	18	17	16
PART NUMBER							

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

7	6	5	4	3	2	1	0
MANUFACTURER IDENTITY							1

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

Set to 0x0.

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

Chip Name	Chip ID
AT91SAM7SE32	0x5B1D
AT91SAM7SE256	0x5B15
AT91SAM7SE512	0x5B14

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

Set to 0x01F.

- **Bit[0] Required by IEEE Std. 1149.1.**

Set to 0x1.

Chip Name	JTAG ID Code
AT91SAM7SE32	05B1_D03F
AT91SAM7SE256	05B1_503F
AT91SAM7SE512	05B1_403F

13. Reset Controller (RSTC)

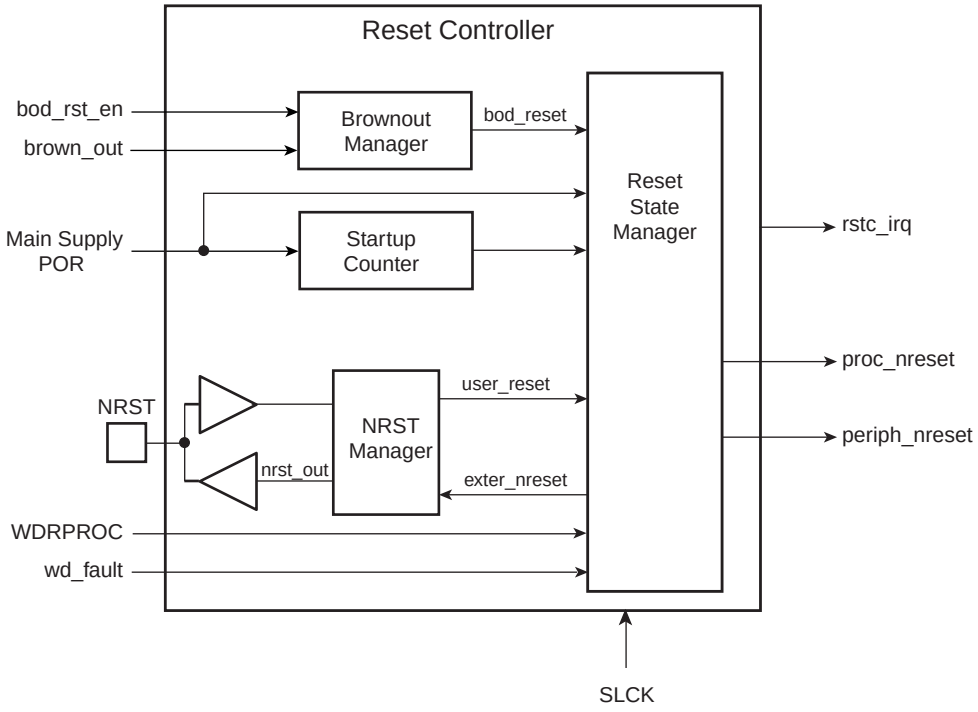
The Reset Controller (RSTC), based on power-on reset cells, handles all the resets of the system without any external components. It reports which reset occurred last.

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

A brownout detection is also available to prevent the processor from falling into an unpredictable state.

13.1 Block Diagram

Figure 13-1. Reset Controller Block Diagram



13.2 Functional Description

13.2.1 Reset Controller Overview

The Reset Controller is made up of an NRST Manager, a Brownout 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.
- periph_nreset: Affects the whole set of embedded peripherals.
- nrst_out: Drives the NRST pin.

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

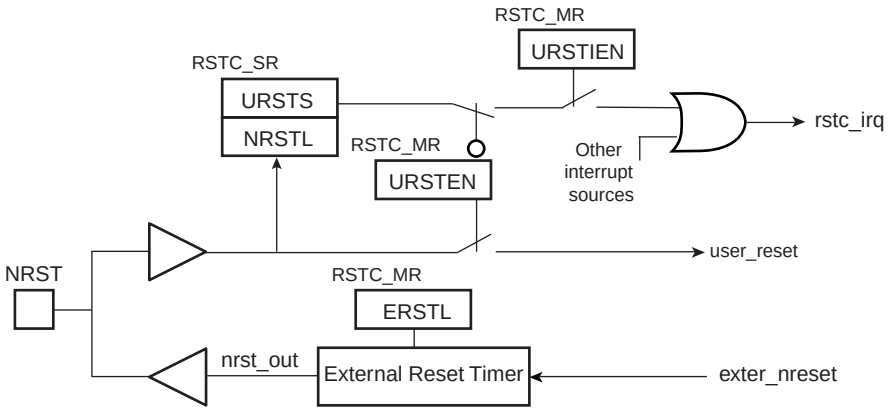
The NRST Manager shapes the NRST assertion during a programmable time, thus controlling external device resets.

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

13.2.2 NRST Manager

The NRST Manager samples the NRST input pin and drives this pin low when required by the Reset State Manager. Figure 13-2 shows the block diagram of the NRST Manager.

Figure 13-2. NRST Manager



13.2.2.1 NRST Signal or Interrupt

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

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

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

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

13.2.2.2 NRST External Reset Control

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

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

13.2.3 Brownout Manager

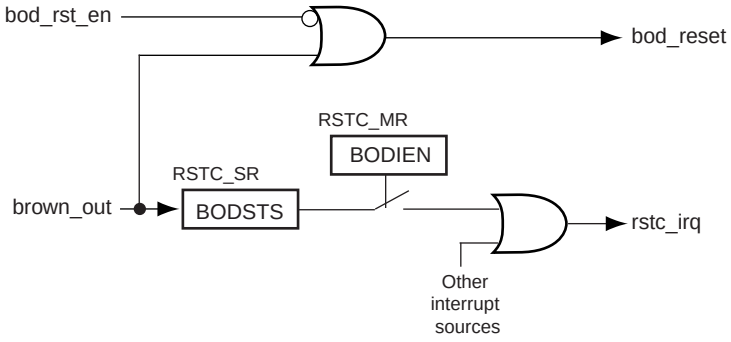
Brownout detection prevents the processor from falling into an unpredictable state if the power supply drops below a certain level. When VDDCORE drops below the brownout threshold, the brownout manager requests a brownout reset by asserting the bod_reset signal.

The programmer can disable the brownout reset by setting low the bod_rst_en input signal, i.e.; by locking the corresponding general-purpose NVM bit in the Flash. When the brownout reset is disabled, no reset is performed. Instead, the brownout detection is reported in the bit BODSTS of RSTC_SR. BODSTS is set and clears only when RSTC_SR is read.

The bit BODSTS can trigger an interrupt if the bit BODIEN is set in the RSTC_MR.

At factory, the brownout reset is disabled.

Figure 13-3. Brownout Manager



13.2.4 Reset States

The Reset State Manager handles the different reset sources and generates the internal reset signals. It reports the reset status in the field RSTTYP of the Status Register (RSTC_SR). The update of the field RSTTYP is performed when the processor reset is released.

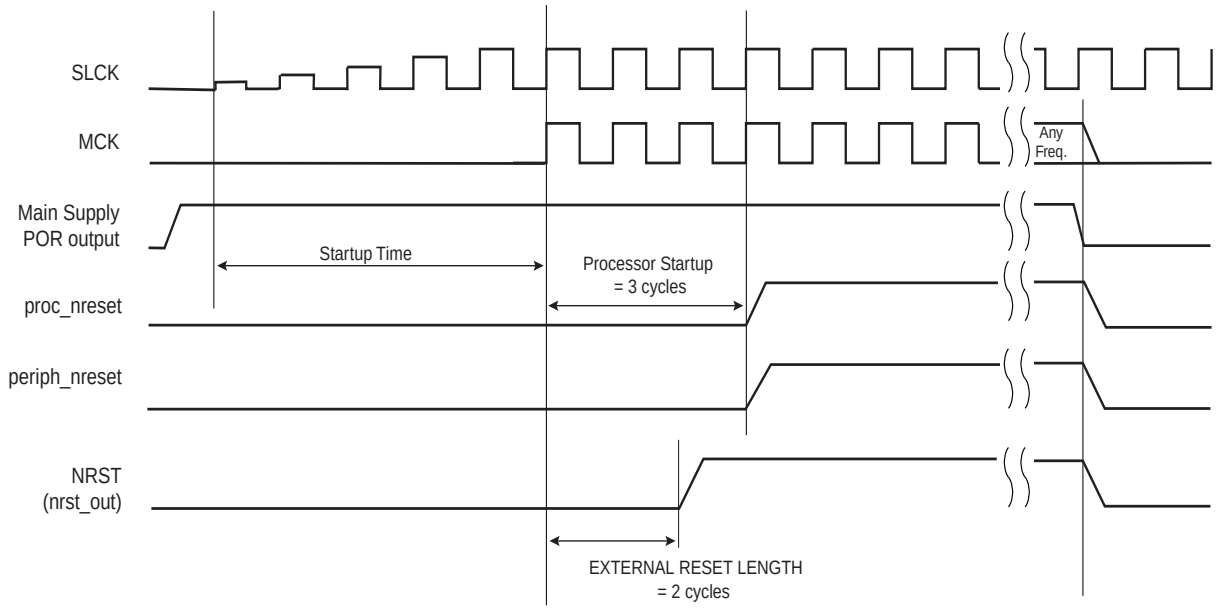
13.2.4.1 Power-up Reset

When VDDCORE is powered on, the Main Supply POR cell output is filtered with a start-up counter that operates at Slow Clock. The purpose of this counter is to ensure that the Slow Clock oscillator is stable before starting up the device.

The startup time, as shown in Figure 13-4, is hardcoded to comply with the Slow Clock Oscillator startup time. After the startup time, the reset signals are released and the field RSTTYP in RSTC_SR reports a Power-up Reset.

When VDDCORE is detected low by the Main Supply POR Cell, all reset signals are asserted immediately.

Figure 13-4. Power-up Reset



13.2.4.2 User Reset

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

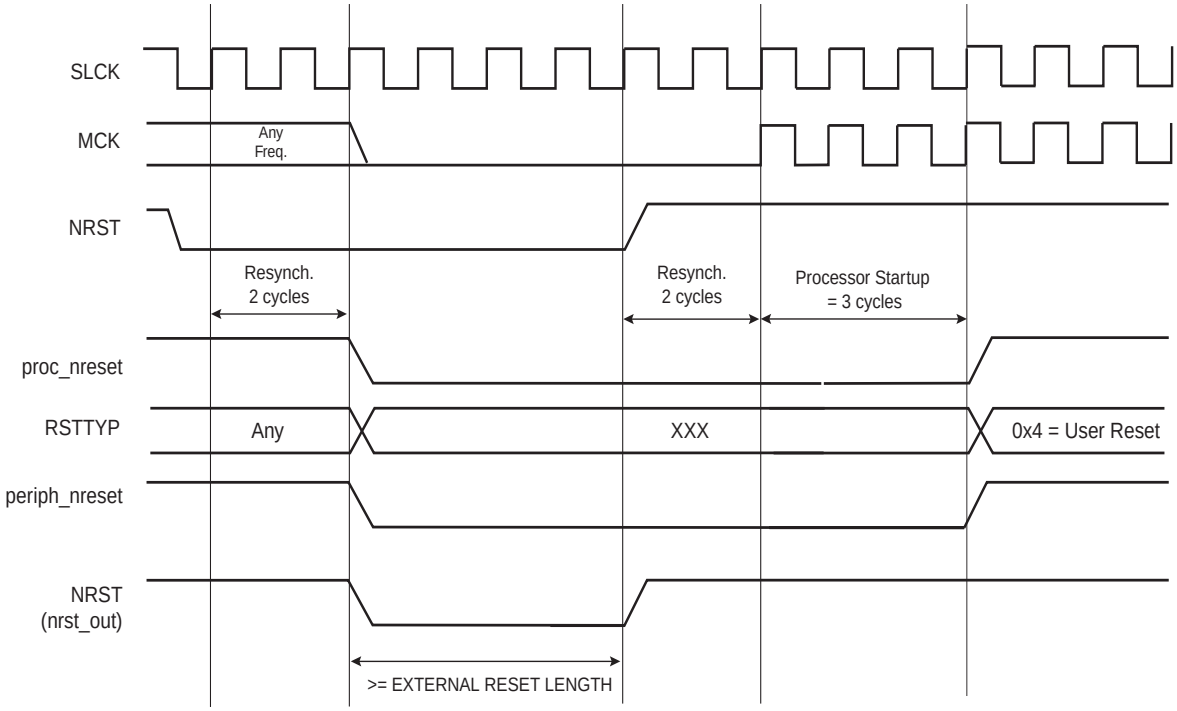
The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

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

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

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

Figure 13-5. User Reset State



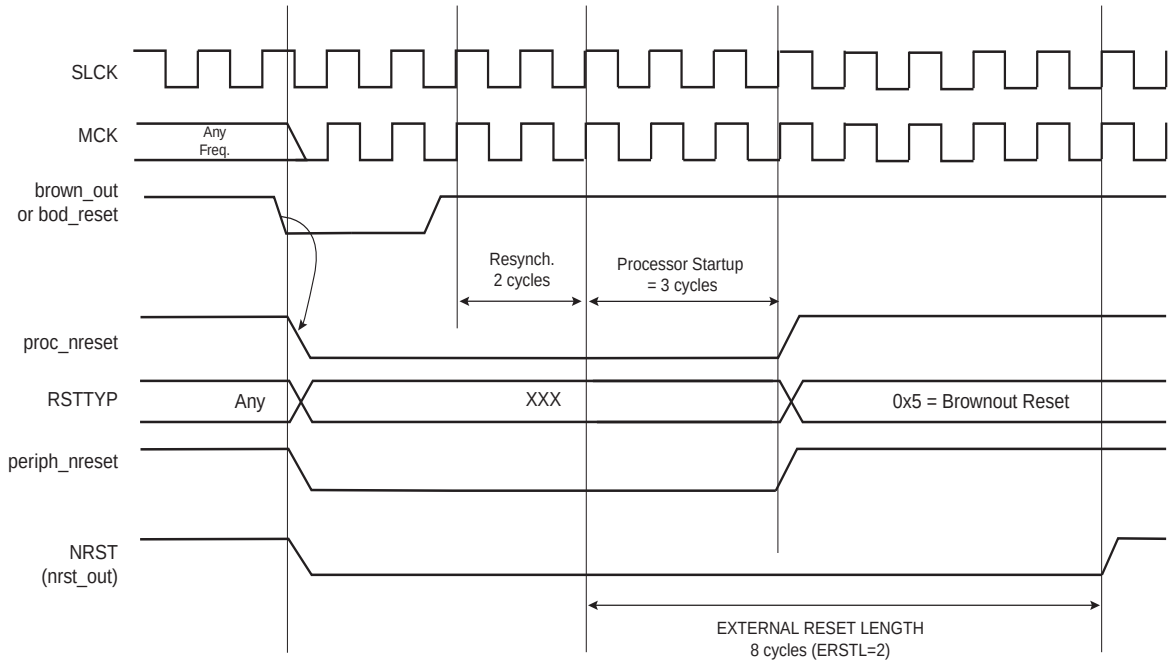
13.2.4.3 Brownout Reset

When the brown_out/bod_reset signal is asserted, the Reset State Manager immediately enters the Brownout Reset. In this state, the processor, the peripheral and the external reset lines are asserted.

The Brownout Reset is left 3 Slow Clock cycles after the rising edge of brown_out/bod_reset after a two-cycle resynchronization. An external reset is also triggered.

When the processor reset is released, the field RSTTYP in RSTC_SR is loaded with the value 0x5, thus indicating that the last reset is a Brownout Reset.

Figure 13-6. Brownout Reset State



13.2.4.4 Software Reset

The Reset Controller offers several commands used to assert the different reset signals. These commands are performed by writing the Control Register (RSTC_CR) with the following bits at 1:

- PROCRST: Writing PROCRST at 1 resets the processor and the watchdog timer.
- PERRST: Writing PERRST at 1 resets all the embedded peripherals, including the memory system and, in particular, the Remap Command. The Peripheral Reset is generally used for debug purposes. Except for Debug purposes, the PERRST must always be used in conjunction with a PROCRST (PERRST and PROCRST both set at 1 simultaneously).
- EXTRST: Writing EXTRST at 1 asserts low the NRST pin during a time defined by the field ERSTL in the Mode Register (RSTC_MR).

The software reset is entered if at least one of these bits is set by the software. All these commands can be performed independently or simultaneously. The software reset lasts 3 Slow Clock cycles.

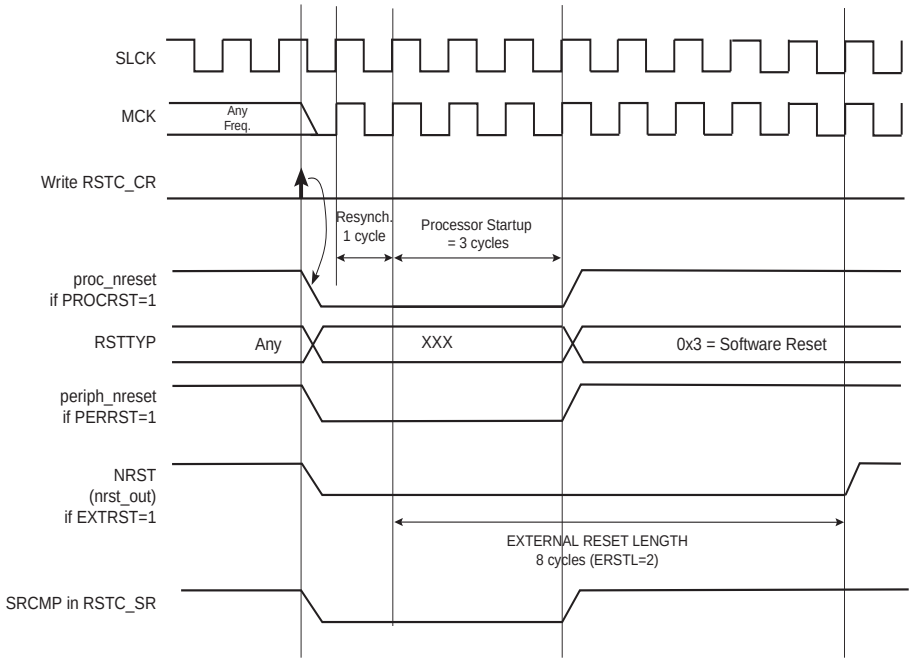
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

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

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

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

Figure 13-7. Software Reset



13.2.4.5 *Watchdog Reset*

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

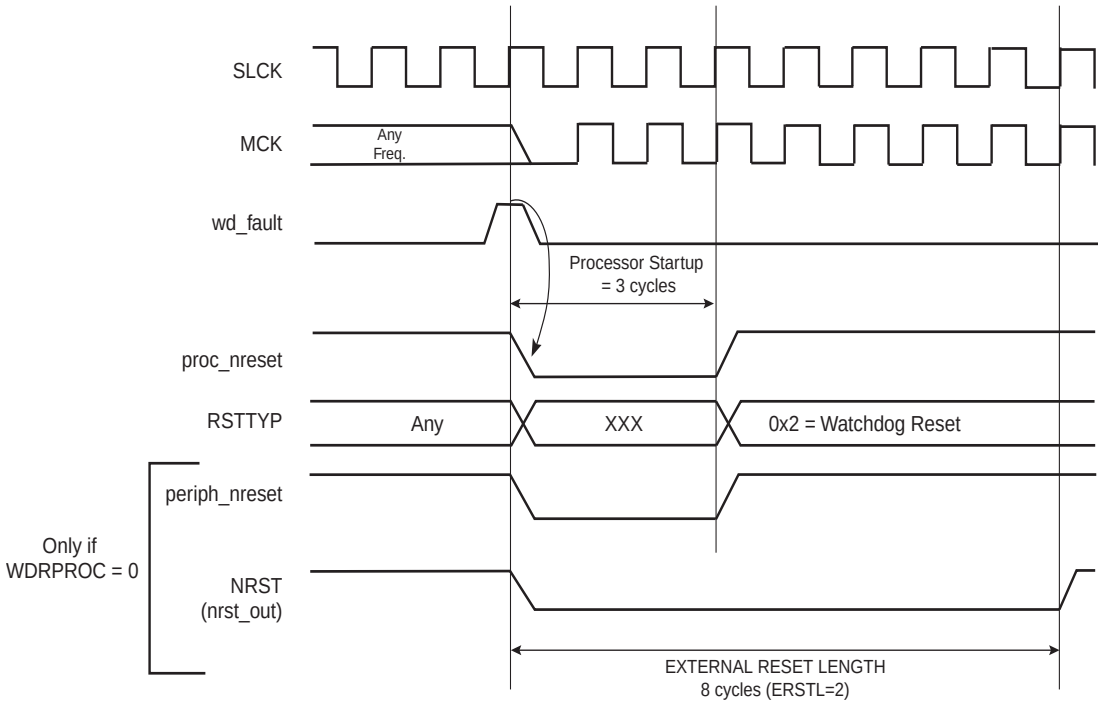
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- 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 13-8. Watchdog Reset



13.2.5 Reset State Priorities

The Reset State Manager manages the following priorities between the different reset sources, given in descending order:

- Power-up Reset
- Brownout Reset
- Watchdog Reset
- Software Reset
- User Reset

Particular cases are listed below:

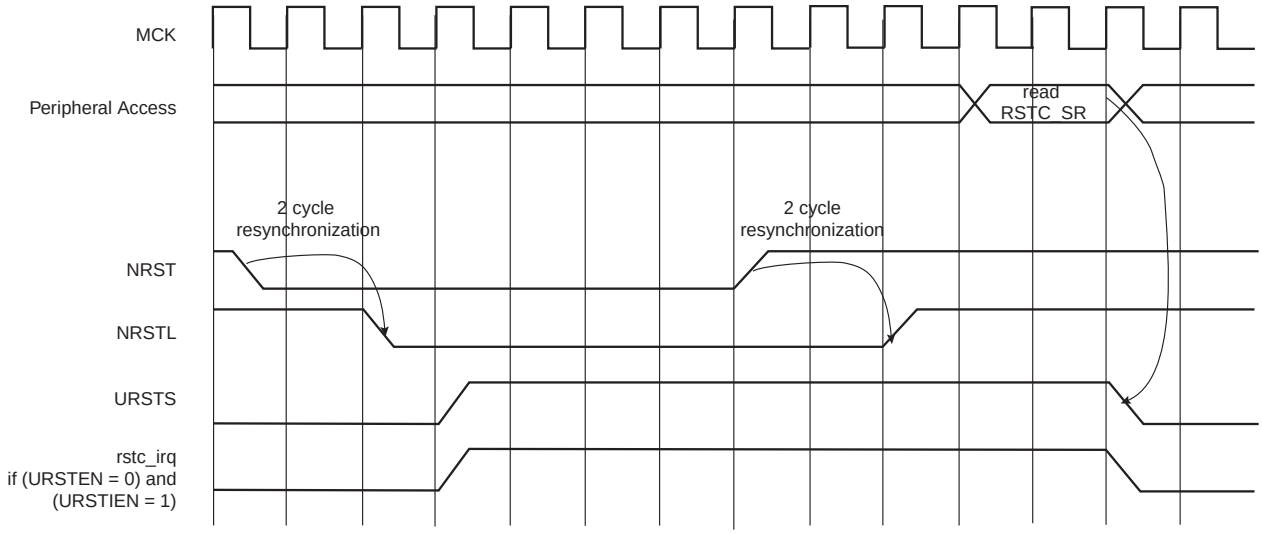
- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST has no effect.
- When in Watchdog Reset:
  - The processor reset is active and so a Software Reset cannot be programmed.
  - A User Reset cannot be entered.

13.2.6 Reset Controller Status Register

The Reset Controller status register (RSTC_SR) provides several status fields:

- RSTTYP field: This field gives the type of the last reset, as explained in previous sections.
- SRCMP bit: This field indicates that a Software Reset Command is in progress and that no further software reset should be performed until the end of the current one. This bit is automatically cleared at the end of the current software reset.
- NRSTL bit: The NRSTL bit of the Status Register gives the level of the NRST pin sampled on each MCK rising edge.
- URSTS bit: A high-to-low transition of the NRST pin sets the URSTS bit of the RSTC_SR register. This transition is also detected on the Master Clock (MCK) rising edge (see Figure 13-9). If the User Reset is disabled (URSTEN = 0) and if the interruption is enabled by the URSTIEN bit in the RSTC_MR register, the URSTS bit triggers an interrupt. Reading the RSTC_SR status register resets the URSTS bit and clears the interrupt.
- BODSTS bit: This bit indicates a brownout detection when the brownout reset is disabled (bod_rst_en = 0). It triggers an interrupt if the bit BODIEN in the RSTC_MR register enables the interrupt. Reading the RSTC_SR register resets the BODSTS bit and clears the interrupt.

Figure 13-9. Reset Controller Status and Interrupt



13.3 Reset Controller (RSTC) User Interface

Table 13-1. Reset Controller (RSTC) Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Control Register	RSTC_CR	Write-only	-
0x04	Status Register	RSTC_SR	Read-only	0x0000_0000
0x08	Mode Register	RSTC_MR	Read/Write	0x0000_0000

13.3.1 Reset Controller Control Register

Name: RSTC_CR

Access: Write-only

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

- **PROCRST: Processor Reset**

0 = No effect.

1 = If KEY is correct, resets the processor.

- **PERRST: Peripheral Reset**

0 = No effect.

1 = If KEY is correct, resets the peripherals.

- **EXTRST: External Reset**

0 = No effect.

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

- **KEY: Password**

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

13.3.2 Reset Controller Status Register

Name: RSTC_SR

Access: Read-only

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

- **BODSTS: Brownout Detection Status**

0 = No brownout high-to-low transition happened since the last read of RSTC_SR.

1 = A brownout high-to-low transition has been detected since the last read of RSTC_SR.

- **RSTTYP: Reset Type**

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

RSTTYP			Reset Type	Comments
0	0	0	Power-up Reset	VDDCORE rising
0	1	0	Watchdog Reset	Watchdog fault occurred
0	1	1	Software Reset	Processor reset required by the software
1	0	0	User Reset	NRST pin detected low
1	0	1	Brownout Reset	Brownout reset occurred

- **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.

13.3.3 Reset Controller Mode Register

Name: RSTC_MR

Access: Read/Write

31	30	29	28	27	26	25	24
KEY							

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

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

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

- **URSTEN: User Reset Enable**

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

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

- **URSTIEN: User Reset Interrupt Enable**

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

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

- **BODIEN: Brownout Detection Interrupt Enable**

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

1 = BODSTS bit in RSTC_SR at 1 asserts rstc_irq.

- **ERSTL: External Reset Length**

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

- **KEY: Password**

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

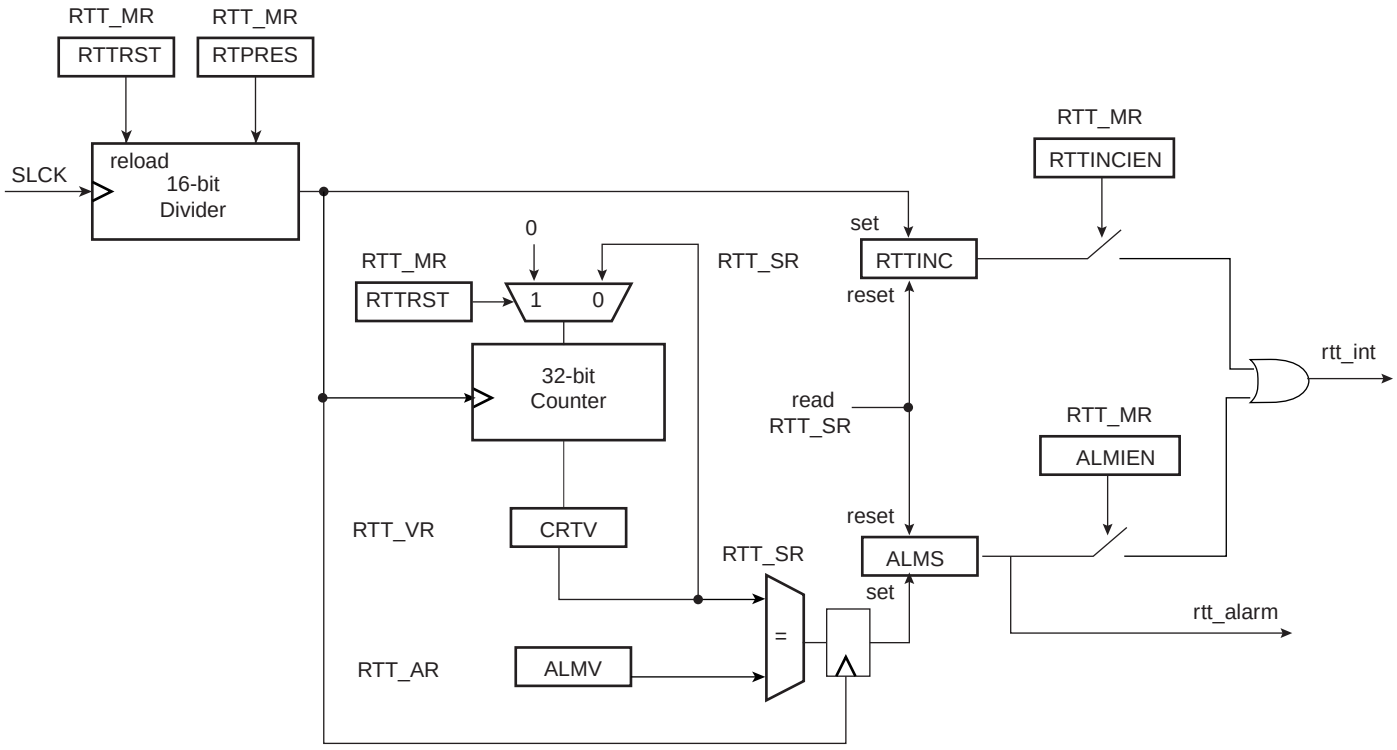
14. Real-time Timer (RTT)

14.1 Overview

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

14.2 Block Diagram

Figure 14-1. Real-time Timer



14.3 Functional Description

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

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

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

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

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

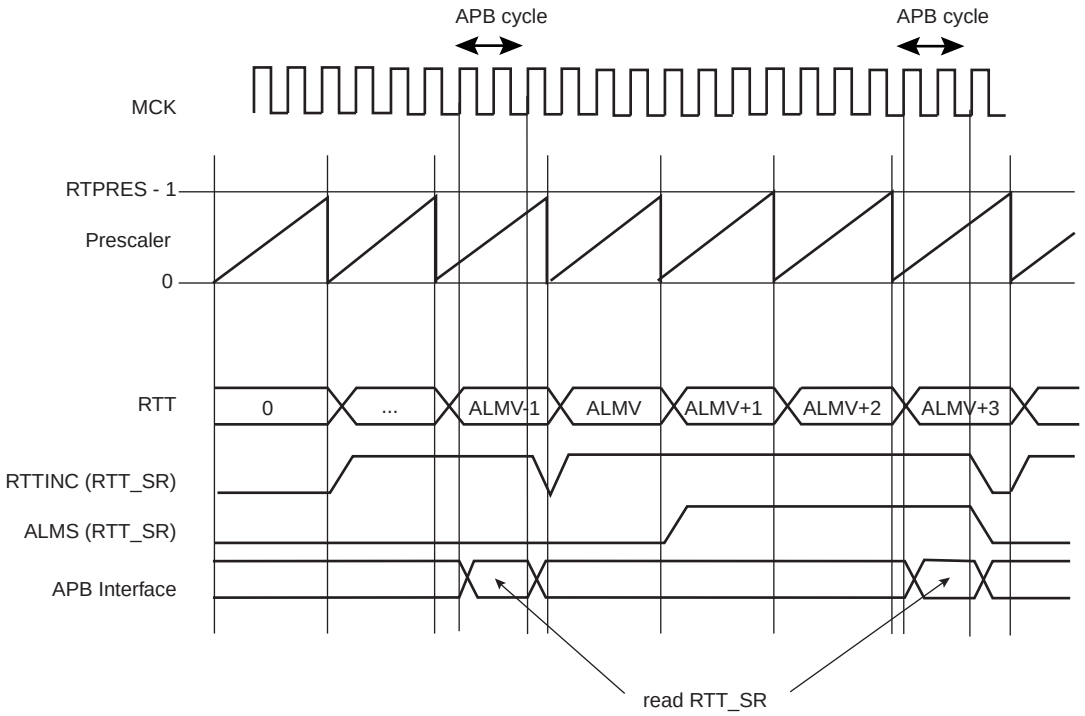
The bit RTTINC in RTT_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32.768 Hz.

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

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

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

Figure 14-2. RTT Counting



14.4 Real-time Timer (RTT) User Interface

Table 14-1. Real-time Timer (RTT) Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Mode Register	RTT_MR	Read/Write	0x0000_8000
0x04	Alarm Register	RTT_AR	Read/Write	0xFFFF_FFFF
0x08	Value Register	RTT_VR	Read-only	0x0000_0000
0x0C	Status Register	RTT_SR	Read-only	0x0000_0000

14.4.1 Real-time Timer Mode Register

Name: RTT_MR

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

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

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

RTPRES = 0: The Prescaler Period is equal to 2^{16}

RTPRES …0: The Prescaler Period is equal to RTPRES.

- **ALMIEN: Alarm Interrupt Enable**

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

1 = The bit ALMS in RTT_SR asserts interrupt.

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

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

1 = The bit RTTINC in RTT_SR asserts interrupt.

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

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

14.4.2 Real-time Timer Alarm Register

Name: RTT_AR

Access: Read/Write

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

7	6	5	4	3	2	1	0
ALMV							

- **ALMV: Alarm Value**

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

14.4.3 Real-time Timer Value Register

Name: RTT_VR

Access: Read-only

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

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

Returns the current value of the Real-time Timer.

14.4.4 Real-time Timer Status Register

Name: RTT_SR

Access: Read-only

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

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

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

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

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

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

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

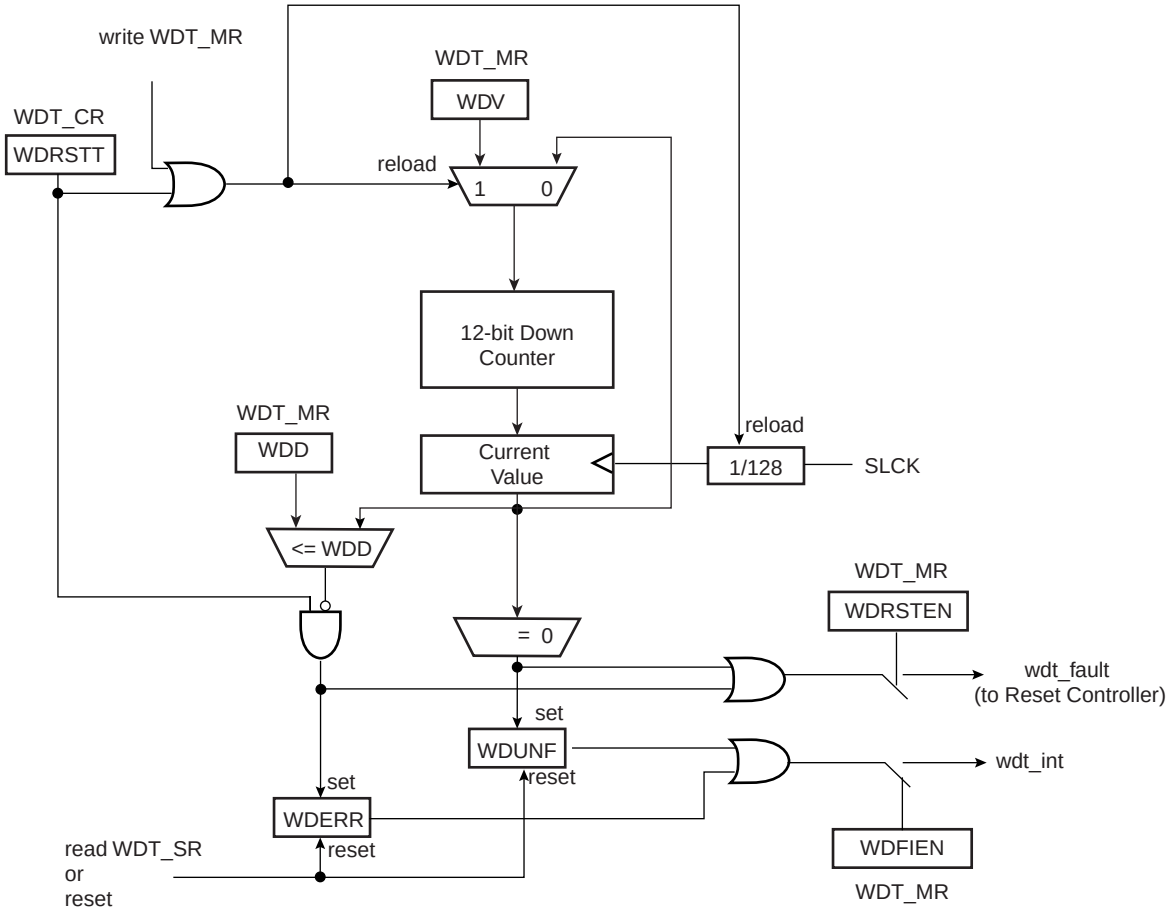
15. Watchdog Timer (WDT)

15.1 Overview

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

15.2 Block Diagram

Figure 15-1. Watchdog Timer Block Diagram



15.3 Functional Description

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

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

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

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

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

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

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

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

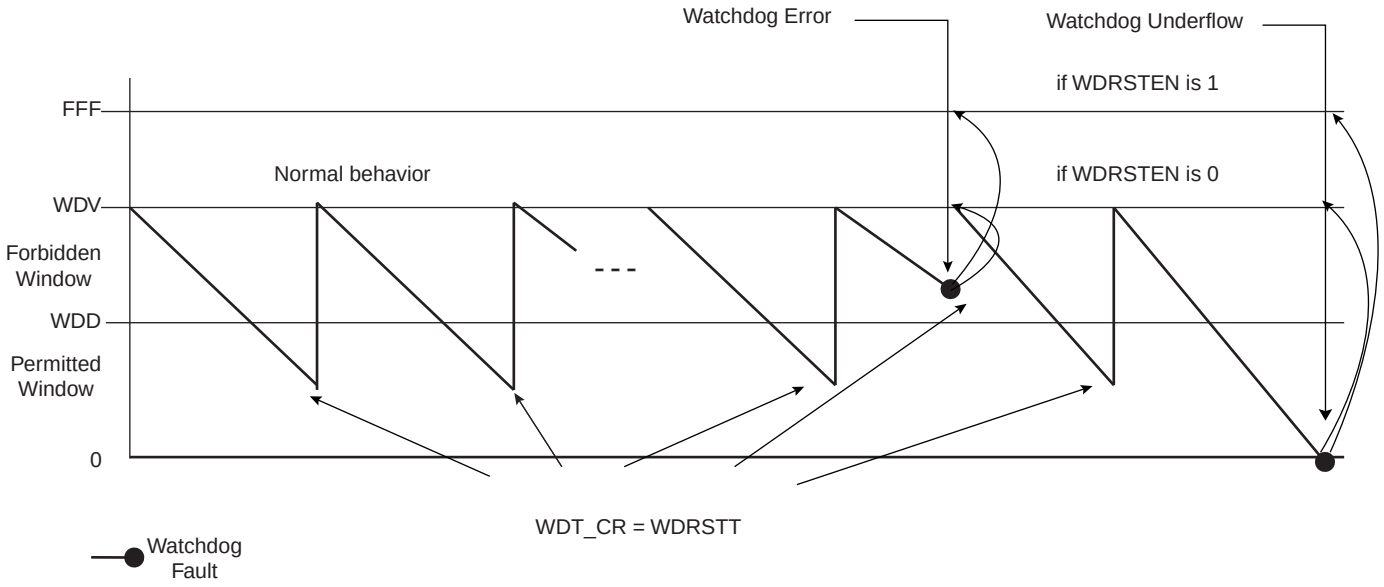
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet.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

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

Writing the WDT_MR reloads and restarts the down counter.

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

Figure 15-2. Watchdog Behavior



15.4 Watchdog Timer (WDT) User Interface

Table 15-1. Watchdog Timer (WDT) Register Mapping

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

15.4.1 Watchdog Timer Control Register

Name: WDT_CR

Access: Write-only

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

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog.

- **KEY: Password**

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

15.4.2 Watchdog Timer Mode Register

Name: WDT_MR

Access: Read/Write Once

31	30	29	28	27	26	25	24
–	–	WDIDLEHLT	WDDBGHLT	WDD			

23	22	21	20	19	18	17	16
WDD							

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

7	6	5	4	3	2	1	0
WDV							

- **WDV: Watchdog Counter Value**

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

- **WDFIEN: Watchdog Fault Interrupt Enable**

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

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

- **WDRSTEN: Watchdog Reset Enable**

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

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

- **WDRPROC: Watchdog Reset Processor**

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

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

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer.

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

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

- **WDDBGHLT: Watchdog Debug Halt**

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

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

- **WDIDLEHLT: Watchdog Idle Halt**

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

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

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

15.4.3 Watchdog Timer Status Register

Name: WDT_SR

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.

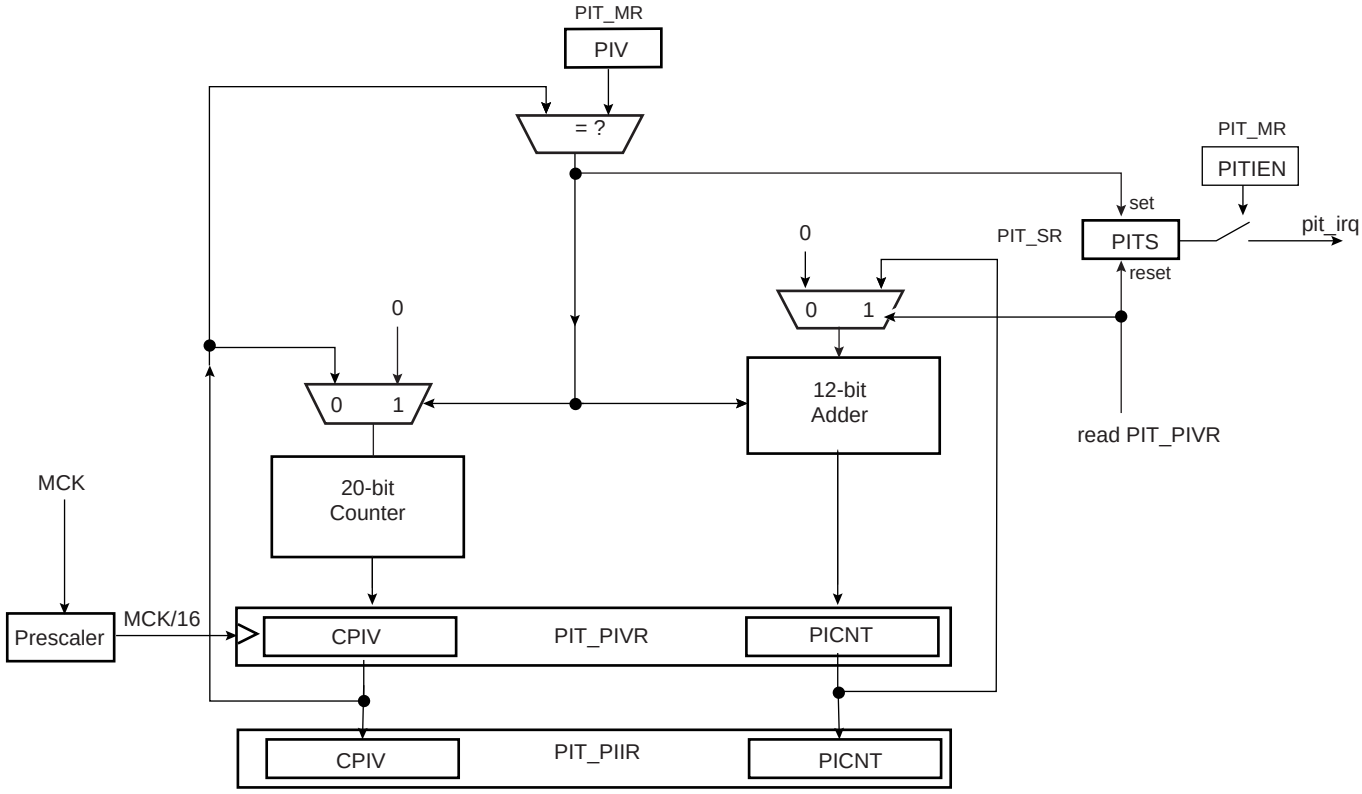
16. Periodic Interval Timer (PIT)

16.1 Overview

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

16.2 Block Diagram

Figure 16-1. Periodic Interval Timer



16.3 Functional Description

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

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

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

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

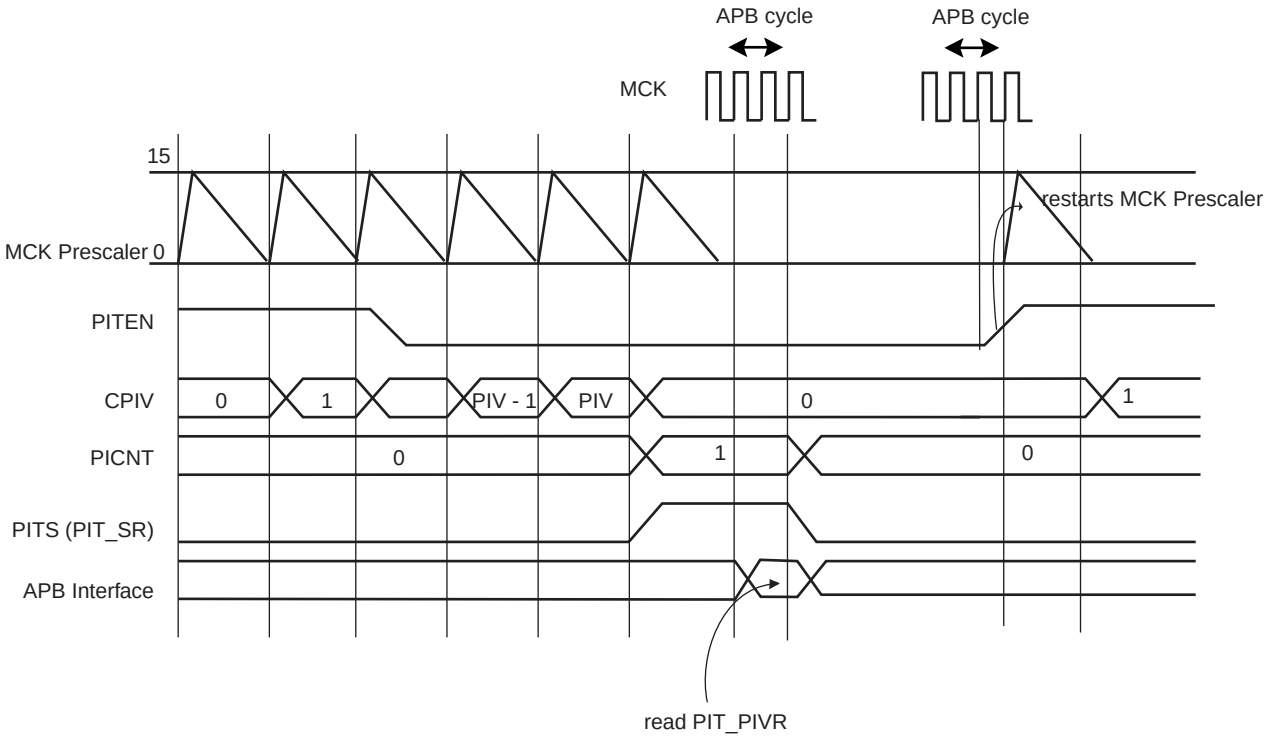
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

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

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

The PIT is stopped when the core enters debug state.

Figure 16-2. Enabling/Disabling PIT with PITEN



16.4 Periodic Interval Timer (PIT) User Interface

Table 16-1. Periodic Interval Timer (PIT) Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Mode Register	PIT_MR	Read/Write	0x000F_FFFF
0x04	Status Register	PIT_SR	Read-only	0x0000_0000
0x08	Periodic Interval Value Register	PIT_PIVR	Read-only	0x0000_0000
0x0C	Periodic Interval Image Register	PIT_PIIR	Read-only	0x0000_0000

16.4.1 Periodic Interval Timer Mode Register

Name: PIT_MR

Access: Read/Write

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

- **PIV: Periodic Interval Value**

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

- **PITEN: Period Interval Timer Enabled**

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

1 = The Periodic Interval Timer is enabled.

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

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

1 = The bit PITS in PIT_SR asserts interrupt.

16.4.2 Periodic Interval Timer Status Register

Name: PIT_SR

Access: Read-only

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

- **PITS: Periodic Interval Timer Status**

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

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

16.4.3 Periodic Interval Timer Value Register

Name: PIT_PIVR

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

16.4.4 Periodic Interval Timer Image Register

Name: PIT_PIIR

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

17. Voltage Regulator Mode Controller (VREG)

17.1 Overview

The Voltage Regulator Mode Controller contains one Read/Write register, the Voltage Regulator Mode Register. Its offset is 0x60 with respect to the System Controller offset.

This register controls the Voltage Regulator Mode. Setting PSTDBY (bit 0) puts the Voltage Regulator in Standby Mode or Low-power Mode. On reset, the PSTDBY is reset, so as to wake up the Voltage Regulator in Normal Mode.

17.2 Voltage Regulator Power Controller (VREG) User Interface

Table 17-1. Voltage Regulator Power Controller Register Mapping

Offset	Register	Name	Access	Reset Value
0x60	Voltage Regulator Mode Register	VREG_MR	Read/Write	0x0

17.2.1 Voltage Regulator Mode Register

Name: VREG_MR

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
–	–	–	–	–	–	–	PSTDBY

- **PSTDBY: Periodic Interval Value**

0 = Voltage regulator in normal mode.

1 = Voltage regulator in standby mode (low-power mode).

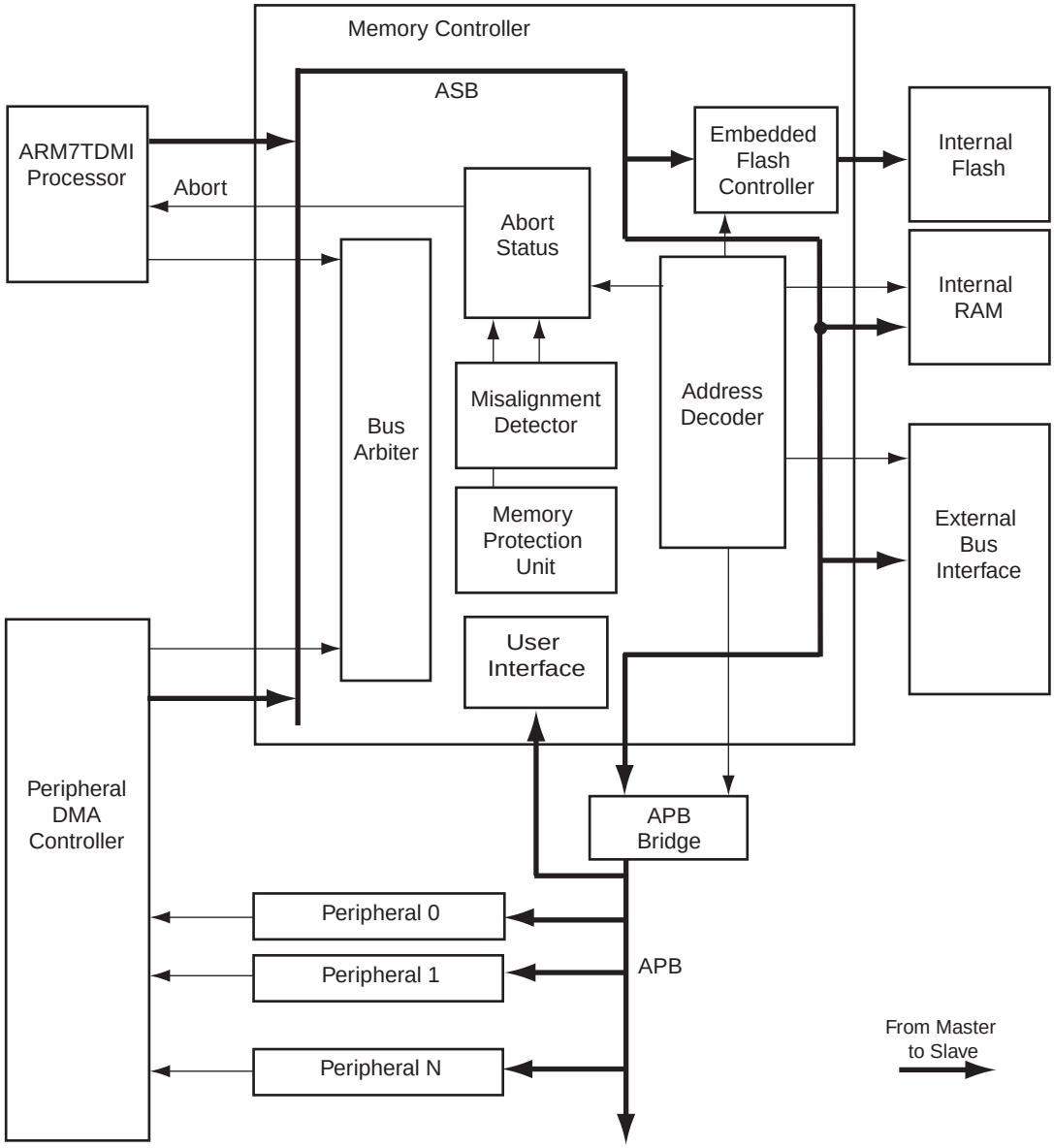
18. Memory Controller (MC)

18.1 Overview

The Memory Controller (MC) manages the ASB bus and controls accesses requested by the masters, typically the ARM7TDMI processor and the Peripheral DMA Controller. It features a simple bus arbiter, an address decoder, an abort status, a misalignment detector and an Embedded Flash Controller. In addition, the MC contains a Memory Protection Unit (MPU) consisting of 16 areas that can be protected against write and/or user accesses. Access to peripherals can be protected in the same way.

18.2 Block Diagram

Figure 18-1. Memory Controller Block Diagram



18.3 Functional Description

The Memory Controller handles the internal ASB bus and arbitrates the accesses of both masters.

It is made up of:

- A bus arbiter
- An address decoder
- An abort status
- A misalignment detector
- A memory protection unit
- An Embedded Flash Controller

The MC handles only little-endian mode accesses. The masters work in little-endian mode only.

18.3.1 Bus Arbiter

The Memory Controller has a simple, hard-wired priority bus arbiter that gives the control of the bus to one of the two masters. The Peripheral Data Controller has the highest priority; the ARM processor has the lowest one.

18.3.2 Address Decoder

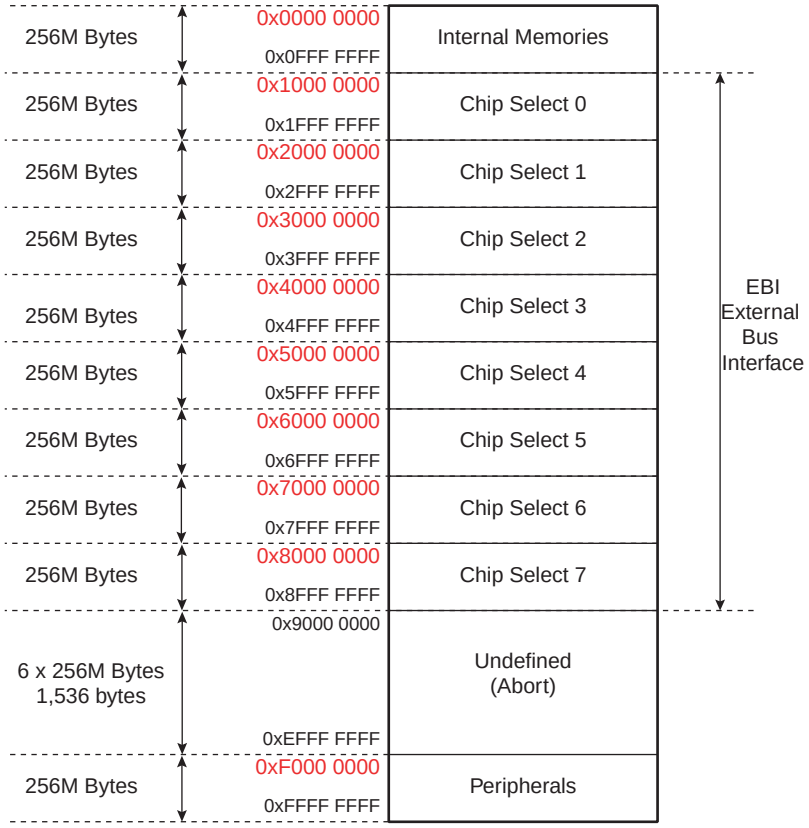
The Memory Controller features an Address Decoder that first decodes the four highest bits of the 32-bit address bus and defines 11 separate areas:

- One 256-Mbyte address space for the internal memories
- Eight 256-Mbyte address spaces, each assigned to one of the eight chip select lines of the External Bus Interface
- One 256-Mbyte address space reserved for the embedded peripherals
- An undefined address space of 1536M bytes that returns an Abort if accessed

18.4 External Memory Areas

Figure 18-2 shows the assignment of the 256-Mbyte memory areas.

Figure 18-2. External Memory Areas



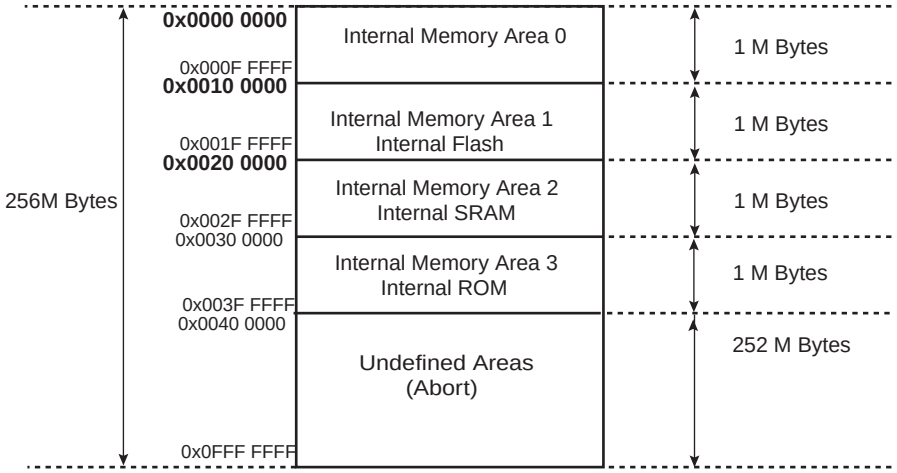
18.4.1 Internal Memory Mapping

Within the Internal Memory address space, the Address Decoder of the Memory Controller decodes eight more address bits to allocate 1-Mbyte address spaces for the embedded memories.

The allocated memories are accessed all along the 1-Mbyte address space and so are repeated n times within this address space, n equaling 1M bytes divided by the size of the memory.

When the address of the access is undefined within the internal memory area, the Address Decoder returns an Abort to the master.

Figure 18-3. Internal Memory Mapping



18.4.2 Internal Memory Area 0

The first 32 bytes of Internal Memory Area 0 contain the ARM processor exception vectors, in particular, the Reset Vector at address 0x0.

Before execution of the remap command, the internal ROM or the on-chip Flash is mapped into Internal Memory Area 0, so that the ARM7TDMI reaches an executable instruction contained in Flash. A general purpose bit (GPNVM Bit 2) is used to boot either on the ROM (default) or from the Flash.

Setting the GPNVM Bit 2 selects the boot from the Flash, clearing it selects the boot from the ROM. Asserting ERASE clears the GPNVM Bit 2 and thus selects the boot from the ROM by default.

After the remap command, the internal SRAM at address 0x0020 0000 is mapped into Internal Memory Area 0. The memory mapped into Internal Memory Area 0 is accessible in both its original location and at address 0x0.

18.4.3 Remap Command

After execution, the Remap Command causes the Internal SRAM to be accessed through the Internal Memory Area 0.

As the ARM vectors (Reset, Abort, Data Abort, Prefetch Abort, Undefined Instruction, Interrupt, and Fast Interrupt) are mapped from address 0x0 to address 0x20, the Remap Command allows the user to redefine dynamically these vectors under software control.

The Remap Command is accessible through the Memory Controller User Interface by writing the MC_RCR (Remap Control Register) RCB field to one.

The Remap Command can be cancelled by writing the MC_RCR RCB field to one, which acts as a toggling command. This allows easy debug of the user-defined boot sequence by offering a simple way to put the chip in the same configuration as after a reset.

18.4.4 Abort Status

There are three reasons for an abort to occur:

- access to an undefined address
- access to a protected area without the permitted state
- an access to a misaligned address.

When an abort occurs, a signal is sent back to all the masters, regardless of which one has generated the access. However, only the ARM7TDMI can take an abort signal into account, and only under the condition that it was generating an access. The Peripheral Data Controller does not handle the abort input signal. Note that the connection is not represented in Figure 18-1.

To facilitate debug or for fault analysis by an operating system, the Memory Controller integrates an Abort Status register set.

The full 32-bit wide abort address is saved in MC_AASR. Parameters of the access are saved in MC_ASR and include:

- the size of the request (field ABTSZ)
- the type of the access, whether it is a data read or write, or a code fetch (field ABTTYP)
- whether the access is due to accessing an undefined address (bit UNDADD), a misaligned address (bit MISADD) or a protection violation (bit MPU)
- the source of the access leading to the last abort (bits MST0 and MST1)
- whether or not an abort occurred for each master since the last read of the register (bit SVMST0 and SVMST1) unless this information is loaded in MST bits

In the case of a Data Abort from the processor, the address of the data access is stored. This is useful, as searching for which address generated the abort would require disassembling the instructions and full knowledge of the processor context.

In the case of a Prefetch Abort, the address may have changed, as the prefetch abort is pipelined in the ARM processor. The ARM processor takes the prefetch abort into account only if the read instruction is executed and it is probable that several aborts have occurred during this time. Thus, in this case, it is preferable to use the content of the Abort Link register of the ARM processor.

18.4.5 Memory Protection Unit

The Memory Protection Unit allows definition of up to 16 memory spaces within the internal memories. Note that the external memories can not be protected.

After reset, the Memory Protection Unit is disabled. Enabling it requires writing the Protection Unit Enable Register (MC_PUER) with the PUEB at 1.

Programming of the 16 memory spaces is done in the registers MC_PUIA0 to MC_PUIA15.

The size of each of the memory spaces is programmable by a power of 2 between 1K bytes and 4M bytes. The base address is also programmable on a number of bits according to the size.

The Memory Protection Unit also allows the protection of the peripherals by programming the Protection Unit Peripheral Register (MC_PUP) with the field PROT at the appropriate value.

The peripheral address space and each internal memory area can be protected against write and non-privileged access of one of the masters. When one of the masters performs a forbidden access, an Abort is generated and the Abort Status traces what has happened.

There is no priority in the protection of the memory spaces. In case of overlap between several memory spaces, the strongest protection is taken into account. If an access is performed to an address which is not contained in any of the 16 memory spaces, the Memory Protection Unit generates an abort.

The reset value of MC_PUIAx registers is 0, which blocks all access to the first 1K of memory starting at address 0, which prevents the core from reading exception vectors. **Therefore, all regions must be programmed to allow read/write access on the first 4M Bytes of the memory range during MPU initialization.**

18.4.6 Embedded Flash Controller

The Embedded Flash Controller is added to the Memory Controller and ensures the interface of the flash block with the 32-bit internal bus. It allows an increase of performance in Thumb Mode for Code Fetch with its system of 32-bit buffers. It also manages with the programming, erasing, locking and unlocking sequences thanks to a full set of commands.

18.4.7 Misalignment Detector

The Memory Controller features a Misalignment Detector that checks the consistency of the accesses.

For each access, regardless of the master, the size of the access and the bits 0 and 1 of the address bus are checked. If the type of access is a word (32-bit) and the bits 0 and 1 are not 0, or if the type of the access is a half-word (16-bit) and the bit 0 is not 0, an abort is returned to the master and the access is cancelled. Note that the accesses of the ARM processor when it is fetching instructions are not checked.

The misalignments are generally due to software bugs leading to wrong pointer handling. These bugs are particularly difficult to detect in the debug phase.

As the requested address is saved in the Abort Status Register and the address of the instruction generating the misalignment is saved in the Abort Link Register of the processor, detection and fix of this kind of software bugs is simplified.

18.5 Memory Controller (MC) User Interface

Base Address: 0xFFFFFF00

Table 18-1. Memory Controller (MC) Memory Mapping

Offset	Register	Name	Access	Reset State
0x00	MC Remap Control Register	MC_RCR	Write-only	
0x04	MC Abort Status Register	MC_ASR	Read-only	0x0
0x08	MC Abort Address Status Register	MC_AASR	Read-only	0x0
0x0C	Reserved			
0x10	MC Protection Unit Area 0	MC_PUIA0	Read/Write	0x0
0x14	MC Protection Unit Area 1	MC_PUIA1	Read/Write	0x0
0x18	MC Protection Unit Area 2	MC_PUIA2	Read/Write	0x0
0x1C	MC Protection Unit Area 3	MC_PUIA3	Read/Write	0x0
0x20	MC Protection Unit Area 4	MC_PUIA4	Read/Write	0x0
0x24	MC Protection Unit Area 5	MC_PUIA5	Read/Write	0x0
0x28	MC Protection Unit Area 6	MC_PUIA6	Read/Write	0x0
0x2C	MC Protection Unit Area 7	MC_PUIA7	Read/Write	0x0
0x30	MC Protection Unit Area 8	MC_PUIA8	Read/Write	0x0
0x34	MC Protection Unit Area 9	MC_PUIA9	Read/Write	0x0
0x38	MC Protection Unit Area 10	MC_PUIA10	Read/Write	0x0
0x3C	MC Protection Unit Area 11	MC_PUIA11	Read/Write	0x0
0x40	MC Protection Unit Area 12	MC_PUIA12	Read/Write	0x0
0x44	MC Protection Unit Area 13	MC_PUIA13	Read/Write	0x0
0x48	MC Protection Unit Area 14	MC_PUIA14	Read/Write	0x0
0x4C	MC Protection Unit Area 15	MC_PUIA15	Read/Write	0x0
0x50	MC Protection Unit Peripherals	MC_PUP	Read/Write	0x0
0x54	MC Protection Unit Enable Register	MC_PUER	Read/Write	0x0
0x60	EFC0 Configuration Registers	See EFC0 User Interface		
0x70	EFC1 Configuration Registers	See EFC1 User Interface		
0x80	External bus Interface Registers	See EBI User Interface		
0x90	SMC Configuration Registers	See SMC User Interface		
0xB0	SDRAMC Configuration Registers	See SDRAMC User Interface		
0xDC	ECC Configuration Registers	See ECC User Interface		

18.5.1 MC Remap Control Register

Name: MC_RCR

Access: Write-only

Absolute Address: 0xFFFF FF00

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

- **RCB: Remap Command Bit**

0: No effect.

1: This Command Bit acts on a toggle basis: writing a 1 alternatively cancels and restores the remapping of the page zero memory devices.

18.5.2 MC Abort Status Register

Name: MC_ASR

Access: Read-only

Reset Value: 0x0

Absolute Address: 0xFFFF FF04

31	30	29	28	27	26	25	24
–	–	–	–	–	–	SVMST1	SVMST0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	MST1	MST0
15	14	13	12	11	10	9	8
–	–	–	–	ABTTYP		ABTSZ	
7	6	5	4	3	2	1	0
–	–	–	–	–	MPU	MISADD	UNDADD

- **UNDADD: Undefined Address Abort Status**

0: The last abort was not due to the access of an undefined address in the address space.

1: The last abort was due to the access of an undefined address in the address space.

- **MISADD: Misaligned Address Abort Status**

0: The last aborted access was not due to an address misalignment.

1: The last aborted access was due to an address misalignment.

- **MPU: Memory Protection Unit Abort Status**

0: The last aborted access was not due to the Memory Protection Unit.

1: The last aborted access was due to the Memory Protection Unit.

- **ABTSZ: Abort Size Status**

ABTSZ		Abort Size
0	0	Byte
0	1	Half-word
1	0	Word
1	1	Reserved

- **ABTTYP: Abort Type Status**

ABTTYP		Abort Type
0	0	Data Read
0	1	Data Write
1	0	Code Fetch
1	1	Reserved

- **MST0: PDC Abort Source**

0: The last aborted access was not due to the PDC.

1: The last aborted access was due to the PDC.

- **MST1: ARM7TDMI Abort Source**

0: The last aborted access was not due to the ARM7TDMI.

1: The last aborted access was due to the ARM7TDMI.

- **SVMST0: Saved PDC Abort Source**

0: No abort due to the PDC occurred.

1: At least one abort due to the PDC occurred.

- **SVMST1:** Saved ARM7TDMI Abort Source

0: No abort due to the ARM7TDMI occurred.

1: At least one abort due to the ARM7TDMI occurred.

18.5.3 MC Abort Address Status Register

Name: MC_AASR

Access: Read-only

Reset Value: 0x0

Absolute Address: 0xFFFF FF08

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

- **ABTADD: Abort Address**

This field contains the address of the last aborted access.

18.5.4 MC Protection Unit Area 0 to 15 Registers

Name: MC_PUIA0 - MC_PUIA15

Access: Read/Write

Reset Value: 0x0

Absolute Address: 0xFFFFFF10 - 0xFFFFFF4C

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

- **PROT: Protection**

PROT		Processor Mode	
		Privilege	User
0	0	No access	No access
0	1	Read/Write	No access
1	0	Read/Write	Read-only
1	1	Read/Write	Read/Write

- **SIZE: Internal Area Size**

SIZE				Area Size	LSB of BA
0	0	0	0	1 KB	10
0	0	0	1	2 KB	11
0	0	1	0	4 KB	12
0	0	1	1	8 KB	13
0	1	0	0	16 KB	14
0	1	0	1	32 KB	15
0	1	1	0	64 KB	16
0	1	1	1	128 KB	17
1	0	0	0	256 KB	18
1	0	0	1	512 KB	19
1	0	1	0	1 MB	20
1	0	1	1	2 MB	21
1	1	0	1	4 MB	22

- **BA: Internal Area Base Address**

These bits define the Base Address of the area. Note that only the most significant bits of BA are significant. The number of significant bits are in respect with the size of the area.

18.5.5 MC Protection Unit Peripheral

Name: MC_PUP

Access: Read/Write

Reset Value: 0x000000000

Absolute Address: 0xFFFFFF50

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

- **PROT: Protection**

PROT		Processor Mode	
		Privilege	User
0	0	Read/Write	No access
0	1	Read/Write	No access
1	0	Read/Write	Read-only
1	1	Read/Write	Read/Write

18.5.6 MC Protection Unit Enable Register

Name: MC_PUER

Access: Read/Write

Reset Value: 0x000000000

Absolute Address: 0xFFFFFF54

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

- **PUEB: Protection Unit Enable Bit**

0: The Memory Controller Protection Unit is disabled.

1: The Memory Controller Protection Unit is enabled.

19. Embedded Flash Controller (EFC)

19.1 Overview

The Embedded Flash Controller (EFC) is a part of the Memory Controller and ensures the interface of the Flash block with the 32-bit internal bus. It increases performance in Thumb Mode for Code Fetch with its system of 32-bit buffers. It also manages the programming, erasing, locking and unlocking sequences using a full set of commands.

The SAM7SE512 is equipped with two EFCs, EFC0 and EFC1. EFC1 does not feature the Security bit and GPNVM bits. The Security bit and GPNVM bits embedded only on EFC0 apply to the two blocks in the SAM7SE512.

The SAM7SE256/32 is equipped with one EFC (EFC0).

19.2 Functional Description

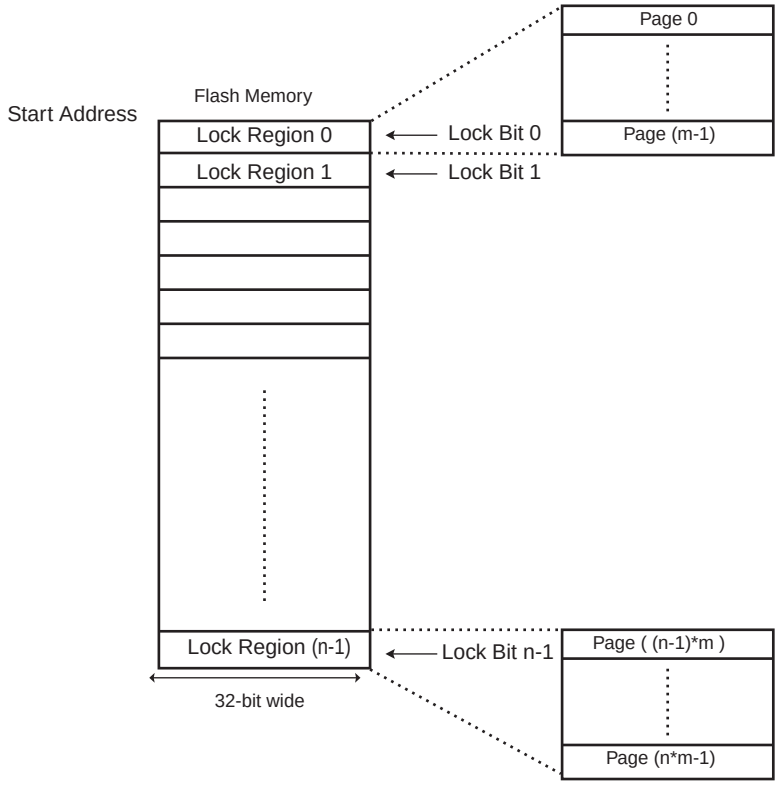
19.2.1 Embedded Flash Organization

The Embedded Flash interfaces directly to the 32-bit internal bus. It is composed of several interfaces:

- One memory plane organized in several pages of the same size
- Two 32-bit read buffers used for code read optimization (see “Read Operations” on page 102).
- One write buffer that manages page programming. The write buffer size is equal to the page size. This buffer is write-only and accessible all along the 1 MByte address space, so that each word can be written to its final address (see “Write Operations” on page 104).
- Several lock bits used to protect write and erase operations on lock regions. A lock region is composed of several consecutive pages, and each lock region has its associated lock bit.
- Several general-purpose NVM bits. Each bit controls a specific feature in the device. Refer to the product definition section to get the GPNVM assignment.

The Embedded Flash size, the page size and the lock region organization are described in the product definition section.

Figure 19-1. Embedded Flash Memory Mapping



19.2.2 Read Operations

An optimized controller manages embedded Flash reads. A system of 2 x 32-bit buffers is added in order to start access at following address during the second read, thus increasing performance when the processor is running in Thumb mode (16-bit instruction set). See Figure 19-2, Figure 19-3 and Figure 19-4.

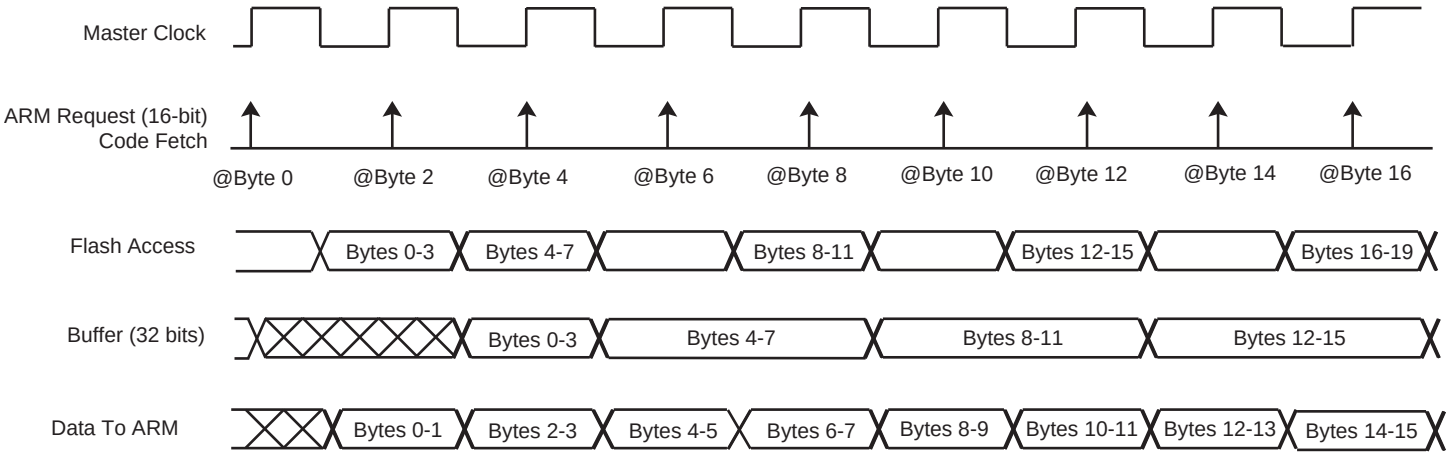
This optimization concerns only Code Fetch and not Data.

The read operations can be performed with or without wait state. Up to 3 wait states can be programmed in the field FWS (Flash Wait State) in the Flash Mode Register MC_FMR (see "MC Flash Mode Register" on page 111). Defining FWS to be 0 enables the single-cycle access of the embedded Flash.

The Flash memory is accessible through 8-, 16- and 32-bit reads.

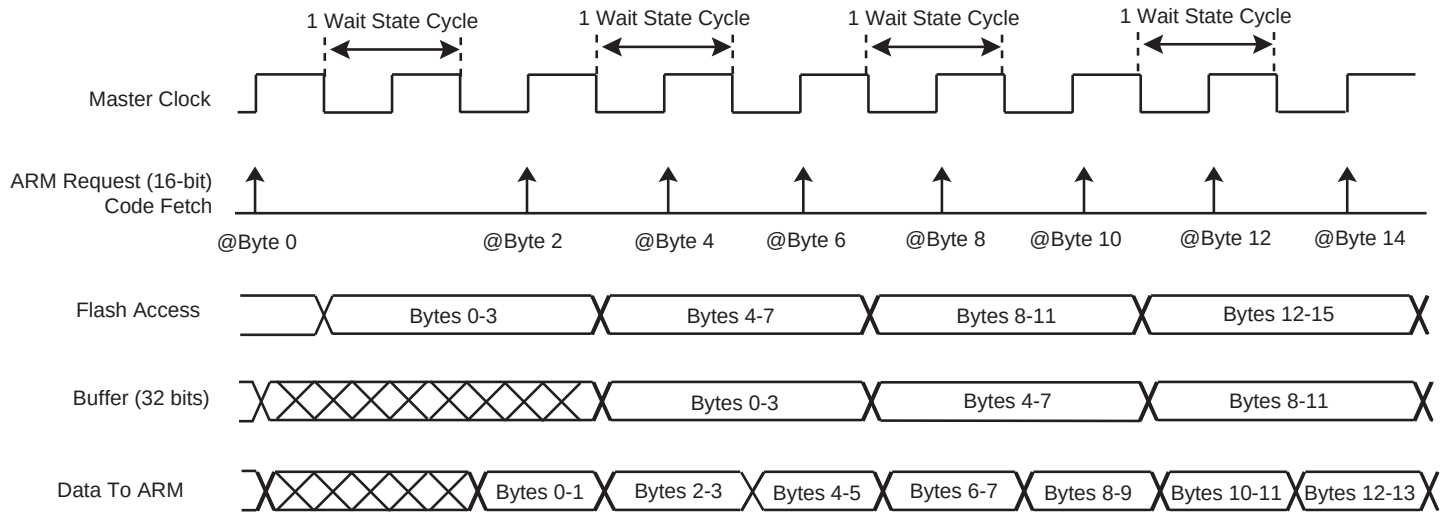
As the Flash block size is smaller than the address space reserved for the internal memory area, the embedded Flash wraps around the address space and appears to be repeated within it.

Figure 19-2. Code Read Optimization in Thumb Mode for FWS = 0



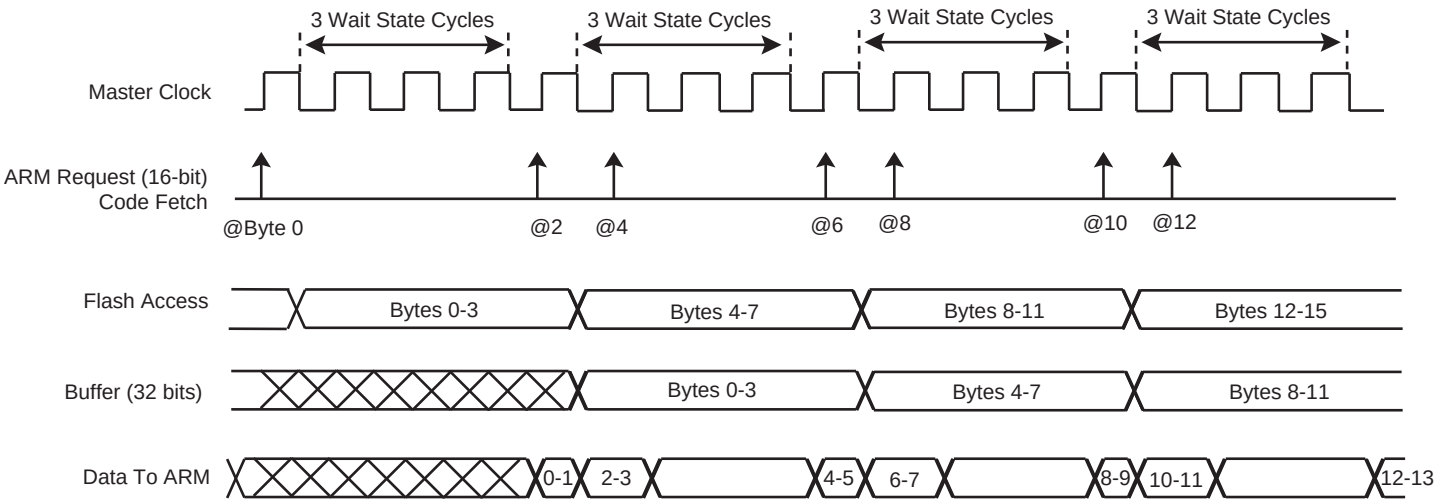
Note: When FWS is equal to 0, all accesses are performed in a single-cycle access.

Figure 19-3. Code Read Optimization in Thumb Mode for FWS = 1



Note: When FWS is equal to 1, in case of sequential reads, all the accesses are performed in a single-cycle access (except for the first one).

Figure 19-4. Code Read Optimization in Thumb Mode for FWS = 3



Note: When FWS is equal to 2 or 3, in case of sequential reads, the first access takes FWS cycles, the second access one cycle, the third access FWS cycles, the fourth access one cycle, etc.

19.2.3 Write Operations

The internal memory area reserved for the embedded Flash can also be written through a write-only latch buffer. Write operations take into account only the 8 lowest address bits and thus wrap around within the internal memory area address space and appear to be repeated 1024 times within it.

Write operations can be prevented by programming the Memory Protection Unit of the product.

Writing 8-bit and 16-bit data is not allowed and may lead to unpredictable data corruption.

Write operations are performed in the number of wait states equal to the number of wait states for read operations + 1, except for FWS = 3 (see “MC Flash Mode Register” on page 111).

19.2.4 Flash Commands

The EFC offers a command set to manage programming the memory flash, locking and unlocking lock sectors, consecutive programming and locking, and full Flash erasing.

Table 19-1. Set of Commands

Command	Value	Mnemonic
Write page	0x01	WP
Set Lock Bit	0x02	SLB
Write Page and Lock	0x03	WPL
Clear Lock Bit	0x04	CLB
Erase all	0x08	EA
Set General-purpose NVM Bit	0x0B	SGPB
Clear General-purpose NVM Bit	0x0D	CGPB
Set Security Bit	0x0F	SSB

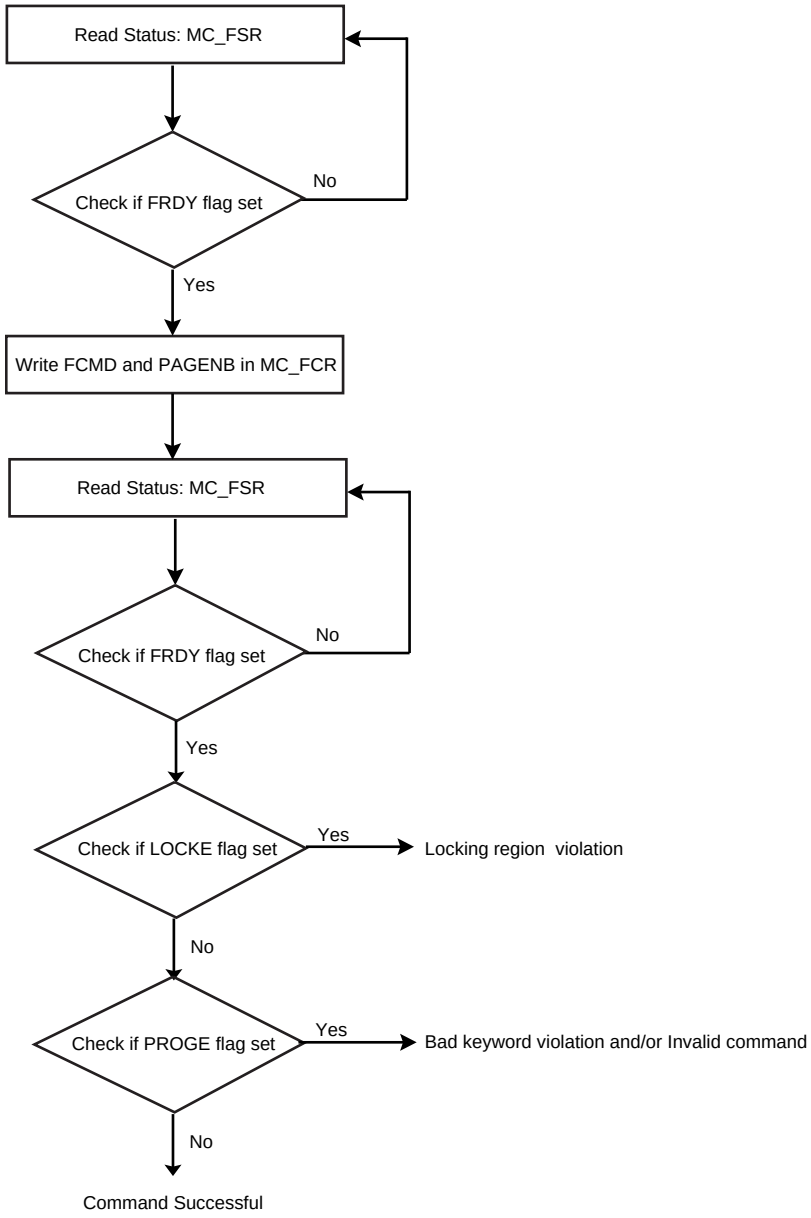
To run one of these commands, the field FCMD of the MC_FCR register has to be written with the command number. As soon as the MC_FCR register is written, the FRDY flag is automatically cleared. Once the current command is achieved, then the FRDY flag is automatically set. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

All the commands are protected by the same keyword, which has to be written in the eight highest bits of the MC_FCR register.

Writing MC_FCR with data that does not contain the correct key and/or with an invalid command has no effect on the memory plane; however, the PROGE flag is set in the MC_FSR register. This flag is automatically cleared by a read access to the MC_FSR register.

When the current command writes or erases a page in a locked region, the command has no effect on the whole memory plane; however, the LOCKE flag is set in the MC_FSR register. This flag is automatically cleared by a read access to the MC_FSR register.

Figure 19-5. Command State Chart



In order to guarantee valid operations on the Flash memory, the field Flash Microsecond Cycle Number (FMCN) in the Flash Mode Register MC_FMR must be correctly programmed (see "MC Flash Mode Register" on page 111).

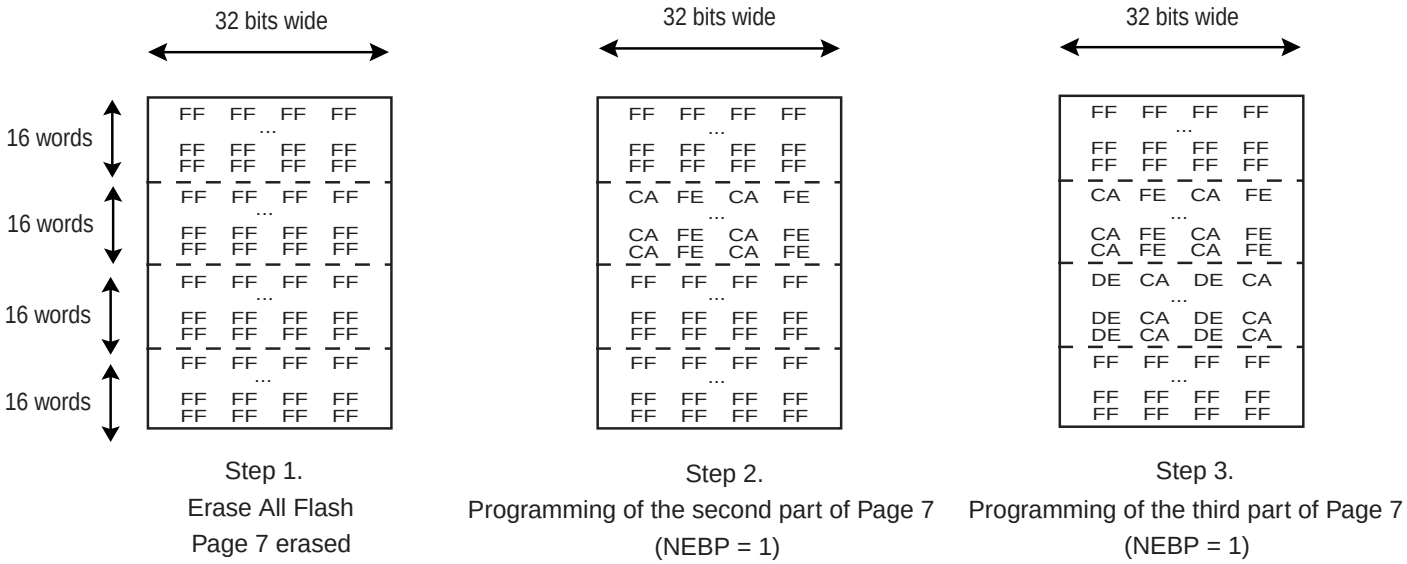
19.2.4.1 Flash Programming

Several commands can be used to program the Flash.

The Flash technology requires that an erase must be done before programming. The entire memory plane can be erased at the same time, or a page can be automatically erased by clearing the NEBP bit in the MC_FMR register before writing the command in the MC_FCR register.

By setting the NEBP bit in the MC_FMR register, a page can be programmed in several steps if it has been erased before (see Figure 19-6).

Figure 19-6. Example of Partial Page Programming:



The Partial Programming mode works only with 32-bit (or higher) boundaries. It cannot be used with boundaries lower than 32 bits (8 or 16-bit for example).

After programming, the page (the whole lock region) can be locked to prevent miscellaneous write or erase sequences. The lock bit can be automatically set after page programming using WPL.

Data to be written is stored in an internal latch buffer. The size of the latch buffer corresponds to the page size. The latch buffer wraps around within the internal memory area address space and appears to be repeated by the number of pages in it.

Note: Writing of 8-bit and 16-bit data is not allowed and may lead to unpredictable data corruption.

Data is written to the latch buffer before the programming command is written to the Flash Command Register MC_FCR. The sequence is as follows:

- Write the full page, at any page address, within the internal memory area address space using only 32-bit access.
- Programming starts as soon as the page number and the programming command are written to the Flash Command Register. The FRDY bit in the Flash Programming Status Register (MC_FSR) is automatically cleared.
- When programming is completed, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt was enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

Two errors can be detected in the MC_FSR register after a programming sequence:

- Programming Error: A bad keyword and/or an invalid command have been written in the MC_FCR register.
- Lock Error: The page to be programmed belongs to a locked region. A command must be previously run to unlock the corresponding region.

19.2.4.2 Erase All Command

The entire memory can be erased if the Erase All Command (EA) in the Flash Command Register MC_FCR is written.

Erase All operation is allowed only if there are no lock bits set. Thus, if at least one lock region is locked, the bit LOCKE in MC_FSR rises and the command is cancelled. If the bit LOCKE has been written at 1 in MC_FMR, the interrupt line rises.

When programming is complete, the bit FRDY bit in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

Two errors can be detected in the MC_FSR register after a programming sequence:

- Programming Error: A bad keyword and/or an invalid command have been written in the MC_FCR register.
- Lock Error: At least one lock region to be erased is protected. The erase command has been refused and no page has been erased. A Clear Lock Bit command must be executed previously to unlock the corresponding lock regions.

19.2.4.3 Lock Bit Protection

Lock bits are associated with several pages in the embedded Flash memory plane. This defines lock regions in the embedded Flash memory plane. They prevent writing/erasing protected pages.

After production, the device may have some embedded Flash lock regions locked. These locked regions are reserved for a default application. Refer to the product definition section for the default embedded Flash mapping. Locked sectors can be unlocked to be erased and then programmed with another application or other data.

The lock sequence is:

- The Flash Command register must be written with the following value:
  (0x5A << 24) | (lockPageNumber << 8 & PAGEN) | SLB
  lockPageNumber is a page of the corresponding lock region.
- When locking completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

A programming error, where a bad keyword and/or an invalid command have been written in the MC_FCR register, may be detected in the MC_FSR register after a programming sequence.

It is possible to clear lock bits that were set previously. Then the locked region can be erased or programmed. The unlock sequence is:

- The Flash Command register must be written with the following value:
  (0x5A << 24) | (lockPageNumber << 8 & PAGEN) | CLB
  lockPageNumber is a page of the corresponding lock region.
- When the unlock completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

A programming error, where a bad keyword and/or an invalid command have been written in the MC_FCR register, may be detected in the MC_FSR register after a programming sequence.

The Unlock command programs the lock bit to 1; the corresponding bit LOCKSx in MC_FSR reads 0. The Lock command programs the lock bit to 0; the corresponding bit LOCKSx in MC_FSR reads 1.

Note: Access to the Flash in Read Mode is permitted when a Lock or Unlock command is performed.

19.2.4.4 General-purpose NVM Bits

General-purpose NVM bits do not interfere with the embedded Flash memory plane. (Does not apply to EFC1 on the SAM7SE512.) These general-purpose bits are dedicated to protect other parts of the product. They can be set (activated) or cleared individually. Refer to the product definition section for the general-purpose NVM bit action.

The activation sequence is:

- Start the Set General Purpose Bit command (SGPB) by writing the Flash Command Register with the SEL command and the number of the general-purpose bit to be set in the PAGEN field.
- When the bit is set, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

Two errors can be detected in the MC_FSR register after a programming sequence:

- Programming Error: A bad keyword and/or an invalid command have been written in the MC_FCR register
- If the general-purpose bit number is greater than the total number of general-purpose bits, then the command has no effect.

It is possible to deactivate a general-purpose NVM bit set previously. The clear sequence is:

- Start the Clear General-purpose Bit command (CGPB) by writing the Flash Command Register with CGPB and the number of the general-purpose bit to be cleared in the PAGEN field.
- When the clear completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

Two errors can be detected in the MC_FSR register after a programming sequence:

- Programming Error: a bad keyword and/or an invalid command have been written in the MC_FCR register
- If the number of the general-purpose bit set in the PAGEN field is greater than the total number of general-purpose bits, then the command has no effect.

The Clear General-purpose Bit command programs the general-purpose NVM bit to 0; the corresponding bit GPNVM0 to GPNVMx in MC_FSR reads 0. The Set General-purpose Bit command programs the general-purpose NVM bit to 1; the corresponding bit GPNVMx in MC_FSR reads 1.

Note: Access to the Flash in read mode is permitted when a Set, Clear or Get General-purpose NVM Bit command is performed.

19.2.4.5 Security Bit

The goal of the security bit is to prevent external access to the internal bus system. (Does not apply to EFC1 on the SAM7SE512.) JTAG, Fast Flash Programming and Flash Serial Test Interface features are disabled. Once set, this bit can be reset only by an external hardware ERASE request to the chip. Refer to the product definition section for the pin name that controls the ERASE. In this case, the full memory plane is erased and all lock and general-purpose NVM bits are cleared. The security bit in the MC_FSR is cleared only after these operations. The activation sequence is:

- Start the Set Security Bit command (SSB) by writing the Flash Command Register.

- When the locking completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

When the security bit is active, the SECURITY bit in the MC_FSR is set.

19.3 Embedded Flash Controller (EFC) User Interface

The User Interface of the EFC is integrated within the Memory Controller with Base Address: 0xFFFF FF00.

The SAM7SE512 is equipped with two EFCs, EFC0 and EFC1, as described in the Register Mapping tables and Register descriptions that follow.

The SAM7SE256/32 is equipped with one EFC (EFC0).

Table 19-2. Embedded Flash Controller (EFC0) Register Mapping

Offset	Register	Name	Access	Reset State
0x60	MC Flash Mode Register	MC_FMR	Read/Write	0x0
0x64	MC Flash Command Register	MC_FCR	Write-only	–
0x68	MC Flash Status Register	MC_FSR	Read-only	–
0x6C	Reserved	–	–	–

Table 19-3. Embedded Flash Controller (EFC1) Register Mapping

Offset	Register	Name	Access	Reset State
0x70	MC Flash Mode Register	MC_FMR	Read/Write	0x0
0x74	MC Flash Command Register	MC_FCR	Write-only	–
0x78	MC Flash Status Register	MC_FSR	Read-only	–
0x7C	Reserved	–	–	–

19.3.1 MC Flash Mode Register

Name: MC_FMR

Access: Read/Write

Offset: (EFC0) 0x60

Offset: (EFC1) 0x70

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

23	22	21	20	19	18	17	16
FMCN							

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

7	6	5	4	3	2	1	0
NEBP	–	–	–	PROGE	LOCKE	–	FRDY

- **FRDY: Flash Ready Interrupt Enable**

0: Flash Ready does not generate an interrupt.

1: Flash Ready generates an interrupt.

- **LOCKE: Lock Error Interrupt Enable**

0: Lock Error does not generate an interrupt.

1: Lock Error generates an interrupt.

- **PROGE: Programming Error Interrupt Enable**

0: Programming Error does not generate an interrupt.

1: Programming Error generates an interrupt.

- **NEBP: No Erase Before Programming**

0: A page erase is performed before programming.

1: No erase is performed before programming.

- **FWS: Flash Wait State**

This field defines the number of wait states for read and write operations:

FWS	Read Operations	Write Operations
0	1 cycle	2 cycles
1	2 cycles	3 cycles
2	3 cycles	4 cycles
3	4 cycles	4 cycles

- **FMCN: Flash Microsecond Cycle Number**

Before writing Non Volatile Memory bits (Lock bits, General Purpose NVM bit and Security bits), this field must be set to the number of Master Clock cycles in one microsecond.

When writing the rest of the Flash, this field defines the number of Master Clock cycles in 1.5 microseconds. This number must be rounded up.

Warning: The value 0 is only allowed for a master clock period superior to 30 microseconds.

Warning: In order to guarantee valid operations on the flash memory, the field Flash Microsecond Cycle Number (FMCN) **must be** correctly programmed.

19.3.2 MC Flash Command Register

Name: MC_FCR

Access: Write-only

Offset: (EFC0) 0x64

Offset: (EFC1) 0x74

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
PAGEN							

7	6	5	4	3	2	1	0
–	–	–	–	FCMD			

- **FCMD: Flash Command**

This field defines the Flash commands:

FCMD	Operations
0000	No command. Does not raise the Programming Error Status flag in the Flash Status Register MC_FSR.
0001	Write Page Command (WP): Starts the programming of the page specified in the PAGEN field.
0010	Set Lock Bit Command (SLB): Starts a set lock bit sequence of the lock region specified in the PAGEN field.
0011	Write Page and Lock Command (WPL): The lock sequence of the lock region associated with the page specified in the field PAGEN occurs automatically after completion of the programming sequence.
0100	Clear Lock Bit Command (CLB): Starts a clear lock bit sequence of the lock region specified in the PAGEN field.
1000	Erase All Command (EA): Starts the erase of the entire Flash. If at least one page is locked, the command is cancelled.
1011	Set General-purpose NVM Bit (SGPB): Activates the general-purpose NVM bit corresponding to the number specified in the PAGEN field.
1101	Clear General Purpose NVM Bit (CGPB): Deactivates the general-purpose NVM bit corresponding to the number specified in the PAGEN field.
1111	Set Security Bit Command (SSB): Sets security bit.
Others	Reserved. Raises the Programming Error Status flag in the Flash Status Register MC_FSR.

- **PAGEN: Page Number**

Command	PAGEN Description
Write Page Command	PAGEN defines the page number to be written.
Write Page and Lock Command	PAGEN defines the page number to be written and its associated lock region.
Erase All Command	This field is meaningless
Set/Clear Lock Bit Command	PAGEN defines one page number of the lock region to be locked or unlocked.
Set/Clear General Purpose NVM Bit Command	PAGEN defines the general-purpose bit number.
Set Security Bit Command	This field is meaningless

Note: Depending on the command, all the possible unused bits of PAGEN are meaningless.

- **KEY: Write Protection Key**

This field should be written with the value 0x5A to enable the command defined by the bits of the register. If the field is written with a different value, the write is not performed and no action is started.

19.3.3 MC Flash Status Register

Name: MC_FSR

Access: Read-only

Offset: (EFC0) 0x68

Offset: (EFC1) 0x78

31	30	29	28	27	26	25	24
LOCKS15	LOCKS14	LOCKS13	LOCKS12	LOCKS11	LOCKS10	LOCKS9	LOCKS8
23	22	21	20	19	18	17	16
LOCKS7	LOCKS6	LOCKS5	LOCKS4	LOCKS3	LOCKS2	LOCKS1	LOCKS0
15	14	13	12	11	10	9	8
–	–	–	–	–	GPNVM2	GPNVM1	GPNVM0
7	6	5	4	3	2	1	0
–	–	–	SECURITY	PROGE	LOCKE	–	FRDY

- **FRDY: Flash Ready Status**

0: The EFC is busy and the application must wait before running a new command.

1: The EFC is ready to run a new command.

- **LOCKE: Lock Error Status**

0: No programming of at least one locked lock region has happened since the last read of MC_FSR.

1: Programming of at least one locked lock region has happened since the last read of MC_FSR.

- **PROGE: Programming Error Status**

0: No invalid commands and no bad keywords were written in the Flash Command Register MC_FCR.

1: An invalid command and/or a bad keyword was/were written in the Flash Command Register MC_FCR.

- **SECURITY: Security Bit Status** (Does not apply to EFC1 on the SAM7SE512.)

0: The security bit is inactive.

1: The security bit is active.

- **GPNVMx: General-purpose NVM Bit Status** (Does not apply to EFC1 on the SAM7SE512.)

0: The corresponding general-purpose NVM bit is inactive.

1: The corresponding general-purpose NVM bit is active.

- **EFC LOCKSx: Lock Region x Lock Status**

0: The corresponding lock region is not locked.

1: The corresponding lock region is locked.

LOCKS 8-15 do not apply to SAM7SE32.

20. Fast Flash Programming Interface (FFPI)

20.1 Overview

The Fast Flash Programming Interface provides two solutions - parallel or serial - for high-volume programming using a standard gang programmer. The parallel interface is fully handshaked and the device is considered to be a standard EEPROM. Additionally, the parallel protocol offers an optimized access to all the embedded Flash functionalities. The serial interface uses the standard IEEE 1149.1 JTAG protocol. It offers an optimized access to all the embedded Flash functionalities.

Although the Fast Flash Programming Mode is a dedicated mode for high volume programming, this mode not designed for in-situ programming.

20.2 Parallel Fast Flash Programming

20.2.1 Device Configuration

In Fast Flash Programming Mode, the device is in a specific test mode. Only a certain set of pins is significant. Other pins must be left unconnected.

Figure 20-1. Parallel Programming Interface

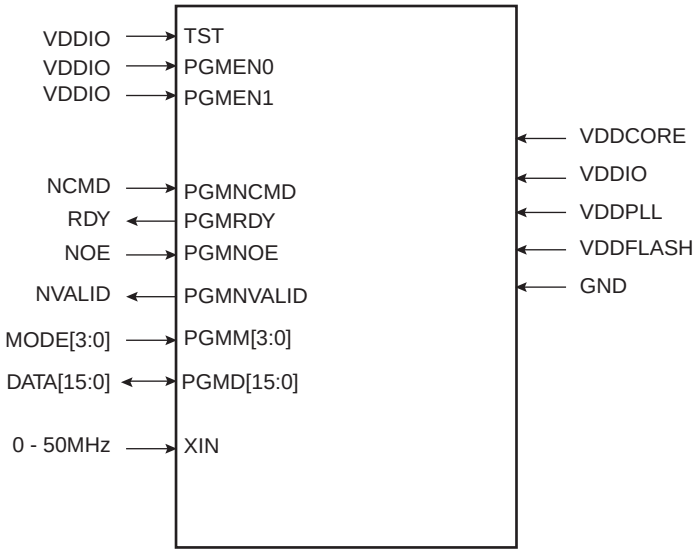


Table 20-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDFLASH	Flash Power Supply	Power		
VDDIO	I/O Lines Power Supply	Power		
VDDCORE	Core Power Supply	Power		
VDDPLL	PLL Power Supply	Power		
GND	Ground	Ground		
Clocks				
XIN	Main Clock Input. This input can be tied to GND. In this case, the device is clocked by the internal RC oscillator.	Input		32KHz to 50MHz
Test				
TST	Test Mode Select	Input	High	Must be connected to VDDIO
PGMEN0	Test Mode Select	Input	High	Must be connected to VDDIO
PGMEN1	Test Mode Select	Input	High	Must be connected to VDDIO

Table 20-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
PIO				
PGMNCMD	Valid command available	Input	Low	Pulled-up input at reset
PGMRDY	0: Device is busy 1: Device is ready for a new command	Output	High	Pulled-up input at reset
PGMNOE	Output Enable (active high)	Input	Low	Pulled-up input at reset
PGMNVALID	0: DATA[15:0] is in input mode 1: DATA[15:0] is in output mode	Output	Low	Pulled-up input at reset
PGMM[3:0]	Specifies DATA type (See Table 20-2)	Input		Pulled-up input at reset
PGMD[15:0]	Bidirectional data bus	Input/Output		Pulled-up input at reset

20.2.2 Signal Names

Depending on the MODE settings, DATA is latched in different internal registers.

Table 20-2. Mode Coding

MODE[3:0]	Symbol	Data
0000	CMDE	Command Register
0001	ADDR0	Address Register LSBs
0010	ADDR1	
0101	DATA	Data Register
Default	IDLE	No register

When MODE is equal to CMDE, then a new command (strobed on DATA[15:0] signals) is stored in the command register.

Table 20-3. Command Bit Coding

DATA[15:0]	Symbol	Command Executed
0x0011	READ	Read Flash
0x0012	WP	Write Page Flash
0x0022	WPL	Write Page and Lock Flash
0x0032	EWP	Erase Page and Write Page
0x0042	EWPL	Erase Page and Write Page then Lock
0x0013	EA	Erase All
0x0014	SLB	Set Lock Bit
0x0024	CLB	Clear Lock Bit
0x0015	GLB	Get Lock Bit
0x0034	SFB	Set General Purpose NVM bit
0x0044	CFB	Clear General Purpose NVM bit
0x0025	GFB	Get General Purpose NVM bit
0x0054	SSE	Set Security Bit
0x0035	GSE	Get Security Bit
0x001F	WRAM	Write Memory
0x0016	SEFC	Select EFC Controller (1)
0x001E	GVE	Get Version

Note: 1. Applies to SAM7SE512.

20.2.3 Entering Programming Mode

The following algorithm puts the device in Parallel Programming Mode:

- Apply GND, VDDIO, VDDCORE, VDDFLASH and VDDPLL.
- Apply XIN clock within T_{POR_RESET} if an external clock is available.
- Wait for T_{POR_RESET}
- Start a read or write handshaking.

Note: After reset, the device is clocked by the internal RC oscillator. Before clearing RDY signal, if an external clock (> 32 kHz) is connected to XIN, then the device switches on the external clock. Else, XIN input is not considered. A higher frequency on XIN speeds up the programmer handshake.

20.2.4 Programmer Handshaking

An handshake is defined for read and write operations. When the device is ready to start a new operation (RDY signal set), the programmer starts the handshake by clearing the NCMD signal. The handshaking is achieved once NCMD signal is high and RDY is high.

20.2.4.1 *Write Handshaking*

For details on the write handshaking sequence, refer to Figure 20-2and Table 20-4.

Figure 20-2. Parallel Programming Timing, Write Sequence

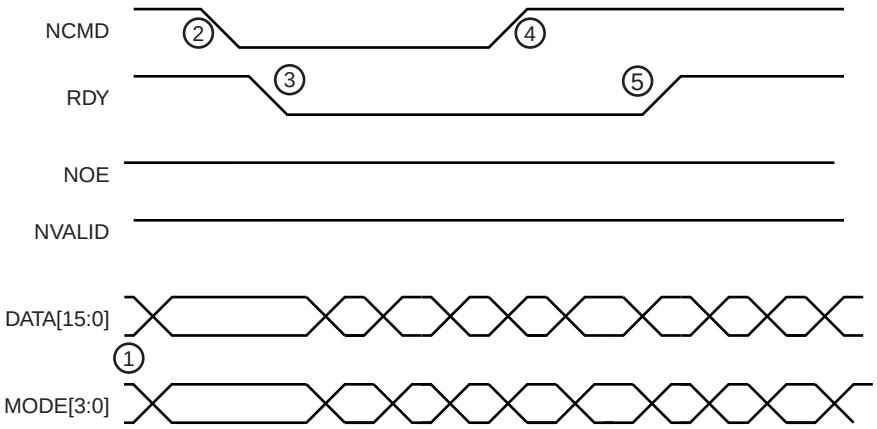


Table 20-4. Write Handshake

Step	Programmer Action	Device Action	Data I/O
1	Sets MODE and DATA signals	Waits for NCMD low	Input
2	Clears NCMD signal	Latches MODE and DATA	Input
3	Waits for RDY low	Clears RDY signal	Input
4	Releases MODE and DATA signals	Executes command and polls NCMD high	Input
5	Sets NCMD signal	Executes command and polls NCMD high	Input
6	Waits for RDY high	Sets RDY	Input

20.2.4.2 *Read Handshaking*

For details on the read handshaking sequence, refer to Figure 20-3 and Table 20-5.

Figure 20-3. Parallel Programming Timing, Read Sequence

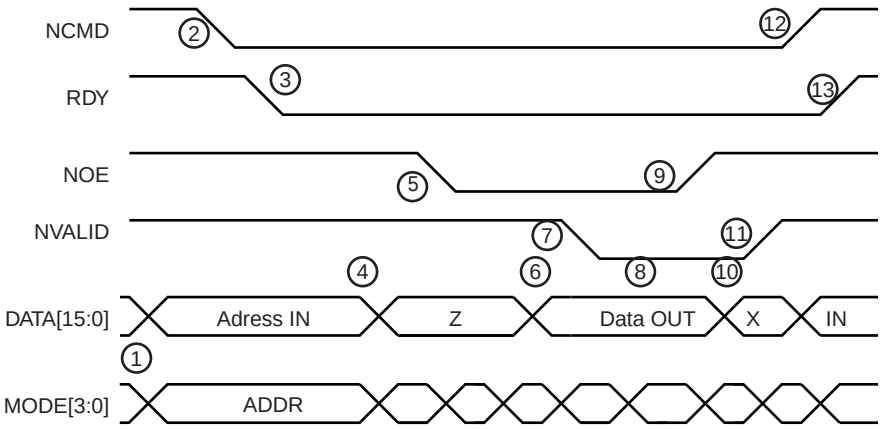


Table 20-5. Read Handshake

Step	Programmer Action	Device Action	DATA I/O
1	Sets MODE and DATA signals	Waits for NCMD low	Input
2	Clears NCMD signal	Latch MODE and DATA	Input
3	Waits for RDY low	Clears RDY signal	Input
4	Sets DATA signal in tristate	Waits for NOE Low	Input
5	Clears NOE signal		Tristate
6	Waits for NVALID low	Sets DATA bus in output mode and outputs the flash contents.	Output
7		Clears NVALID signal	Output
8	Reads value on DATA Bus	Waits for NOE high	Output
9	Sets NOE signal		Output
10	Waits for NVALID high	Sets DATA bus in input mode	X
11	Sets DATA in output mode	Sets NVALID signal	Input
12	Sets NCMD signal	Waits for NCMD high	Input
13	Waits for RDY high	Sets RDY signal	Input

20.2.5 Device Operations

Several commands on the Flash memory are available. These commands are summarized in Table 20-3 on page 120. Each command is driven by the programmer through the parallel interface running several read/write handshaking sequences.

When a new command is executed, the previous one is automatically achieved. Thus, chaining a read command after a write automatically flushes the load buffer in the Flash.

20.2.5.1 Flash Read Command

This command is used to read the contents of the Flash memory. The read command can start at any valid address in the memory plane and is optimized for consecutive reads. Read handshaking can be chained; an internal address buffer is automatically increased.

Table 20-6. Read Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	READ
2	Write handshaking	ADDR0	32-bit Memory Address First byte
3	Write handshaking	ADDR1	32-bit Flash Address
6	Read handshaking	DATA	*Memory Address++
7	Read handshaking	DATA	*Memory Address++
...	...	...	...
n	Write handshaking	ADDR0	32-bit Memory Address First byte
n+1	Write handshaking	ADDR1	32-bit Flash Address
n+2	Write handshaking	ADDR2	32-bit Flash Address
n+3	Write handshaking	ADDR3	32-bit Flash Address Last Byte
n+4	Read handshaking	DATA	*Memory Address++
n+5	Read handshaking	DATA	*Memory Address++
...	...	...	...

20.2.5.2 Flash Write Command

This command is used to write the Flash contents.

The Flash memory plane is organized into several pages. Data to be written are stored in a load buffer that corresponds to a Flash memory page. The load buffer is automatically flushed to the Flash:

- before access to any page other than the current one
- when a new command is validated (MODE = CMDE)

The **Write Page** command **(WP)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

Table 20-7. Write Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	WP or WPL or EWP or EWPL
2	Write handshaking	ADDR0	32-bit Memory Address First byte
3	Write handshaking	ADDR1	32-bit Flash Address
4	Write handshaking	ADDR2	32-bit Flash Address
5	Write handshaking	ADDR3	32-bit Flash Address Last Byte
6	Write handshaking	DATA	*Memory Address++
7	Write handshaking	DATA	*Memory Address++
...	...	...	...
n	Write handshaking	ADDR0	32-bit Memory Address First byte
n+1	Write handshaking	ADDR1	32-bit Flash Address
n+2	Write handshaking	ADDR2	32-bit Flash Address
n+3	Write handshaking	ADDR3	32-bit Flash Address Last Byte
n+4	Write handshaking	DATA	*Memory Address++
n+5	Write handshaking	DATA	*Memory Address++
...	...	...	...

The Flash command **Write Page and Lock (WPL)** is equivalent to the Flash Write Command. However, the lock bit is automatically set at the end of the Flash write operation. As a lock region is composed of several pages, the programmer writes to the first pages of the lock region using Flash write commands and writes to the last page of the lock region using a Flash write and lock command.

The Flash command **Erase Page and Write (EWP)** is equivalent to the Flash Write Command. However, before programming the load buffer, the page is erased.

The Flash command **Erase Page and Write the Lock (EWPL)** combines EWP and WPL commands.

20.2.5.3 Flash Full Erase Command

This command is used to erase the Flash memory planes.

All lock regions must be unlocked before the Full Erase command by using the CLB command. Otherwise, the erase command is aborted and no page is erased.

Table 20-8. Full Erase Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	EA
2	Write handshaking	DATA	0

20.2.5.4 Flash Lock Commands

Lock bits can be set using WPL or EWPL commands. They can also be set by using the **Set Lock** command **(SLB)**. With this command, several lock bits can be activated. A Bit Mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first lock bit is activated.

In the same way, the **Clear Lock** command **(CLB)** is used to clear lock bits. All the lock bits are also cleared by the EA command.

Table 20-9. Set and Clear Lock Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SLB or CLB
2	Write handshaking	DATA	Bit Mask

Lock bits can be read using **Get Lock Bit** command **(GLB)**. The n^{th} lock bit is active when the bit n of the bit mask is set..

Table 20-10. Get Lock Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GLB
2	Read handshaking	DATA	Lock Bit Mask Status 0 = Lock bit is cleared 1 = Lock bit is set

20.2.5.5 Flash General-purpose NVM Commands

General-purpose NVM bits (GP NVM bits) can be set using the **Set Fuse** command **(SFB)**. This command also activates GP NVM bits. A bit mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first GP NVM bit is activated.

In the same way, the **Clear Fuse** command **(CFB)** is used to clear general-purpose NVM bits. All the general-purpose NVM bits are also cleared by the EA command. The general-purpose NVM bit is deactivated when the corresponding bit in the pattern value is set to 1.

Table 20-11. Set/Clear GP NVM Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SFB or CFB
2	Write handshaking	DATA	GP NVM bit pattern value

General-purpose NVM bits can be read using the **Get Fuse Bit** command **(GFB)**. The n^{th} GP NVM bit is active when bit n of the bit mask is set..

Table 20-12. Get GP NVM Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GFB
2	Read handshaking	DATA	GP NVM Bit Mask Status 0 = GP NVM bit is cleared 1 = GP NVM bit is set

20.2.5.6 *Flash Security Bit Command*

A security bit can be set using the **Set Security Bit** command (SSE). Once the security bit is active, the Fast Flash programming is disabled. No other command can be run. An event on the Erase pin can erase the security bit once the contents of the Flash have been erased.

The SAM7SE512 security bit is controlled by the EFC0. To use the Set Security Bit command, the EFC0 must be selected using the Select EFC command.

Table 20-13. Set Security Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SSE
2	Write handshaking	DATA	0

20.2.5.7 SAM7SE512 *Select EFC Command*

The commands WPx, EA, xLB, xFB are executed using the current EFC controller. The default EFC controller is EFC0. The **Select EFC** command (SEFC) allows selection of the current EFC controller.

Table 20-14. Select EFC Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SEFC
2	Write handshaking	DATA	0 = Select EFC0 1 = Select EFC1

20.2.5.8 *Memory Write Command*

This command is used to perform a write access to any memory location.

The **Memory Write** command **(WRAM)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

Table 20-15. Write Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	WRAM
2	Write handshaking	ADDR0	32-bit Memory Address First byte
3	Write handshaking	ADDR1	32-bit Flash Address
4	Write handshaking	ADDR2	32-bit Flash Address
5	Write handshaking	ADDR3	32-bit Flash Address Last Byte
6	Write handshaking	DATA	*Memory Address++
7	Write handshaking	DATA	*Memory Address++
...	...	...	...
n	Write handshaking	ADDR0	32-bit Memory Address First byte
n+1	Write handshaking	ADDR1	32-bit Flash Address
n+2	Write handshaking	ADDR2	32-bit Flash Address
n+3	Write handshaking	ADDR3	32-bit Flash Address Last Byte

Table 20-15. Write Command (Continued)

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
n+4	Write handshaking	DATA	*Memory Address++
n+5	Write handshaking	DATA	*Memory Address++
...	...	...	...

20.2.5.9 Get Version Command

The **Get Version** (GVE) command retrieves the version of the FFPI interface.

Table 20-16. Get Version Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GVE
2	Write handshaking	DATA	Version

20.3 Serial Fast Flash Programming

The Serial Fast Flash programming interface is based on IEEE Std. 1149.1 “Standard Test Access Port and Boundary-Scan Architecture”. Refer to this standard for an explanation of terms used in this chapter and for a description of the TAP controller states.

In this mode, data read/written from/to the embedded Flash of the device are transmitted through the JTAG interface of the device.

20.3.1 Device Configuration

In Serial Fast Flash Programming Mode, the device is in a specific test mode. Only a distinct set of pins is significant. Other pins must be left unconnected.

Figure 20-4. Serial Programing

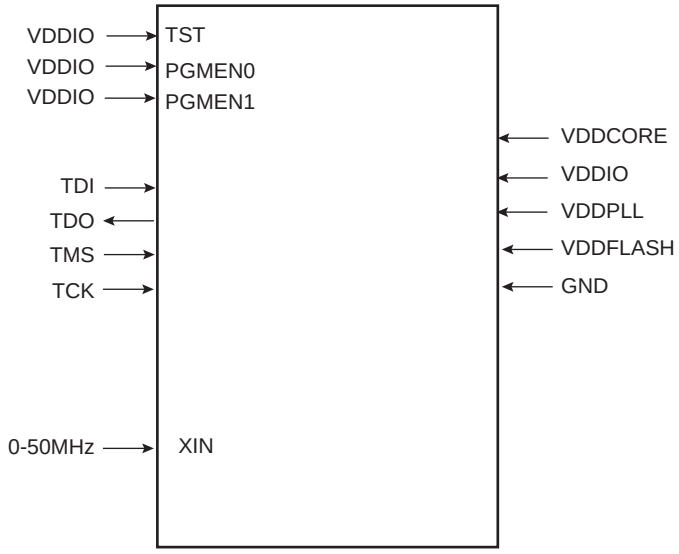


Table 20-17. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDFLASH	Flash Power Supply	Power		
VDDIO	I/O Lines Power Supply	Power		
VDDCORE	Core Power Supply	Power		
VDDPLL	PLL Power Supply	Power		
GND	Ground	Ground		
Clocks				
XIN	Main Clock Input. This input can be tied to GND. In this case, the device is clocked by the internal RC oscillator.	Input		32 kHz to 50 MHz

Table 20-17. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
Test				
TST	Test Mode Select	Input	High	Must be connected to VDDIO.
PGMEN0	Test Mode Select	Input	High	Must be connected to VDDIO
PGMEN1	Test Mode Select	Input	High	Must be connected to VDDIO
JTAG				
TCK	JTAG TCK	Input	-	Pulled-up input at reset
TDI	JTAG Test Data In	Input	-	Pulled-up input at reset
TDO	JTAG Test Data Out	Output	-	
TMS	JTAG Test Mode Select	Input	-	Pulled-up input at reset

20.3.2 Entering Serial Programming Mode

The following algorithm puts the device in Serial Programming Mode:

- Apply GND, VDDIO, VDDCORE, VDDFLASH and VDDPLL.
- Apply XIN clock within T_{POR_RESET} + 32(T_{SCLK}) if an external clock is available.
- Wait for T_{POR_RESET}.
- Reset the TAP controller clocking 5 TCK pulses with TMS set.
- Shift 0x2 into the IR register (IR is 4 bits long, LSB first) without going through the Run-Test-Idle state.
- Shift 0x2 into the DR register (DR is 4 bits long, LSB first) without going through the Run-Test-Idle state.
- Shift 0xC into the IR register (IR is 4 bits long, LSB first) without going through the Run-Test-Idle state.

Note: After reset, the device is clocked by the internal RC oscillator. Before clearing RDY signal, if an external clock (> 32 kHz) is connected to XIN, then the device will switch on the external clock. Else, XIN input is not considered. An higher frequency on XIN speeds up the programmer handshake.

Table 20-18. Reset TAP Controller and Go to Select-DR-Scan

TDI	TMS	TAP Controller State
X	1	
X	1	
X	1	
X	1	
X	1	Test-Logic Reset
X	0	Run-Test/Idle
Xt	1	Select-DR-Scan

20.3.3 Read/Write Handshake

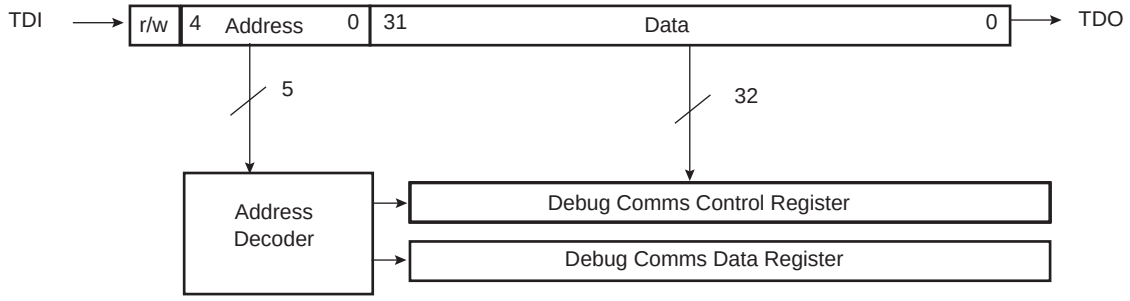
The read/write handshake is done by carrying out read/write operations on two registers of the device that are accessible through the JTAG:

- Debug Comms Control Register: DCCR
- Debug Comms Data Register: DCDR

Access to these registers is done through the TAP 38-bit DR register comprising a 32-bit data field, a 5-bit address field and a read/write bit. The data to be written is scanned into the 32-bit data field with the address of the register to the 5-bit address field and 1 to the read/write bit. A register is read by scanning its address into the address field and 0 into the read/write bit, going through the UPDATE-DR TAP state, then scanning out the data.

Refer to the ARM7TDMI reference manuel for more information on Comm channel operations.

Figure 20-5. TAP 8-bit DR Register



A read or write takes place when the TAP controller enters UPDATE-DR state. Refer to the IEEE 1149.1 for more details on JTAG operations.

- The address of the Debug Comms Control Register is 0x04.
- The address of the Debug Comms Data Register is 0x05.
  The Debug Comms Control Register is read-only and allows synchronized handshaking between the processor and the debugger.
  - Bit 1 (W): Denotes whether the programmer can read a data through the Debug Comms Data Register. If the device is busy W = 0, then the programmer must poll until W = 1.
  - Bit 0 (R): Denotes whether the programmer can send data from the Debug Comms Data Register. If R = 1, data previously placed there through the scan chain has not been collected by the device and so the programmer must wait.

The write handshake is done by polling the Debug Comms Control Register until the R bit is cleared. Once cleared, data can be written to the Debug Comms Data Register.

The read handshake is done by polling the Debug Comms Control Register until the W bit is set. Once set, data can be read in the Debug Comms Data Register.

20.3.4 Device Operations

Several commands on the Flash memory are available. These commands are summarized in Table 20-3 on page 120. Commands are run by the programmer through the serial interface that is reading and writing the Debug Comms Registers.

20.3.4.1 Flash Read Command

This command is used to read the Flash contents. The memory map is accessible through this command. Memory is seen as an array of words (32-bit wide). The read command can start at any valid address in the memory plane. **This address must be word-aligned**. The address is automatically incremented.

Table 20-19. Read Command

Read/Write	DR Data
Write	(Number of Words to Read) << 16 | READ
Write	Address
Read	Memory [address]
Read	Memory [address+4]
...	...
Read	Memory [address+(Number of Words to Read - 1)* 4]

20.3.4.2 Flash Write Command

This command is used to write the Flash contents. The address transmitted must be a valid Flash address in the memory plane.

The Flash memory plane is organized into several pages. Data to be written is stored in a load buffer that corresponds to a Flash memory page. The load buffer is automatically flushed to the Flash:

- before access to any page than the current one
- at the end of the number of words transmitted

The **Write Page** command **(WP)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

Table 20-20. Write Command

Read/Write	DR Data
Write	(Number of Words to Write) << 16 | (WP or WPL or EWP or EWPL)
Write	Address
Write	Memory [address]
Write	Memory [address+4]
Write	Memory [address+8]
Write	Memory [address+(Number of Words to Write - 1)* 4]

Flash **Write Page and Lock** command **(WPL)** is equivalent to the Flash Write Command. However, the lock bit is automatically set at the end of the Flash write operation. As a lock region is composed of several pages, the programmer writes to the first pages of the lock region using Flash write commands and writes to the last page of the lock region using a Flash write and lock command.

Flash **Erase Page and Write** command **(EWP)** is equivalent to the Flash Write Command. However, before programming the load buffer, the page is erased.

Flash **Erase Page and Write the Lock** command **(EWPL)** combines EWP and WPL commands.

20.3.4.3 Flash Full Erase Command

This command is used to erase the Flash memory planes.

All lock bits must be deactivated before using the **Full Erase** command. This can be done by using the CLB command.

Table 20-21. Full Erase Command

Read/Write	DR Data
Write	EA

20.3.4.4 Flash Lock Commands

Lock bits can be set using WPL or EWPL commands. They can also be set by using the **Set Lock** command **(SLB)**. With this command, several lock bits can be activated at the same time. Bit 0 of Bit Mask corresponds to the first lock bit and so on.

In the same way, the **Clear Lock** command **(CLB)** is used to clear lock bits. All the lock bits can also be cleared by the EA command.

Table 20-22. Set and Clear Lock Bit Command

Read/Write	DR Data
Write	SLB or CLB
Write	Bit Mask

Lock bits can be read using **Get Lock Bit** command **(GLB)**. When a bit set in the Bit Mask is returned, then the corresponding lock bit is active.

Table 20-23. Get Lock Bit Command

Read/Write	DR Data
Write	GLB
Read	Bit Mask

20.3.4.5 Flash General-purpose NVM Commands

General-purpose NVM bits (GP NVM) can be set with the **Set Fuse** command **(SFB)**. Using this command, several GP NVM bits can be activated at the same time. Bit 0 of Bit Mask corresponds to the first fuse bit and so on.

In the same way, the **Clear Fuse** command **(CFB)** is used to clear GP NVM bits. All the general-purpose NVM bits are also cleared by the EA command.

Table 20-24. Set and Clear General-purpose NVM Bit Command

Read/Write	DR Data
Write	SFB or CFB
Write	Bit Mask

GP NVM bits can be read using **Get Fuse Bit** command **(GFB)**. When a bit set in the Bit Mask is returned, then the corresponding fuse bit is set.

Table 20-25. Get General-purpose NVM Bit Command

Read/Write	DR Data
Write	GFB
Read	Bit Mask

20.3.4.6 *Flash Security Bit Command*

Security bits can be set using **Set Security Bit** command (SSE). Once the security bit is active, the Fast Flash programming is disabled. No other command can be run. Only an event on the Erase pin can erase the security bit once the contents of the Flash have been erased.

The SAM7SE512 security bit is controlled by the EFC0. To use the **Set Security Bit** command, the EFC0 must be selected using the **Select EFC** command.

Table 20-26. Set Security Bit Command

Read/Write	DR Data
Write	SSE

20.3.4.7 SAM7SE512 *Select EFC Command*

The commands WPx, EA, xLB, xFB are executed using the current EFC controller. The default EFC controller is EFC0. The **Select EFC** command (SEFC) allows selection of the current EFC controller.

Table 20-27. Select EFC Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SEFC
2	Write handshaking	DATA	0 = Select EFC0 1 = Select EFC1

20.3.4.8 *Memory Write Command*

This command is used to perform a write access to any memory location.

The **Memory Write** command **(WRAM)** is optimized for consecutive writes. An internal address buffer is automatically increased.

Table 20-28. Write Command

Read/Write	DR Data
Write	(Number of Words to Write) << 16 | (WRAM)
Write	Address
Write	Memory [address]
Write	Memory [address+4]
Write	Memory [address+8]
Write	Memory [address+(Number of Words to Write - 1)* 4]

20.3.4.9 Get Version Command

The **Get Version** (GVE) command retrieves the version of the FFPI interface.

Table 20-29. Get Version Command

Read/Write	DR Data
Write	GVE
Read	Version

21. External Bus Interface (EBI)

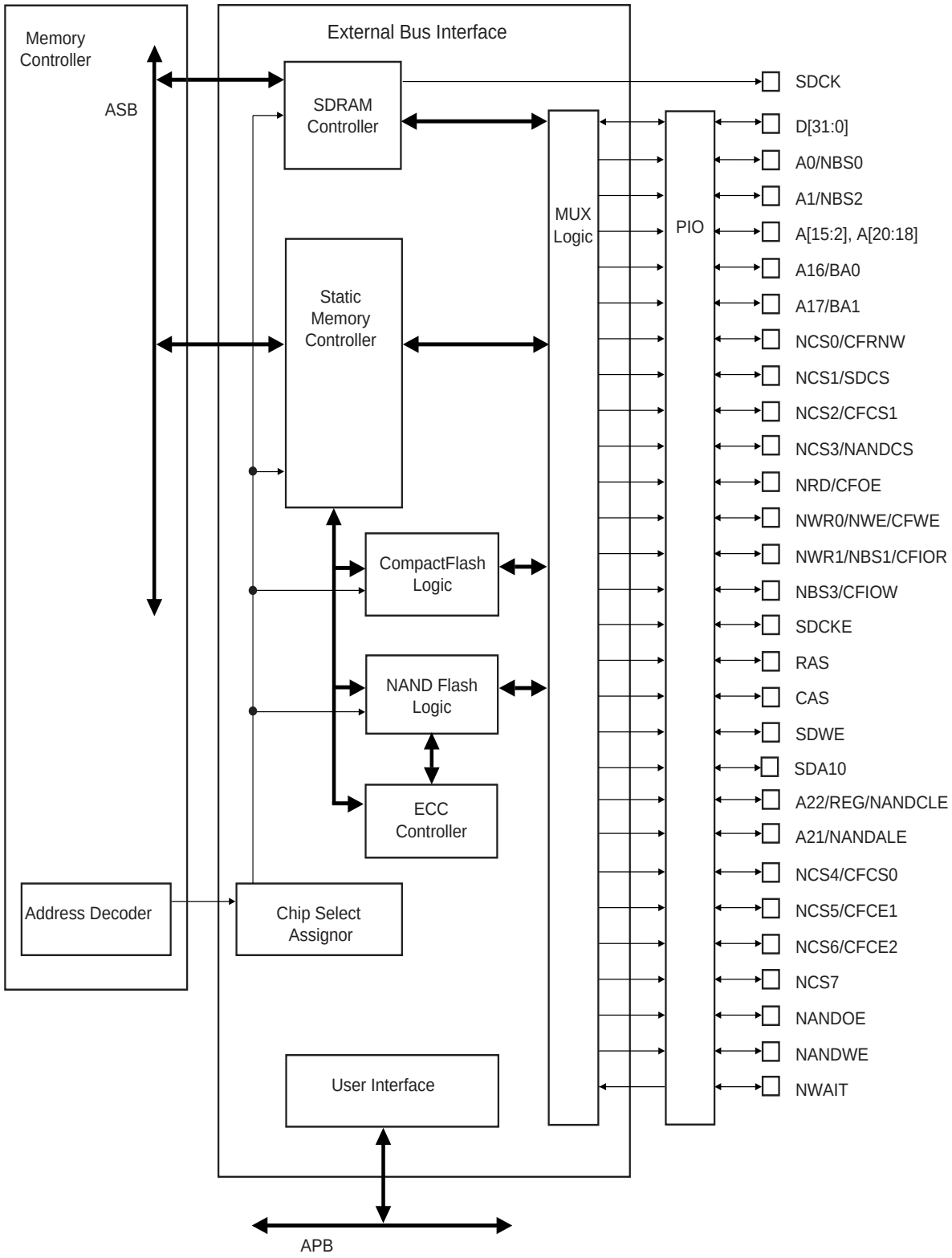
21.1 Overview

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

The EBI also supports the CompactFlash® and the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to eight external devices, each assigned to eight address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 23 bits, up to eight chip select lines (NCS[7:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

21.2 Block Diagram

Figure 21-1. Organization of the External Bus Interface



21.3 I/O Lines Description

Table 21-1. I/O Lines Description

Name	Function	Type	Active Level
EBI			
D[31:0]	Data Bus	I/O	
A[22:0]	Address Bus	Output	
NWAIT	External Wait Signal	Input	Low
SMC			
NCS[7:0]	Chip Select Lines	Output	Low
NWR[1:0]	Write Signals	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NUB	NUB: Upper Byte Select	Output	Low
NLB	NLB: Lower Byte Select	Output	Low
EBI for CompactFlash Support			
CFCE[2:1]	CompactFlash Chip Enable	Output	Low
CFOE	CompactFlash Output Enable	Output	Low
CFWE	CompactFlash Write Enable	Output	Low
CFIOR	CompactFlash I/O Read Signal	Output	Low
CFIOW	CompactFlash I/O Write Signal	Output	Low
CFRNW	CompactFlash Read Not Write Signal	Output	
CFCS[1:0]	CompactFlash Chip Select Lines	Output	Low
EBI for NAND Flash Support			
NANDCS	NAND Flash Chip Select Line	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
NANDCLE	NAND Flash Command Line Enable	Output	Low
NANDALE	NAND Flash Address Line Enable	Output	Low
SDRAM Controller			
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select Line	Output	Low
BA[1:0]	Bank Select	Output	
SDWE	SDRAM Write Enable	Output	Low
RAS - CAS	Row and Column Signal	Output	Low
NBS[3:0]	Byte Mask Signals	Output	Low
SDA10	SDRAM Address 10 Line	Output	

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

Table 21-2 details the connections between the two Memory Controllers and the EBI pins.

Table 21-2. EBI Pins and Memory Controllers I/O Lines Connections

EBI Pins	SDRAMC I/O Lines	SMC I/O Lines
NWR1/NBS1/CFIOR	NBS1	NWR1/NUB
A0/NBS0	Not Supported	A0/NLB
A1/NBS2	Not Supported	A1
A[11:2]	A[9:0]	A[11:2]
SDA10	A10	Not Supported
A12	Not Supported	A12
A[14:13]	A[12:11]	A[14:13]
A[22:15]	Not Supported	A[22:15]
D[31:16]	D[31:16]	Not Supported
D[15:0]	D[15:0]	D[15:0]

21.4 Application Example

21.4.1 Hardware Interface

Table 21-3 details the connections to be applied between the EBI pins and the external devices for each Memory Controller

Table 21-3. EBI Pins and External Static Device Connections

	Pins of the Interfaced Device						
Pin	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	SDRAM[1]	CompactFlash	CompactFlash True IDE Mode	NAND Flash[2]
Controller	**SMC**			**SDRAMC**	**SMC**		
D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	I/O0 - I/O7
D8 - D15	–	D8 - D15	D8 - D15	D8 - D15	D8 - 15	D8 - 15	I/O8 - I/O15[3]
D16 - D31	–	–	–	D16 - D31	–	–	–
A0/NBS0	A0	–	NLB	DQM0	A0	A0	–
A1/NBS2	A1	A0	A0	DQM2	A1	A1	–
A2 - A9	A2 - A9	A1 - A8	A1 - A8	A0 - A7	A2 - A9	A2 - A9	–
A10	A10	A9	A9	A8	A10	A10	–
A11	A11	A10	A10	A9	–	–	–
SDA10	–	–	–	A10	–	–	–
A12	A12	A11	A11	–	–	–	–
A13 - A14	A13 - A14	A12 - A13	A12 - A13	A11 - A12	–	–	–
A15	A15	A14	A14	–	–	–	–
A16/BA0	A16	A15	A15	BA0	–	–	–
A17/BA1	A17	A16	A16	BA1	–	–	–

Table 21-3. EBI Pins and External Static Device Connections (Continued)

Pin	Pins of the Interfaced Device						
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	SDRAM[1]	CompactFlash	CompactFlash True IDE Mode	NAND Flash[2]
Controller	**SMC**			**SDRAMC**	**SMC**		
A18 - A20	A18 - A20	A17 - A19	A17 - A19	–	–	–	–
A21/NANDALE	A21	A20	A20	–	–	–	ALE
A22/REG/NANDCLE	A22	A21	A21	–	REG	–	CLE
NCS0	CS	CS	CS	–	CFRNW[4]	CFRNW[4]	–
NCS1/SDCS	CS	CS	CS	CS	–	–	–
NCS2/CFCS1	CS	CS	CS	–	CFCS1[4]	CFCS1[4]	–
NCS3/NANDCS	CS	CS	CS	–	–	–	CE[7]
NCS4/CFCS0	CS	CS	CS	–	CFCS0[4]	CFCS0[4]	–
NCS5/CFCE1	CS	CS	CS	–	CE1	CS0	–
NCS6/CFCE2	CS	CS	CS	–	CE2	CS1	–
NCS7	CS	CS	CS	–	–	–	–
NANDOE	–	–	–	–	–	–	RE
NANDWE	–	–	–	–	–	–	WE
NRD/CFOE	OE	OE	OE	–	OE	–	
NWR0/NWE/CFWE	WE	WE[5]	WE	–	WE	–	[8]
NWR1/NBS1/CFIOR	WE	WE[5]	NUB	DQM1	IOR	IOR	–
NBS3/CFIOW	–	–	–	DQM3	IOW	IOW	–
SDCK	–	–	–	CLK	–	–	–
SDCKE	–	–	–	CKE	–	–	–
RAS	–	–	–	RAS	–	–	–
CAS	–	–	–	CAS	–	–	–
SDWE	–	–	–	WE	–	–	–
NWAIT	–	–	–	–	WAIT	WAIT	–
Pxx[6]	–	–	–	–	CD1 or CD2	CD1 or CD2	–
Pxx[6]	–	–	–	–	–	–	CE[7]
Pxx[6]	–	–	–	–	–	–	RDY

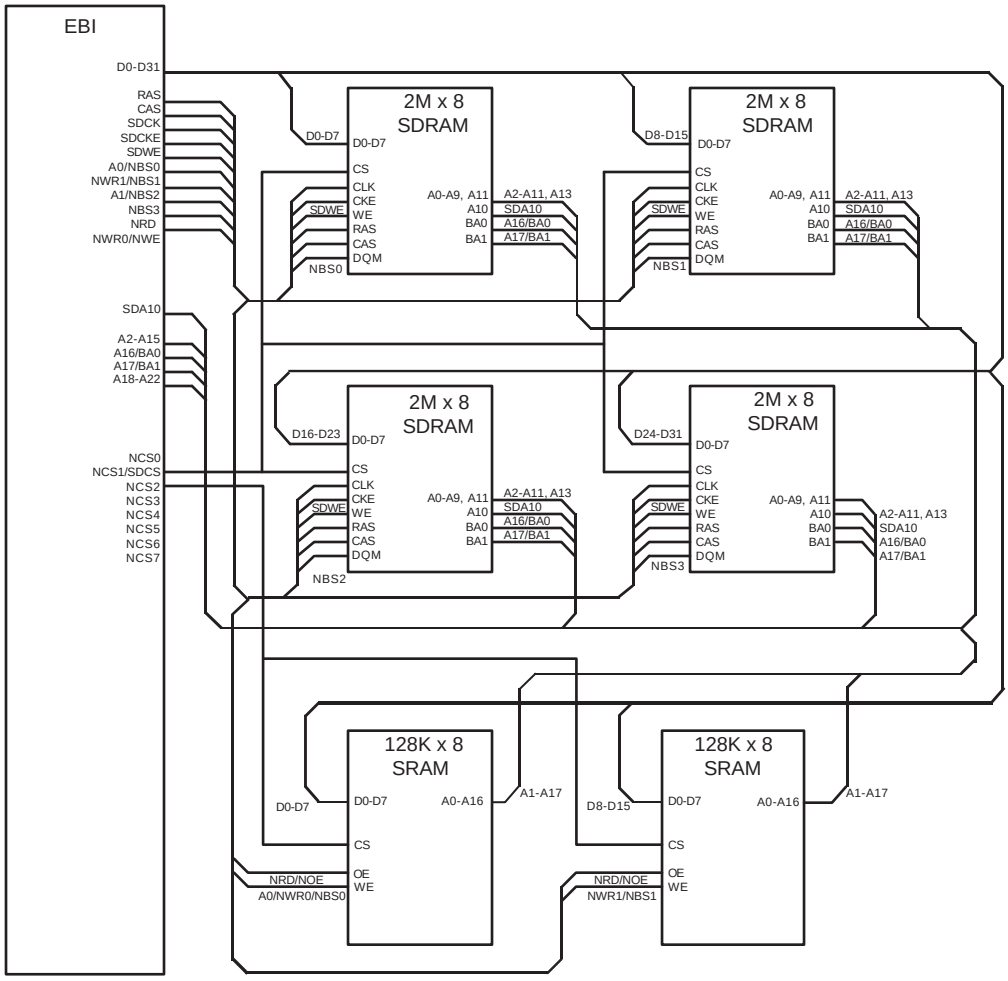
Notes:
1. For SDRAM connection examples, refer to "Using SDRAM on AT91SAM7SE Microcontrollers", application note.
2. For NAND Flash connection examples, refer to "NAND Flash Support on AT91SAM7SE Microcontrollers", application note.
3. I/O8 - I/O15 bits used only for 16-bit NAND Flash.
4. Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.
5. NWR1 enables upper byte writes. NWR0 enables lower byte writes.
6. Any free PIO line.
7. CE connection depends on the Nand Flash.
   For standard Nand Flash devices, it must be connected to any free PIO line.
   For "CE don't care" 8-bit Nand Flash devices, it can be either connected to NCS3/NANDCS or to any free PIO line.
   For "CE don't care" 16-bit Nand Flash devices, it must be connected to any free PIO line.

8. When the NAND Flash Logic is used, NWR0/NWE/CFWE must be kept as PIO Input Mode with Pull-up enabled (default state after reset) or as PIO Output set at logic level 1. The PIO cannot be used in PIO Mode.

21.4.2 Connection Examples

Figure 21-2 shows an example of connections between the EBI and external devices.

Figure 21-2. EBI Connections to Memory Devices



21.5 Product Dependencies

21.5.1 I/O Lines

The pins used for interfacing the External Bus Interface may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the External Bus Interface pins to their peripheral function. If I/O lines of the External Bus Interface are not used by the application, they can be used for other purposes by the PIO Controller.

21.6 Functional Description

The EBI transfers data between the internal ASB Bus (handled by the Memory Controller) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control busses and is composed of the following elements:

- The Static Memory Controller (SMC)
- The SDRAM Controller (SDRAMC)
- The ECC Controller (ECC)
- A chip select assignment feature that assigns an ASB address space to the external devices
- A multiplex controller circuit that shares the pins between the different Memory Controllers
- Programmable CompactFlash support logic
- Programmable NAND Flash support logic

21.6.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 32-bit data lines, the address lines of up to 23 bits and the control signals through a multiplex logic operating in function of the memory area requests.

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

21.6.2 Static Memory Controller

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

21.6.3 SDRAM Controller

For information on the SDRAM Controller, refer to the SDRAMC Section.

21.6.4 ECC Controller

For information on the ECC Controller, refer to the ECCC Section.

21.6.5 CompactFlash Support

The External Bus Interface integrates circuitry that interfaces to CompactFlash devices.

The CompactFlash logic is driven by the Static Memory Controller (SMC) on the NCS4 and/or NCS2 address space. Programming the CS4A and/or CS2A bit of the Chip Select Assignment Register (See “EBI Chip Select Assignment Register” on page 158.) to the appropriate value enables this logic. Access to an external CompactFlash device is then made by accessing the

address space reserved to NCS4 and/or NCS2 (i.e., between 0x5000 0000 and 0x5FFF FFFF for NCS4 and between 0x3000 0000 and 0x3FFF FFFF for NCS2).

When multiplexed with CFCE1 and CFCE2 signals, the NCS5 and NCS6 signals become unavailable. Performing an access within the address space reserved to NCS5 and NCS6 (i.e., between 0x6000 0000 and 0x7FFF FFFF) may lead to an unpredictable outcome.

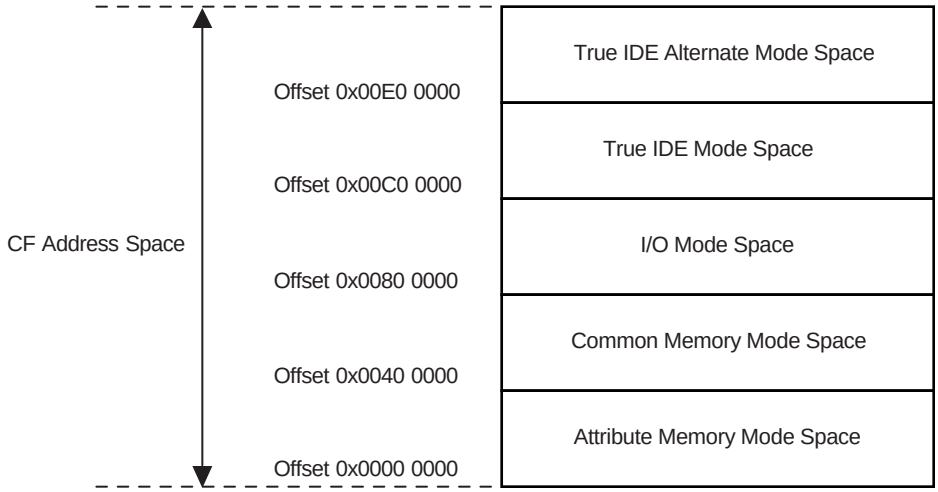
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

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

Within the NCS4 and/or NCS2 address space, the current transfer address is used to distinguish I/O mode, common memory mode, attribute memory mode and True IDE mode.

The different modes are accessed through a specific memory mapping as illustrated in Figure 21-3.

Figure 21-3. CompactFlash Memory Mapping



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

21.6.5.2 CFCE1 and CFCE2 signals

To cover all types of access, the SMC must be alternatively set to drive the 8-bit data bus or 16-bit data bus. The odd byte access on the D[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the corresponding NCS pin (NCS4 and/or NCS2). The DBW field in the corresponding Chip Select Register of the NCS4 and/or NCS2 address space must be set as shown in Table 21-4 to enable the required access type.

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

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

Table 21-4. CFCE1 and CFCE2 Truth Table

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

21.6.5.3 Read/Write Signals

In I/O mode and True IDE mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and CFIOW signals, while the CFOE and CFWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFOE and CFWE signals, while the CFIOR and CFIOW are deactivated. Figure 21-4 on page 144 shows a schematic representation of this logic and Table 21-5 on page 144 presents the signal decoding.

Attribute memory mode, common memory mode and I/O mode are supported by setting the address setup and hold time on the NCS4 (and/or NCS2) chip select to the appropriate values. For details on these signal waveforms, please refer to the section: Setup and Hold Cycles of the Static Memory Controller Section.

Figure 21-4. CompactFlash Read/Write Control Signals

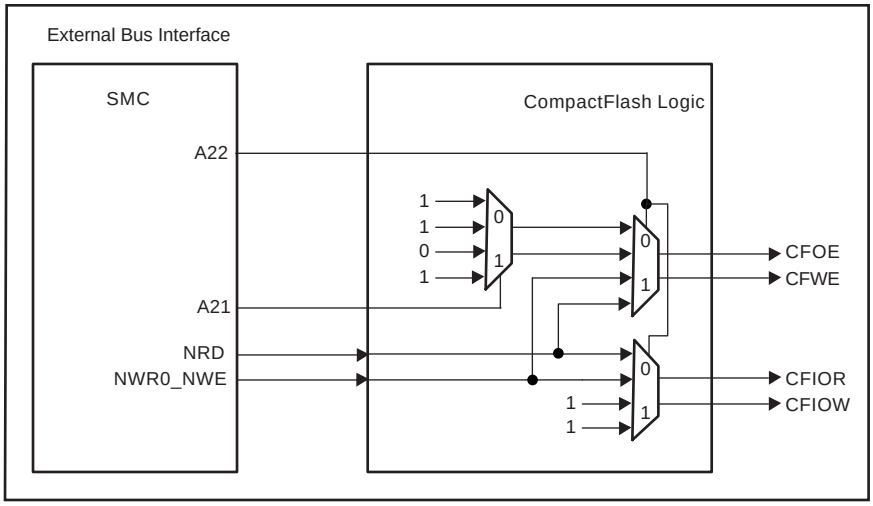


Table 21-5. CompactFlash Mode Selection

Mode Base Address	CFOE	CFWE	CFIOR	CFIOW
Attribute Memory Common Memory	NRD	NWR0_NWE	1	1
I/O Mode	1	1	NRD	NWR0_NWE
True IDE Mode	0	1	NRD	NWR0_NWE

21.6.5.4 Multiplexing of CompactFlash Signals on EBI Pins

Table 21-6 and Table 21-7 on page 145 describe the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 21-6 are strictly dedicated to the CompactFlash interface as soon as the CS4A and/or CS2A field of the Chip Select Assignment Register is set (See “EBI Chip Select Assignment Register” on page 158.). These pins must not be used to drive any other memory devices.

The EBI pins in Table 21-7 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (CS4A = 1 and/or CS2A = 1).

Table 21-6. Dedicated CompactFlash Interface Multiplexing

	CompactFlash Signals		EBI Signals	
Pins	**CS4A = 1**	**CS2A = 1**	**CS4A = 0**	**CS2A = 0**
NCS4/CFCS0	CFCS0		NCS4	
NCS2/CFCS1		CFCS1		NCS2

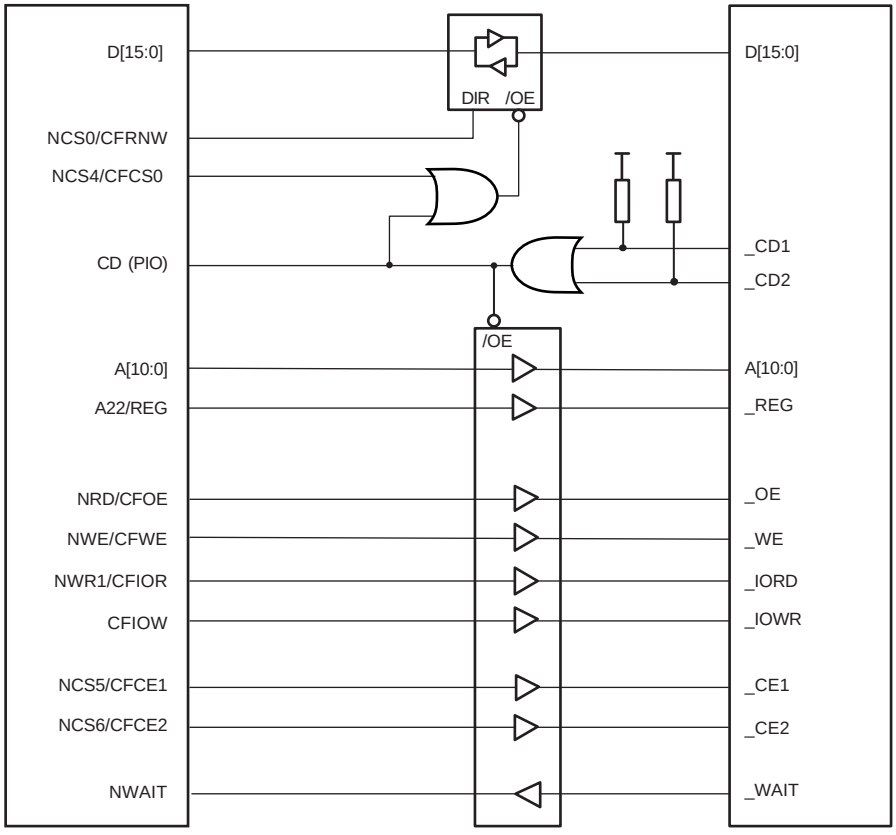
Table 21-7. Shared CompactFlash Interface Multiplexing

Pins	Access to CompactFlash Device CompactFlash Signals	Access to Other EBI Devices EBI Signals
NRD/CFOE	CFOE	NRD
NWR0/NWE/CFWE	CFWE	NWR0/NWE
NWR1/NBS1/CFIOR	CFIOR	NWR1/NBS1
NBS3/CFIOW	CFIOW	NBS3
NCS0/CFRNW	CFRNW	NCS0

21.6.5.5 Application Example

Figure 21-5 on page 145 illustrates an example of a CompactFlash application. CFCS0 and CFRNW signals are not directly connected to the CompactFlash slot 0, but do control the direction and the output enable of the buffers between the EBI and the CompactFlash Device. The timing of the CFCS0 signal is identical to the NCS4 signal. Moreover, the CFRNW signal remains valid throughout the transfer, as does the address bus. The CompactFlash _WAIT signal is connected to the NWAIT input of the Static Memory Controller. For details on these waveforms and timings, refer to the Static Memory Controller Section.

Figure 21-5. CompactFlash Application Example



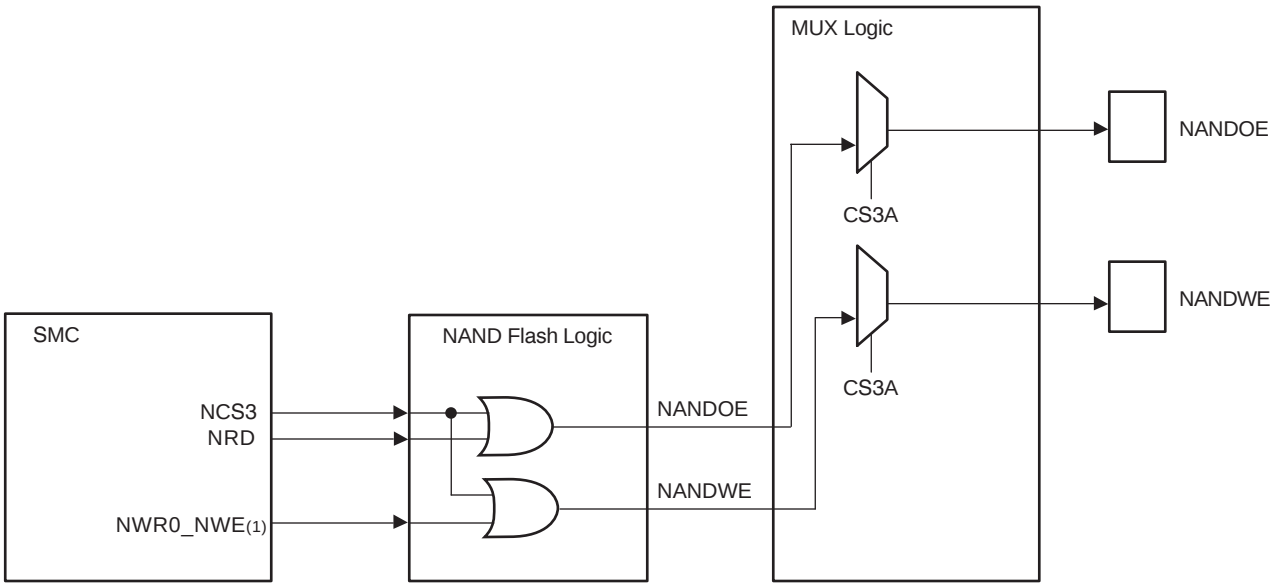
21.6.6 NAND Flash Support

The EBI integrates circuitry that interfaces to NAND Flash devices.

The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the CS3A field in the Chip Select Assignment Register to the appropriate value enables the NAND Flash logic (See “EBI Chip Select Assignment Register” on page 158.). Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x4000 0000 and 0x4FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. For details on these waveforms, refer to the Static Memory Controller Section.

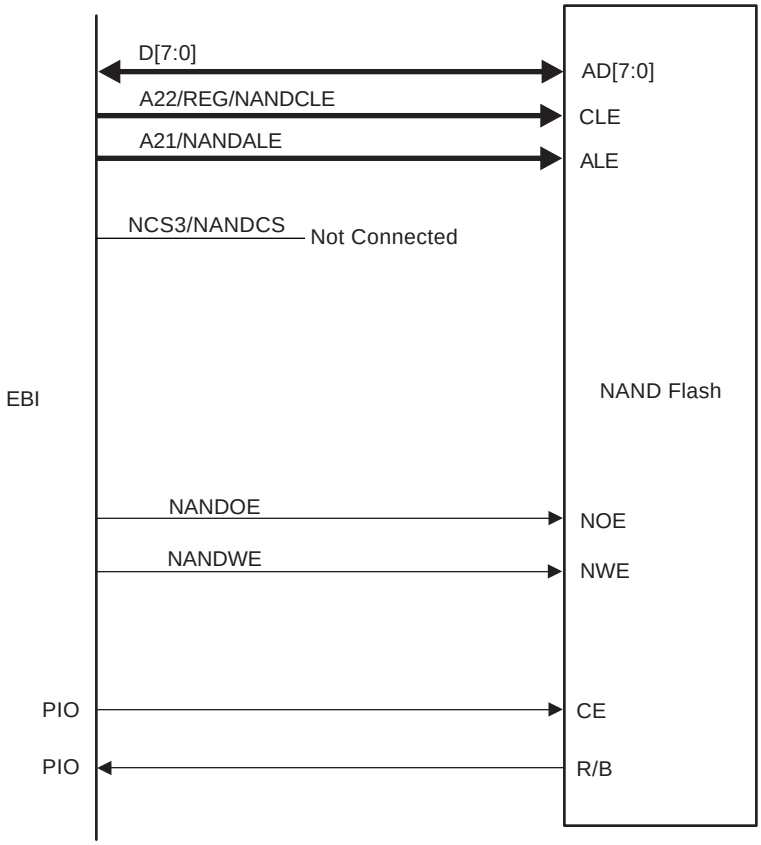
Figure 21-6. NAND Flash Signal Multiplexing on EBI Pins



(1) When the NAND Flash Logic is used, NWR0/NWE/CFWE must be kept as PIO Input Mode with Pull-up enabled (default state after reset) or as PIO Output set at logic level 1. The PIO cannot be used in PIO Mode.

The address latch enable and command latch enable signals on the NAND Flash device are driven respectively by address bits A21 and A22 of the EBI address bus. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCS3 address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCS3 is not selected, preventing the device from returning to standby mode.

Figure 21-7. NAND Flash Application Example

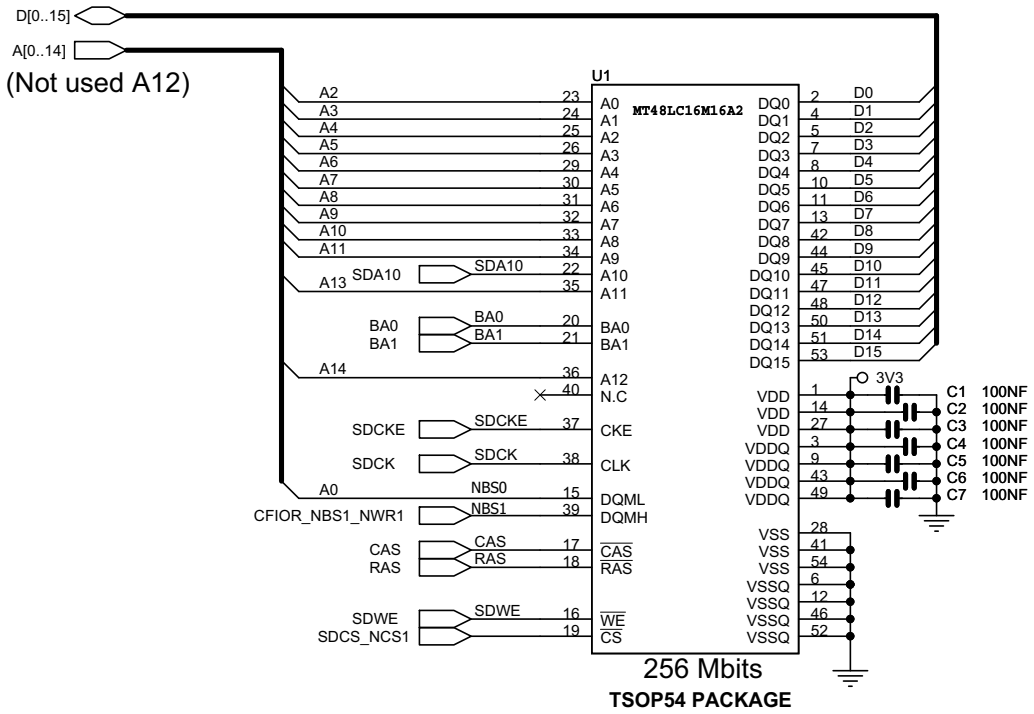


Note: The External Bus Interface is also able to support 16-bit devices.

21.7 Implementation Examples

21.7.1 16-bit SDRAM

21.7.1.1 Hardware Configuration



21.7.1.2 Software Configuration

The following configuration must be respected:

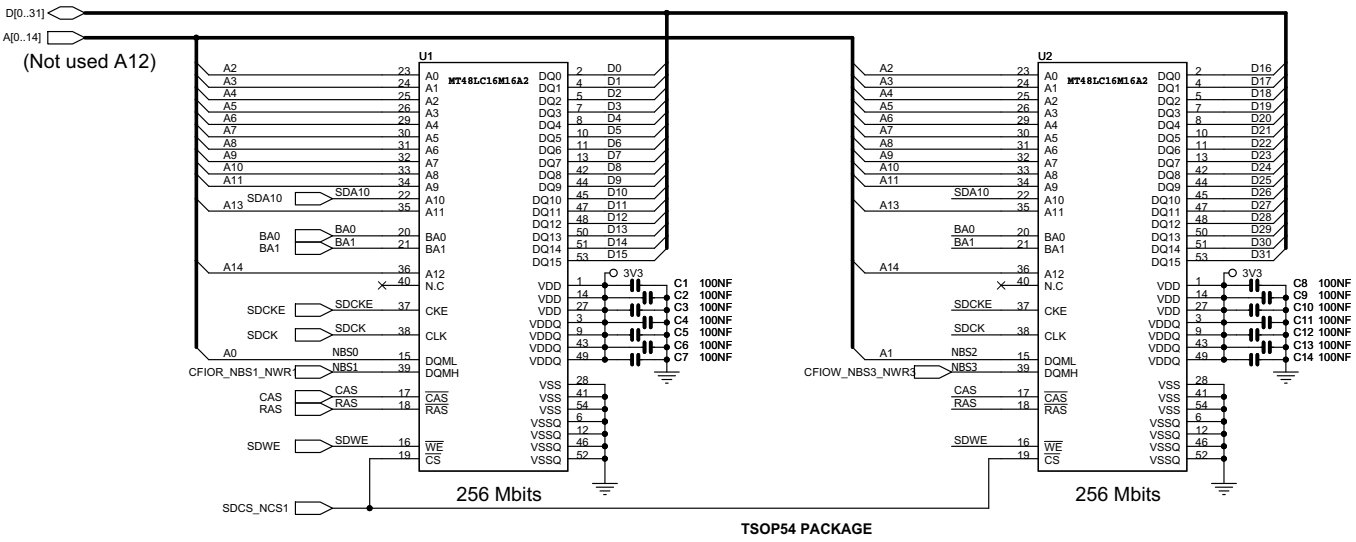
- Address lines A[0..11], A[13-14], BA0, BA1, SDA10, SDCS_NCS1, SDWE, SDCKE, NBS1, RAS, CAS, and data lines D[8..15] are multiplexed with PIO lines and thus dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The data bus width is to be programmed to 16 bits.

The SDRAM initialization sequence is described in the “SDRAM Device Initialization” section of the SDRAM Controller.

21.7.2 32-bit SDRAM

21.7.2.1 Hardware Configuration



21.7.2.2 Software Configuration

The following configuration must be respected:

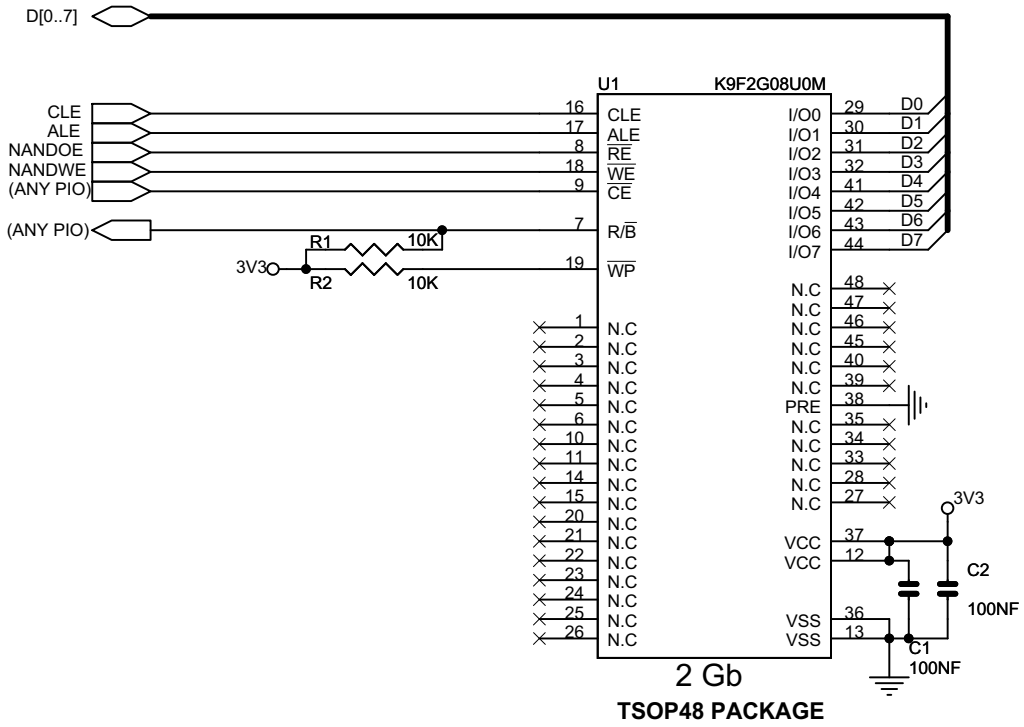
- Address lines A[0..11], A[13-14], BA0, BA1, SDA10, SDCS_NCS1, SDWE, SDCKE, NBS1, RAS, CAS, and data lines D[8..31] are multiplexed with PIO lines and thus dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The data bus width is to be programmed to 32 bits.

The SDRAM initialization sequence is described in the “SDRAM Device Initialization” section of the SDRAM Controller.

21.7.3 8-bit NAND Flash

21.7.3.1 Hardware Configuration



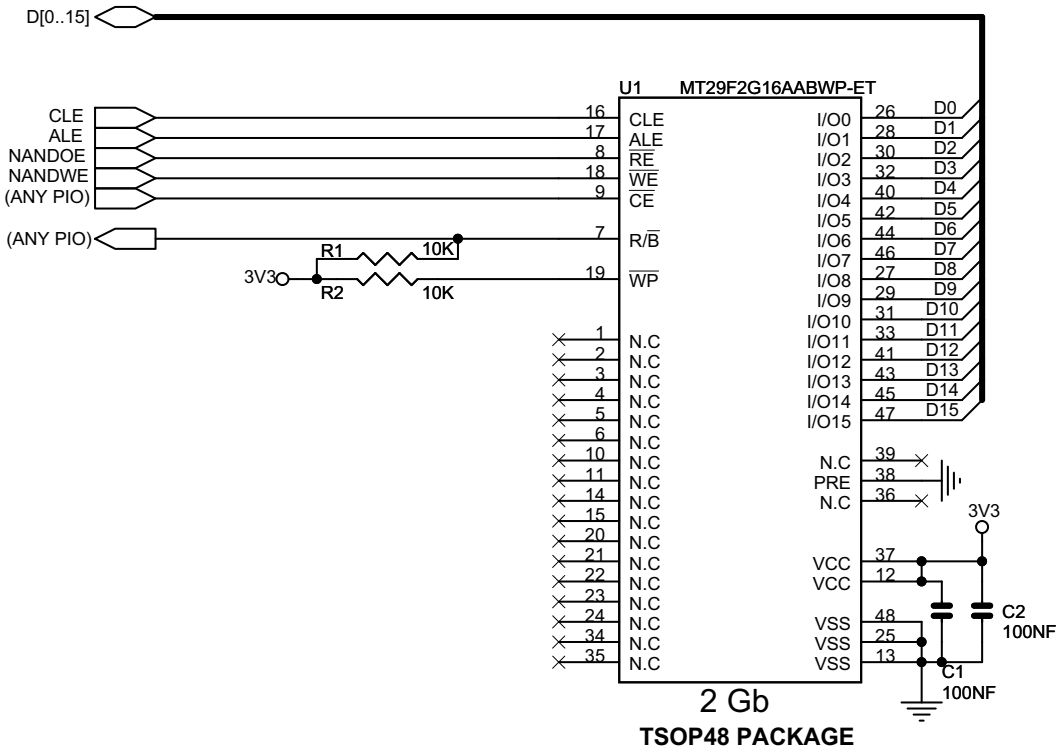
21.7.3.2 Software Configuration

The following configuration must be respected:

- CLE, ALE, NANDOE and NANDWE signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register.
- Reserve A21/A22 for ALE/CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bit A21 and A22 during accesses.
- Configure a PIO line as an input to manage the Ready/Busy signal.
- Configure Static Memory Controller CS3 Setup, Pulse, Cycle and Mode according to NAND Flash timings, the data bus width and the system bus frequency.

21.7.4 16-bit NAND Flash

21.7.4.1 Hardware Configuration

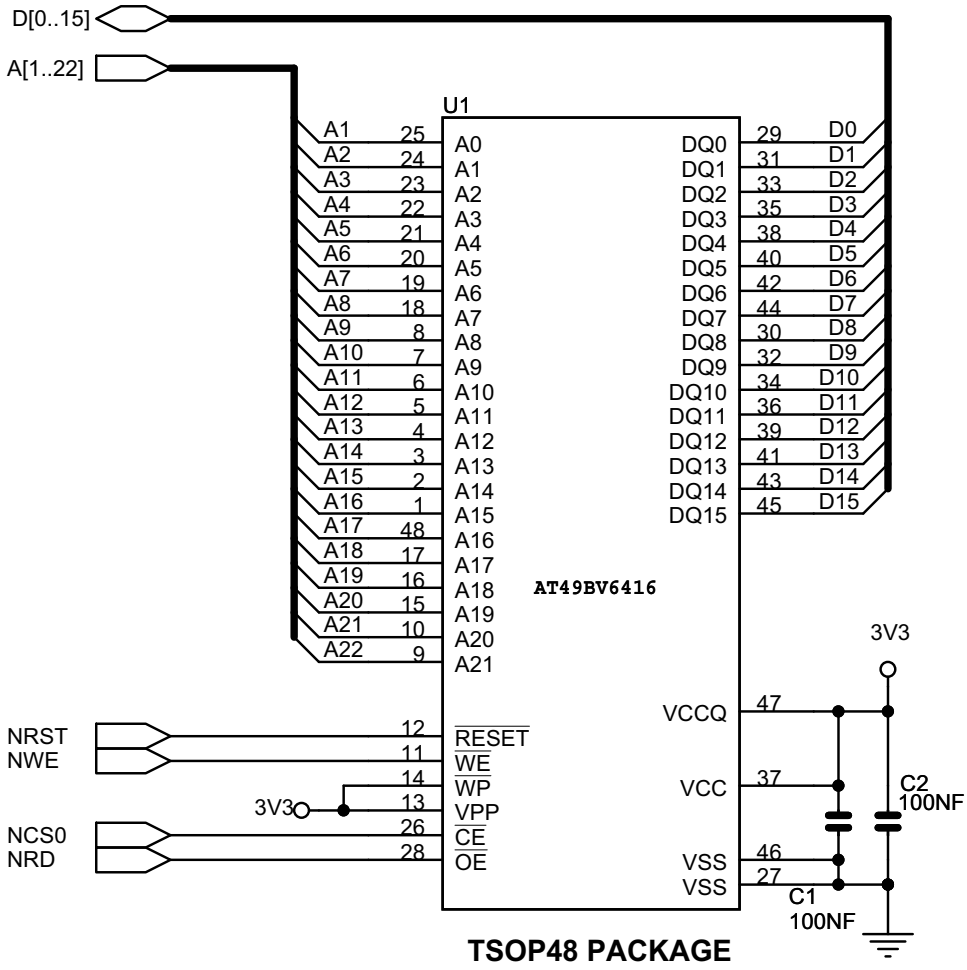


21.7.4.2 Software Configuration

The software configuration is the same as for an 8-bit NAND Flash except the data bus width programmed in the mode register of the Static Memory Controller and the selection of D[8..15] in the PIO controller.

21.7.5 NOR Flash on NCS0

21.7.5.1 Hardware Configuration



21.7.5.2 Software Configuration

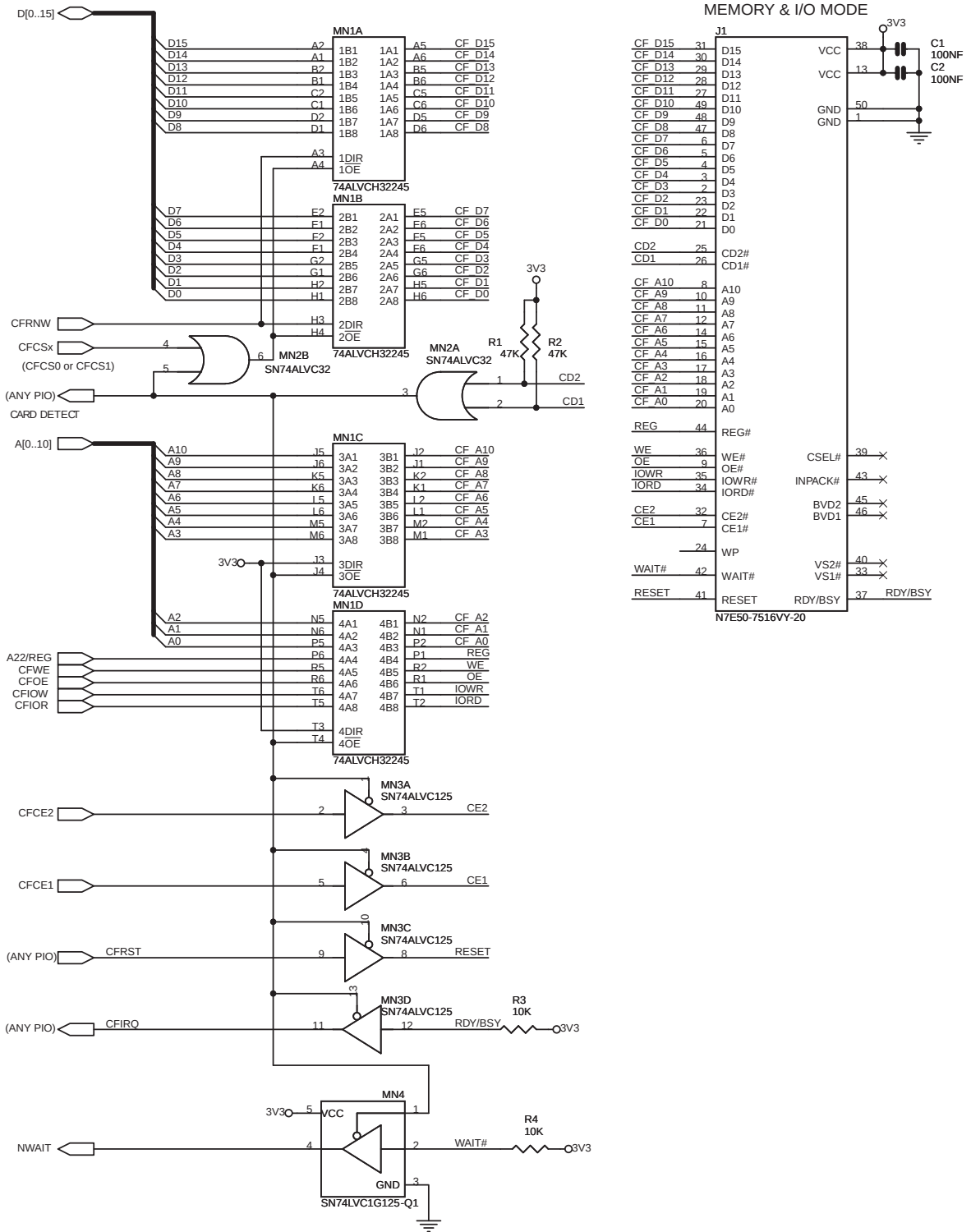
- Address lines A[1..22], NCS0, NRD, NWE and data lines D[8..15] are multiplexed with PIO lines and thus dedicated PIOs must be programmed in peripheral mode in the PIO controller.

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

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

21.7.6 CompactFlash

21.7.6.1 Hardware Configuration



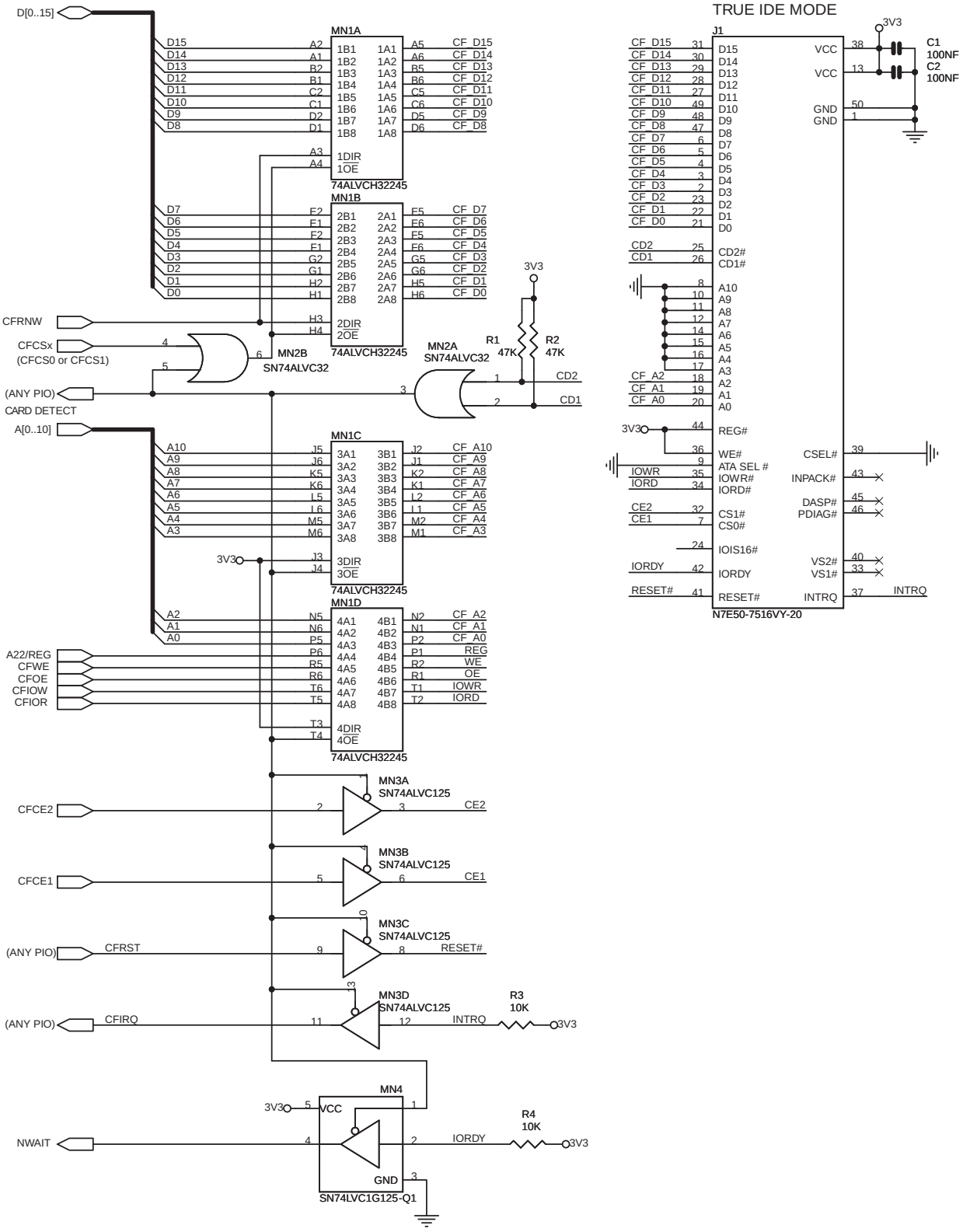
21.7.6.2 Software Configuration

The following configuration must be respected:

- Assign the EBI CS4 and/or EBI_CS5 to the CompactFlash Slot 0 and/or Slot 1 by setting the bit EBI_CS4A and/or EBI_CS5A in the EBI Chip Select Assignment Register.
- Select the mode by using the corresponding address (refer to Figure 21.3).
- Address lines A[0..10], A22, CFWE, CFOE, CFIOW, CFIOR, NWAIT, CFRNW, CFS0, CFCS1, CFCE1, CFCE2 and data lines D[8..15] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO Controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to CompactFlash timings and system bus frequency.

21.7.7 CompactFlash True IDE

21.7.7.1 Hardware Configuration



21.7.7.2 Software Configuration

The following configuration must be respected:

- Address lines A[0..10], A22, CFWE, CFOE, CFIOW, CFIOR, NWAIT, CFRNW, CFS0, CFCS1, CFCE1, CFCE2 and data lines D[8..15] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Assign the EBI CS4 and/or EBI CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI_CS4A and/or EBI_CS5A in the EBI Chip Select Assignment Register.
- Select the mode by using the corresponding address (refer to Figure 21-3).
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode according to CompactFlash timings and system bus frequency.

21.8 External Bus Interface (EBI) User Interface

EBI User Interface Base Address: 0xFFFF FF80

Table 21-8. External Bus Interface Memory Map

Offset	Register	Name	Access	Reset State
0x00	Chip Select Assignment Register	EBI_CSA	Read/Write	0x0
0x04	Reserved	–		
0x08	Reserved	–		
0x0C	Reserved	–		
0x10 - 0x2C	SMC User Interface	Refer to the Static Memory Controller User Interface		
0x30 - 0x58	SDRAMC User Interface	Refer to the SDRAM Controller User Interface		
0x5C - 0x6C	ECC User Interface	Refer to the Error Code Corrected Controller User interface		
0x70 - 0x7C	Reserved	–		

21.8.1 EBI Chip Select Assignment Register

Name: EBI_CSA

Access: Read/Write

Reset Value: 0x0

Offset: 0x0

Absolute Address: 0xFFFF FF80

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	CS4A	CS3A	CS2A	CS1A	–

- **CS1A: Chip Select 1 Assignment**

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

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

- **CS2A: Chip Select 2 Assignment**

0 = Chip Select 2 is assigned to the Static Memory Controller and NCS2, NCS5 and NCS6 behave as defined by the SMC.

1 = Chip Select 2 is assigned to the Static Memory Controller and the CompactFlash Logic (second slot) is activated.

Accessing the address space reserved to NCS5 and NCS6 may lead to an unpredictable outcome.

- **CS3A: Chip Select 3 Assignment**

0 = Chip Select 3 is only assigned to the Static Memory Controller and NCS3 behave as defined by the SMC.

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

- **CS4A: Chip Select 4 Assignment**

0 = Chip Select 4 is assigned to the Static Memory Controller and NCS4, NCS5 and NCS6 behave as defined by the SMC.

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

Accessing the address space reserved to NCS5 and NCS6 may lead to an unpredictable outcome.

- **NWPC: NWAIT Pin Configuration**

0 = The NWAIT device pin is not connected to the External Wait Request input of the Static Memory Controller, this multiplexed pin can be used as a PIO.

1 = The NWAIT device pin is connected to the External Wait Request input of the Static Memory Controller.

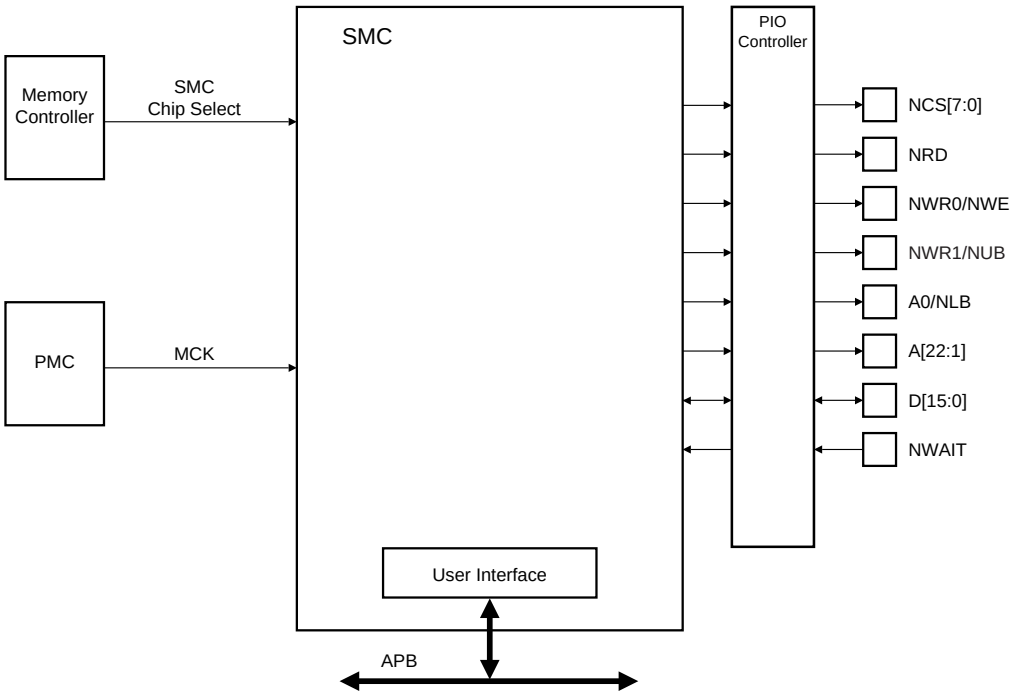
22. Static Memory Controller (SMC)

22.1 Overview

The Static Memory Controller (SMC) generates the signals that control the access to external static memory or peripheral devices. The SMC is fully programmable. It has eight chip selects and a 23-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. The SMC supports different access protocols allowing single clock cycle memory accesses. It also provides an external wait request capability.

22.2 Block Diagram

Figure 22-1. Static Memory Controller Block Diagram



22.3 I/O Lines Description

Table 22-1. I/O Lines Description

Name	Description	Type	Active Level
NCS[7:0]	Static Memory Controller Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
NWR1/NUB	Write 1/Upper Byte Select Signal	Output	Low
A0/NLB	Address Bit 0/Lower Byte Select Signal	Output	Low
A[22:1]	Address Bus	Output	
D[15:0]	Data Bus	I/O	
NWAIT	External Wait Signal	Input	Low

22.4 Multiplexed Signals

Table 22-2. Static Memory Controller Multiplexed Signals

Multiplexed Signals		Related Function
A0	NLB	8-bit or 16-bit data bus, see 22.6.1.3 “Data Bus Width” on page 164.
NWR0	NWE	Byte-write or byte-select access, see 22.6.2.1 “Write Access Type” on page 165.
NWR1	NUB	Byte-write or byte-select access, see 22.6.2.1 “Write Access Type” on page 165.

22.5 Product Dependencies

22.5.1 I/O Lines

The pins used for interfacing the Static Memory Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the Static Memory Controller pins to their peripheral function. If I/O lines of the Static Memory Controller are not used by the application, they can be used for other purposes by the PIO Controller.

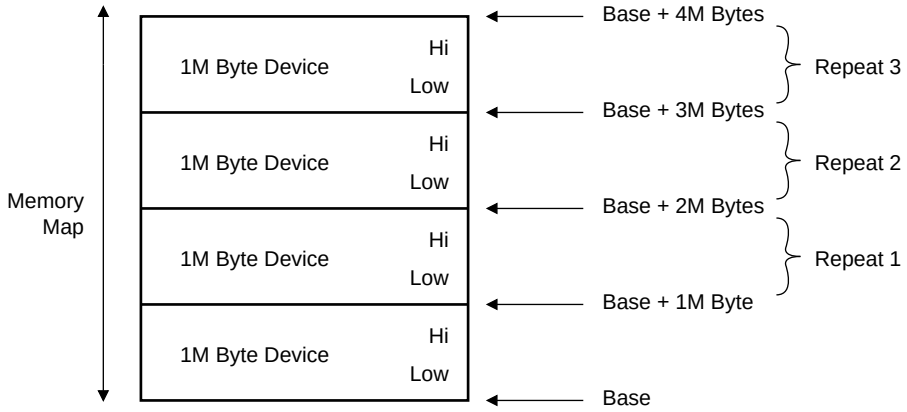
22.6 Functional Description

22.6.1 External Memory Interface

22.6.1.1 External Memory Mapping

The memory map is defined by hardware and associates the internal 32-bit address space with the external 23-bit address bus. Note that A[22:0] is only significant for 8-bit memory. A[22:1] is used for 16-bit memory. If the physical memory device is smaller than the page size, it wraps around and appears to be repeated within the page. The SMC correctly handles any valid access to the memory device within the page. See Figure 22-2.

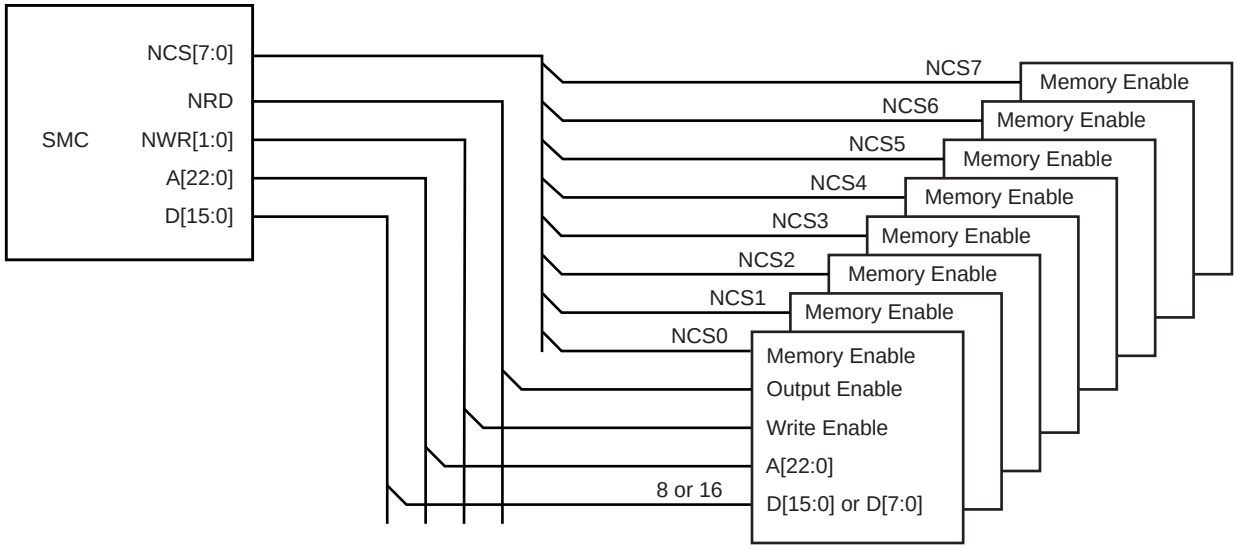
Figure 22-2. Case of an External Memory Smaller than Page Size



22.6.1.2 Chip Select Lines

The Static Memory Controller provides up to eight chip select lines: NCS0 to NCS7.

Figure 22-3. Memory Connections for Eight External Devices [1]



Note: 1. The maximum address space per device is 8 Mbytes.

22.6.1.3 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 DBW field in the SMC_CSR for the corresponding chip select. See "SMC Chip Select Registers" on page 196.

Figure 22-4 shows how to connect a 512K x 8-bit memory on NCS2 (DBW = 10).

Figure 22-4. Memory Connection for an 8-bit Data Path Device

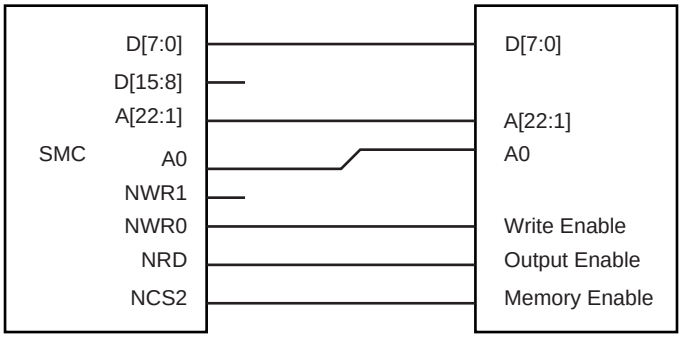
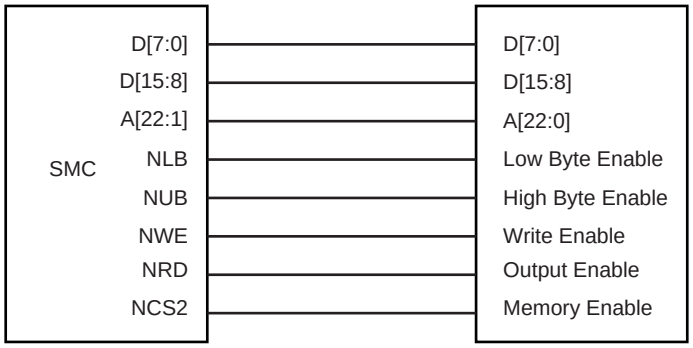


Figure 22-5 shows how to connect a 512K x 16-bit memory on NCS2 (DBW = 01).

Figure 22-5. Memory Connection for a 16-bit Data Path Device



22.6.2 Write Access

22.6.2.1 Write Access Type

Each chip select with a 16-bit data bus can operate with one of two different types of write access:

- Byte Write Access supports two byte write and a single read signal.
- Byte Select Access selects upper and/or lower byte with two byte select lines, and separate read and write signals.

This option is controlled by the BAT field in the SMC_CSR for the corresponding chip select. See “SMC Chip Select Registers” on page 196.

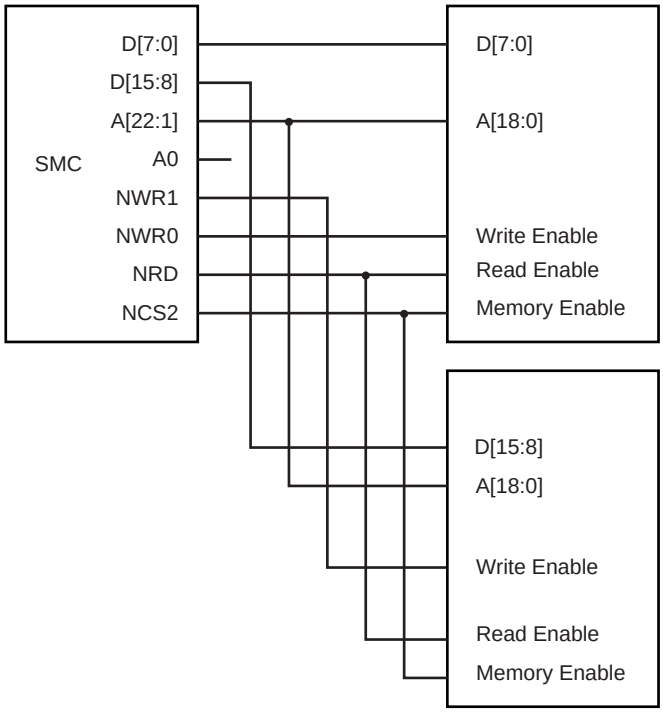
22.6.2.2 Byte Write Access

Byte Write Access is used to connect 2 x 8-bit devices as a 16-bit memory page.

- The signal A0/NLB is not used.
- The signal NWR1/NUB is used as NWR1 and enables upper byte writes.
- The signal NWR0/NWE is used as NWR0 and enables lower byte writes.
- The signal NRD enables half-word and byte reads.

Figure 22-6 shows how to connect two 512K x 8-bit devices in parallel on NCS2 (BAT = 0)

Figure 22-6. Memory Connection for 2 x 8-bit Data Path Devices



22.6.2.3 *Byte Select Access*

Byte Select Access is used to connect 16-bit devices in a memory page.

- The signal A0/NLB is used as NLB and enables the lower byte for both read and write operations.
- The signal NWR1/NUB is used as NUB and enables the upper byte for both read and write operations.
- The signal NWR0/NWE is used as NWE and enables writing for byte or half-word.
- The signal NRD enables reading for byte or half-word.

Figure 22-7 shows how to connect a 16-bit device with byte and half-word access (e.g., SRAM device type) on NCS2 (BAT = 1).

Figure 22-7. Connection to a 16-bit Data Path Device with Byte and Half-word Access

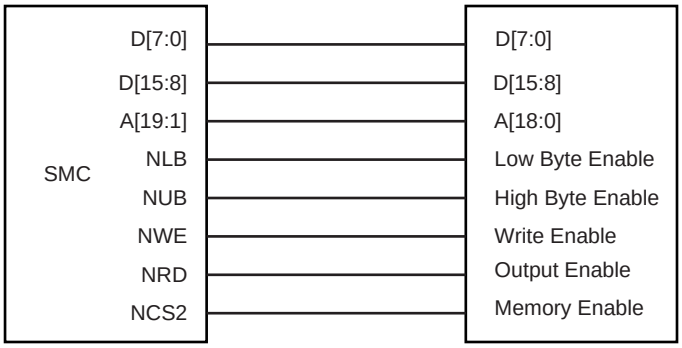
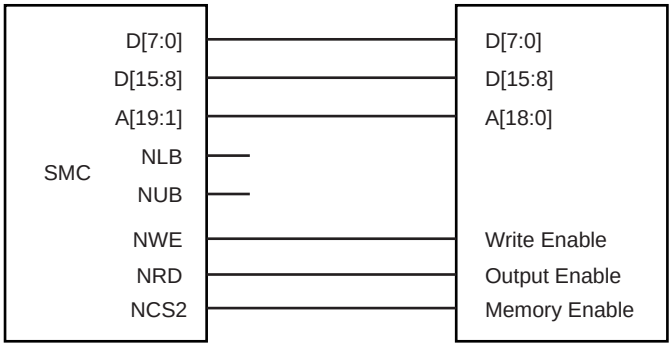


Figure 22-8 shows how to connect a 16-bit device without byte access (e.g., Flash device type) on NCS2 (BAT = 1).

Figure 22-8. Connection to a 16-bit Data Path Device without Byte Write Capability



22.6.2.4 Write Data Hold Time

During write cycles, data output becomes valid after the rising edge of MCK and remains valid after the rising edge of NWE. During a write access, the data remain on the bus 1/2 period of MCK after the rising edge of NWE. See Figure 22-9 and Figure 22-10.

Figure 22-9. Write Access with 0 Wait State

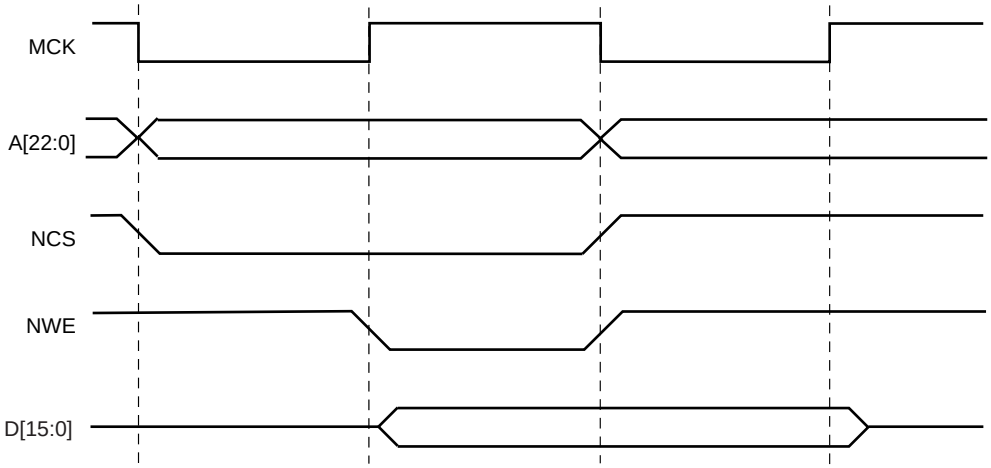
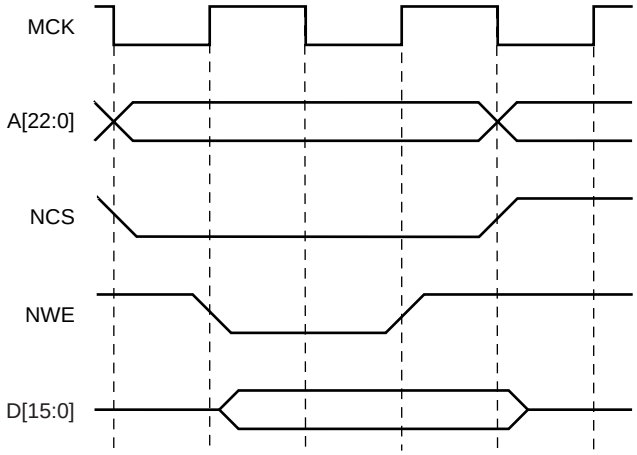


Figure 22-10. Write Access with 1 Wait State



22.6.3 Read Access

22.6.3.1 Read Protocols

The SMC provides two alternative protocols for external memory read accesses: standard and early read. The difference between the two protocols lies in the behavior of the NRD signal.

For write accesses, in both protocols, NWE has the same behavior. In the second half of the master clock cycle, NWE always goes low (see Figure 22-18 on page 173).

The protocol is selected by the DRP field in SMC_CSR (See “SMC Chip Select Registers” on page 196.). Standard read protocol is the default protocol after reset.

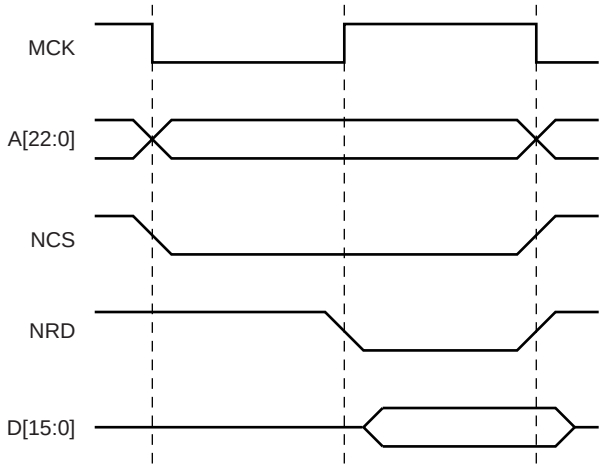
Note: In the following waveforms and descriptions NWE represents NWE, NWR0 and NWR1 unless NWR0 and NWR1 are otherwise represented. In addition, NCS represents NCS[7:0] (see 22.5.1 “I/O Lines” on page 163, Table 22-1 and Table 22-2).

22.6.3.2 Standard Read Protocol

Standard read protocol implements a read cycle during which NRD and NWE are similar. Both are active during the second half of the clock cycle. The first half of the clock cycle allows time to ensure completion of the previous access as well as the output of address lines and NCS before the read cycle begins.

During a standard read protocol, NCS is set low and address lines are valid at the beginning of the external memory access, while NRD goes low only in the second half of the master clock cycle to avoid bus conflict. See Figure 22-11.

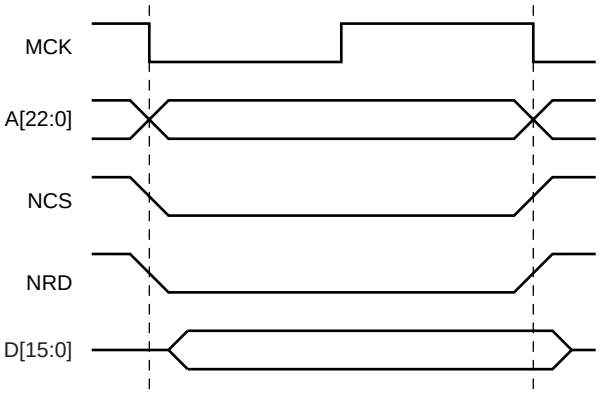
Figure 22-11. Standard Read Protocol



22.6.3.3 Early Read Protocol

Early read protocol provides more time for a read access from the memory by asserting NRD at the beginning of the clock cycle. In the case of successive read cycles in the same memory, NRD remains active continuously. Since a read cycle normally limits the speed of operation of the external memory system, early read protocol can allow a faster clock frequency to be used. However, an extra wait state is required in some cases to avoid contentions on the external bus.

Figure 22-12. Early Read Protocol



22.6.4 Wait State Management

The SMC can automatically insert wait states. The different types of wait states managed are listed below:

- Standard wait states
- External wait states
- Data float wait states
- Chip select change wait states
- Early Read wait states

22.6.4.1 Standard Wait States

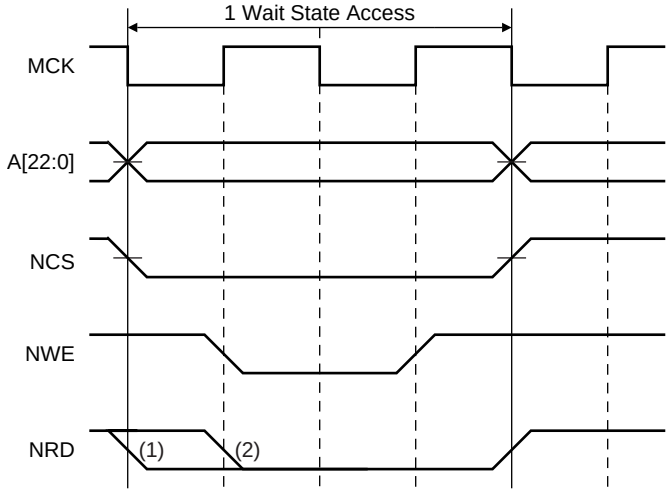
Each chip select can be programmed to insert one or more wait states during an access on the corresponding memory area. This is done by setting the WSEN field in the corresponding SMC_CSR ("SMC Chip Select Registers" on page 196). The number of cycles to insert is programmed in the NWS field in the same register.

Below is the correspondence between the number of standard wait states programmed and the number of clock cycles during which the NWE pulse is held low:

0 wait states	1/2 clock cycle
1 wait state	1 clock cycle

For each additional wait state programmed, an additional cycle is added.

Figure 22-13. One Standard Wait State Access



Notes: 1. Early Read Protocol
2. Standard Read Protocol

22.6.4.2 External Wait States

The NWAIT input pin is used to insert wait states beyond the maximum standard wait states programmable or in addition to. If NWAIT is asserted low, then the SMC adds a wait state and no changes are made to the output signals, the internal counters or the state. When NWAIT is deasserted, the SMC completes the access sequence.

WARNING: Asserting NWAIT low stops the core's clock and thus stops program execution.

The input of the NWAIT signal is an asynchronous input. To avoid any metastability problems, NWAIT is synchronized before using it. This operation results in a two-cycle delay.

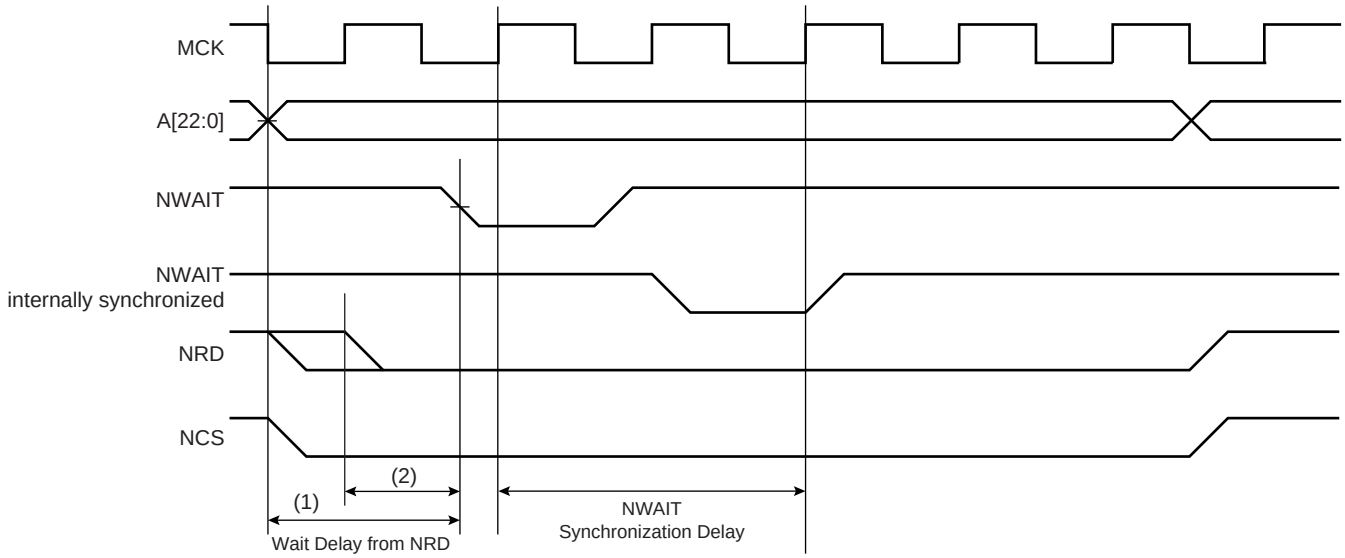
NWS must be programmed as a function of synchronization time and delay between NWAIT falling and control signals falling (NRD/NWE), otherwise SMC will not function correctly.

$$NWS \geq \text{Wait Delay from nrd/nwe} + \text{external\_nwait Synchronization Delay} + 1$$

Note: Where external NWAIT synchronization is equal to 2 cycles.
The minimum value for NWS if NWAIT is used, is 3.

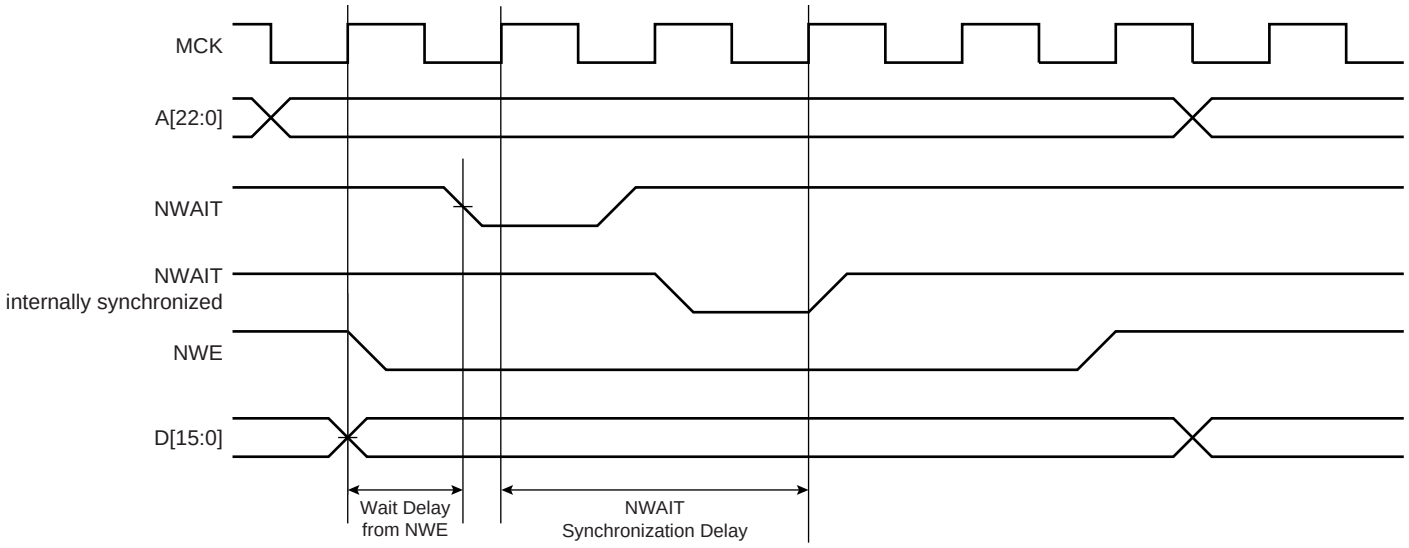
WARNING: If NWAIT is asserted during a setup or hold timing, the SMC does not function correctly.

Figure 22-14. NWAIT Behavior in Read Access [NWS = 3]



Notes: 1. Early Read Protocol
2. Standard Read Protocol

Figure 22-15. NWAIT Behavior in Write Access [NWS = 3]



22.6.4.3 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 write access or a read access to a different external memory.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF field of the SMC_CSR register for the corresponding chip select ("SMC Chip Select Registers" on page 196). The value of TDF indicates the number of data float wait cycles (between 0 and

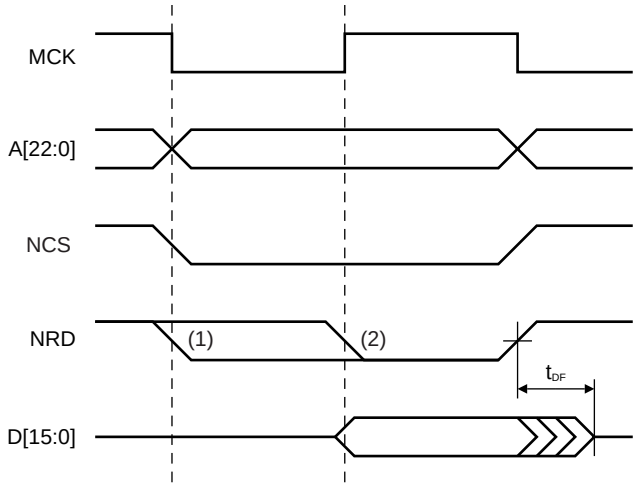
15) to be inserted and represents the time allowed for the data output to go to high impedance after the memory is disabled.

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.

To ensure that the external memory system is not accessed while it is still busy, the SMC keeps track of the programmed external data float time during internal accesses.

Internal memory accesses and consecutive read accesses to the same external memory do not add data float wait states.

Figure 22-16. Data Float Output Delay

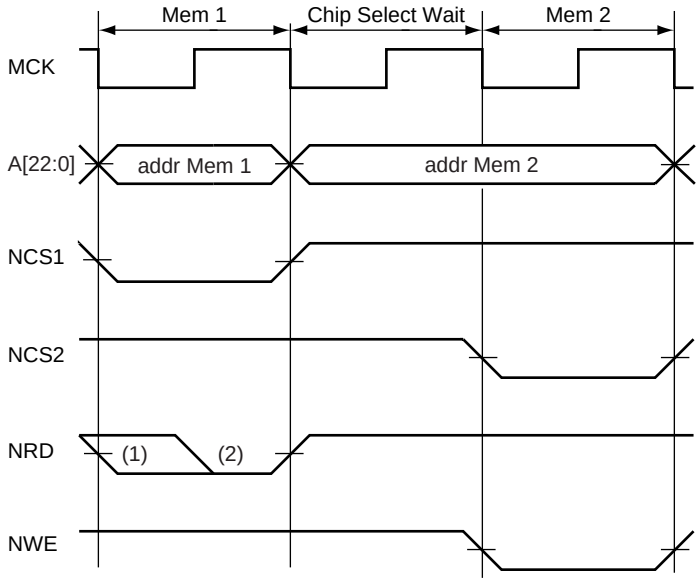


Notes: 1. Early Read Protocol
2. Standard Read Protocol

22.6.4.4 Chip Select Change Wait State

A chip select wait state is automatically inserted when consecutive accesses are made to two different external memories (if no other type of wait state has already been inserted). If a wait state has already been inserted (e.g., data float wait state), then no more wait states are added.

Figure 22-17. Chip Select Wait State



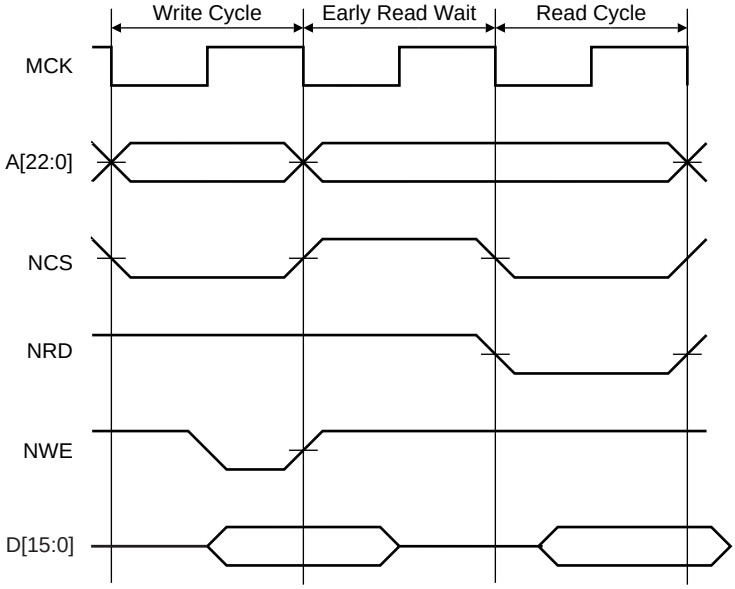
Notes: 1. Early Read Protocol
2. Standard Read Protocol

22.6.4.5 *Early Read Wait State*

In early read protocol, an early read wait state is automatically inserted when an external write cycle is followed by a read cycle to allow time for the write cycle to end before the subsequent read cycle begins (see Figure 22-18). This wait state is generated in addition to any other programmed wait states (i.e., data float wait state).

No wait state is added when a read cycle is followed by a write cycle, between consecutive accesses of the same type, or between external and internal memory accesses.

Figure 22-18. Early Read Wait States



22.6.5 Setup and Hold Cycles

The SMC allows some memory devices to be interfaced with different setup, hold and pulse delays. These parameters are programmable and define the timing of each portion of the read and write cycles. However, it is not possible to use this feature in early read protocol.

If an attempt is made to program the setup parameter as not equal to zero and the hold parameter as equal to zero with WSEN = 0 (0 standard wait state), the SMC does not operate correctly.

If consecutive accesses are made to two different external memories and the second memory is programmed with setup cycles, then no chip select change wait state is inserted (see Figure 22-23 on page 176).

When a data float wait state (t_{DF}) is programmed on the first memory bank and when the second memory bank is programmed with setup cycles, the SMC behaves as follows:

- If the number of t_{DF} is higher or equal to the number of setup cycles, the number of setup cycles inserted is equal to 0 (see Figure 22-24 on page 176).
- If the number of the setup cycle is higher than the number of $t_{DF,}$ the number of t_{DF} inserted is 0 (see Figure 22-25 on page 177).

22.6.5.1 Read Access

The read cycle can be divided into a setup, a pulse length and a hold. The setup parameter can have a value between 1.5 and 7.5 clock cycles, the hold parameter between 0 and 7 clock cycles and the pulse length between 1.5 and 128.5 clock cycles, by increments of one.

Figure 22-19. Read Access with Setup and Hold

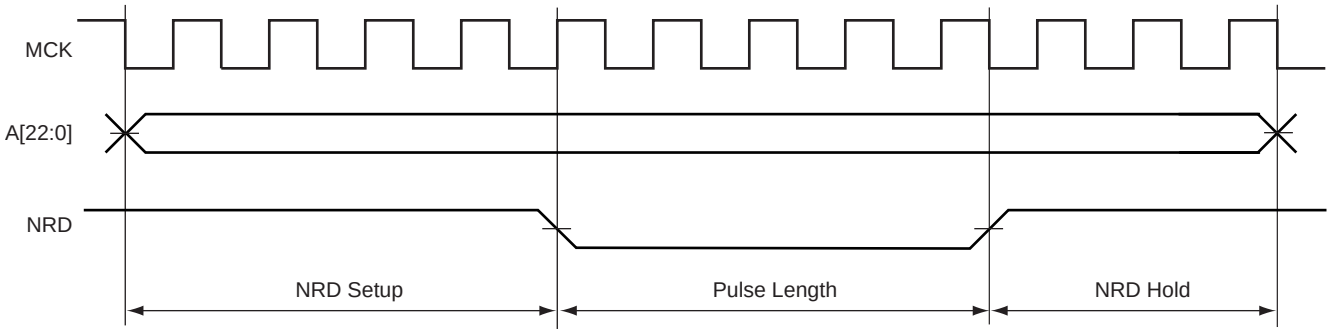
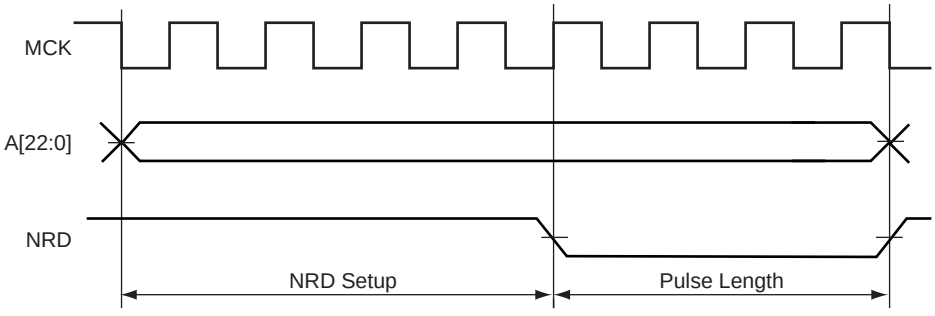


Figure 22-20. Read Access with Setup



22.6.5.2 *Write Access*

The write cycle can be divided into a setup, a pulse length and a hold. The setup parameter can have a value between 1.5 and 7.5 clock cycles, the hold parameter between 0.5 and 7 clock cycles and the pulse length between 1 and 128 clock cycles by increments of one.

Figure 22-21. Write Access with Setup and Hold

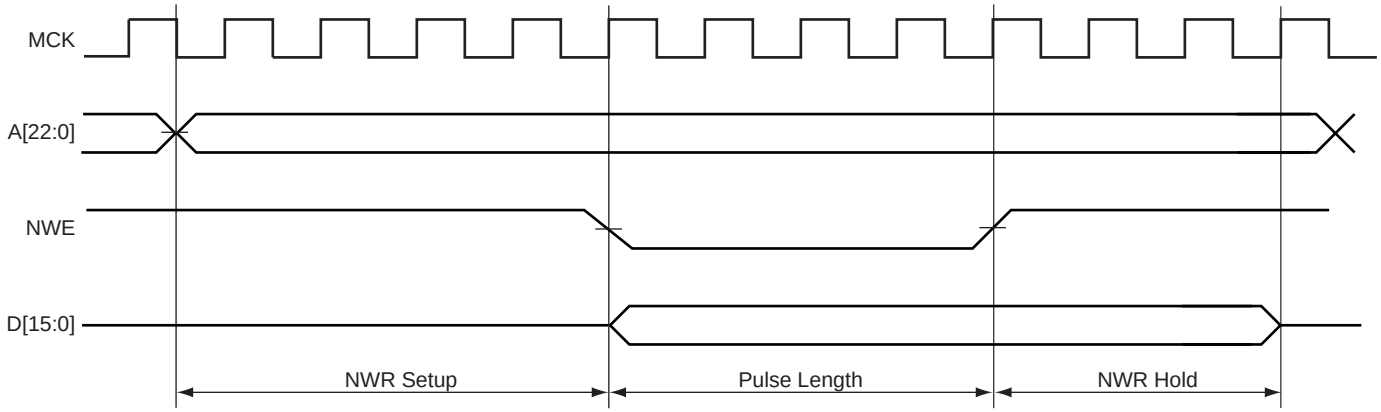
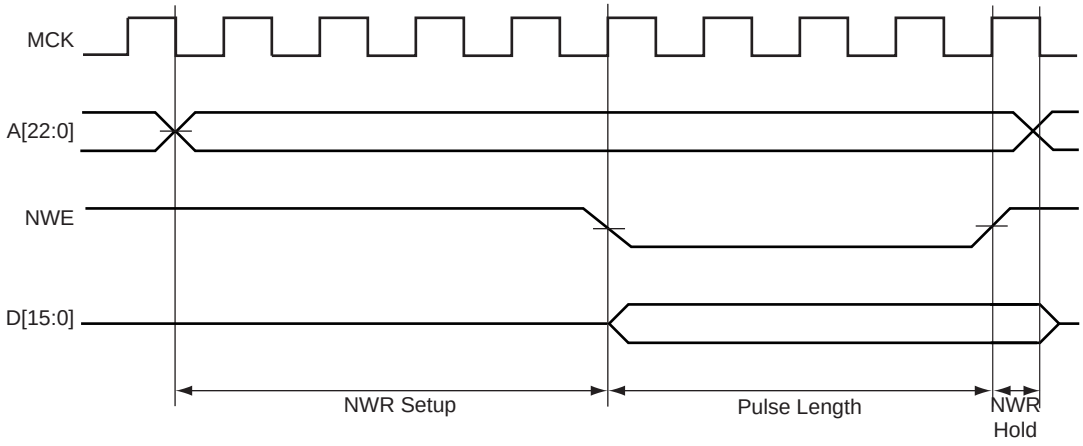


Figure 22-22. Write Access with Setup



22.6.5.3 *Data Float Wait States with Setup Cycles*

Figure 22-23. Consecutive Accesses with Setup Programmed on the Second Access

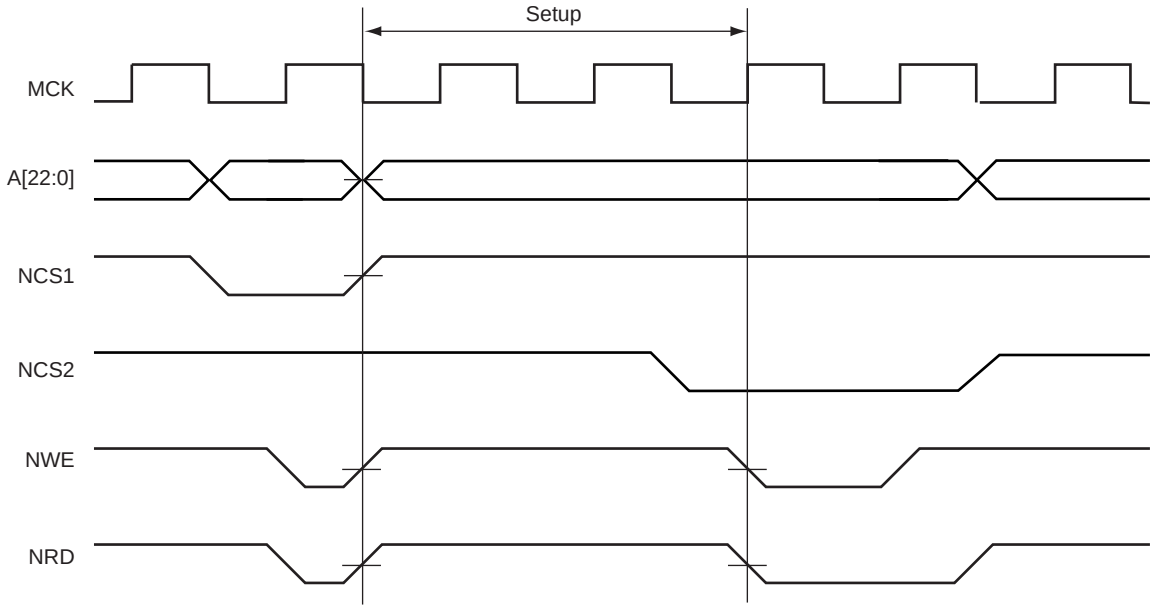


Figure 22-24. First Access with Data Float Wait States (TDF = 2) and Second Access with Setup (NRDSETUP = 1)

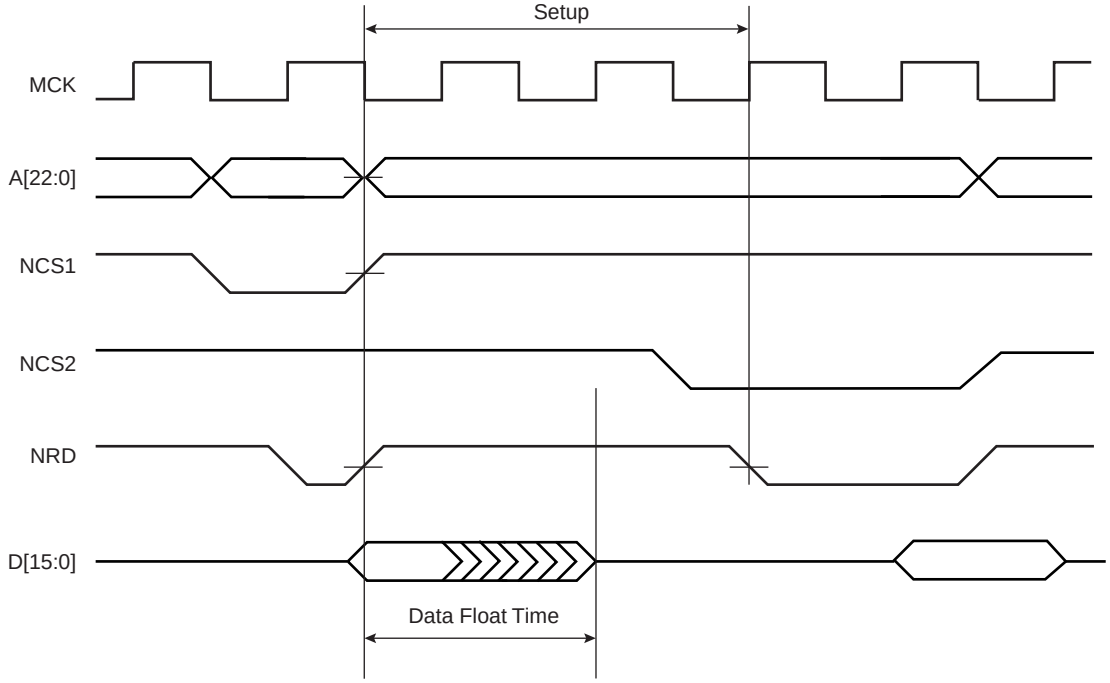
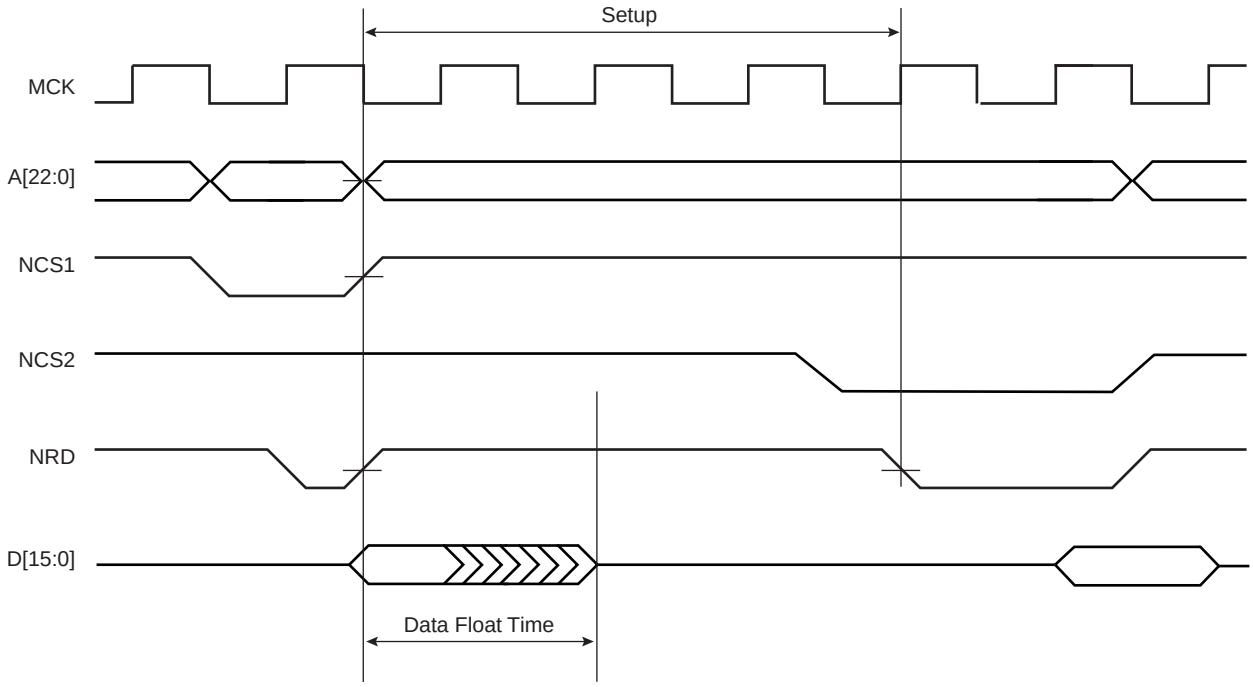


Figure 22-25. First Access with Data Float Wait States (TDF = 2) and Second Access with Setup (NRDSETUP = 3)



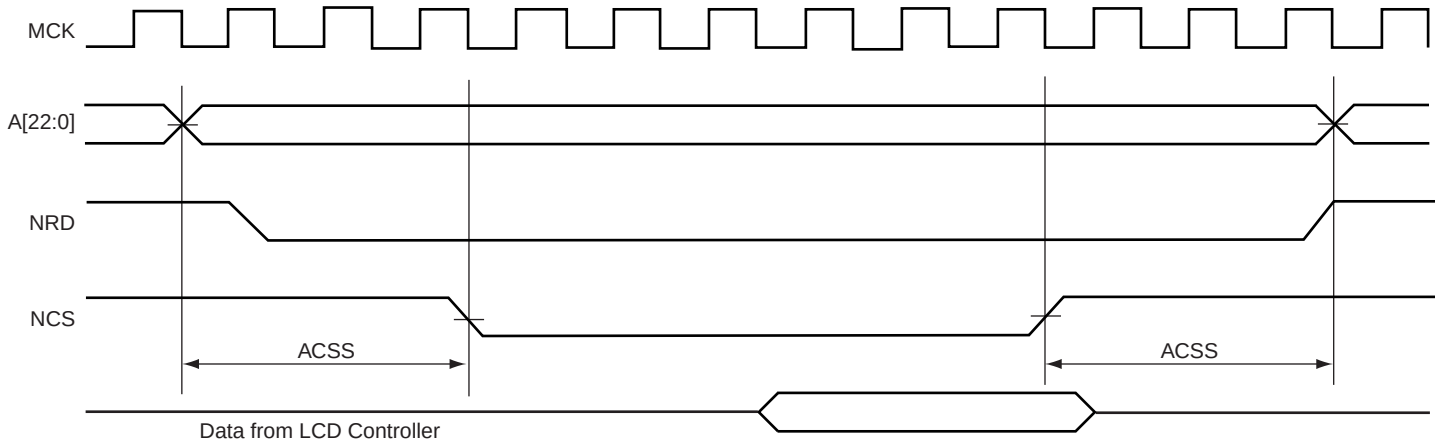
22.6.6 LCD Interface Mode

The SMC can be configured to work with an external liquid crystal display (LCD) controller by setting the ACSS (Address to Chip Select Setup) bit in the SMC_CSR registers ("SMC Chip Select Registers" on page 196).

In LCD mode, NCS is shortened by one/two/three clock cycles at the leading and trailing edges, providing positive address setup and hold. For read accesses, the data is latched in the SMC when NCS is raised at the end of the access.

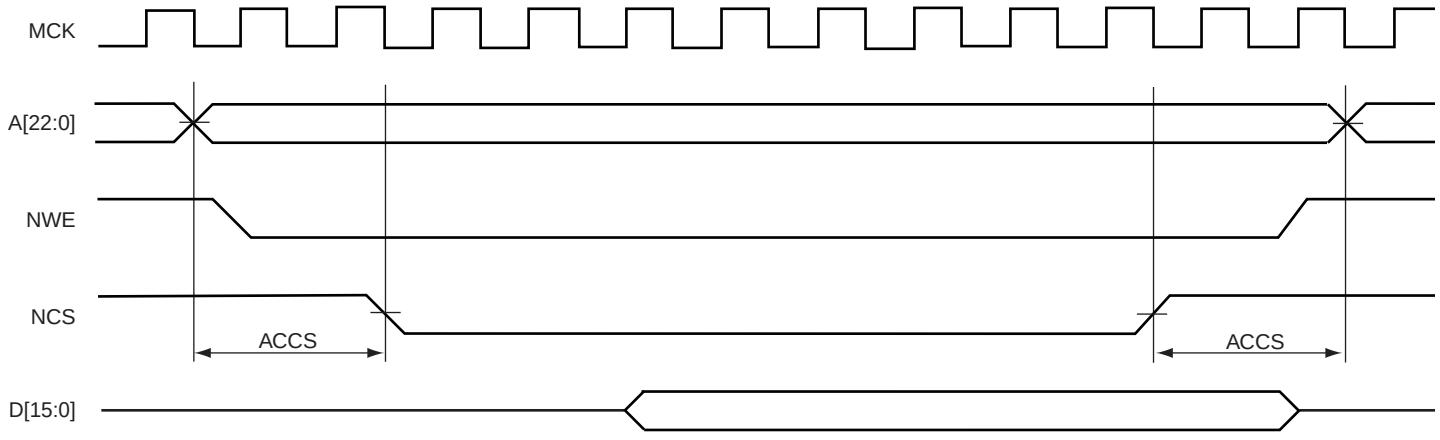
Additionally, WSEN must be set and NWS programmed with a value of two or more superior to ACSS. In LCD mode, it is not recommended to use RWHOLD or RWSETUP. If the above conditions are not satisfied, SMC does not operate correctly.

Figure 22-26. Read Access in LCD Interface Mode



ACSS = 3, NWEN = 1, NWS = 10

Figure 22-27. Write Access in LCD Interface Mode



ACCS = 2, NWEN = 1, NWS = 10

22.6.7 Memory Access Waveforms

22.6.7.1 Read Accesses in Standard and Early Protocols

Figure 22-28 on page 179 through Figure 22-31 on page 182 show examples of the alternatives for external memory read protocol.

Figure 22-28. Standard Read Protocol without t_{DF}

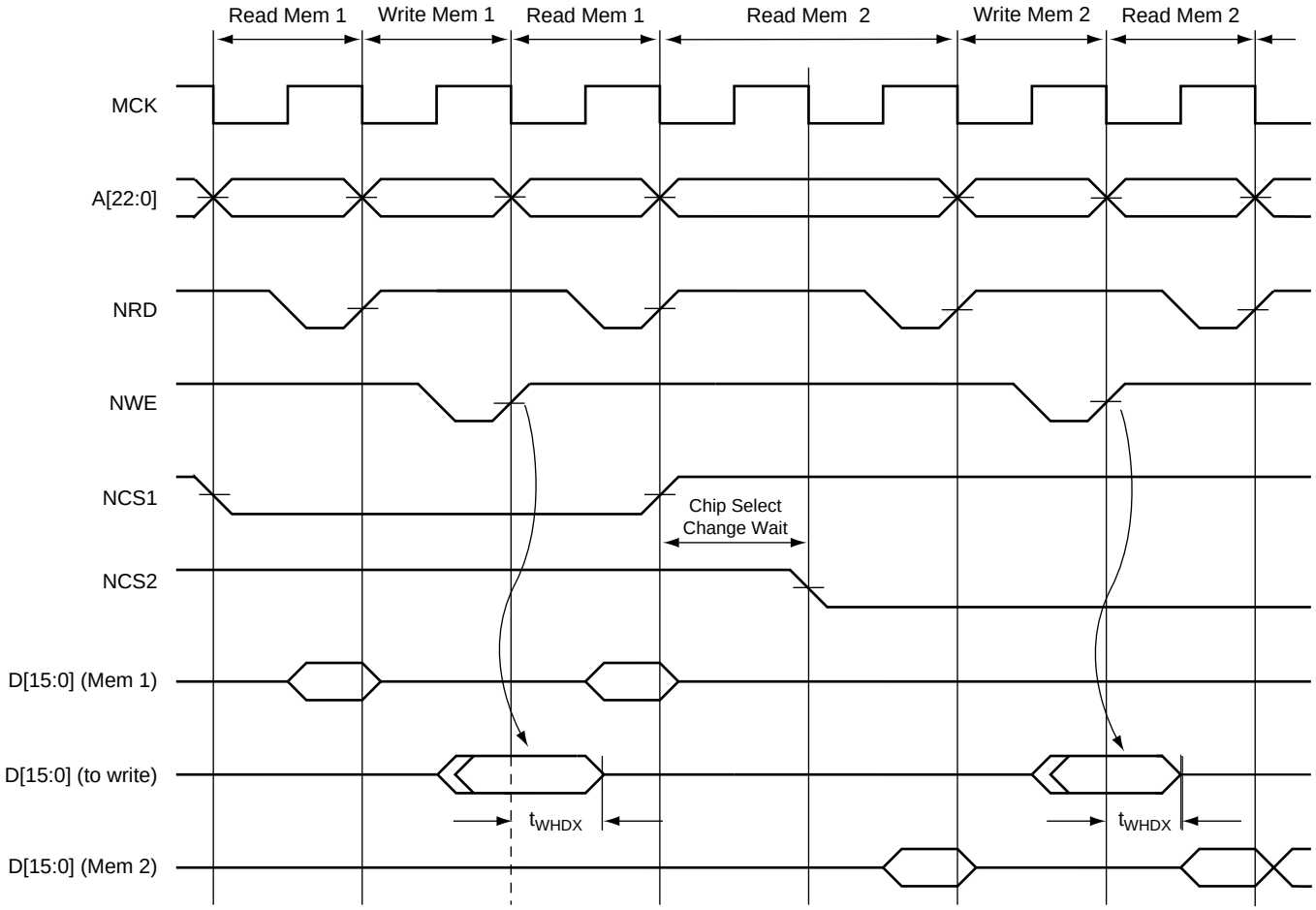


Figure 22-29. Early Read Protocol without t_{DF}

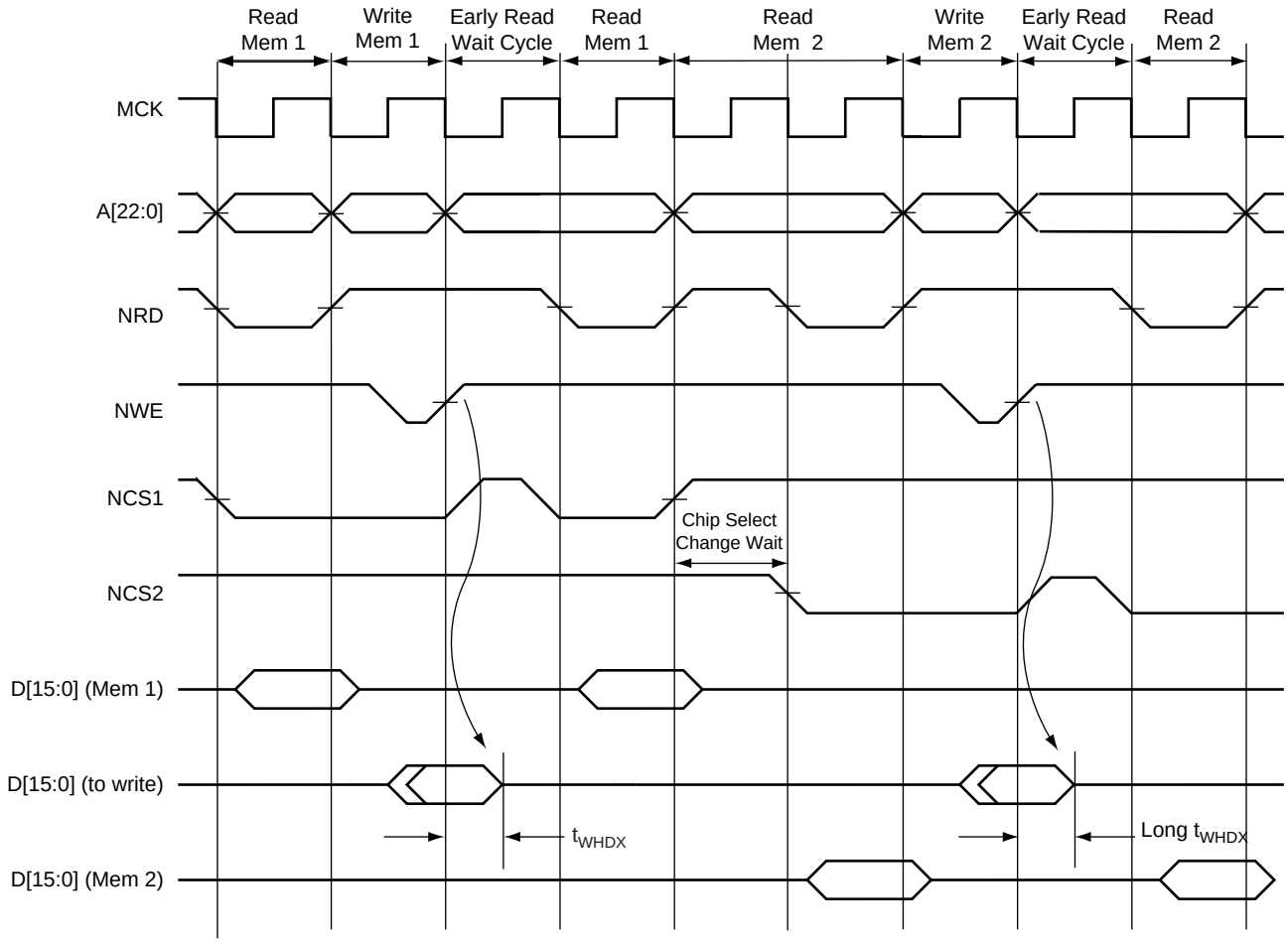


Figure 22-30. Standard Read Protocol with t_{DF}

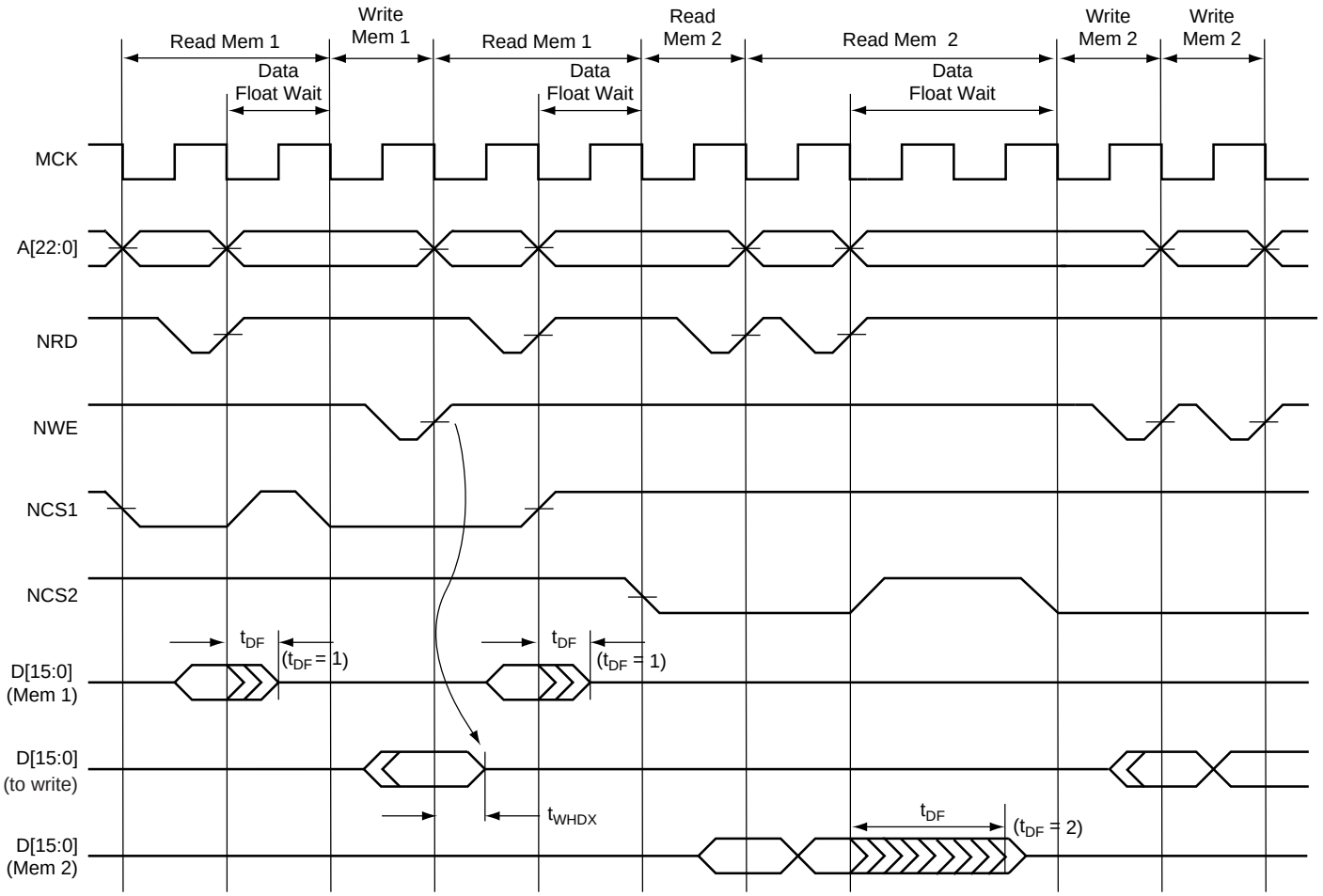
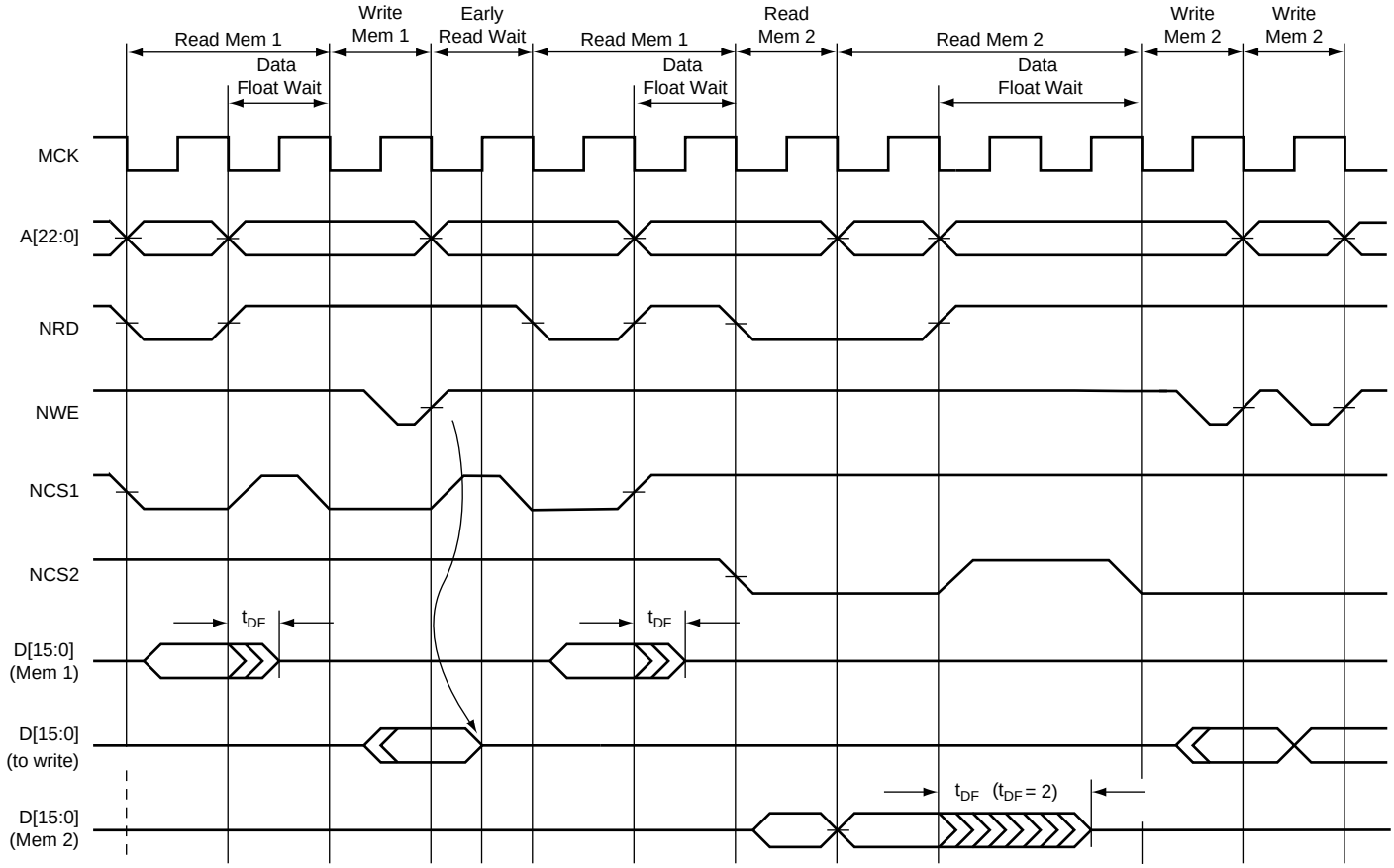


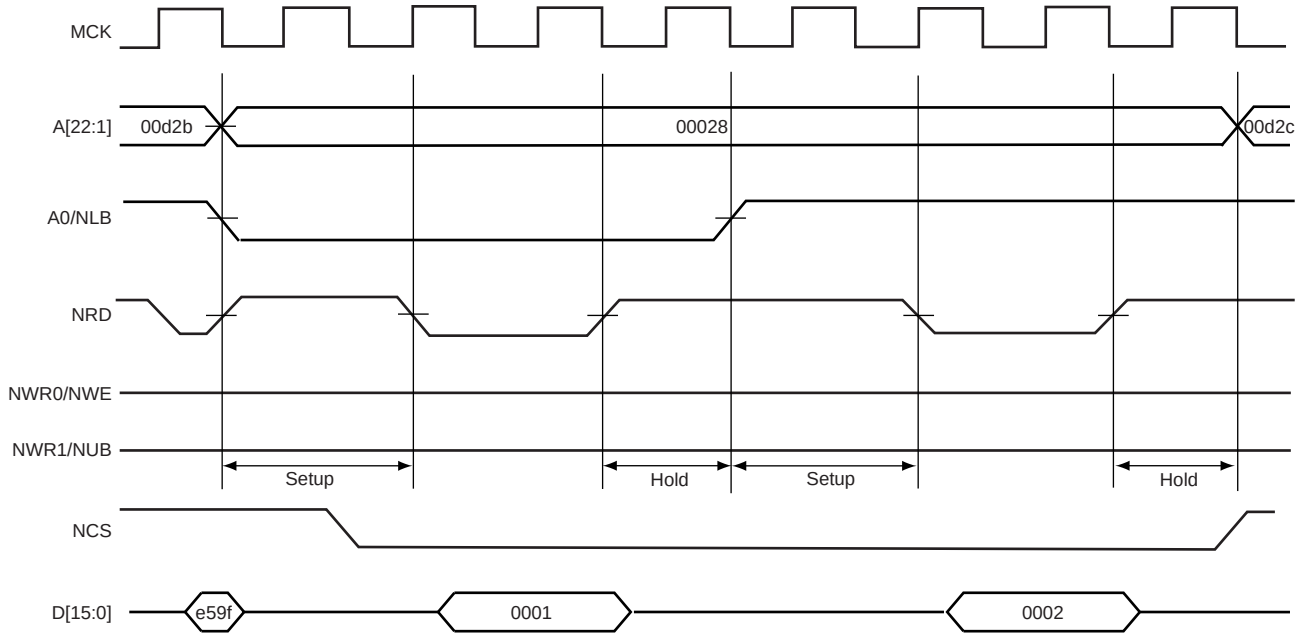
Figure 22-31. Early Read Protocol with t_{DF}



22.6.7.2 Accesses with Setup and Hold

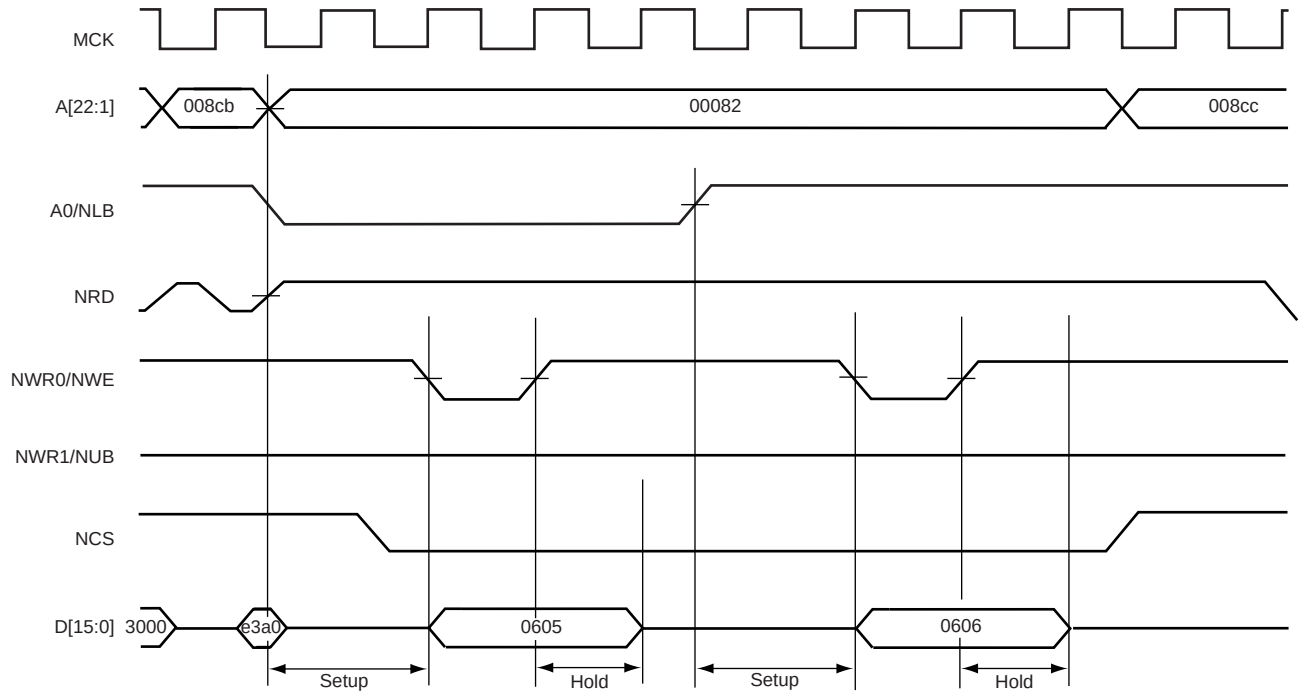
Figure 22-32 and Figure 22-33 show an example of read and write accesses with Setup and Hold Cycles.

Figure 22-32. Read Accesses in Standard Read Protocol with Setup and Hold[1]



Note: 1. Read access, memory data bus width = 8, RWSETUP = 1, RWHOLD = 1,WSEN= 1, NWS = 0

Figure 22-33. Write Accesses with Setup and Hold[1]

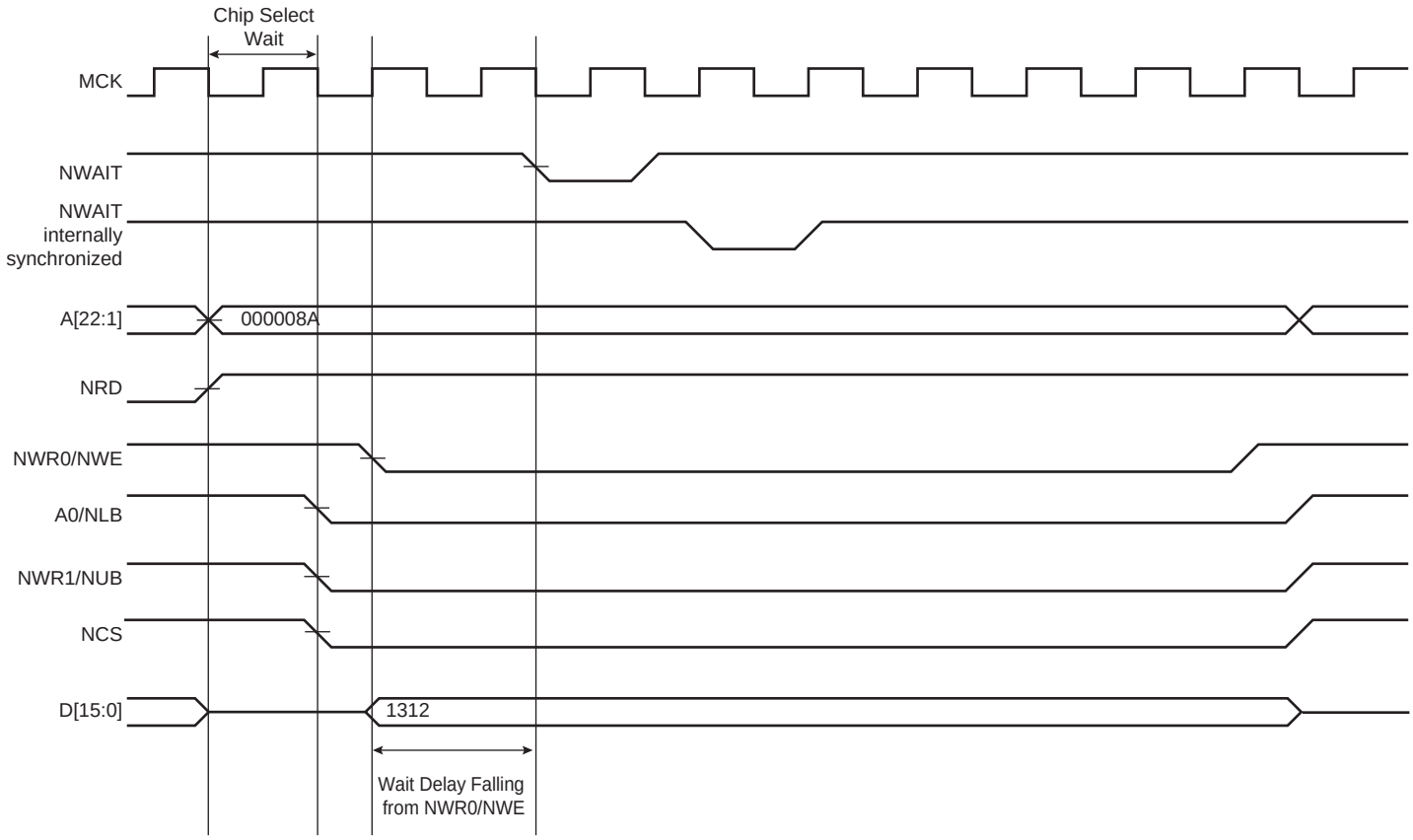


Note: 1. Write access, memory data bus width = 8, RWSETUP = 1, RWHOLD = 1, WSEN = 1, NWS = 0

22.6.7.3 *Accesses Using NWAIT Input Signal*

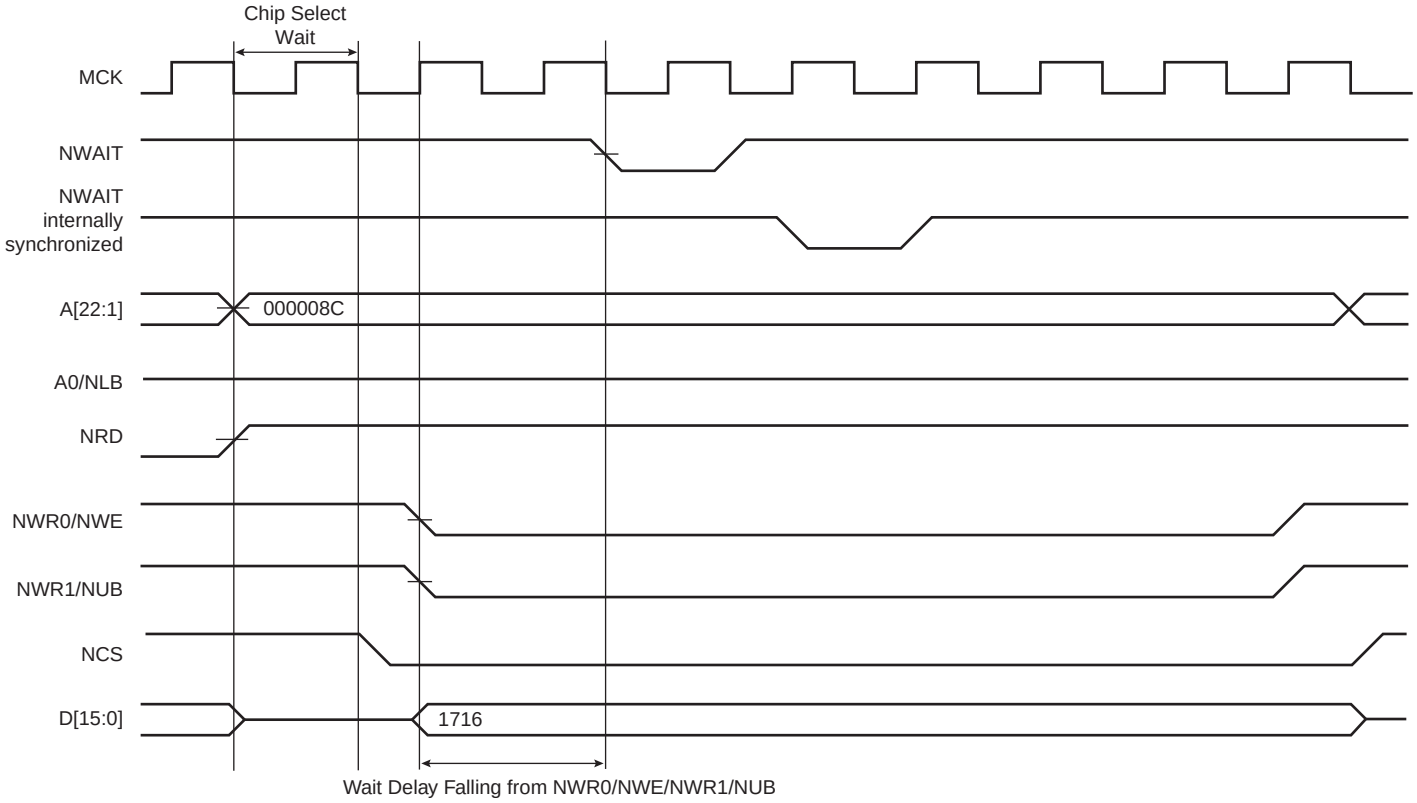
Figure 22-34 on page 184 through Figure 22-37 on page 187 show examples of accesses using NWAIT.

Figure 22-34. Write Access using NWAIT in Byte Select Type Access[1]



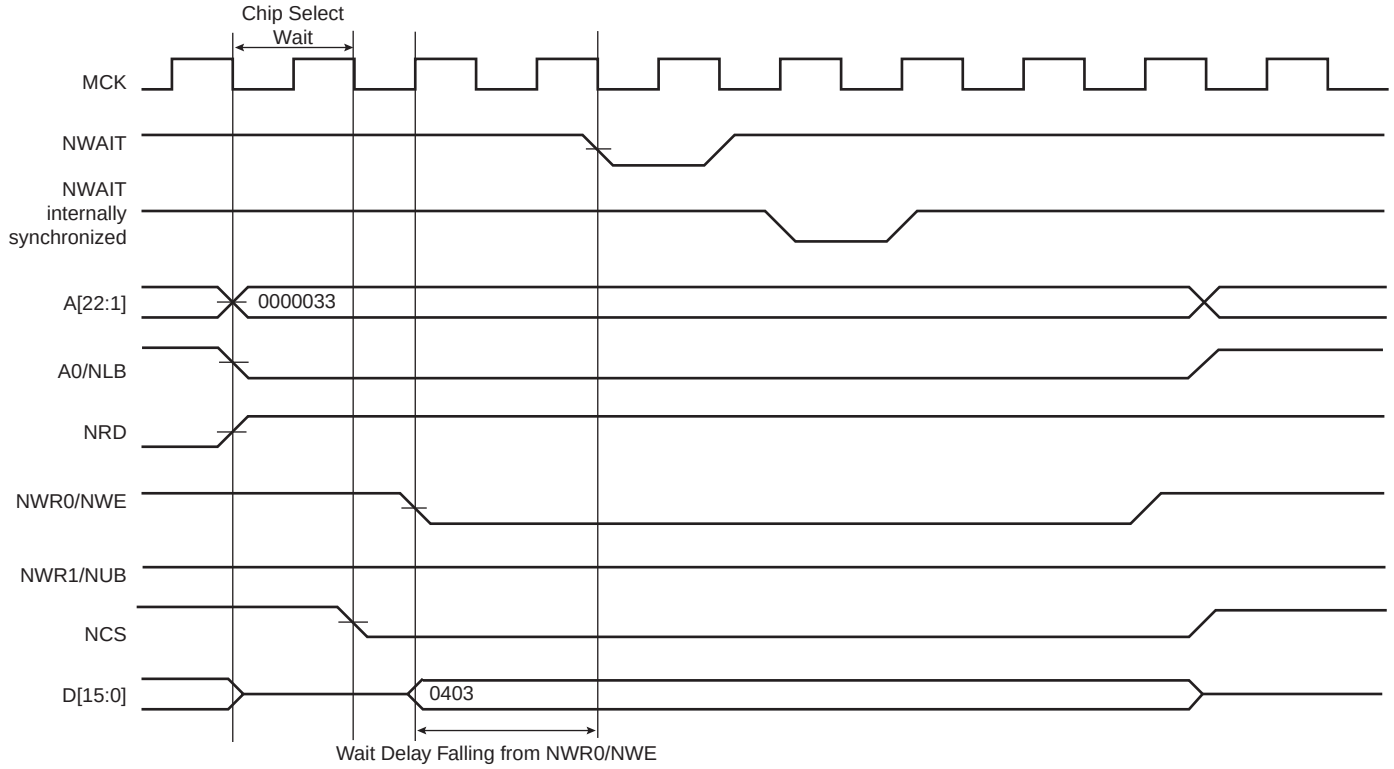
Note: 1. Write access memory, data bus width = 16 bits, WSEN = 1, NWS = 6

Figure 22-35. Write Access using NWAIT in Byte Write Type Access[1]



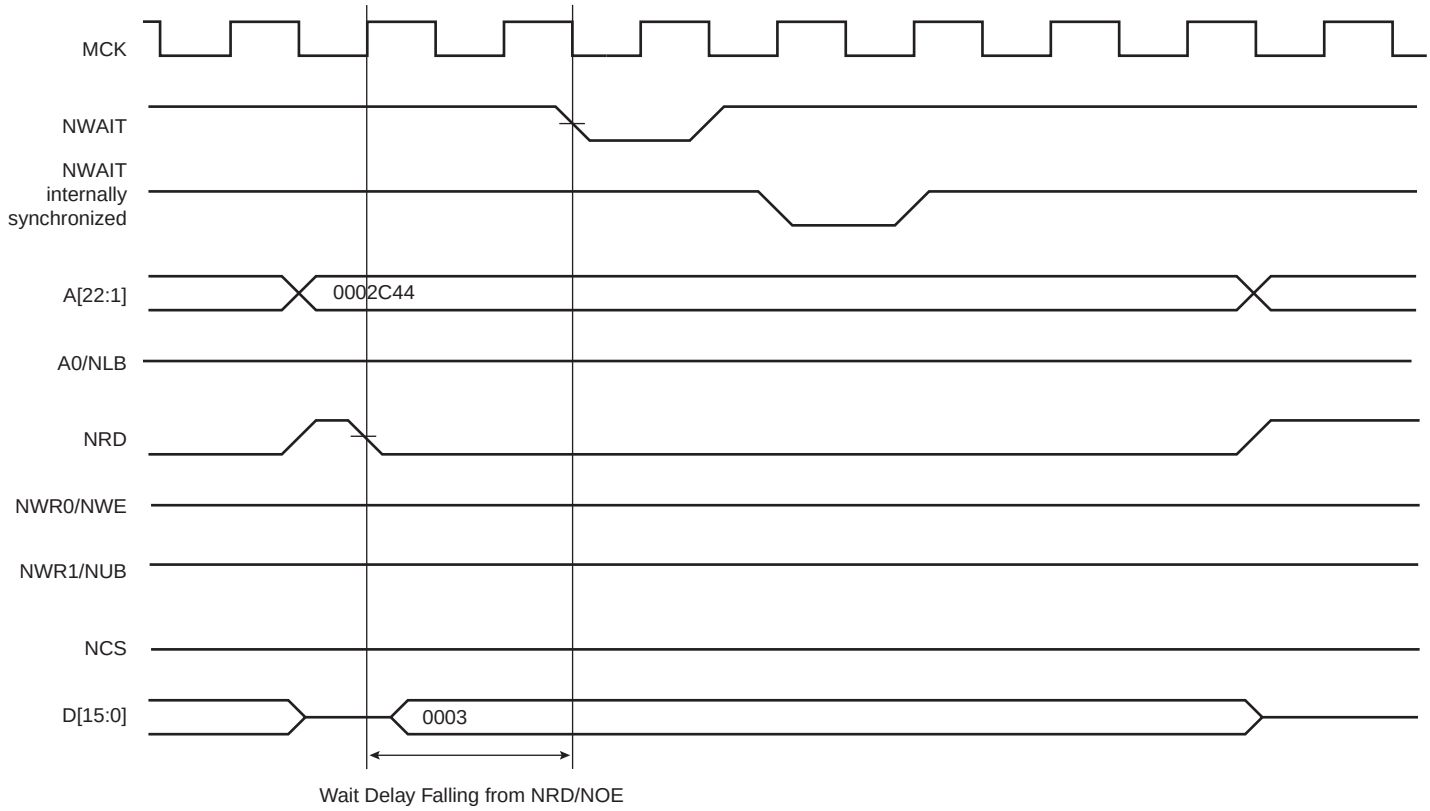
Note: 1. Write access memory, data bus width = 16 bits, WSEN = 1, NWS = 5

Figure 22-36. Write Access using NWAIT[(1)]



Note: 1. Write access memory, data bus width = 8 bits, WSEN = 1, NWS = 4

Figure 22-37. Read Access in Standard Protocol using NWAIT[1]



Note: 1. Read access, memory data bus width = 16, NWS = 5, WSEN = 1

22.6.7.4 Memory Access Example Waveforms

Figure 22-38 on page 188 through Figure 22-44 on page 194 show the waveforms for read and write accesses to the various associated external memory devices. The configurations described are shown in Table 22-3.

Table 22-3. Memory Access Waveforms

Figure Number	Number of Wait States	Bus Width	Size of Data Transfer
Figure 22-38	0	16	Word
Figure 22-39	1	16	Word
Figure 22-40	1	16	Half-word
Figure 22-41	0	8	Word
Figure 22-42	1	8	Half-word
Figure 22-43	1	8	Byte
Figure 22-44	0	16	Byte

Figure 22-38. 0 Wait State, 16-bit Bus Width, Word Transfer

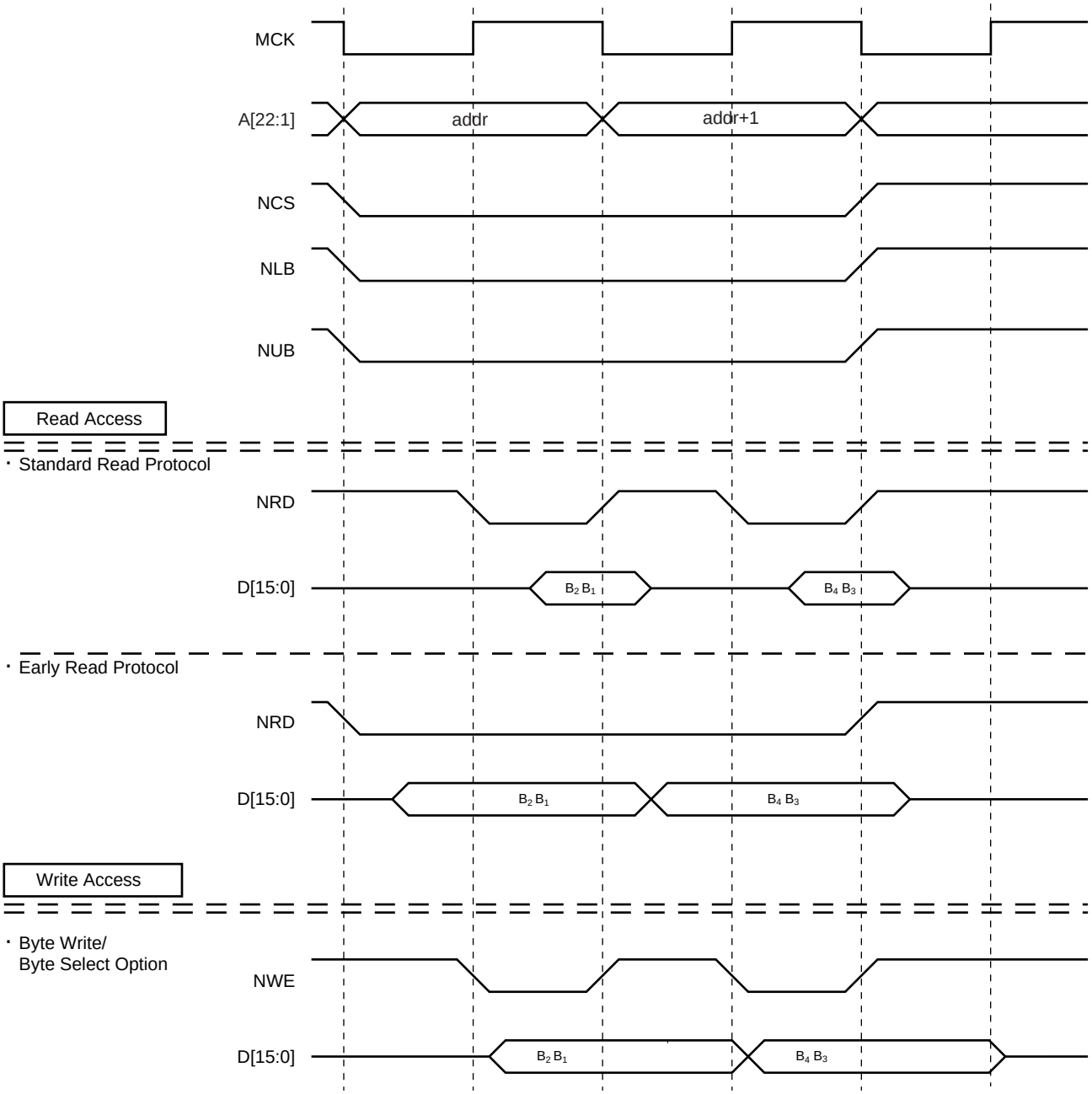


Figure 22-39. 1 Wait State, 16-bit Bus Width, Word Transfer

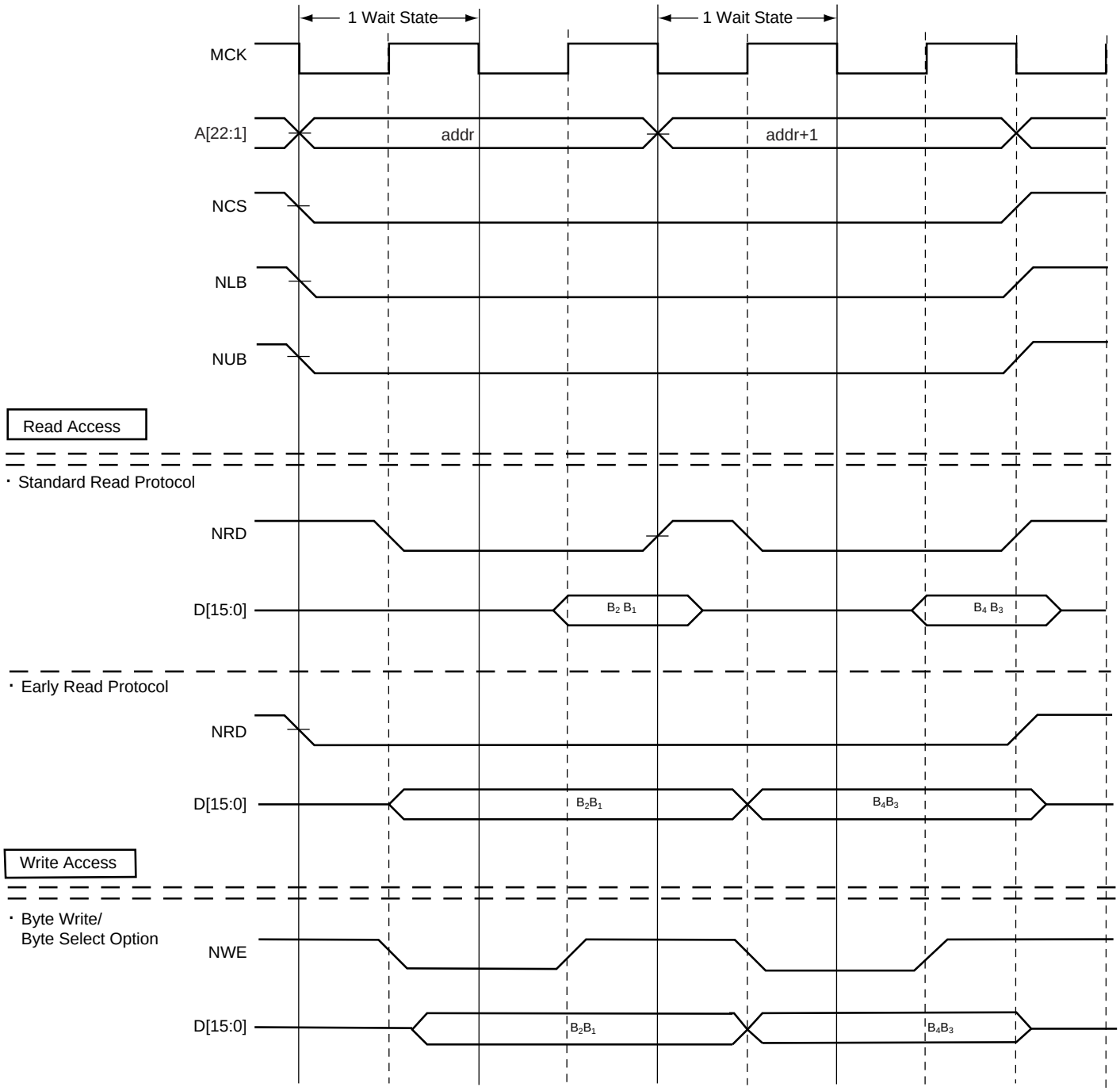


Figure 22-40. 1 Wait State, 16-bit Bus Width, Half-Word Transfer

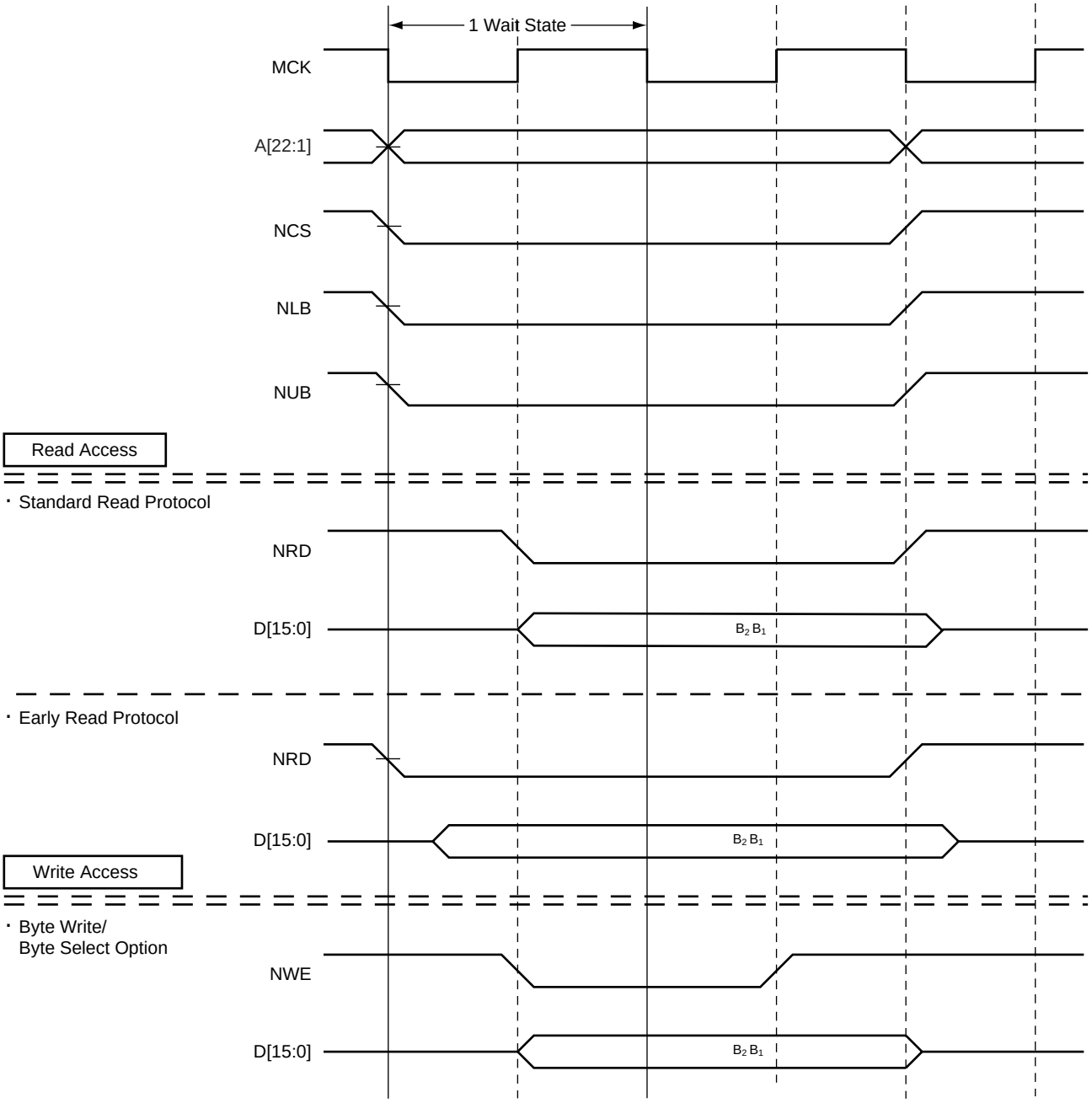


Figure 22-41. 0 Wait State, 8-bit Bus Width, Word Transfer

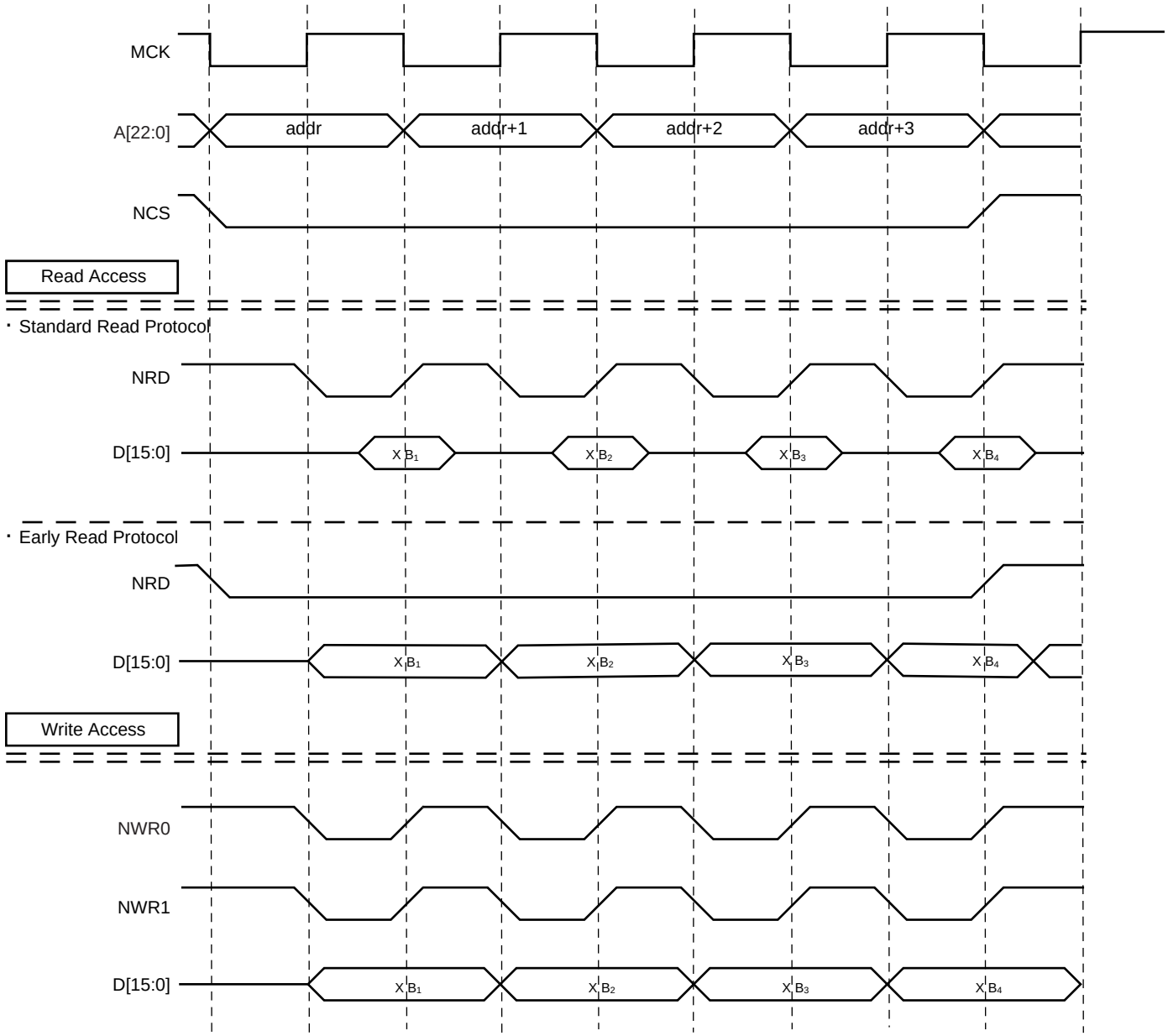


Figure 22-42. 1 Wait State, 8-bit Bus Width, Half-Word Transfer

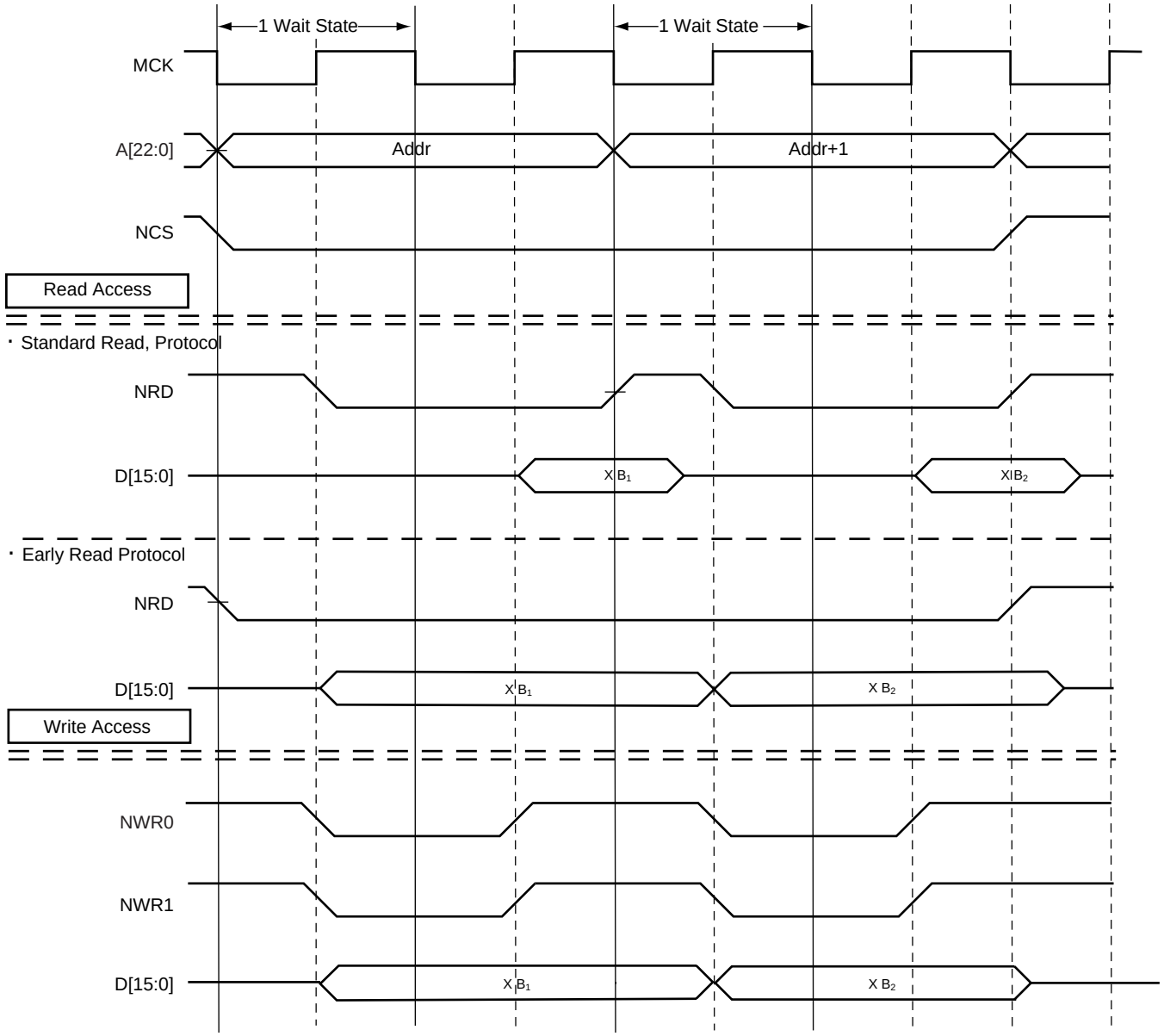


Figure 22-43. 1 Wait State, 8-bit Bus Width, Byte Transfer

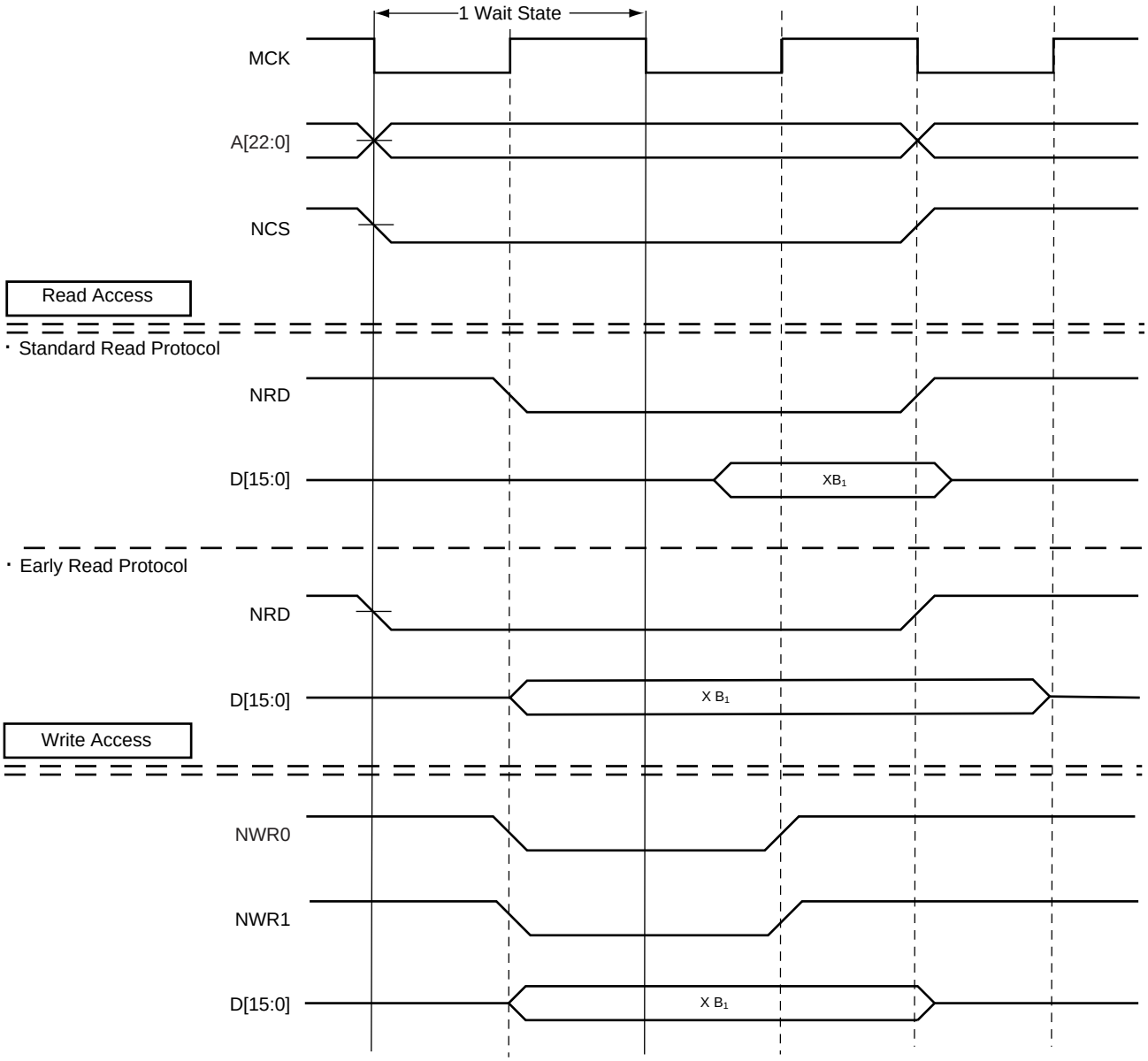
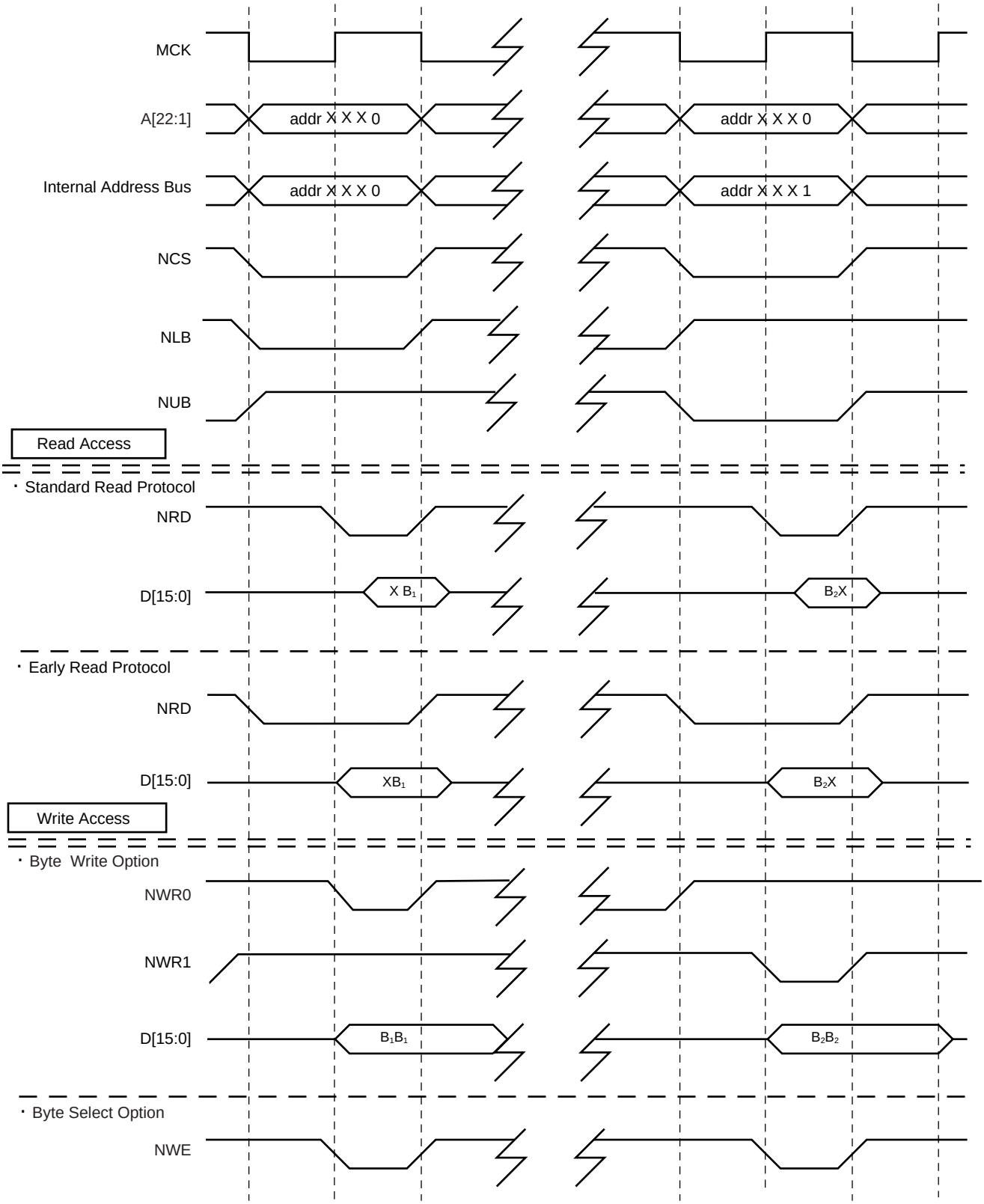


Figure 22-44. 0 Wait State, 16-bit Bus Width, Byte Transfer



22.7 Static Memory Controller (SMC) User Interface

The Static Memory Controller is programmed using the registers listed in Table 22-4. Eight Chip Select Registers (SMC_CSR0 to SMC_CSR7) are used to program the parameters for the individual external memories.

Table 22-4. Static Memory Controller Register Mapping

Offset	Register	Name	Access	Reset State
0x00	SMC Chip Select Register 0	SMC_CSR0	Read/Write	0x00002000
0x04	SMC Chip Select Register 1	SMC_CSR1	Read/Write	0x00002000
0x08	SMC Chip Select Register 2	SMC_CSR2	Read/Write	0x00002000
0x0C	SMC Chip Select Register 3	SMC_CSR3	Read/Write	0x00002000
0x10	SMC Chip Select Register 4	SMC_CSR4	Read/Write	0x00002000
0x14	SMC Chip Select Register 5	SMC_CSR5	Read/Write	0x00002000
0x18	SMC Chip Select Register 6	SMC_CSR6	Read/Write	0x00002000
0x1C	SMC Chip Select Register 7	SMC_CSR7	Read/Write	0x00002000

22.7.1 SMC Chip Select Registers

Name: SMC_CSR0..SMC_CSR7

Access: Read/Write

Reset Value: See Table 22-4 on page 195

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

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

15	14	13	12	11	10	9	8
DRP	DBW		BAT	TDF			

7	6	5	4	3	2	1	0
WSEN	NWS						

- **NWS: Number of Wait States**

This field defines the Read and Write signal pulse length from 1 cycle up to 128 cycles.

Note: When WSEN is 0, NWS will be read to 0 whichever the previous programmed value should be.

Number of Wait States	NWS Field	NRD Pulse Length Standard Read Protocol	NRD Pulse Length Early Read Protocol	NWR Pulse Length
0[1]	Don't Care	½ cycle	1 cycle	½ cycle
1	0	1 + ½ cycles	2 cycles	1 cycle
2	1	2 + ½ cycles	3 cycles	2 cycles
X + 1	Up to X = 127	X + 1+ ½ cycles	X + 2 cycles	X + 1 cycle

Note: 1. Assuming WSEN Field = 0.

- **WSEN: Wait State Enable**

0: Wait states are disabled.

1: Wait states are enabled.

- **TDF: Data Float Time**

The external bus is marked occupied and cannot be used by another chip select during TDF cycles. Up to 15 cycles can be defined and represents the time allowed for the data output to go to high impedance after the memory is disabled.

- **BAT: Byte Access Type**

This field is used only if DBW defines a 16-bit data bus.

0: Chip select line is connected to two 8-bit wide devices.

1: Chip select line is connected to a 16-bit wide device.

- **DBW: Data Bus Width**

DBW		Data Bus Width
0	0	Reserved
0	1	16-bit
1	0	8-bit
1	1	Reserved

- **DRP: Data Read Protocol**

0: Standard Read Protocol is used.

1: Early Read Protocol is used.

- **ACSS: Address to Chip Select Setup**

ACSS		Chip Select Waveform
0	0	Standard, asserted at the beginning of the access and deasserted at the end.
0	1	One cycle less at the beginning and the end of the access.
1	0	Two cycles less at the beginning and the end of the access.
1	1	Three cycles less at the beginning and the end of the access.

- **RWSETUP: Read and Write Signal Setup Time**

See definition and description below.

- **RWHOLD: Read and Write Signal Hold Time**

See definition and description below.

RWSETUP[1]			NRD Setup	NWR Setup
0	0	0	½ cycle[2] or 0 cycles[3]	½ cycle
0	0	1	1 + ½ cycles	1 + ½ cycles
0	1	0	2 + ½ cycles	2 + ½ cycles
0	1	1	3 + ½ cycles	3 + ½ cycles
1	0	0	4 + ½ cycles	4 + ½ cycles
1	0	1	5 + ½ cycles	5 + ½ cycles
1	1	0	6 + ½ cycles	6 + ½ cycles
1	1	1	7 + ½ cycles	7 + ½ cycles

RWHOLD[1] [4]			NRD Hold	NWR Hold
0	0	0	0	½ cycle
0	0	1	1 cycles	1 cycle
0	1	0	2 cycles	2 cycles
0	1	1	3 cycles	3 cycles
1	0	0	4 cycles	4 cycles
1	0	1	5 cycles	5 cycles
1	1	0	6 cycles	6 cycles
1	1	1	7 cycles	7 cycles

Notes:
1. For a visual description, please refer to “Setup and Hold Cycles” on page 174 and the diagrams in Figure 22-45 and Figure 22-46 and Figure 22-47 on page 198.
2. In Standard Read Protocol.
3. In Early Read Protocol. (It is not possible to use the parameters RWSETUP or RWHOLD in this mode.)
4. When the ECC Controller is used, RWHOLD must be programmed to 1 at least.

Figure 22-45. Read/Write Setup

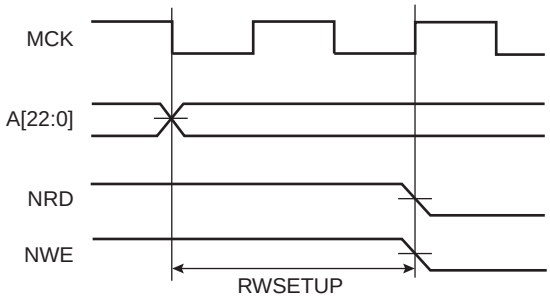


Figure 22-46. Read Hold

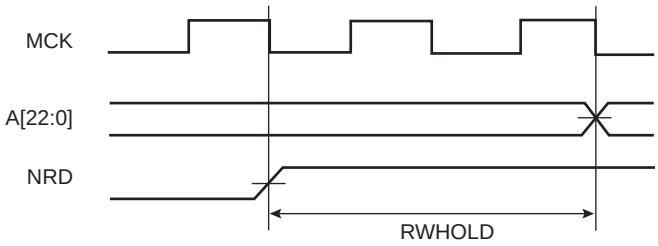
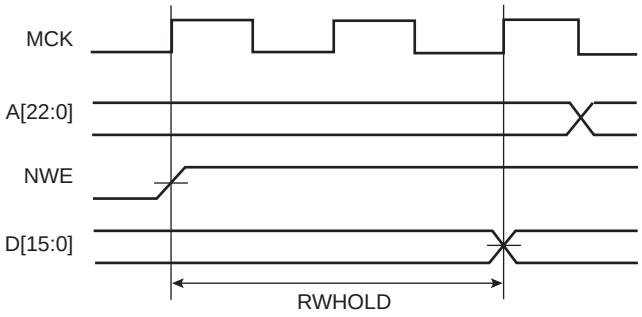


Figure 22-47. Write Hold



23. SDRAM Controller (SDRAMC)

23.1 Overview

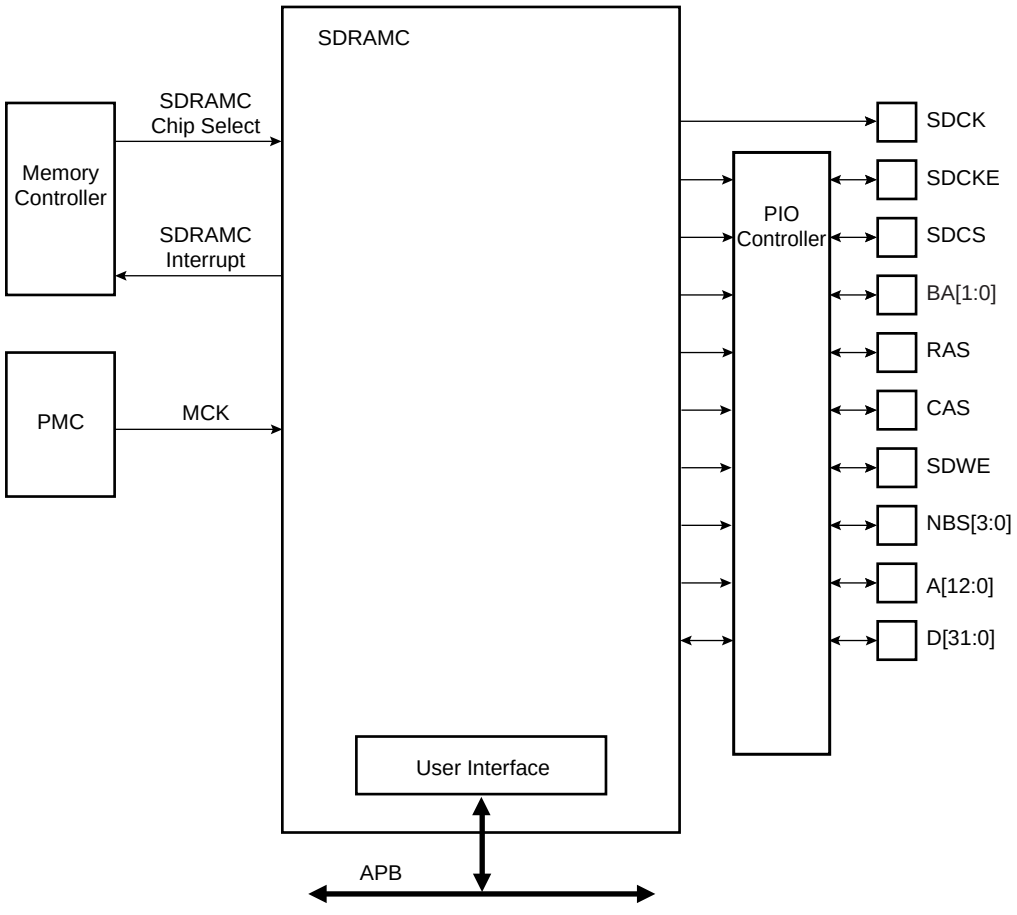
The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

The SDRAM Controller supports a read or write burst length of one location. It does not support byte Read/Write bursts or half-word write bursts. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAM Controller also supports Mobile SDRAM if VDDIO is set at 1.8V with the frequency limitation as given in the product Electrical Characteristics. However, the SDRAMC does not support the low-power extended mode register and deep power-down mode.

23.2 Block Diagram

Figure 23-1. SDRAM Controller Block Diagram



23.3 I/O Lines Description

Table 23-1. I/O Line Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output[1]	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
NBS[3:0]	Data Mask Enable Signals	Output	Low
A[12:0]	Address Bus	Output	
D[31:0]	Data Bus	I/O	

Note: 1. SDCK is tied low after reset.

23.4 Application Example

23.4.1 Software Interface

The SDRAM Controller's function is to make the SDRAM device access protocol transparent to the user. Table 23-2 to Table 23-7 illustrate the SDRAM device memory mapping therefore seen by the user in correlation with the device structure. Various configurations are illustrated.

23.4.1.1 32-bit Memory Data Bus Width

Table 23-2. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[10:0]											Column[7:0]								M[1:0]	
				Bk[1:0]			Row[10:0]											Column[8:0]								M[1:0]	
			Bk[1:0]				Row[10:0]											Column[9:0]								M[1:0]	
		Bk[1:0]					Row[10:0]											Column[10:0]								M[1:0]	

Table 23-3. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]			Row[11:0]											Column[7:0]								M[1:0]	
			Bk[1:0]				Row[11:0]											Column[8:0]								M[1:0]	
		Bk[1:0]					Row[11:0]											Column[9:0]								M[1:0]	
	Bk[1:0]						Row[11:0]											Column[10:0]								M[1:0]	

Table 23-4. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
			Bk[1:0]		Row[12:0]													Column[7:0]								M[1:0]	
		Bk[1:0]			Row[12:0]													Column[8:0]								M[1:0]	
	Bk[1:0]				Row[12:0]													Column[9:0]								M[1:0]	
Bk[1:0]					Row[12:0]													Column[10:0]								M[1:0]	

Notes: 1. M[1:0] is the byte address inside a 32-bit word.
2. Bk[1] = BA1, Bk[0] = BA0.

23.4.1.2 16-bit Memory Data Bus Width

Table 23-5. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
						Bk[1:0]		Row[10:0]										Column[7:0]									M0
					Bk[1:0]			Row[10:0]										Column[8:0]									M0
				Bk[1:0]				Row[10:0]										Column[9:0]									M0
			Bk[1:0]					Row[10:0]										Column[10:0]									M0

Table 23-6. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Bk[1:0]			Row[11:0]										Column[7:0]									M0
				Bk[1:0]				Row[11:0]										Column[8:0]									M0
			Bk[1:0]					Row[11:0]										Column[9:0]									M0
		Bk[1:0]						Row[11:0]										Column[10:0]									M0

Table 23-7. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]				Row[12:0]										Column[7:0]									M0
			Bk[1:0]					Row[12:0]										Column[8:0]									M0
		Bk[1:0]						Row[12:0]										Column[9:0]									M0
	Bk[1:0]							Row[12:0]										Column[10:0]									M0

Notes: 1. M0 is the byte address inside a 16-bit half-word.
2. Bk[1] = BA1, Bk[0] = BA0.

23.5 Product Dependencies

23.5.1 SDRAM Device Initialization

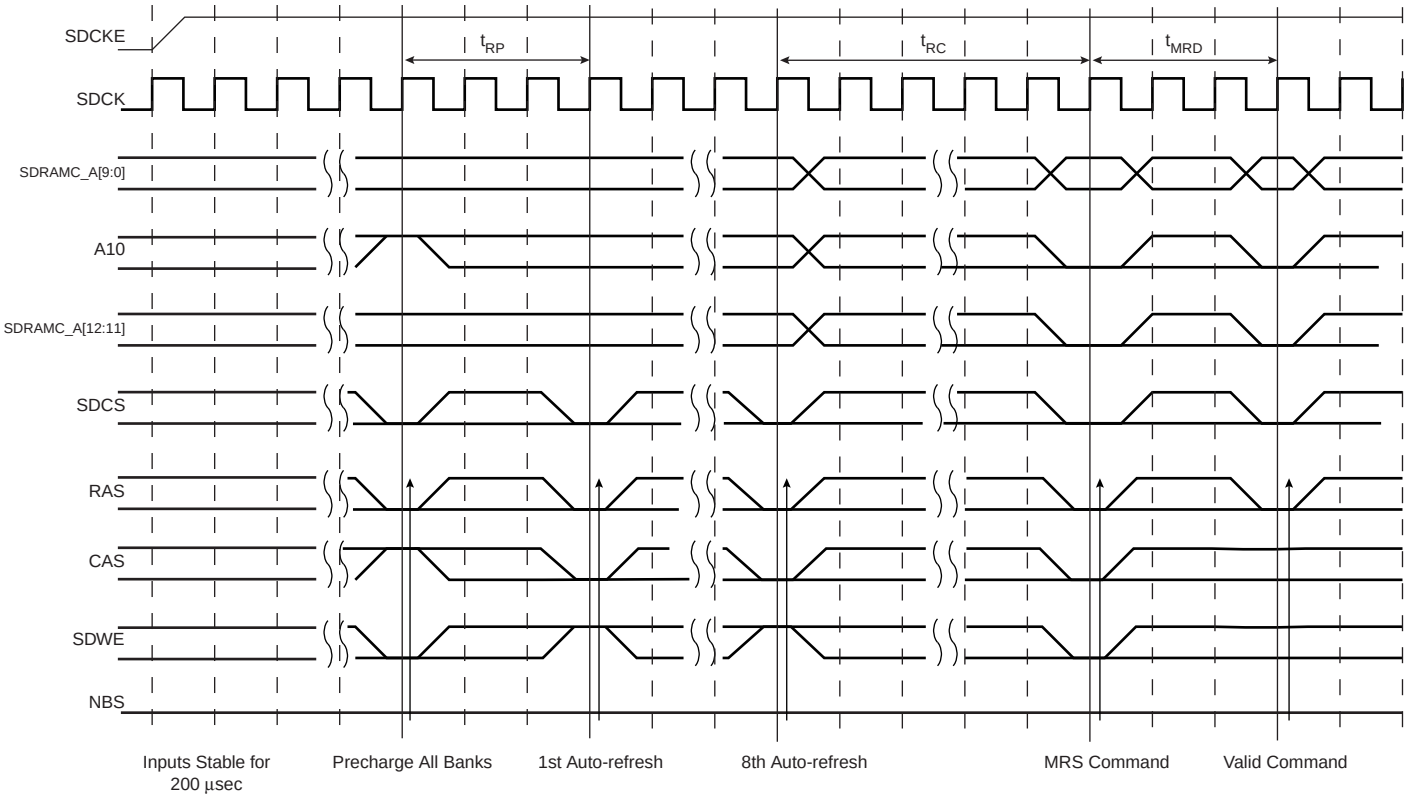
The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

1. SDRAM Characteristics must be set in the Configuration Register: asynchronous timings (TRC, TRAS,...), number of columns, rows, and CAS latency. The data bus width must be set in the Mode Register.
2. A minimum pause of 200 µs is provided to precede any signal toggle.
3. (1) A NOP command is issued to the SDRAM devices. The application must set Mode to 1 in the Mode Register and perform a write access to any SDRAM address.
4. An All Banks Precharge command is issued to the SDRAM devices. The application must set Mode to 2 in the Mode Register and perform a write access to any SDRAM address.
5. Eight auto-refresh (CBR) cycles are provided. The application must set the Mode to 4 in the Mode Register and performs a write access to any SDRAM location height times.
6. A Mode Register set (MRS) cycle is issued to program the parameters of the SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register and perform a write access to the SDRAM.
7. The application must go into Normal Mode, setting Mode to 0 in the Mode Register and performing a write access at any location in the SDRAM.
8. Write the refresh rate into the count field in the SDRAMC Refresh Timer Register. (Refresh rate = delay between refresh cycles). The SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562(15.652 µs x 100 MHz) or 781(7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

Note: 1. It is strongly recommended to respect the instructions stated in step 3 of the initialization process in order to be certain that the following commands issued by the SDRAMC will be well taken into account.

Figure 23-2. SDRAM Devices Initialization Sequence



23.5.2 I/O Lines

The pins used for interfacing the SDRAM Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the SDRAM Controller pins to their peripheral function. If I/O lines of the SDRAM Controller are not used by the application, they can be used for other purposes by the PIO Controller.

23.5.3 Interrupt

The SDRAM Controller interrupt (Refresh Error notification) is connected to the Memory Controller. This interrupt may be ORed with other System Peripheral interrupt lines and is finally provided as the System Interrupt Source (Source 1) to the AIC (Advanced Interrupt Controller).

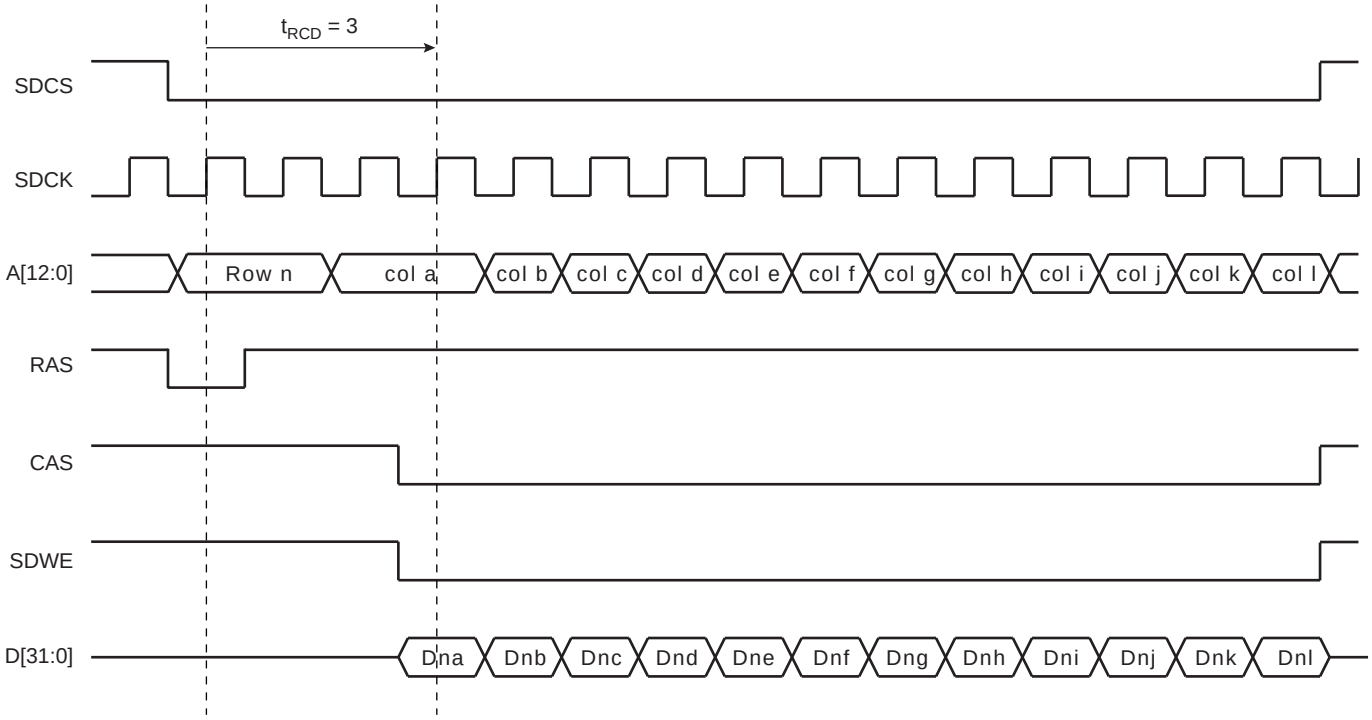
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

23.6 Functional Description

23.6.1 SDRAM Controller Write Cycle

The SDRAM Controller allows burst access or single access. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/write (t_{RCD}) commands. For definition of these timing parameters, refer to the "SDRAMC Configuration Register" on page 213. This is described in Figure 23-3 below.

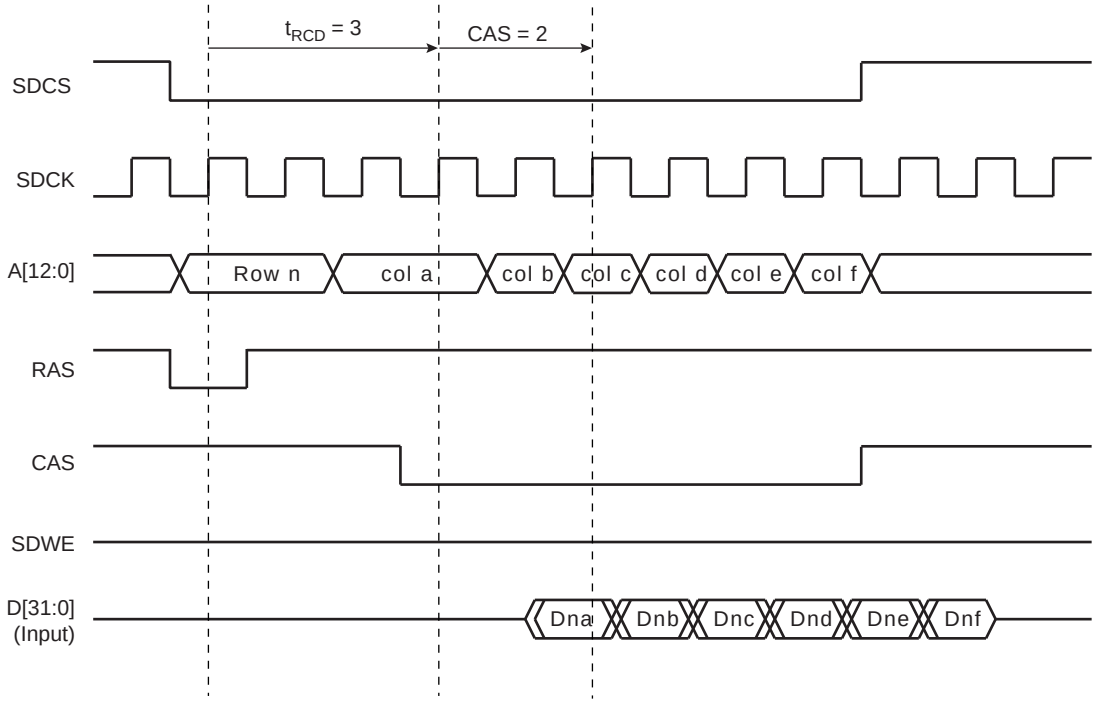
Figure 23-3. Write Burst, 32-bit SDRAM Access



23.6.2 SDRAM Controller Read Cycle

The SDRAM Controller allows burst access or single access. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential read access, reading to the SDRAM device is carried out. If the next access is a sequential read access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a read command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active (t_{RP}) command and the active/read (t_{RCD}) command, After a read command, additional wait states are generated to comply with CAS latency. The SDRAM Controller supports a CAS latency of two. For definition of these timing parameters, refer to "SDRAMC Configuration Register" on page 213. This is described in Figure 23-4 below.

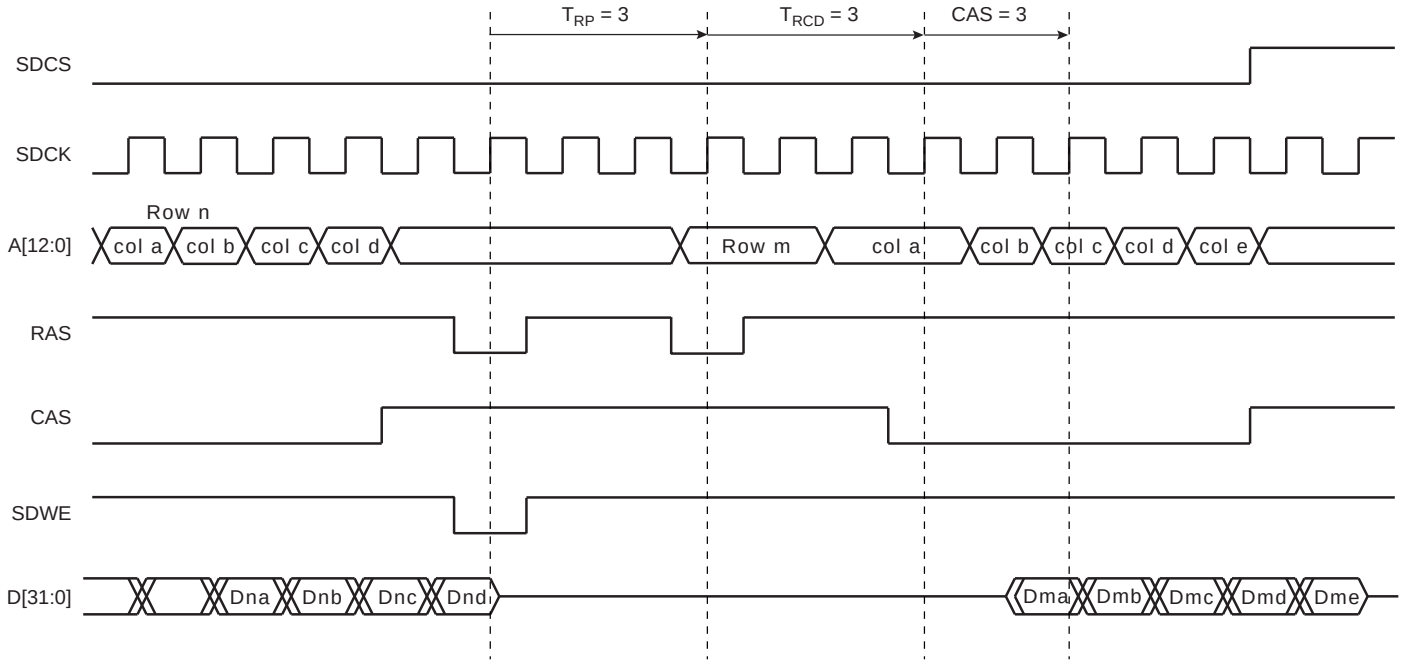
Figure 23-4. Read Burst, 32-bit SDRAM access



23.6.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAM controller generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active (t_{RP}) command and the active/read (t_{RCD}) command. This is described in Figure 23-5 below.

Figure 23-5. Read Burst with Boundary Row Access



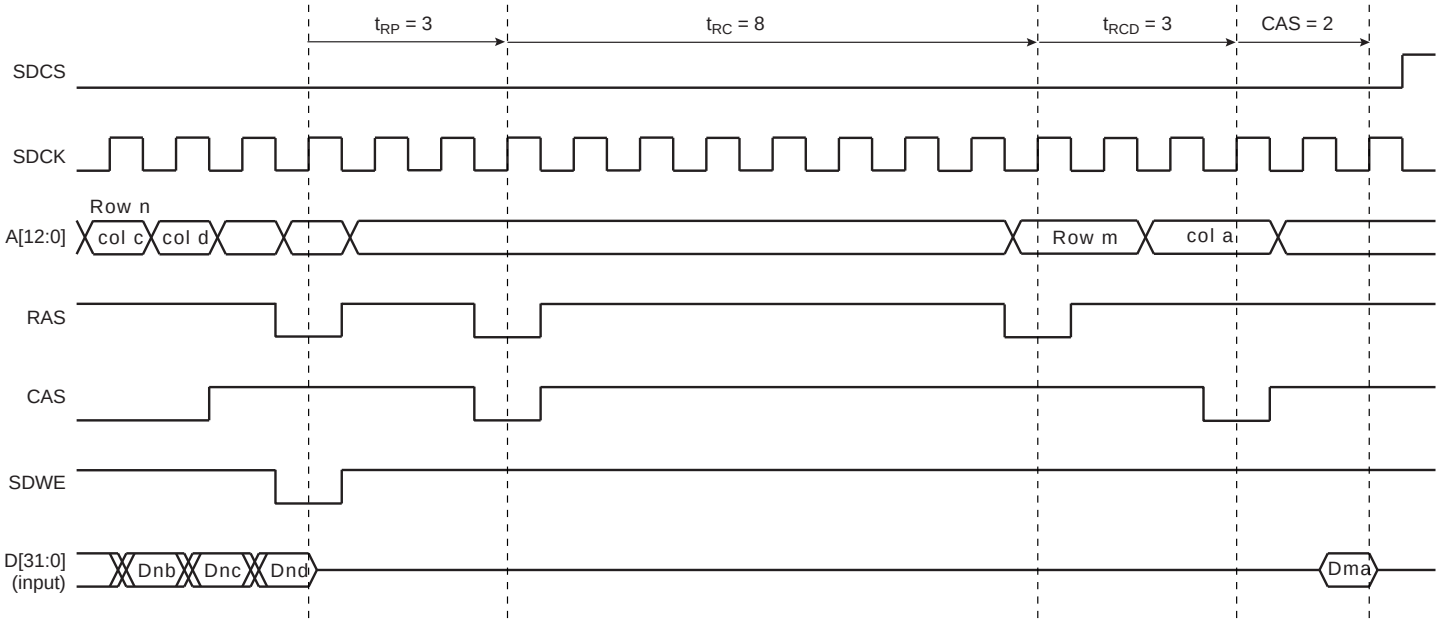
23.6.4 SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAM Controller generates these auto-refresh commands periodically. A timer is loaded with the value in the register SDRAMC_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It will be acknowledged by reading the Interrupt Status Register (SDRAMC_ISR).

When the SDRAM Controller initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave will indicate that the device is busy and the ARM BWAIT signal will be asserted. See Figure 23-6 below.

Figure 23-6. Refresh Cycle Followed by a Read Access



23.6.5 Power Management

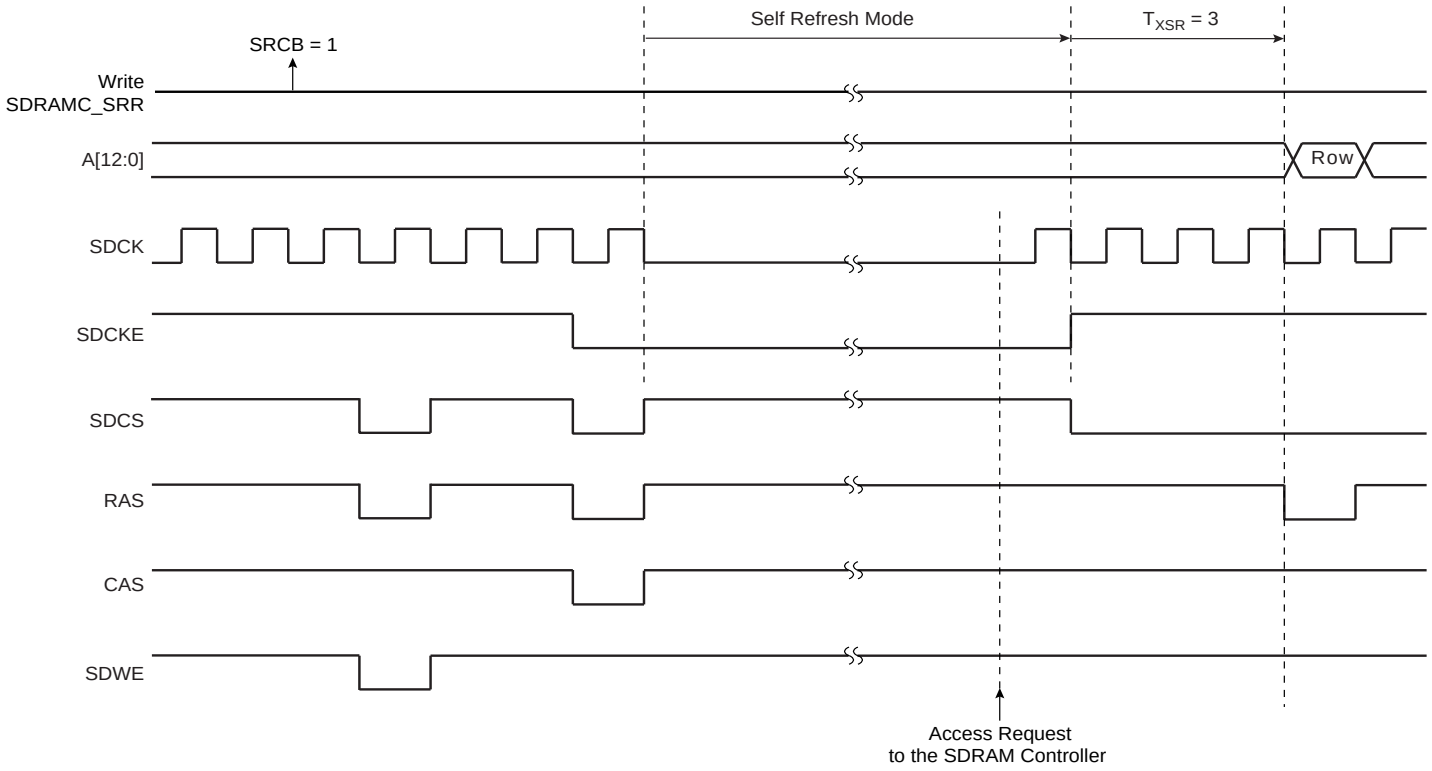
23.6.5.1 Self-refresh Mode

Self-refresh mode is used in power-down mode, i.e., when no access to the SDRAM device is possible. In this case, power consumption is very low. The mode is activated by programming the self-refresh command bit (SRCB) in SDRAMC_SRR. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAM Controller provides a sequence of commands and exits self-refresh mode, so the self-refresh command bit is disabled.

To re-activate this mode, the self-refresh command bit must be re-programmed.

The SDRAM device must remain in self-refresh mode for a minimum period of t_{RAS} and may remain in self-refresh mode for an indefinite period. This is described in Figure 23-7 below.

Figure 23-7. Self-refresh Mode Behavior



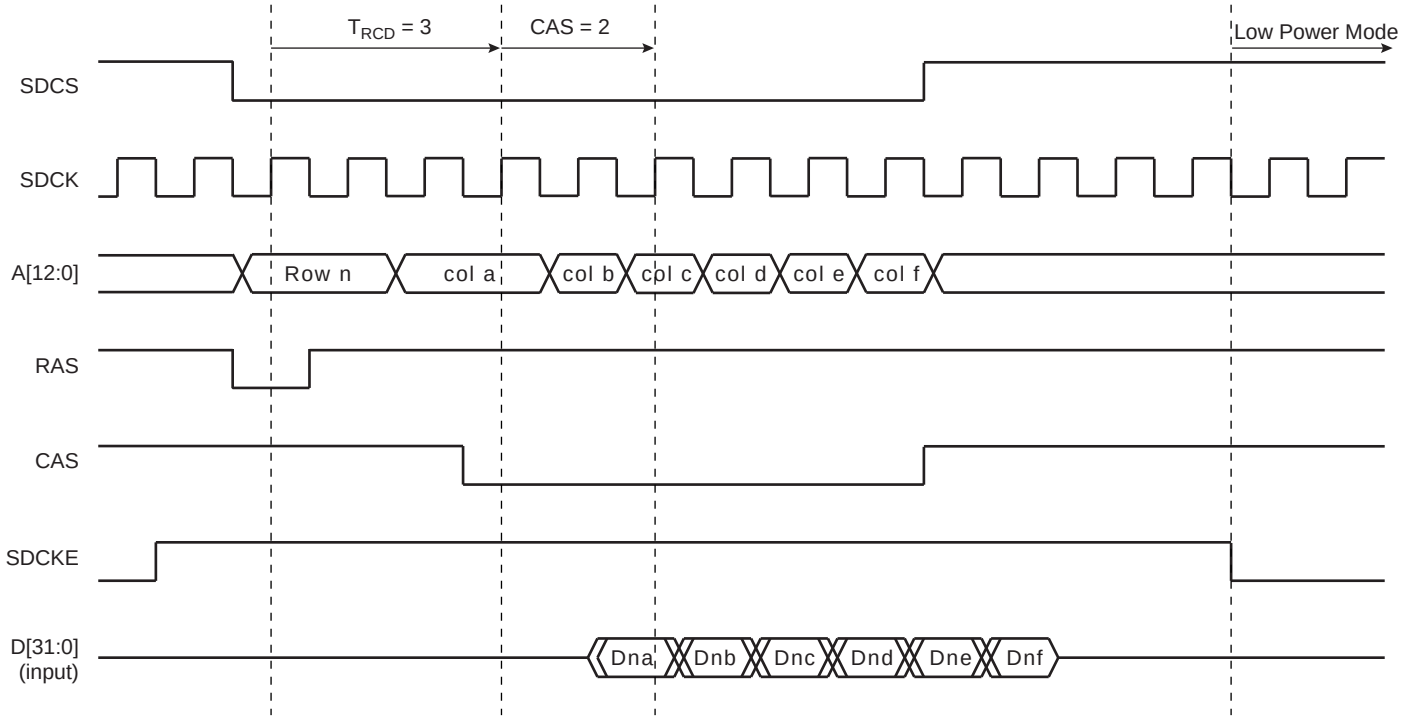
23.6.5.2 Low-power Mode

Low-power mode is used in power-down mode, i.e., when no access to the SDRAM device is possible. In this mode, power consumption is greater than in self-refresh mode. This state is similar to normal mode (No low-power mode/No self-refresh mode), but the SDCKE pin is low and the input and output buffers are deactivated as soon as the SDRAM device is no longer accessible. In contrast to self-refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed in this mode, the SDRAM Controller carries out the refresh operation. In order to exit low-power mode, a NOP command is required. The exit procedure is faster than in self-refresh mode.

When self-refresh mode is enabled, it is recommended to avoid enabling low-power mode. When low-power mode is enabled, it is recommended to avoid enabling self-refresh mode.

This is described in Figure 23-8 below.

Figure 23-8. Low-power Mode Behavior



23.7 SDRAM Controller (SDRAMC) User Interface

Table 23-8. SDRAM Controller Memory Mapping

Offset	Register	Name	Access	Reset State
0x00	SDRAMC Mode Register	SDRAMC_MR	Read/Write	0x00000010
0x04	SDRAMC Refresh Timer Register	SDRAMC_TR	Read/Write	0x00000800
0x08	SDRAMC Configuration Register	SDRAMC_CR	Read/Write	0x2A99C140
0x0C	SDRAMC Self Refresh Register	SDRAMC_SRR	Write-only	–
0x10	SDRAMC Low Power Register	SDRAMC_LPR	Read/Write	0x0
0x14	SDRAMC Interrupt Enable Register	SDRAMC_IER	Write-only	–
0x18	SDRAMC Interrupt Disable Register	SDRAMC_IDR	Write-only	–
0x1C	SDRAMC Interrupt Mask Register	SDRAMC_IMR	Read-only	0x0
0x20	SDRAMC Interrupt Status Register	SDRAMC_ISR	Read-only	0x0
0x24 - 0xFC	Reserved	–	–	–

23.7.1 SDRAMC Mode Register

Name: SDRAMC_MR

Access: Read/Write

Reset Value: 0x00000010

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	DBW	MODE			

- **MODE: SDRAMC Command Mode**

This field defines the command issued by the SDRAM Controller when the SDRAM device is accessed.

MODE				Description
0	0	0	0	Normal mode. Any access to the SDRAM is decoded normally.
0	0	0	1	The SDRAM Controller issues a NOP command when the SDRAM device is accessed regardless of the cycle.
0	0	1	0	The SDRAM Controller issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle.
0	0	1	1	The SDRAM Controller issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. The address offset with respect to the SDRAM device base address is used to program the Mode Register. For instance, when this mode is activated, an access to the “SDRAM_Base + offset” address generates a “Load Mode Register” command with the value “offset” written to the SDRAM device Mode Register.
0	1	0	0	The SDRAM Controller issues a “Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued.

- **DBW: Data Bus Width**

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

23.7.2 SDRAMC Refresh Timer Register

Name: SDRAMC_TR

Access: Read/Write

Reset Value: 0x00000800

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

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

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

7	6	5	4	3	2	1	0
COUNT							

- **COUNT: SDRAMC Refresh Timer Count**

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The value to be loaded depends on the SDRAMC clock frequency (MCK: Master Clock), the refresh rate of the SDRAM device and the refresh burst length where 15.6 µs per row is a typical value for a burst of length one.

To refresh the SDRAM device even if the reset value is not equal to 0, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

23.7.3 SDRAMC Configuration Register

Name: SDRAMC_CR

Access: Read/Write

Reset Value: 0x2A99C140

31	30	29	28	27	26	25	24
–	TXSR				TRAS		
23	22	21	20	19	18	17	16
TRAS	TRCD				TRP		
15	14	13	12	11	10	9	8
TRP	TRC				TWR		
7	6	5	4	3	2	1	0
TWR	CAS		NB	NR		NC	

- **NC: Number of Column Bits**

Reset value is 8 column bits.

NC		Column Bits
0	0	8
0	1	9
1	0	10
1	1	11

- **NR: Number of Row Bits**

Reset value is 11 row bits.

NR		Row Bits
0	0	11
0	1	12
1	0	13
1	1	Reserved

- **NB: Number of Banks**

Reset value is two banks.

NB	Number of Banks
0	2
1	4

- **CAS: CAS Latency**

Reset value is two cycles.

In the SDRAMC, only a CAS latency of two cycles is managed. In any case, another value must be programmed.

CAS		CAS Latency (Cycles)
0	0	Reserved
0	1	Reserved
1	0	2
1	1	Reserved

- **TWR: Write Recovery Delay**

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 2 and 15.

If TWR is less than or equal to 2, two clock periods are inserted by default.

- **TRC: Row Cycle Delay**

Reset value is eight cycles.

This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 2 and 15.

If TRC is less than or equal to 2, two clock periods are inserted by default.

- **TRP: Row Precharge Delay**

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 2 and 15.

If TRP is less than or equal to 2, two clock periods are inserted by default.

- **TRCD: Row to Column Delay**

Reset value is three cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 2 and 15.

If TRCD is less than or equal to 2, two clock periods are inserted by default.

- **TRAS: Active to Precharge Delay**

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 2 and 15.

If TRAS is less than or equal to 2, two clock periods are inserted by default.

- **TXSR: Exit Self Refresh to Active Delay**

Reset value is five cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 1/2 and 15.5.

If TXSR is equal to 0, 1/2 clock period is inserted by default.

23.7.4 SDRAMC Self-refresh Register

Name: SDRAMC_SRR

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
–	–	–	–	–	–	–	SRCB

- **SRCB: Self-refresh Command Bit**

0: No effect.

1: The SDRAM Controller issues a self-refresh command to the SDRAM device, the SDCK clock is inactivated and the SDCKE signal is set low. The SDRAM device leaves self-refresh mode when accessed again.

23.7.5 SDRAMC Low-power Register

Name: SDRAMC_LPR

Access: Read/Write

Reset Value: 0x0

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

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

0: The SDRAM Controller low-power feature is inhibited: no low-power command is issued to the SDRAM device.

1: The SDRAM Controller issues a low-power command to the SDRAM device after each burst access, the SDCKE signal is set low. The SDRAM device will leave low-power mode when accessed and enter it after the access.

23.7.6 SDRAMC Interrupt Enable Register

Name: SDRAMC_IER

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
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Enables the refresh error interrupt.

23.7.7 SDRAMC Interrupt Disable Register

Name: SDRAMC_IDR

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
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Disables the refresh error interrupt.

23.7.8 SDRAMC Interrupt Mask Register

Name: SDRAMC_IMR

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
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

23.7.9 SDRAMC Interrupt Status Register

Name: SDRAMC_ISR

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
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.

24. Error Corrected Code Controller (ECC)

24.1 Overview

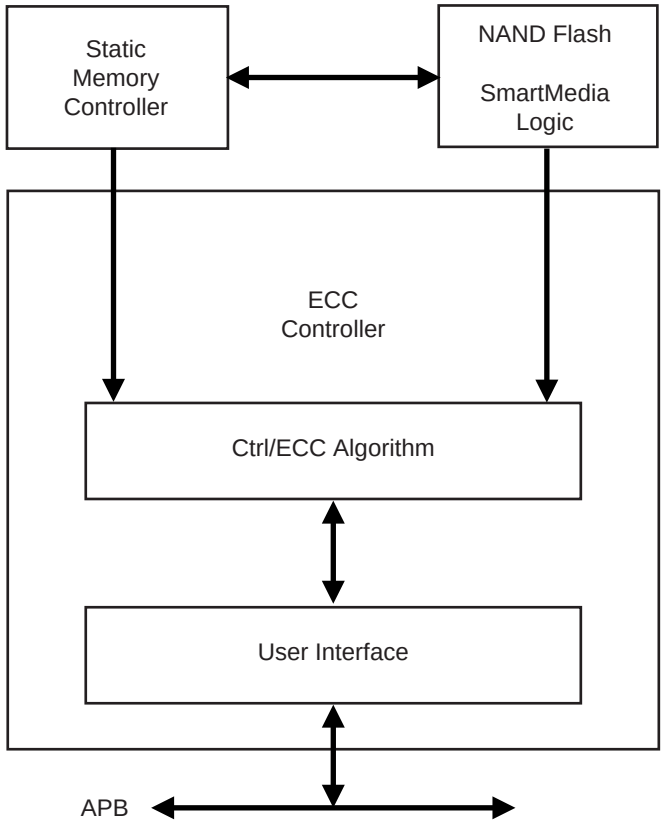
NAND Flash/SmartMedia devices contain by default invalid blocks which have one or more invalid bits. Over the NAND Flash/SmartMedia lifetime, additional invalid blocks may occur which can be detected/corrected by ECC code.

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

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

24.2 Block Diagram

Figure 24-1. Block Diagram



24.3 Functional Description

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

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

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

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

24.3.1 Write Access

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

24.3.2 Read Access

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

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

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

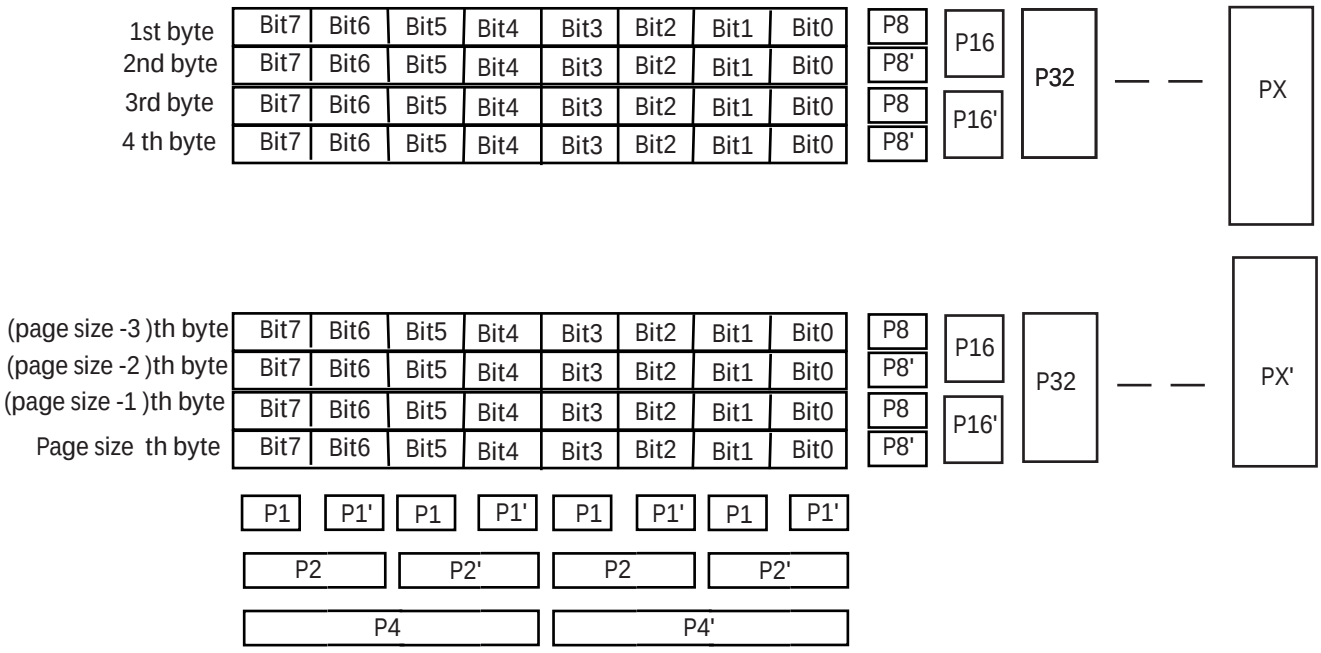
- No error: XOR between the ECC computation and the ECC code stored at the end of the NAND Flash or SmartMedia page is equal to 0. No error flags in the ECC Status Register (ECC_SR).
- Recoverable error: Only the RECERR flag in the ECC Status register (ECC_SR) is set. The corrupted word offset in the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

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

For Single-bit Error Correction and Double-bit Error Detection (SEC-DED) hsiao code is used. 32-bit ECC is generated in order to perform one bit correction per 512/1024/2048/4096 8- or 16-

bit words. Of the 32 ECC bits, 26 bits are for line parity and 6 bits are for column parity. They are generated according to the schemes shown in Figure 24-2 and Figure 24-3.

Figure 24-2. Parity Generation for 512/1024/2048/4096 8-bit Words1



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

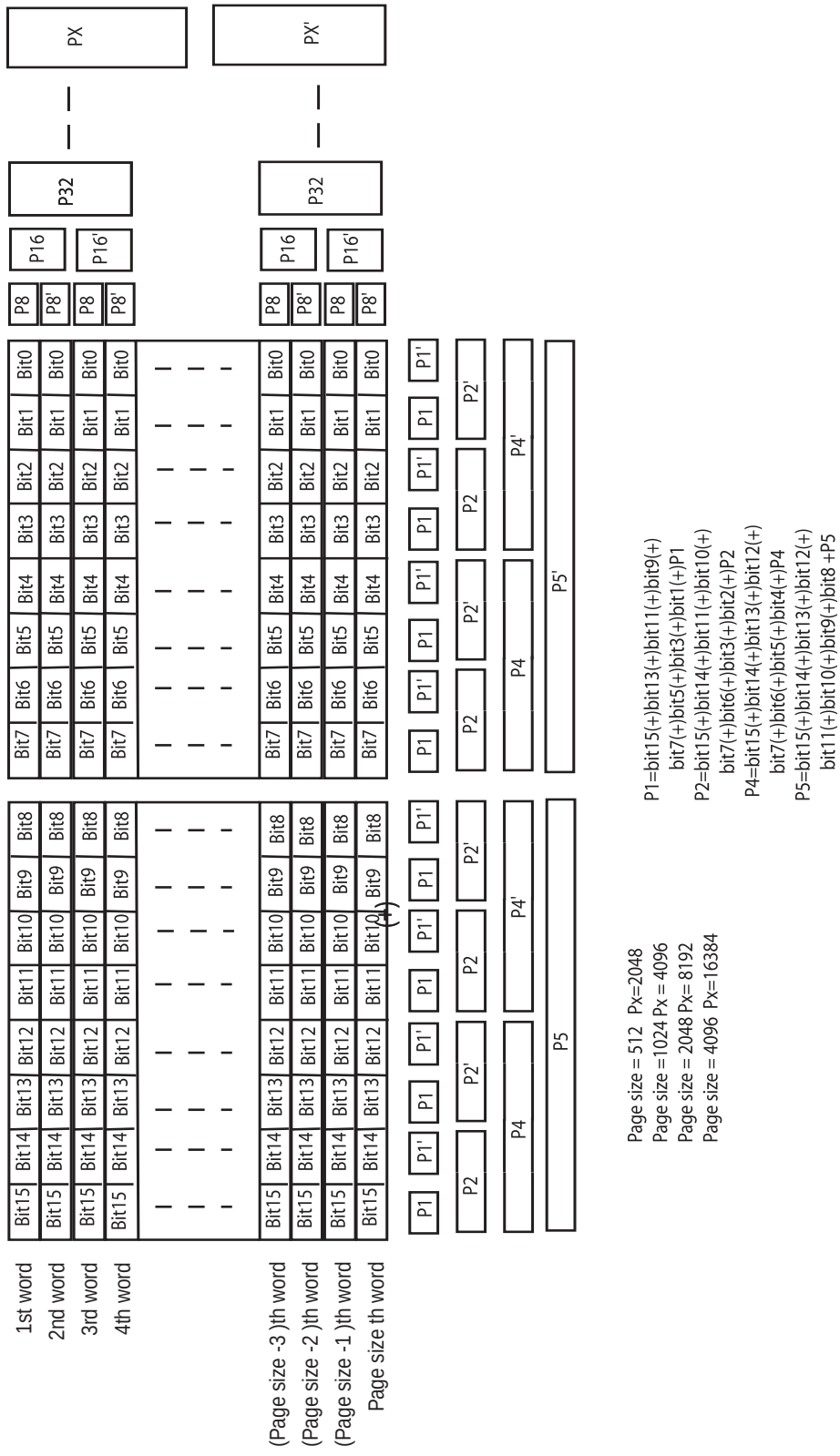
P1=bit7(+)bit5(+)bit3(+)bit1(+)P1
P2=bit7(+)bit6(+)bit3(+)bit2(+)P2
P4=bit7(+)bit6(+)bit5(+)bit4(+)P4
P1'=bit6(+)bit4(+)bit2(+)bit0(+)P1'
P2'=bit5(+)bit4(+)bit1(+)bit0(+)P2'
P4'=bit7(+)bit6(+)bit5(+)bit4(+)P4'

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

```
Page size = 2^n

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

Figure 24-3. Parity Generation for 512/1024/2048/4096 16-bit Words



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

Page size = 2^n

```
for i =0 to n
begin
 for (j = 0 to page_size_word)
 begin
  if(j[i] ==1)
   P[2^(i+3)]= bit15(+)bit14(+)bit13(+)bit12(+)
               bit11(+)bit10(+)bit9(+)bit8(+)
               bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
               bit2(+)bit1(+)bit0(+)P[2^(n+3)]
  else
   P[2^(i+3)]’=bit15(+)bit14(+)bit13(+)bit12(+)
               bit11(+)bit10(+)bit9(+)bit8(+)
               bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
               bit2(+)bit1(+)bit0(+)P[2^(i+3)]'
  end
end
```

24.4 ECC User Interface

Table 24-1. ECC Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	ECC Control Register	ECC_CR	Write-only	0x0
0x04	ECC Mode Register	ECC_MR	Read/Write	0x0
0x8	ECC Status Register	ECC_SR	Read-only	0x0
0x0C	ECC Parity Register	ECC_PR	Read-only	0x0
0x10	ECC NParity Register	ECC_NPR	Read-only	0x0
0x14 - 0xFC	Reserved	–	–	–

24.4.1 ECC Control Register

Name: ECC_CR

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
–	–	–	–	–	–	–	RST

- **RST: RESET Parity**

Provides reset to current ECC by software.

1: Reset sECC Parity and ECC NParity register

0: No effect

24.4.2 ECC Mode Register

Name: ECC_MR

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
–	–	–	–	–	–	PAGESIZE	

- **PAGESIZE: Page Size**

This field defines the page size of the NAND Flash device.

Page Size	Description
00	528 words
01	1056 words
10	2112 words
11	4224 words

A word has a value of 8 bits or 16 bits, depending on the NAND Flash or Smartmedia memory organization.

24.4.3 ECC Status Register

Name: ECC_SR

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
–	–	–	–	–	MULERR	ECCERR	RECERR

- **RECERR: Recoverable Error**

0 = No Errors Detected

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

- **ECCERR: ECC Error**

0 = No Errors Detected

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

Read both ECC Parity and ECC NParity register, the error occurred at the location which contains a 1 in the least significant 16 bits.

- **MULERR: Multiple Error**

0 = No Multiple Errors Detected

1 = Multiple Errors Detected

24.4.4 ECC Parity Register

Name: ECC_PR

Access: Read-only

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

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

- **BITADDR**

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **WORDADDR**

During a page read, this value contains the word address (8-bit or 16-bit word depending on the memory plane organization) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

24.4.5 ECC NParity Register

Name: ECC_NPR

Access: Read-only

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

- **NPARITY:**

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

25. AT91SAM Boot Program

25.1 Overview

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

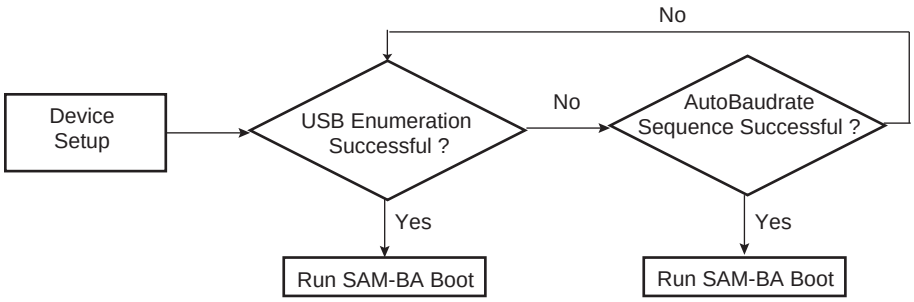
First, it initializes the Debug Unit serial port (DBGU) and the USB Device Port.

SAM-BA® Boot is then executed. It waits for transactions either on the USB device, or on the DBGU serial port.

25.2 Flow Diagram

The Boot Program implements the algorithm in Figure 25-1.

Figure 25-1. Boot Program Algorithm Flow Diagram



25.3 Device Initialization

Initialization follows the steps described below:

1. FIQ initialization
1. Stack setup for ARM supervisor mode
2. Setup the Embedded Flash Controller
3. External Clock detection
4. Main oscillator frequency detection if no external clock detected
5. Switch Master Clock on Main Oscillator
6. Copy code into SRAM
7. C variable initialization
8. PLL setup: PLL is initialized to generate a 48 MHz clock necessary to use the USB Device
9. Disable of the Watchdog and enable of the user reset
10. Initialization of the USB Device Port
11. Jump to SAM-BA Boot sequence (see “SAM-BA Boot” on page 230)

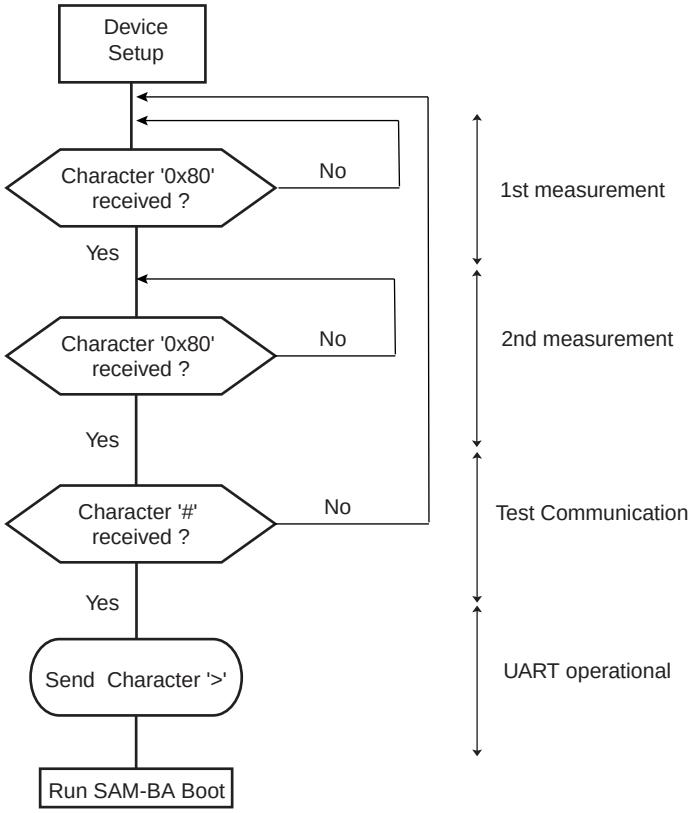
25.4 SAM-BA Boot

The SAM-BA boot principle is to:

– Check if USB Device enumeration has occurred

– Check if the AutoBaudrate sequence has succeeded (see Figure 25-2)

Figure 25-2. AutoBaudrate Flow Diagram



– Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as in Table 25-1.

Table 25-1. Commands Available through the SAM-BA Boot

Command	Action	Argument(s)	Example
O	write a byte	Address, Value#	**O**200001,CA#
o	read a byte	Address,#	**o**200001,#
H	write a half word	Address, Value#	**H**200002,CAFE#
h	**read a half word**	**Address,#**	**h**200002,#
W	**write a word**	**Address, Value#**	**W**200000,CAFEDECA#
w	**read a word**	**Address,#**	**w**200000,#
S	**send a file**	**Address,#**	**S**200000,#
R	**receive a file**	**Address, NbOfBytes#**	**R**200000,1234#
G	**go**	**Address#**	**G**200200#
V	**display version**	**No argument**	**V**#

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

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

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

25.4.1 DBGU Serial Port

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

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

the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory to work.

25.4.2 Xmodem Protocol

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

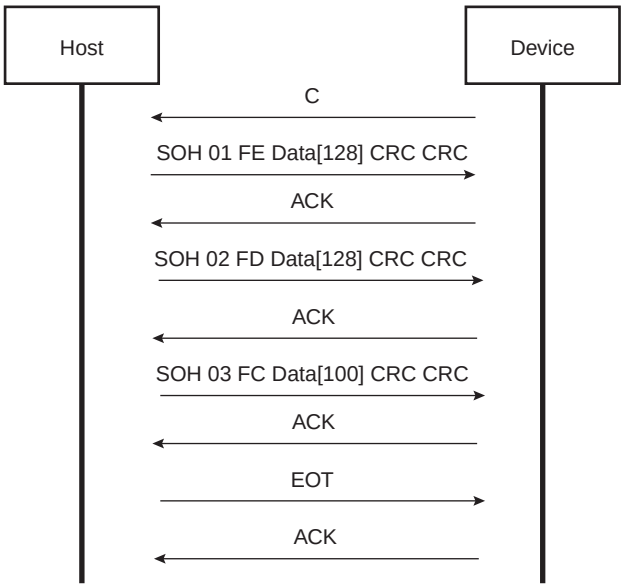
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

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

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

Figure 25-3 shows a transmission using this protocol.

Figure 25-3. Xmodem Transfer Example



25.4.3 USB Device Port

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

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

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

25.4.3.1 Enumeration Process

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

Table 25-2. 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 25-3. Handled Class Requests

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

Unhandled requests are STALLed.

25.4.3.2 Communication Endpoints

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint and endpoint 2 is a 64-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through the endpoint 1. If required, the message is split by the host into several data payloads by the host driver.

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

25.5 Hardware and Software Constraints

- SAM-BA boot copies itself in the SRAM and uses a block of internal SRAM for variables and stacks. The remaining available size for the user code is 24576 bytes for SAM7SE512/256, 8192 bytes for SAM7SE32.
  - The SAM7SE512/256 user area extends from address 0x202000 to address 0x208000.
  - The SAM7SE32 user area extends from address 0x201400 to address 0x201C00.
- USB requirements:
  - 18.432 MHz Quartz

Table 25-4. Pins Driven during Boot Program Execution

Peripheral	Pin	PIO Line
DBGU	DRXD	PA9
DBGU	DTXD	PA10

26. Peripheral DMA Controller (PDC)

26.1 Overview

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals such as the UART, USART, SSC, SPI, MCI and the on- and off-chip memories. Using the Peripheral DMA Controller avoids processor intervention and removes the processor interrupt-handling overhead. This significantly reduces the number of clock cycles required for a data transfer and, as a result, improves the performance of the microcontroller and makes it more power efficient.

The PDC channels are implemented in pairs, each pair being dedicated to a particular peripheral. One channel in the pair is dedicated to the receiving channel and one to the transmitting channel of each UART, USART, SSC and SPI.

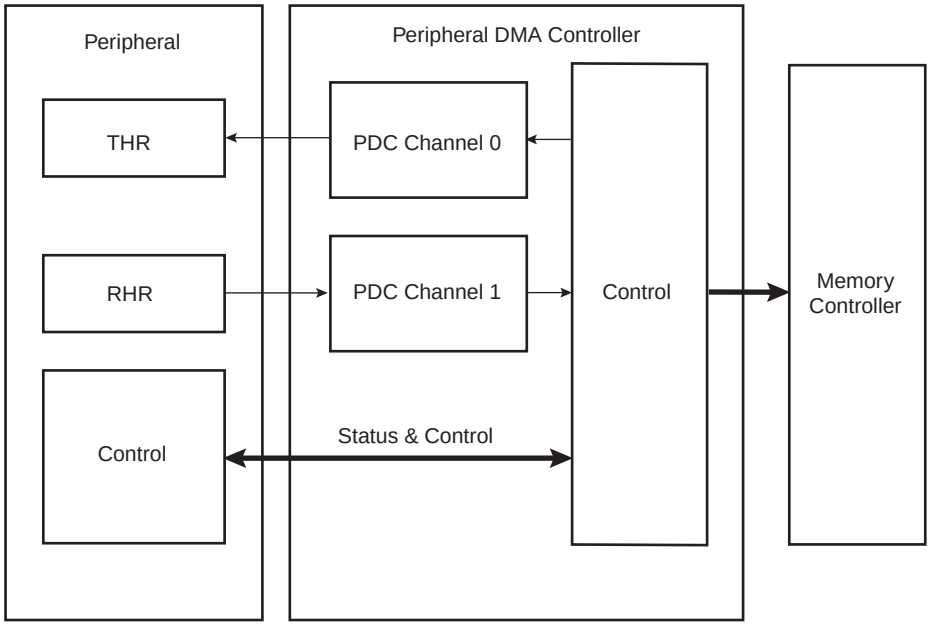
The user interface of a PDC channel is integrated in the memory space of each peripheral. It contains:

- two 32-bit memory pointer registers (send and receive)
- two 16-bit transfer count register (send and receive)
- two 32-bit register for next memory pointer (send and receive)
- two 16-bit register for next transfer count (send and receive)

The peripheral triggers PDC transfers using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the corresponding peripheral.

26.2 Block Diagram

Figure 26-1. Block Diagram



26.3 Functional Description

26.3.1 Configuration

The PDC channels user interface enables the user to configure and control the data transfers for each channel. The user interface of a PDC channel is integrated into the user interface of the peripheral (offset 0x100), which it is related to.

Per peripheral, it contains four 32-bit Pointer Registers (RPR, RNPR, TPR, and TNPR) and four 16-bit Counter Registers (RCR, RNCR, TCR, and TNCR).

The size of the buffer (number of transfers) is configured in an internal 16-bit transfer counter register, and it is possible, at any moment, to read the number of transfers left for each channel.

The memory base address is configured in a 32-bit memory pointer by defining the location of the first address to access in the memory. It is possible, at any moment, to read the location in memory of the current transfer and the number of remaining transfers. The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in PDC Transfer Control Register. These control bits enable reading the pointer and counter registers safely without any risk of their changing between both reads.

The PDC sends status flags to the peripheral visible in its status-register (ENDRX, ENDTX, RXBUFF, and TXBUFE).

ENDRX flag is set when the PERIPH_RCR register reaches zero.

RXBUFF flag is set when both PERIPH_RCR and PERIPH_RNCR reach zero.

ENDTX flag is set when the PERIPH_TCR register reaches zero.

TXBUFE flag is set when both PERIPH_TCR and PERIPH_TNCR reach zero.

These status flags are described in the peripheral status register.

26.3.2 Memory Pointers

Each peripheral is connected to the PDC by a receiver data channel and a transmitter data channel. Each channel has an internal 32-bit memory pointer. Each memory pointer points to a location anywhere in the memory space (on-chip memory or external bus interface memory).

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented by 1, 2 or 4, respectively for peripheral transfers.

If a memory pointer is reprogrammed while the PDC is in operation, the transfer address is changed, and the PDC performs transfers using the new address.

26.3.3 Transfer Counters

There is one internal 16-bit transfer counter for each channel used to count the size of the block already transferred by its associated peripheral. These counters are decremented after each data transfer. When the counter reaches zero, the transfer is complete and the PDC stops transferring data.

If the Next Counter Register is equal to zero, the PDC disables the trigger while activating the related peripheral end flag.

If the counter is reprogrammed while the PDC is operating, the number of transfers is updated and the PDC counts transfers from the new value.

Programming the Next Counter/Pointer registers chains the buffers. The counters are decremented after each data transfer as stated above, but when the transfer counter reaches zero, the values of the Next Counter/Pointer are loaded into the Counter/Pointer registers in order to re-enable the triggers.

For each channel, two status bits indicate the end of the current buffer (ENDRX, ENDTX) and the end of both current and next buffer (RXBUFF, TXBUFE). These bits are directly mapped to the peripheral status register and can trigger an interrupt request to the AIC.

The peripheral end flag is automatically cleared when one of the counter-registers (Counter or Next Counter Register) is written.

Note: When the Next Counter Register is loaded into the Counter Register, it is set to zero.

26.3.4 Data Transfers

The peripheral triggers PDC transfers using transmit (TXRDY) and receive (RXRDY) signals.

When the peripheral receives an external character, it sends a Receive Ready signal to the PDC which then requests access to the system bus. When access is granted, the PDC starts a read of the peripheral Receive Holding Register (RHR) and then triggers a write in the memory.

After each transfer, the relevant PDC memory pointer is incremented and the number of transfers left is decremented. When the memory block size is reached, a signal is sent to the peripheral and the transfer stops.

The same procedure is followed, in reverse, for transmit transfers.

26.3.5 Priority of PDC Transfer Requests

The Peripheral DMA Controller handles transfer requests from the channel according to priorities fixed for each product.These priorities are defined in the product datasheet.

If simultaneous requests of the same type (receiver or transmitter) occur on identical peripherals, the priority is determined by the numbering of the peripherals.

If transfer requests are not simultaneous, they are treated in the order they occurred. Requests from the receivers are handled first and then followed by transmitter requests.

26.4 Peripheral DMA Controller (PDC) User Interface

Table 26-1. Register Mapping

Offset	Register	Register Name	Read/Write	Reset
0x100	Receive Pointer Register	PERIPH[(1)]_RPR	Read/Write	0x0
0x104	Receive Counter Register	PERIPH_RCR	Read/Write	0x0
0x108	Transmit Pointer Register	PERIPH_TPR	Read/Write	0x0
0x10C	Transmit Counter Register	PERIPH_TCR	Read/Write	0x0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read/Write	0x0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read/Write	0x0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read/Write	0x0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read/Write	0x0
0x120	PDC Transfer Control Register	PERIPH_PTCR	Write-only	-
0x124	PDC Transfer Status Register	PERIPH_PTSR	Read-only	0x0

Note: 1. PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SSC, SPI, MCI, etc).

26.4.1 PDC Receive Pointer Register

Name: PERIPH_RPR

Access: Read/Write

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

- **RXPTR: Receive Pointer Address**

Address of the next receive transfer.

26.4.2 PDC Receive Counter Register

Name: PERIPH_RCR

Access: Read/Write

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

- **RXCTR: Receive Counter Value**

Number of receive transfers to be performed.

26.4.3 PDC Transmit Pointer Register

Name: PERIPH_TPR

Access: Read/Write

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

- **TXPTR: Transmit Pointer Address**

Address of the transmit buffer.

26.4.4 PDC Transmit Counter Register

Name: PERIPH_TCR

Access: Read/Write

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

- **TXCTR: Transmit Counter Value**

TXCTR is the size of the transmit transfer to be performed. At zero, the peripheral data transfer is stopped.

26.4.5 PDC Receive Next Pointer Register

Name: PERIPH_RNPR

Access: Read/Write

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

- **RXNPTR: Receive Next Pointer Address**

RXNPTR is the address of the next buffer to fill with received data when the current buffer is full.

26.4.6 PDC Receive Next Counter Register

Name: PERIPH_RNCR

Access: Read/Write

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

- **RXNCR: Receive Next Counter Value**

RXNCR is the size of the next buffer to receive.

26.4.7 PDC Transmit Next Pointer Register

Name: PERIPH_TNPR

Access: Read/Write

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

- **TXNPTR: Transmit Next Pointer Address**

TXNPTR is the address of the next buffer to transmit when the current buffer is empty.

26.4.8 PDC Transmit Next Counter Register

Name: PERIPH_TNCR

Access: Read/Write

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

- **TXNCR: Transmit Next Counter Value**

TXNCR is the size of the next buffer to transmit.

26.4.9 PDC Transfer Control Register

Name: PERIPH_PTCR

Access: Write-only

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

- **RXTEN: Receiver Transfer Enable**

0 = No effect.

1 = Enables the receiver PDC transfer requests if RXTDIS is not set.

- **RXTDIS: Receiver Transfer Disable**

0 = No effect.

1 = Disables the receiver PDC transfer requests.

- **TXTEN: Transmitter Transfer Enable**

0 = No effect.

1 = Enables the transmitter PDC transfer requests.

- **TXTDIS: Transmitter Transfer Disable**

0 = No effect.

1 = Disables the transmitter PDC transfer requests.

26.4.10 PDC Transfer Status Register

Name: PERIPH_PTSR

Access: Read-only

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

- **RXTEN: Receiver Transfer Enable**

0 = Receiver PDC transfer requests are disabled.

1 = Receiver PDC transfer requests are enabled.

- **TXTEN: Transmitter Transfer Enable**

0 = Transmitter PDC transfer requests are disabled.

1 = Transmitter PDC transfer requests are enabled.

27. Advanced Interrupt Controller (AIC)

27.1 Overview

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

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

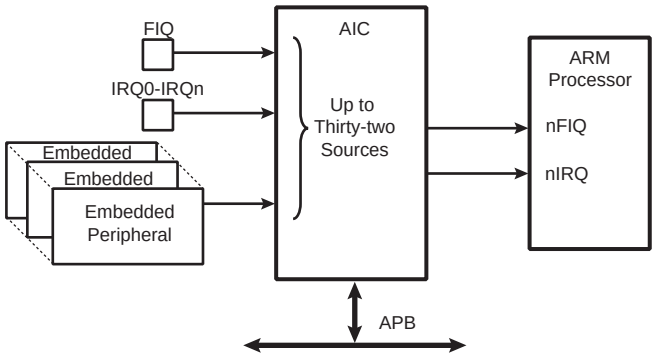
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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

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

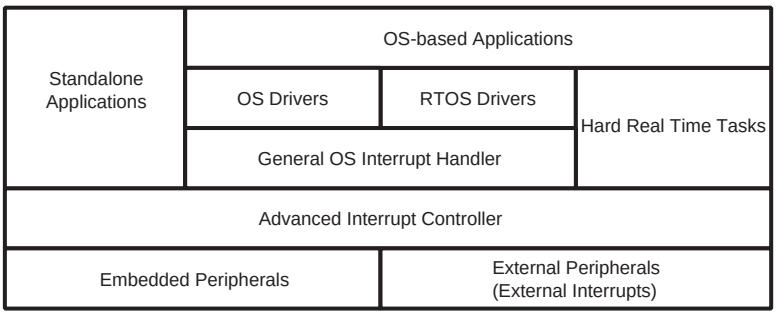
27.2 Block Diagram

Figure 27-1. Block Diagram



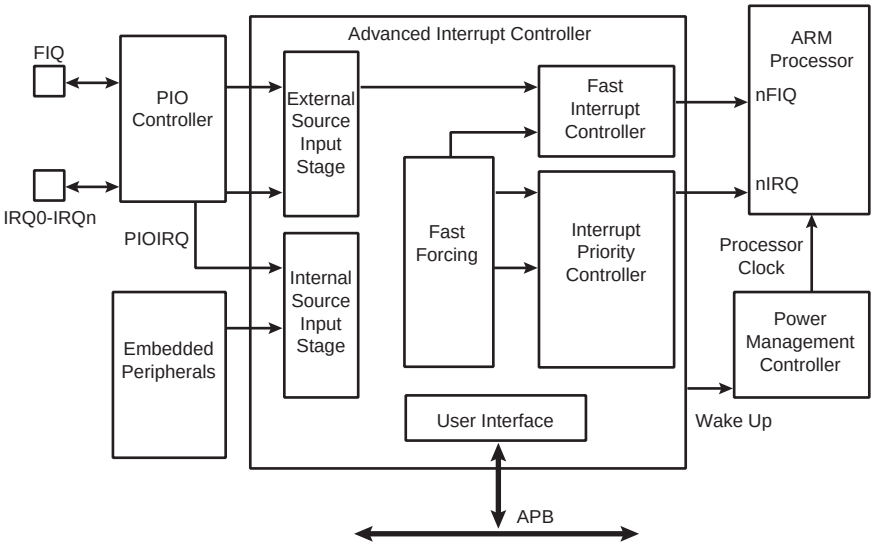
27.3 Application Block Diagram

Figure 27-2. Description of the Application Block



27.4 AIC Detailed Block Diagram

Figure 27-3. AIC Detailed Block Diagram



27.5 I/O Line Description

Table 27-1. I/O Line Description

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

27.6 Product Dependencies

27.6.1 I/O Lines

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

27.6.2 Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the ARM processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

27.6.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines, such as the System Timer, the Real Time Clock, the Power Management Controller and the Memory Controller. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

27.7 Functional Description

27.7.1 Interrupt Source Control

27.7.1.1 Interrupt Source Mode

The Advanced Interrupt Controller independently programs each interrupt source. The SRCTYPE field of the corresponding AIC_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

27.7.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the AIC_IMR register. A disabled interrupt does not affect servicing of other interrupts.

27.7.1.3 Interrupt Clearing and Setting

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the AIC_ISCR and AIC_ICCR registers. Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the “memorization” circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC-implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current interrupt is affected by this operation. (See “Priority Controller” on page 252.) The automatic clear reduces the operations required by the interrupt service routine entry code to reading the AIC_IVR. Note that the automatic interrupt clear is disabled if the interrupt source has the Fast Forcing feature enabled as it is considered uniquely as a FIQ source. (For further details, See “Fast Forcing” on page 256.)

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

27.7.1.4 Interrupt Status

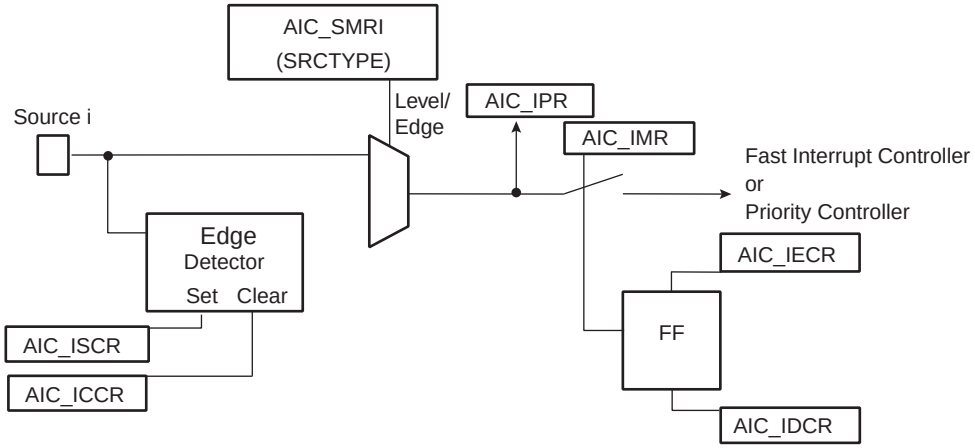
For each interrupt, the AIC operation originates in AIC_IPR (Interrupt Pending Register) and its mask in AIC_IMR (Interrupt Mask Register). AIC_IPR enables the actual activity of the sources, whether masked or not.

The AIC_ISR register reads the number of the current interrupt (see “Priority Controller” on page 252) 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.

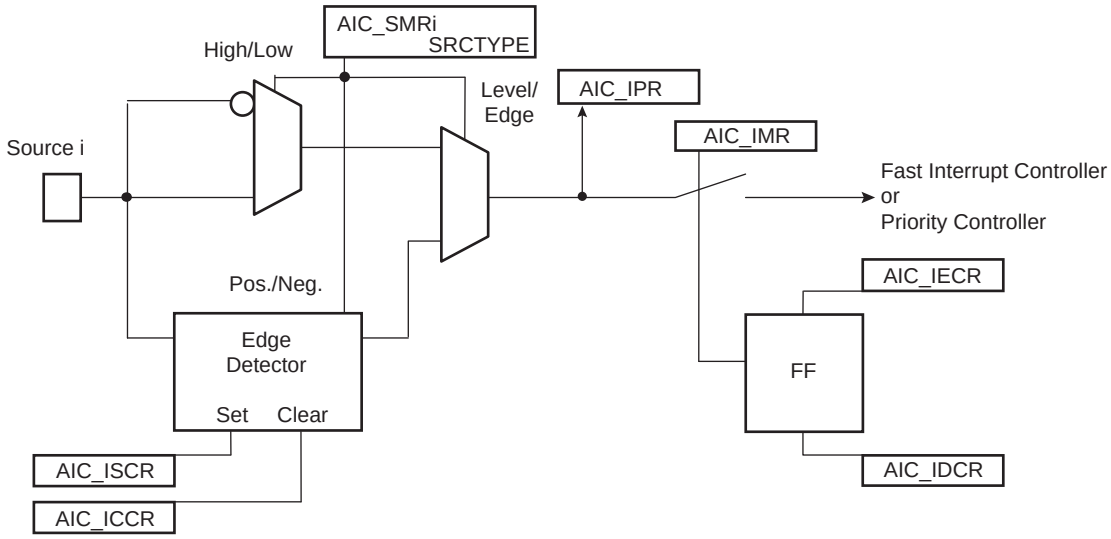
27.7.1.5 *Internal Interrupt Source Input Stage*

Figure 27-4. Internal Interrupt Source Input Stage



27.7.1.6 *External Interrupt Source Input Stage*

Figure 27-5. External Interrupt Source Input Stage



27.7.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

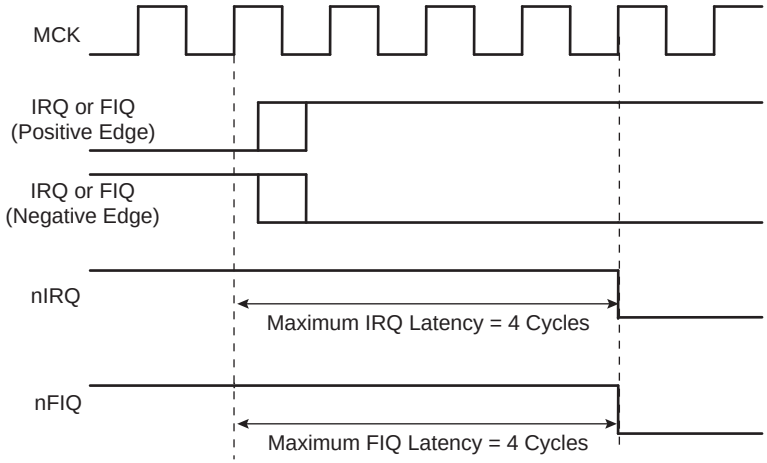
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

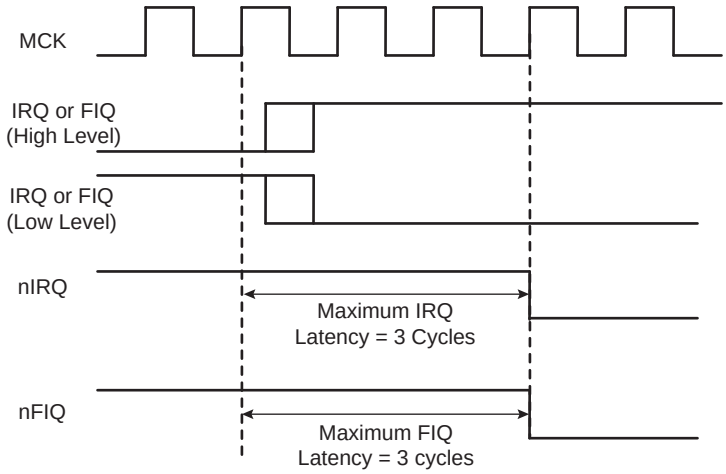
27.7.2.1 External Interrupt Edge Triggered Source

Figure 27-6. External Interrupt Edge Triggered Source



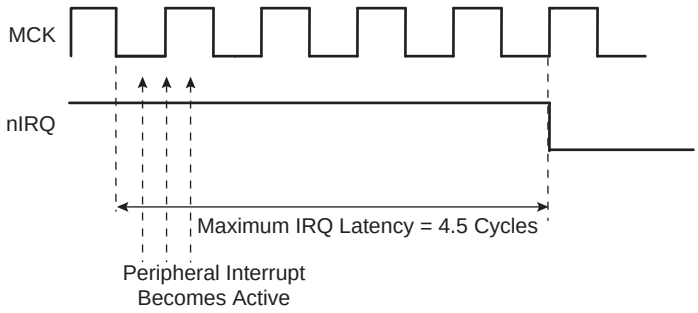
27.7.2.2 External Interrupt Level Sensitive Source

Figure 27-7. External Interrupt Level Sensitive Source



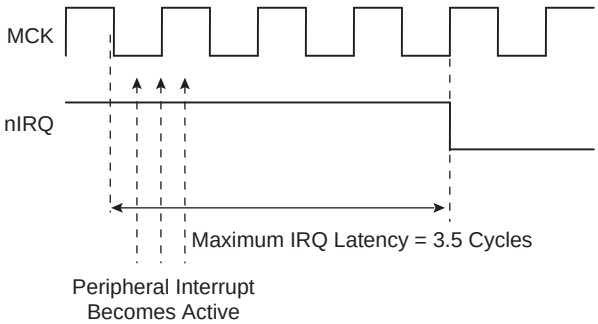
27.7.2.3 Internal Interrupt Edge Triggered Source

Figure 27-8. Internal Interrupt Edge Triggered Source



27.7.2.4 Internal Interrupt Level Sensitive Source

Figure 27-9. Internal Interrupt Level Sensitive Source



27.7.3 Normal Interrupt

27.7.3.1 Priority Controller

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC_SMR (Source Mode Register). Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC_SMR (Source Mode Register), the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the AIC_IVR (Interrupt Vector Register) is read. **The read of AIC_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the AIC_EOICR (End of Interrupt Command Register). **The write of AIC_EOICR is the exit point of the interrupt handling**.

27.7.3.2 Interrupt Nesting

The priority controller utilizes interrupt nesting in order for the high priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

27.7.3.3 Interrupt Vectoring

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC_SVR1 to AIC_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC_IVR (Interrupt Vector Register), the value written into AIC_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC_IVR is mapped at the absolute address 0xFFFF F100 and thus accessible from the ARM interrupt vector at address 0x0000 0018 through the following instruction:

```
LDR   PC,[PC,# -&F20]
```

When the processor executes this instruction, it loads the read value in AIC_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

27.7.3.4 Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
2. The instruction at the ARM interrupt exception vector address is required to work with the vectoring

```
LDR PC, [PC, # -&F20]
```

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by four.
2. The ARM core enters Interrupt mode, if it has not already done so.
3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   - Pushes the current level and the current interrupt number on to the stack.
   - Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.

5. Further interrupts can then be unmasked by clearing the “I” bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The “I” bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The End of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the “I” bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in SPSR_irq.

Note: The “I” bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

27.7.4 Fast Interrupt

27.7.4.1 Fast Interrupt Source

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if fast forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

27.7.4.2 Fast Interrupt Control

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC_SMR0 and the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR0 enables programming the fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register) respectively enables and disables the fast interrupt. The bit 0 of AIC_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

27.7.4.3 Fast Interrupt Vectoring

The fast interrupt handler address can be stored in AIC_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC_FVR (Fast Vector Register). This offers a way to branch in one single instruction to the interrupt handler, as AIC_FVR is mapped at the absolute address 0xFFFF F104 and thus accessible from the ARM fast interrupt vector at address 0x0000 001C through the following instruction:

```
LDR   PC,[PC,# -&F20]
```

When the processor executes this instruction it loads the value read in AIC_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

27.7.4.4 Fast Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and associated status bits.

Assuming that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
2. The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:

```
LDR PC, [PC, # -&F20]
```

3. The user does not need nested fast interrupts.

When nFIQ is asserted, if the bit “F” of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the program counter is loaded in the FIQ link register (R14_FIQ) and the program counter (R15) is loaded with 0x1C. In the following cycle, during fetch at address 0x20, the ARM core adjusts R14_fiq, decrementing it by four.
2. The ARM core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the program counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automati-

cally clearing the fast interrupt, if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.

4. The previous step enables branching to the corresponding interrupt service routine. It is not necessary to save the link register R14_fiq and SPSR_fiq if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers R8 to R13 because FIQ mode has its own dedicated registers and the user R8 to R13 are banked. The other registers, R0 to R7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the interrupt source 0.
6. Finally, the Link Register R14_fiq is restored into the PC after decrementing it by four (with instruction `SUB PC, LR, #4` for example). This has the effect of returning from the interrupt to whatever was being executed before, loading the CPSR with the SPSR and masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The “F” bit in SPSR is significant. If it is set, it indicates that the ARM core was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the ARM vector 0x1C. This method does not use the vectoring, so that reading AIC_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

27.7.4.5 Fast Forcing

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC_FFER) and the Fast Forcing Disable Register (AIC_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the Interrupt Pending Register (AIC_IPR).

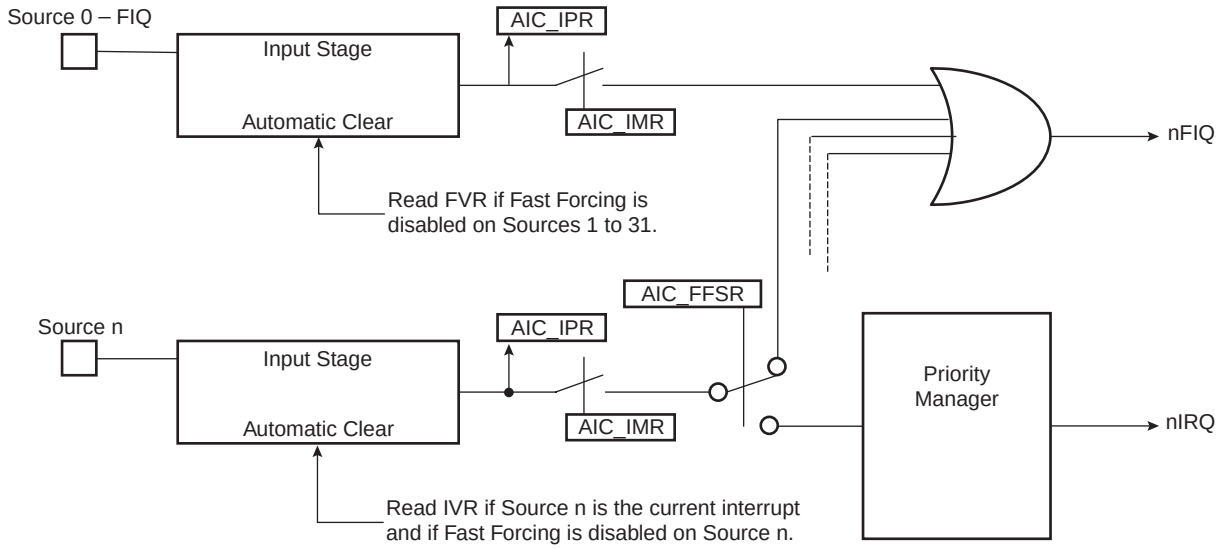
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt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

All enabled and pending interrupt sources that have the fast forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the Interrupt Clear Command Register. In doing so, they are cleared independently and thus lost interrupts are prevented.

The read of AIC_IVR does not clear the source that has the fast forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.

Figure 27-10. Fast Forcing



27.7.5 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

- If an enabled interrupt with a higher priority than the current one is pending, it is stacked.
- If there is no enabled pending interrupt, the spurious vector is returned.

In either case, an End of Interrupt command is necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system as the debug system would become strongly intrusive and cause the application to enter an undesired state.

This is avoided by using the Protect Mode. Writing DBGM in AIC_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it. The new context of the AIC, including the value of the Interrupt Status Register (AIC_ISR), is updated with the current interrupt only when AIC_IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR. Extra AIC_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC_IVR write has no effect and can be removed to optimize the code.

27.7.6 Spurious Interrupt

The Advanced Interrupt Controller features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.

- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time. (As in the case for the Watchdog.)
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in AIC_SPU (Spurious Vector Register). The programmer must store the address of a spurious interrupt handler in AIC_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC_EOICR and performs a return from interrupt.

27.7.7 General Interrupt Mask

The AIC features a General Interrupt Mask bit to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if the bit GMSK in AIC_DCR (Debug Control Register) is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

27.8 Advanced Interrupt Controller (AIC) User Interface

27.8.1 Base Address

The AIC is mapped at the address **0xFFFF F000**. It has a total 4-Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the ARM processor support only an ± 4-Kbyte offset.

27.8.2 Register Mapping

Table 27-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0000	Source Mode Register 0	AIC_SMR0	Read/Write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read/Write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read/Write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read/Write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read/Write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read/Write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register(2)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(2)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118	Reserved	---	---	---
0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register(2)	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register(2)	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register(2)	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register(2)	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read/Write	0x0
0x138	Debug Control Register	AIC_DCR	Read/Write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register(2)	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register(2)	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register(2)	AIC_FFSR	Read-only	0x0

Notes: 1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.
2. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.

27.8.3 AIC Source Mode Register

Name: AIC_SMR0..AIC_SMR31

Access: Read/Write

Reset Value: 0x0

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

- **PRIOR: Priority Level**

Programs the priority level for all sources except FIQ source (source 0).

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

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

- **SRCTYPE: Interrupt Source Type**

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

SRCTYPE		Internal Interrupt Sources	External Interrupt Sources
0	0	High level Sensitive	Low level Sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level Sensitive	High level Sensitive
1	1	Positive edge triggered	Positive edge triggered

27.8.4 AIC Source Vector Register

Name: AIC_SVR0..AIC_SVR31

Access: Read/Write

Reset Value: 0x0

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

- **VECTOR: Source Vector**

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

27.8.5 AIC Interrupt Vector Register

Name: AIC_IVR

Access: Read-only

Reset Value: 0x0

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

- **IRQV: Interrupt Vector Register**

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

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

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

27.8.6 AIC FIQ Vector Register

Name: AIC_FVR

Access: Read-only

Reset Value: 0 x0

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

- **FIQV: FIQ Vector Register**

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

27.8.7 AIC Interrupt Status Register

Name: AIC_ISR

Access: Read-only

Reset Value: 0x0

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

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

27.8.8 AIC Interrupt Pending Register

Name: AIC_IPR

Access: Read-only

Reset Value: 0x0

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

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

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

27.8.9 AIC Interrupt Mask Register

Name: AIC_IMR

Access: Read-only

Reset Value: 0x0

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

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

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

27.8.10 AIC Core Interrupt Status Register

Name: AIC_CISR

Access: Read-only

Reset Value: 0x0

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

- **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.

27.8.11 AIC Interrupt Enable Command Register

Name: AIC_IECR

Access: Write-only

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

- **FIQ, SYS, PID2-PID3: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

27.8.12 AIC Interrupt Disable Command Register

Name: AIC_IDCR

Access: Write-only

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

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

0 = No effect.

1 = Disables corresponding interrupt.

27.8.13 AIC Interrupt Clear Command Register

Name: AIC_ICCR

Access: Write-only

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

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

0 = No effect.

1 = Clears corresponding interrupt.

27.8.14 AIC Interrupt Set Command Register

Name: AIC_ISCR

Access: Write-only

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

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

0 = No effect.

1 = Sets corresponding interrupt.

27.8.15 AIC End of Interrupt Command Register

Name: AIC_EOICR

Access: Write-only

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

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

27.8.16 AIC Spurious Interrupt Vector Register

Name: AIC_SPU

Access: Read/Write

Reset Value: 0x0

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

- **SIQV: Spurious Interrupt Vector Register**

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

27.8.17 AIC Debug Control Register

Name: AIC_DEBUG

Access: Read/Write

Reset Value: 0x0

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

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

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

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

27.8.18 AIC Fast Forcing Enable Register

Name: AIC_FFER

Access: Write-only

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

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

0 = No effect.

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

27.8.19 AIC Fast Forcing Disable Register

Name: AIC_FFDR

Access: Write-only

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

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

0 = No effect.

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

27.8.20 AIC Fast Forcing Status Register

Name: AIC_FFSR

Access: Read-only

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

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

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

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

28. Clock Generator

28.1 Overview

The Clock Generator is made up of 1 PLL, a Main Oscillator, as well as an RC Oscillator.

It provides the following clocks:

- SLCK, the Slow Clock, which is the only permanent clock within the system
- MAINCK is the output of the Main Oscillator
- PLLCK is the output of the Divider and PLL block

The Clock Generator User Interface is embedded within the Power Management Controller and is described in Section 29.9. However, the Clock Generator registers are named CKGR_.

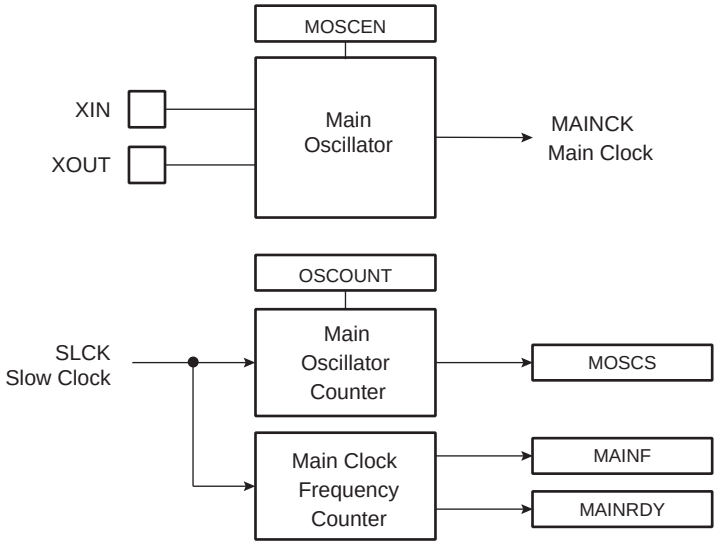
28.2 Slow Clock RC Oscillator

The user has to take into account the possible drifts of the RC Oscillator. More details are given in the section “DC Characteristics” of the product datasheet.

28.3 Main Oscillator

Figure 28-1 shows the Main Oscillator block diagram.

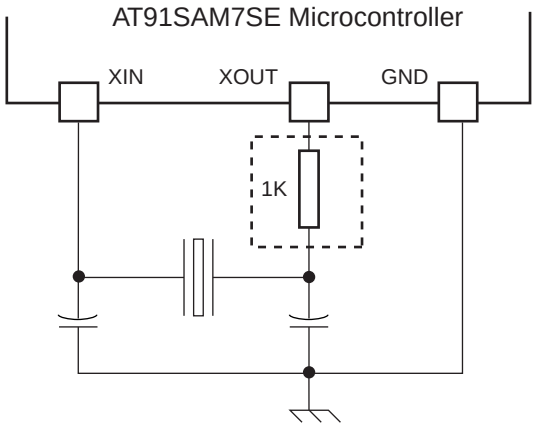
Figure 28-1. Main Oscillator Block Diagram



28.3.1 Main Oscillator Connections

The Clock Generator integrates a Main Oscillator that is designed for a 3 to 20 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 28-2. For further details on the electrical characteristics of the Main Oscillator, see the section “DC Characteristics” of the product datasheet.

Figure 28-2. Typical Crystal Connection



28.3.2 Main Oscillator Startup Time

The startup time of the Main Oscillator is given in the DC Characteristics section of the product datasheet. The startup time depends on the crystal frequency and decreases when the frequency rises.

28.3.3 Main Oscillator Control

To minimize the power required to start up the system, the main oscillator is disabled after reset and slow clock is selected.

The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCEN bit in the Main Oscillator Register (CKGR_MOR).

When disabling the main oscillator by clearing the MOSCEN bit in CKGR_MOR, the MOSCS bit in PMC_SR is automatically cleared, indicating the main clock is off.

When enabling the main oscillator, the user must initiate the main oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the main oscillator.

When the MOSCEN bit and the OSCOUNT are written in CKGR_MOR to enable the main oscillator, the MOSCS bit in PMC_SR (Status Register) is cleared and the counter starts counting down on the slow clock divided by 8 from the OSCOUNT value. Since the OSCOUNT value is coded with 8 bits, the maximum startup time is about 62 ms.

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

28.3.4 Main Clock Frequency Counter

The Main Oscillator features a Main Clock frequency counter that provides the quartz frequency connected to the Main Oscillator. Generally, this value is known by the system designer; however, it is useful for the boot program to configure the device with the correct clock speed, independently of the application.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as the Main Oscillator is stable, i.e., as soon as the MOSCS bit is set. Then, at the 16th falling edge of Slow Clock, the MAINRDY bit in CKGR_MCFR (Main Clock Frequency Register) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of

Slow Clock, so that the frequency of the crystal connected on the Main Oscillator can be determined.

28.3.5 Main Oscillator Bypass

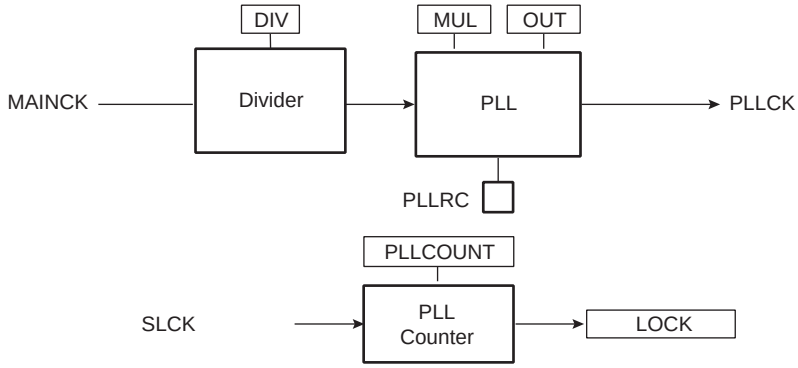
The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN pin. The input characteristics of the XIN pin under these conditions are given in the product electrical characteristics section. The programmer has to be sure to set the OSCBYPASS bit to 1 and the MOSCEN bit to 0 in the Main OSC register (CKGR_MOR) for the external clock to operate properly.

28.4 Divider and PLL Block

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

Figure 28-3 shows the block diagram of the divider and PLL block.

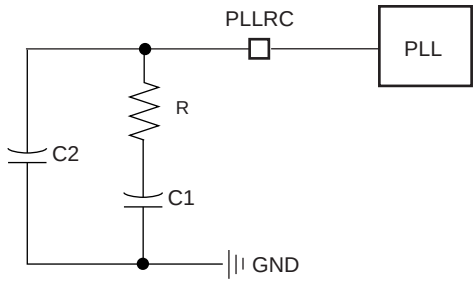
Figure 28-3. Divider and PLL Block Diagram



28.4.1 PLL Filter

The PLL requires connection to an external second-order filter through the PLLRC pin. Figure 28-4 shows a schematic of these filters.

Figure 28-4. PLL Capacitors and Resistors



Values of R, C1 and C2 to be connected to the PLLRC pin must be calculated as a function of the PLL input frequency, the PLL output frequency and the phase margin. A trade-off has to be found between output signal overshoot and startup time.

28.4.2 Divider and Phase Lock Loop Programming

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is set to 0, thus the corresponding PLL input clock is set to 0.

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

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK bit in PMC_SR is automatically cleared. The values written in the PLLCOUNT field in CKGR_PLLR are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field. The transient time depends on the PLL filter. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

29. Power Management Controller (PMC)

29.1 Overview

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

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the AIC and the Memory Controller.
- Processor Clock (PCK), switched off when entering processor in idle mode.
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, MCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- UDP Clock (UDPCK), required by USB Device Port operations.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

29.2 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

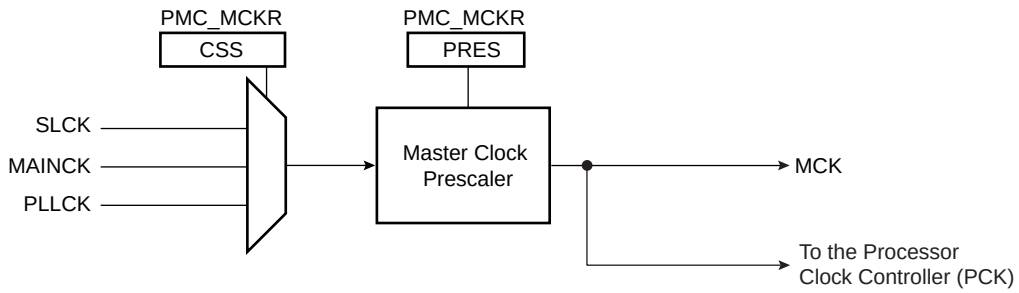
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLL.

The Master Clock Controller is made up of a clock selector and a prescaler.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC_MCKR programs the prescaler.

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 29-1. Master Clock Controller



29.3 Processor Clock Controller

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

The Processor Clock PCK is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock, 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.

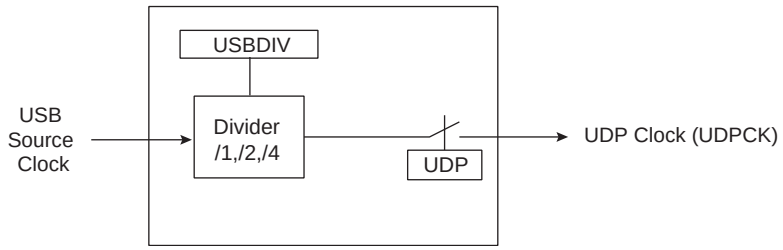
29.4 USB Clock Controller

The USB Source Clock is the PLL output. If using the USB, the user must program the PLL to generate a 48 MHz, a 96 MHz or a 192 MHz signal with an accuracy of ± 0.25% depending on the USBDIV bit in CKGR_PLLR.

When the PLL output is stable, i.e., the LOCK bit is set:

- The USB device clock can be enabled by setting the UDP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UDP bit in PMC_SCDR. The UDP bit in PMC_SCSR gives the activity of this clock. The USB device port require both the 48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.

Figure 29-2. USB Clock Controller



29.5 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by the way of the Peripheral Clock Controller. The user can individually enable and disable the Master Clock on the peripherals by writing into the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

In order to stop a peripheral, it is recommended that the system software wait until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

The bit number within the Peripheral Clock Control registers (PMC_PCER, PMC_PCDR, and PMC_PCSR) is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

29.6 Programmable Clock Output Controller

The PMC controls 3 signals to be output on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

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

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

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

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

29.7 Programming Sequence

1. Enabling the Main Oscillator:

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

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

   Code Example:

   ```
   write_register(CKGR_MOR,0x00000701)
   ```

   Start Up Time = 8 * OSCOUNT / SLCK = 56 Slow Clock Cycles.

   So, the main oscillator will be enabled (MOSCS bit set) after 56 Slow Clock Cycles.

2. Checking the Main Oscillator Frequency (Optional):

   In some situations the user may need an accurate measure of the main oscillator frequency. This measure can be accomplished via the CKGR_MCFR register.

   Once the MAINRDY field is set in CKGR_MCFR register, the user may read the MAINF field in CKGR_MCFR register. This provides the number of main clock cycles within sixteen slow clock cycles.

3. Setting PLL and divider:

All parameters needed to configure PLL and the divider are located in the CKGR_PLLR register.

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

The OUT field is used to select the PLL B output frequency range.

The MUL field is the PLL multiplier factor. This parameter can be programmed between 0 and 2047. If MUL is set to 0, PLL will be turned off, otherwise the PLL output frequency is PLL input frequency multiplied by (MUL + 1).

The PLLCOUNT field specifies the number of slow clock cycles before LOCK bit is set in the PMC_SR register after CKGR_PLLR register has been written.

Once the PMC_PLL register has been written, the user must wait for the LOCK bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCK has been enabled in the PMC_IER register. All parameters in CKGR_PLLR can be programmed in a single write operation. If at some stage one of the following parameters, MUL, DIV is modified, LOCK bit will go low to indicate that PLL is not ready yet. When PLL is locked, LOCK will be set again. The user is constrained to wait for LOCK bit to be set before using the PLL output clock.

The USBDIV field is used to control the additional divider by 1, 2 or 4, which generates the USB clock(s).

Code Example:

```
write_register(CKGR_PLLR,0x00040805)
```

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

4. Selection of Master Clock and Processor Clock

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

The CSS field is used to select the Master Clock divider source. By default, the selected clock source is slow clock.

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

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

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

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

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

Note: IF PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR, the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK goes high and MCKRDY is set.
While PLL is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 29.8.2. “Clock Switching Waveforms” on page 281.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

5. Selection of Programmable clocks

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

Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR registers. Depending on the system used, 3 Programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which Programmable clock is enabled. By default all Programmable clocks are disabled.

PMC_PCKx registers are used to configure Programmable clocks.

The CSS field is used to select the Programmable clock divider source. Four clock options are available: main clock, slow clock, PLLCK. By default, the clock source selected is slow clock.

The PRES field is used to control the Programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler

input divided by PRES parameter. By default, the PRES parameter is set to 1 which means that master clock is equal to slow clock.

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

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

Code Example:

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

Programmable clock 0 is main clock divided by 32.

6. Enabling Peripheral Clocks

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

Depending on the system used, AT91SAM7SE512,14 peripheral clocks and for AT91SAM7SE256/32,12 peripheral clocks can be enabled or disabled. The PMC_PCSR provides a clear view as to which peripheral clock is enabled.

Note: Each enabled peripheral clock corresponds to Master Clock.

Code Examples:

```
write_register(PMC_PCER,0x00000110)
```

Peripheral clocks 4 and 8 are enabled.

```
write_register(PMC_PCDR,0x00000010)
```

Peripheral clock 4 is disabled.

29.8 Clock Switching Details

29.8.1 Master Clock Switching Timings

Table 29-1 gives the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

Table 29-1. Clock Switching Timings (Worst Case)

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

29.8.2 Clock Switching Waveforms

Figure 29-3. Switch Master Clock from Slow Clock to PLL Clock

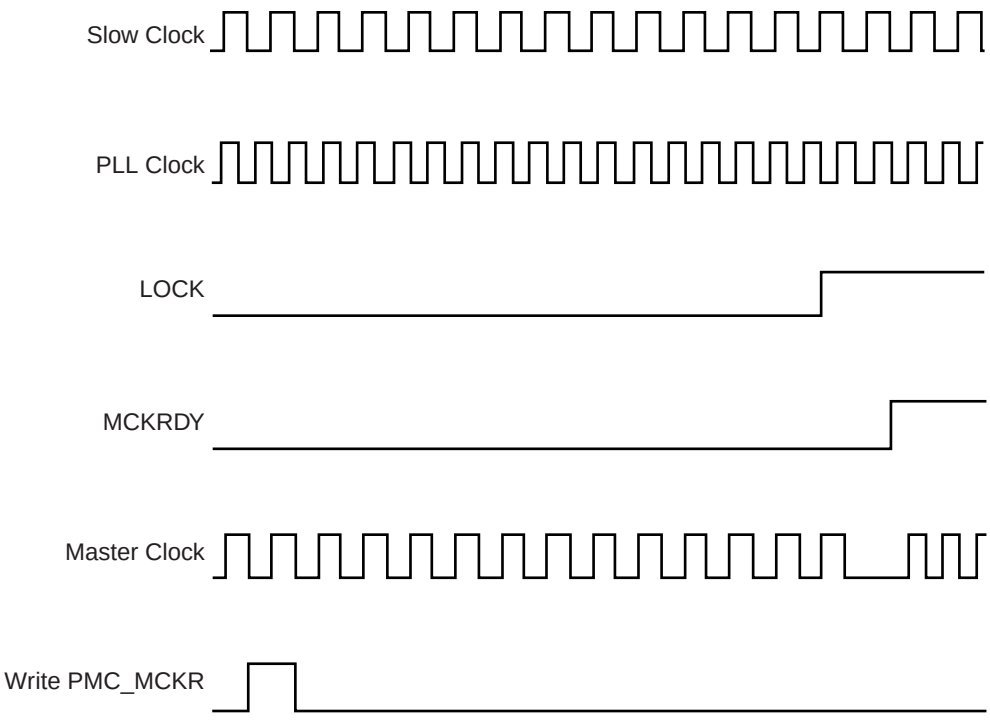


Figure 29-4. Switch Master Clock from Main Clock to Slow Clock

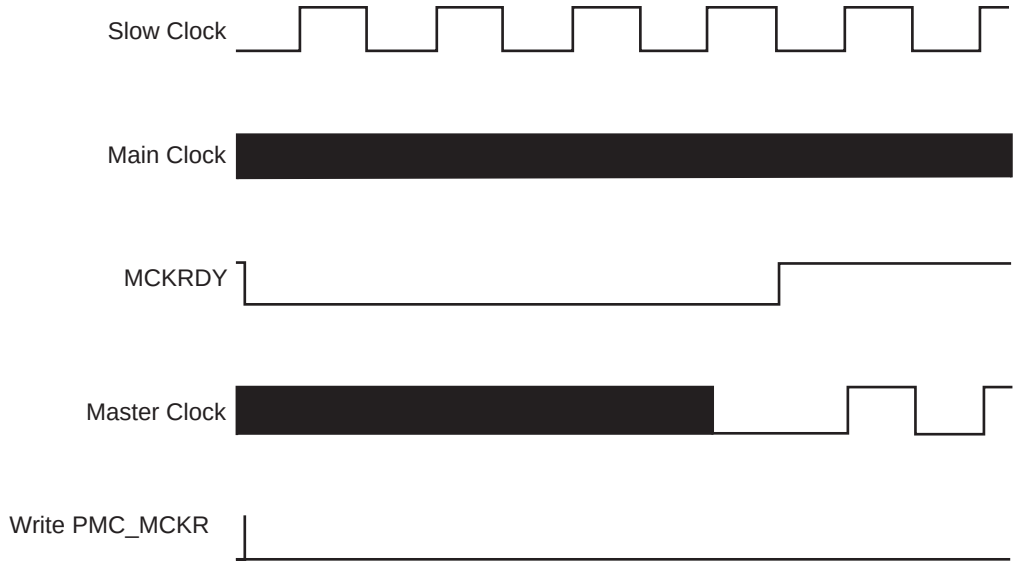


Figure 29-5. Change PLL Programming

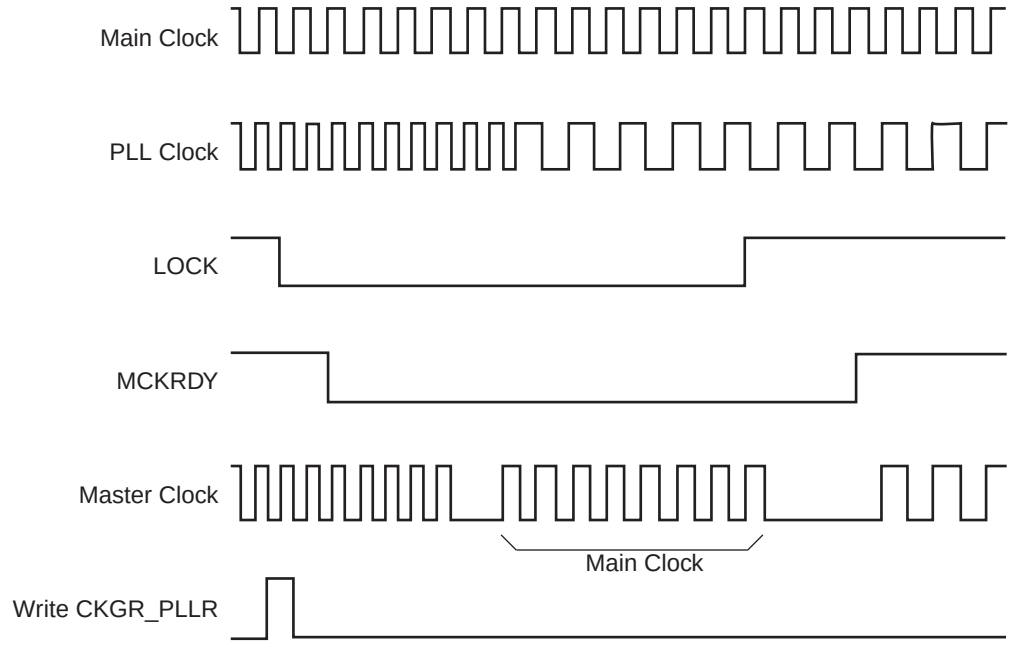
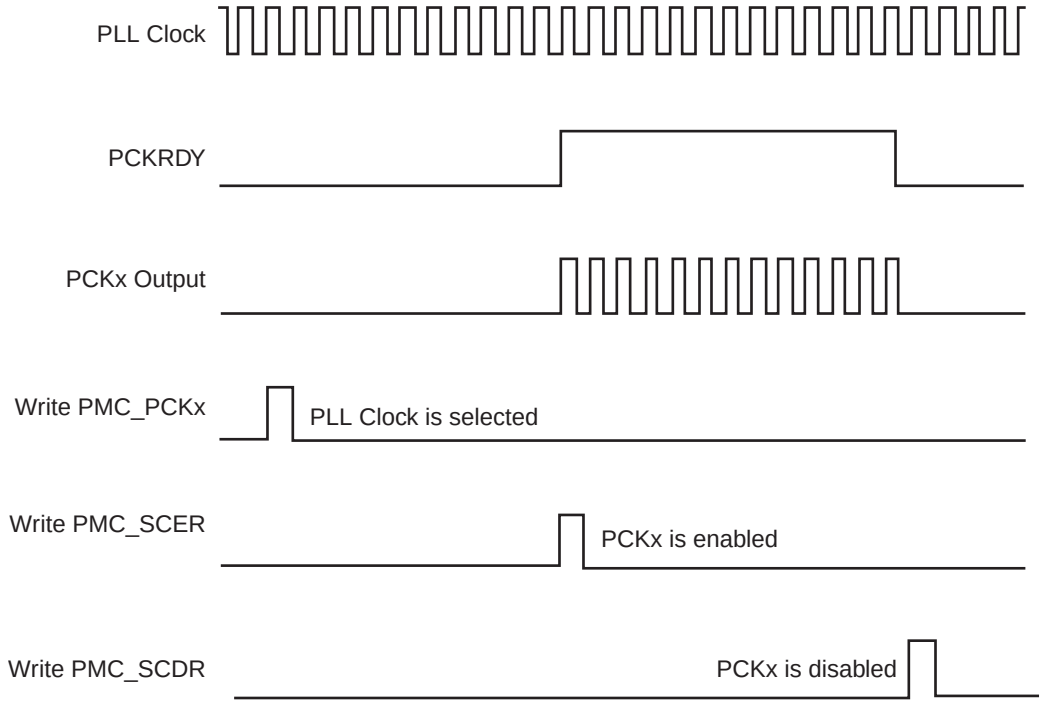


Figure 29-6. Programmable Clock Output Programming



29.9 Power Management Controller (PMC) User Interface

Table 29-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC _SCSR	Read-only	0x01
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	–
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0
0x001C	Reserved	–	–	–
0x0020	Main Oscillator Register	CKGR_MOR	Read-write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	Reserved	–	–	–
0x002C	PLL Register	CKGR_PLLR	Read-write	0x3F00
0x0030	Master Clock Register	PMC_MCKR	Read-write	0x0
0x0038	Reserved	–	–	–
0x003C	Reserved	–	–	–
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read-write	0x0
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read-write	0x0
...	...	...	...	...
0x0060	Interrupt Enable Register	PMC_IER	Write-only	--
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	--
0x0068	Status Register	PMC_SR	Read-only	0x08
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0
0x0070 - 0x007C	Reserved	–	–	–

29.9.1 PMC System Clock Enable Register

Name: PMC_SCER

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	–	–	–	–	–	–	–

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

0 = No effect.

1 = Enables the 48 MHz clock of the USB Device Port.

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

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

29.9.2 PMC System Clock Disable Register

Name: PMC_SCDR

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	–	–	–	–	–	–	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

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

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

0 = No effect.

1 = Disables the 48 MHz clock of the USB Device Port.

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

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

29.9.3 PMC System Clock Status Register

Name: PMC_SCSR

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	–	–	–	–	–	–	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

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

0 = The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

1 = The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

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

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

29.9.4 PMC Peripheral Clock Enable Register

Name: PMC_PCER

Access: Write-only

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

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

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

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

29.9.5 PMC Peripheral Clock Disable Register

Name: PMC_PCDR

Access: Write-only

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

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

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

29.9.6 PMC Peripheral Clock Status Register

Name: PMC_PCSR

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.

29.9.7 PMC Clock Generator Main Oscillator Register

Name: CKGR_MOR

Access: Read-write

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

- **MOSCEN: Main Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0 = The Main Oscillator is disabled.

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

When MOSCEN is set, the MOSCS flag is set once the Main Oscillator startup time is achieved.

- **OSCBYPASS: Oscillator Bypass**

0 = No effect.

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

When OSCBYPASS is set, the MOSCS flag in PMC_SR is automatically set.

Clearing MOSCEN and OSCBYPASS bits allows resetting the MOSCS flag.

- **OSCOUNT: Main Oscillator Start-up Time**

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

29.9.8 PMC Clock Generator Main Clock Frequency Register

Name: CKGR_MCFR

Access: Read-only

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

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled.

1 = The Main Oscillator has been enabled previously and MAINF value is available.

29.9.9 PMC Clock Generator PLL Register

Name: CKGR_PLLR

Access: Read-write

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

23	22	21	20	19	18	17	16
MUL							

15	14	13	12	11	10	9	8
OUT		PLLCOUNT					

7	6	5	4	3	2	1	0
DIV							

Possible limitations on PLL input frequencies and multiplier factors should be checked before using the PMC.

- **DIV: Divider**

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

- **PLLCOUNT: PLL Counter**

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

- **OUT: PLL Clock Frequency Range**

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

- **MUL: PLL Multiplier**

0 = The PLL is deactivated.

1 up to 2047 = The PLL Clock frequency is the PLL input frequency multiplied by MUL+ 1.

- **USBDIV: Divider for USB Clock**

USBDIV		Divider for USB Clock(s)
0	0	Divider output is PLL clock output.
0	1	Divider output is PLL clock output divided by 2.
1	0	Divider output is PLL clock output divided by 4.
1	1	Reserved.

29.9.10 PMC Master Clock Register

Name: PMC_MCKR

Access: Read-write

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

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	Reserved
1	1	PLL Clock is selected.

- **PRES: Processor Clock Prescaler**

PRES			Processor Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

29.9.11 PMC Programmable Clock Register

Name: PMC_PCKx

Access: Read-write

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

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	Reserved
1	1	PLL Clock is selected

- **PRES: Programmable Clock Prescaler**

PRES			Programmable Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

29.9.12 PMC Interrupt Enable Register

Name: PMC_IER

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
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**
- **LOCK: PLL Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

29.9.13 PMC Interrupt Disable Register

Name: PMC_IDR

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
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Disable**
- **LOCK: PLL Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

29.9.14 PMC Status Register

Name: PMC_SR

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
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: MOSCS Flag Status**

0 = Main oscillator is not stabilized.

1 = Main oscillator is stabilized.

- **LOCK: PLL Lock Status**

0 = PLL is not locked

1 = PLL is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

29.9.15 PMC Interrupt Mask Register

Name: PMC_IMR

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
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Mask**
- **LOCK: PLL Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

30. Debug Unit (DBGU)

30.1 Overview

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers inform as to the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

30.2 Block Diagram

Figure 30-1. Debug Unit Functional Block Diagram

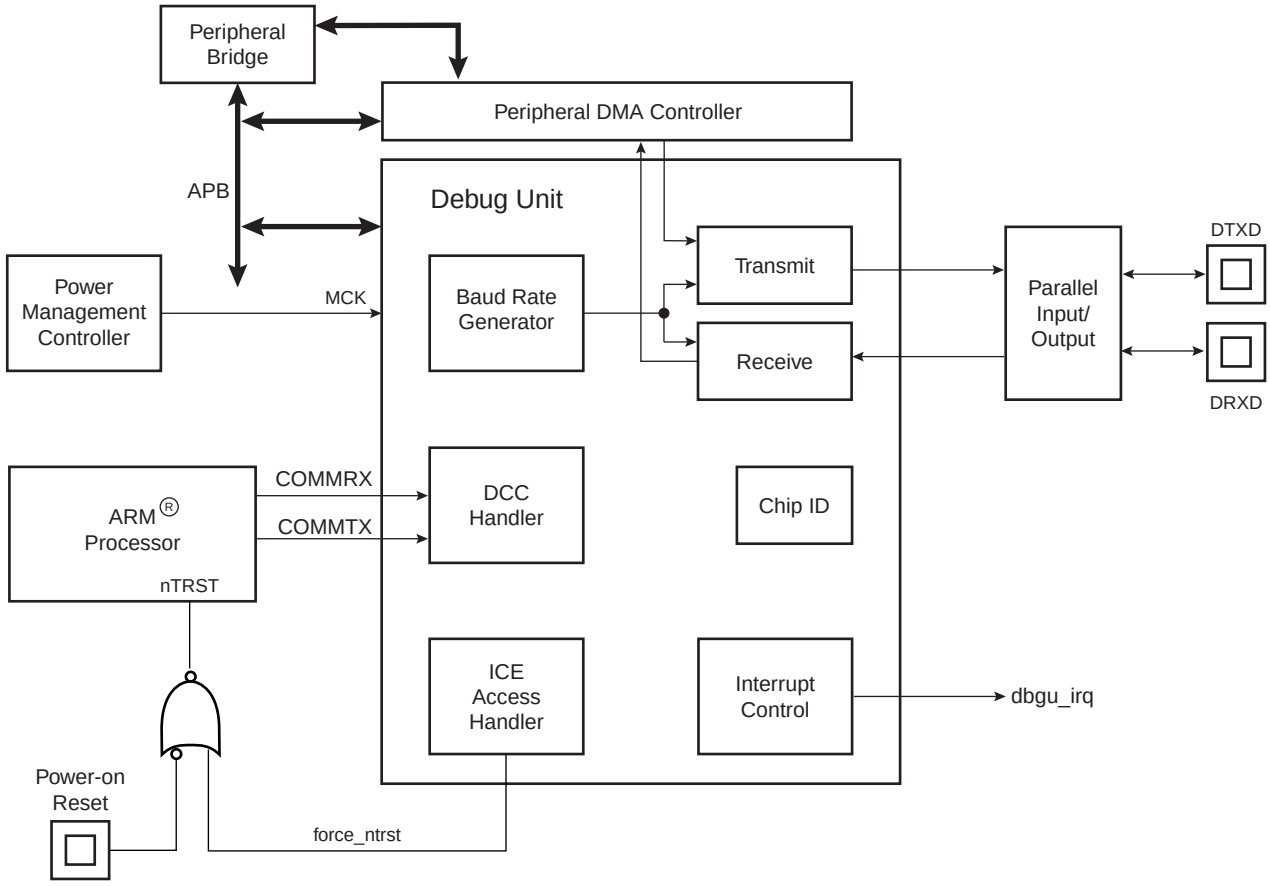
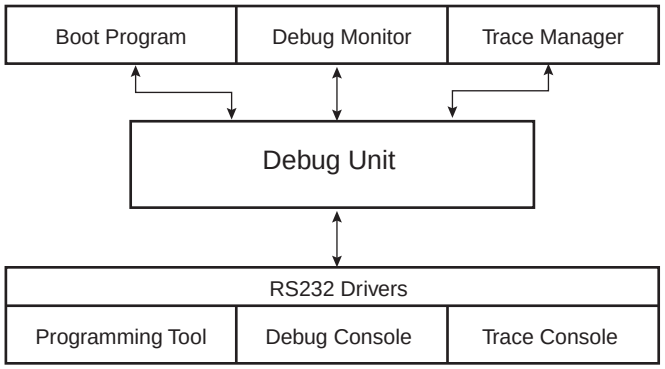


Table 30-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Debug Unit Application Example



30.3 Product Dependencies

30.3.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

30.3.2 Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

30.3.3 Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 30-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

30.4 UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

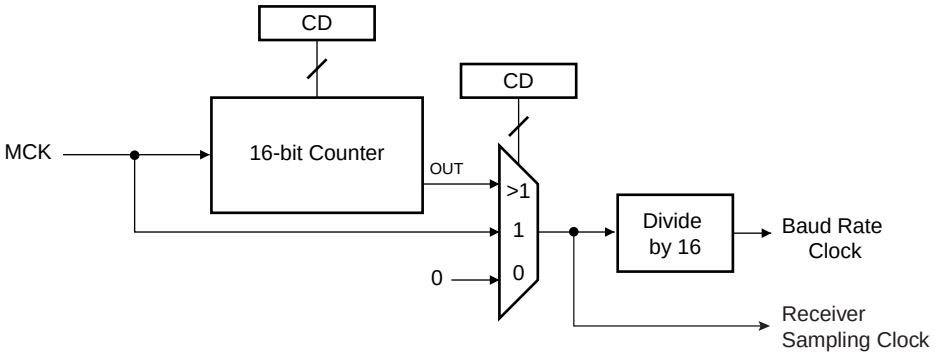
30.4.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in DBGU_BRGR (Baud Rate Generator Register). If DBGU_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

$$\text{Baud Rate} = \frac{\text{MCK}}{16 \times \text{CD}}$$

Figure 30-2. Baud Rate Generator



30.4.2 Receiver

30.4.2.1 Receiver Reset, Enable and Disable

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register DBGU_CR with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing DBGU_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing DBGU_CR with the bit RSTRX at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

30.4.2.2 Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the DRXD at the theoretical mid-point of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after the falling edge of the start bit was detected.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 30-3. Start Bit Detection

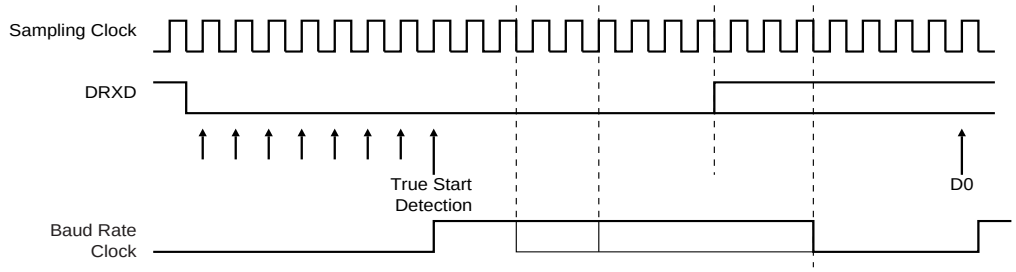
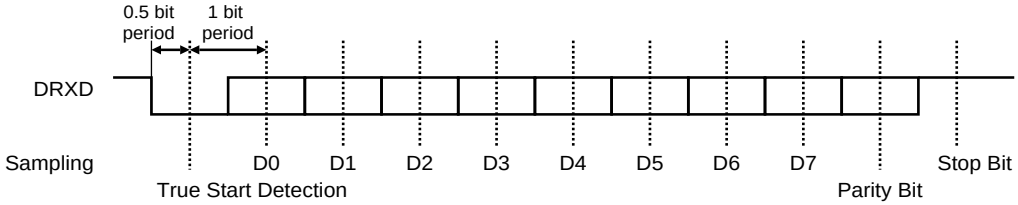


Figure 30-4. Character Reception

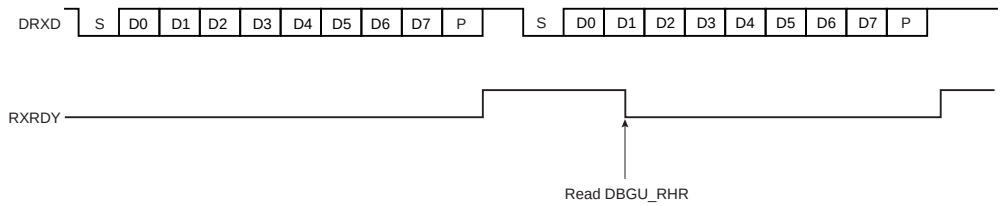
Example: 8-bit, parity enabled 1 stop



30.4.2.3 *Receiver Ready*

When a complete character is received, it is transferred to the DBGU_RHR and the RXRDY status bit in DBGU_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU_RHR is read.

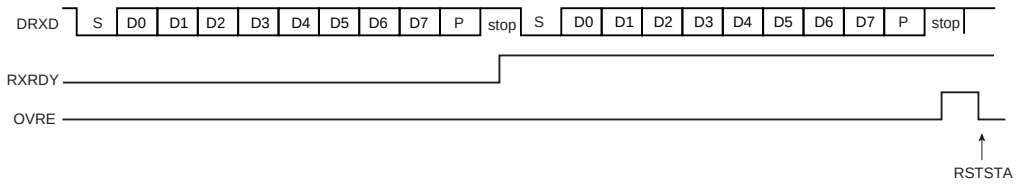
Figure 30-5. Receiver Ready



30.4.2.4 *Receiver Overrun*

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU_SR is set. OVRE is cleared when the software writes the control register DBGU_CR with the bit RSTSTA (Reset Status) at 1.

Figure 30-6. Receiver Overrun

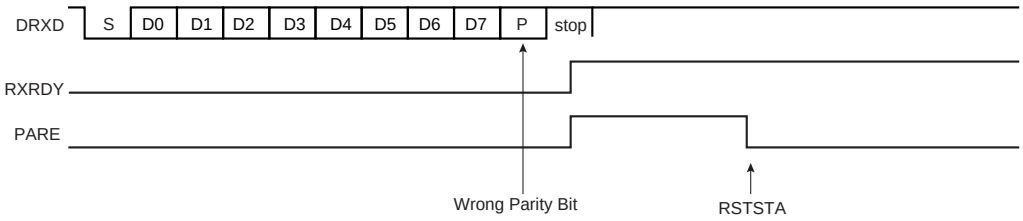


30.4.2.5 *Parity Error*

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in DBGU_MR. It then compares the result with the received parity

bit. If different, the parity error bit PARE in DBGU_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register DBGU_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

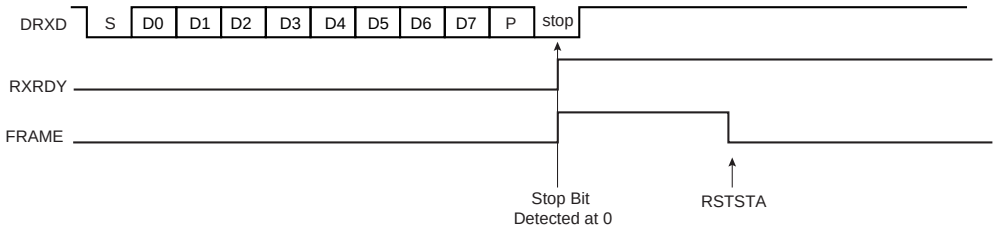
Figure 30-7. Parity Error



30.4.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time the RXRDY bit is set. The bit FRAME remains high until the control register DBGU_CR is written with the bit RSTSTA at 1.

Figure 30-8. Receiver Framing Error



30.4.3 Transmitter

30.4.3.1 Transmitter Reset, Enable and Disable

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register DBGU_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register DBGU_THR before actually starting the transmission.

The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

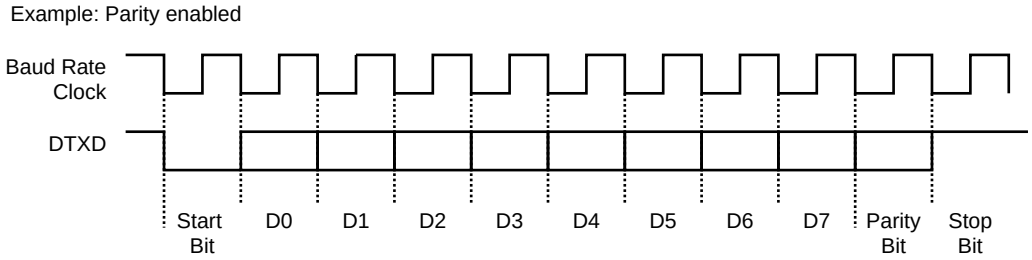
The programmer can also put the transmitter in its reset state by writing the DBGU_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

30.4.3.2 Transmit Format

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in the Mode Register and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field

PARE in the mode register DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 30-9. Character Transmission

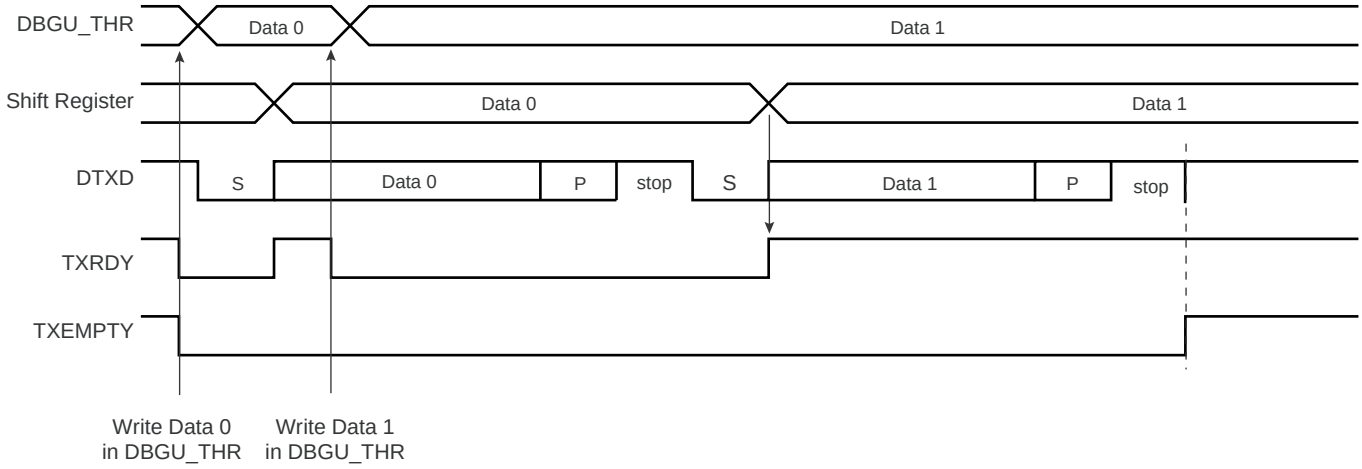


30.4.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register DBGU_SR. The transmission starts when the programmer writes in the Transmit Holding Register DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

Figure 30-10. Transmitter Control



30.4.4 Peripheral Data Controller

Both the receiver and the transmitter of the Debug Unit's UART are generally connected to a Peripheral Data Controller (PDC) channel.

The peripheral data controller channels are programmed via registers that are mapped within the Debug Unit user interface from the offset 0x100. The status bits are reported in the Debug Unit status register DBGU_SR and can generate an interrupt.

The RXRDY bit triggers the PDC channel data transfer of the receiver. This results in a read of the data in DBGU_RHR. The TXRDY bit triggers the PDC channel data transfer of the transmitter. This results in a write of a data in DBGU_THR.

30.4.5 Test Modes

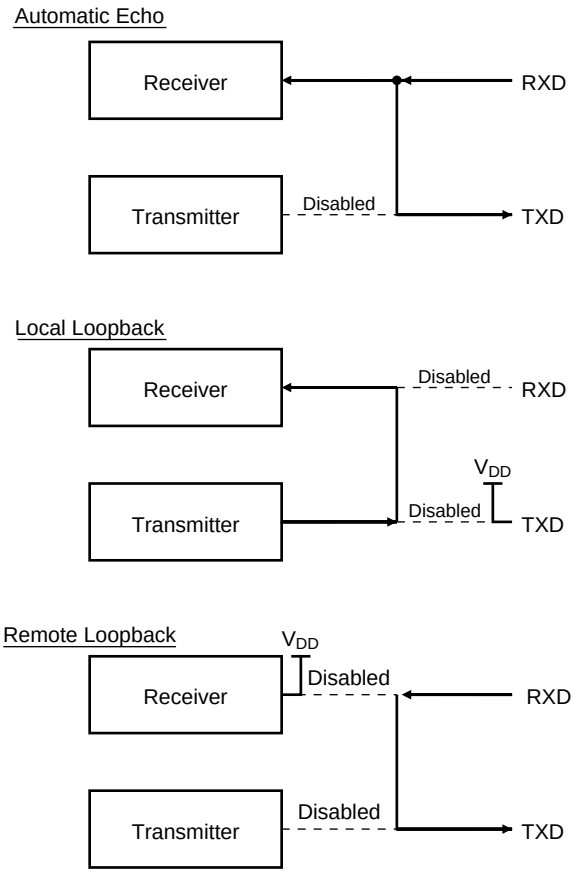
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

The Local Loopback mode allows the transmitted characters to be received. DTXD and DRXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The DRXD pin level has no effect and the DTXD line is held high, as in idle state.

The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Figure 30-11. Test Modes



30.4.6 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC    p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR    p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

30.4.7 Chip Identifier

The Debug Unit features two chip identifier registers, DBGU_CIDR (Chip ID Register) and DBGU_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register
- NVPTYP and NVPSIZ - identifies the type of embedded non-volatile memory and its size
- ARCH - identifies the set of embedded peripheral
- SRAMSIZ - indicates the size of the embedded SRAM
- EPROC - indicates the embedded ARM processor
- VERSION - gives the revision of the silicon

The second register is device-dependent and reads 0 if the bit EXT is 0.

30.4.8 ICE Access Prevention

The Debug Unit allows blockage of access to the system through the ARM processor's ICE interface. This feature is implemented via the register Force NTRST (DBGU_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the FNTRST bit resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

30.5 Debug Unit User Interface

Table 30-2. Debug Unit Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Control Register	DBGU_CR	Write-only	–
0x0004	Mode Register	DBGU_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	DBGU_IER	Write-only	–
0x000C	Interrupt Disable Register	DBGU_IDR	Write-only	–
0x0010	Interrupt Mask Register	DBGU_IMR	Read-only	0x0
0x0014	Status Register	DBGU_SR	Read-only	–
0x0018	Receive Holding Register	DBGU_RHR	Read-only	0x0
0x001C	Transmit Holding Register	DBGU_THR	Write-only	–
0x0020	Baud Rate Generator Register	DBGU_BRGR	Read/Write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x0040	Chip ID Register	DBGU_CIDR	Read-only	–
0x0044	Chip ID Extension Register	DBGU_EXID	Read-only	–
0x0048	Force NTRST Register	DBGU_FNR	Read/Write	0x0
0x004C - 0x00FC	Reserved	–	–	–
0x0100 - 0x0124	PDC Area	–	–	–

30.5.1 Debug Unit Control Register

Name: DBGU_CR

Access: Write-only

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

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME and OVRE in the DBGU_SR.

30.5.2 Debug Unit Mode Register

Name: DBGU_MR

Access: Read/Write

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

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Space: parity forced to 0
0	1	1	Mark: parity forced to 1
1	x	x	No parity

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo
1	0	Local Loopback
1	1	Remote Loopback

30.5.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Access: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

30.5.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Access: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

30.5.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Access: Read-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Mask End of Receive Transfer Interrupt**
- **ENDTX: Mask End of Transmit Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **TXBUFE: Mask TXBUFE Interrupt**
- **RXBUFF: Mask RXBUFF Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

30.5.6 Debug Unit Status Register

Name: DBGU_SR

Access: Read-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the DBGU_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to DBGU_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

0 = The End of Transfer signal from the receiver Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the receiver Peripheral Data Controller channel is active.

- **ENDTX: End of Transmitter Transfer**

0 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is active.

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

- **TXBUFE: Transmission Buffer Empty**

0 = The buffer empty signal from the transmitter PDC channel is inactive.

1 = The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0 = The buffer full signal from the receiver PDC channel is inactive.

1 = The buffer full signal from the receiver PDC channel is active.

- **COMMTX: Debug Communication Channel Write Status**

0 = COMMTX from the ARM processor is inactive.

1 = COMMTX from the ARM processor is active.

- **COMMRX: Debug Communication Channel Read Status**

0 = COMMRX from the ARM processor is inactive.

1 = COMMRX from the ARM processor is active.

30.5.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Access: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

30.5.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Access: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

30.5.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Access: Read/Write

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

- **CD: Clock Divisor**

CD	Baud Rate Clock
0	Disabled
1	MCK
2 to 65535	MCK / (CD x 16)

30.5.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

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**
- **EPROC: Embedded Processor**

EPROC			Processor
0	0	1	ARM946ES™
0	1	0	ARM7TDMI®
1	0	0	ARM920T™
1	0	1	ARM926EJS™

- **NVPSIZ: Nonvolatile Program Memory Size**

NVPSIZ				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

NVPSIZ2				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **SRAMSIZ: Internal SRAM Size**

SRAMSIZ				Size
0	0	0	0	Reserved
0	0	0	1	1K bytes
0	0	1	0	2K bytes
0	0	1	1	6K bytes
0	1	0	0	112K bytes
0	1	0	1	4K bytes
0	1	1	0	80K bytes
0	1	1	1	160K bytes
1	0	0	0	8K bytes
1	0	0	1	16K bytes
1	0	1	0	32K bytes
1	0	1	1	64K bytes
1	1	0	0	128K bytes
1	1	0	1	256K bytes
1	1	1	0	96K bytes
1	1	1	1	512K bytes

- **ARCH: Architecture Identifier**

ARCH		
Hex	**Bin**	**Architecture**
0x19	0001 1001	AT91SAM9xx Series
0x29	0010 1001	AT91SAM9XExx Series
0x34	0011 0100	AT91x34 Series
0x37	0011 0111	CAP7 Series
0x39	0011 1001	CAP9 Series
0x3B	0011 1011	CAP11 Series
0x40	0100 0000	AT91x40 Series
0x42	0100 0010	AT91x42 Series
0x55	0101 0101	AT91x55 Series
0x60	0110 0000	AT91SAM7Axx Series
0x61	0110 0001	AT91SAM7AQxx Series
0x63	0110 0011	AT91x63 Series
0x70	0111 0000	AT91SAM7Sxx Series
0x71	0111 0001	AT91SAM7XCxx Series
0x72	0111 0010	AT91SAM7SExx Series
0x73	0111 0011	AT91SAM Lxx Series
0x75	0111 0101	AT91SAM7Xxx Series
0x92	1001 0010	AT91x92 Series
0xF0	1111 0000	AT75Cxx Series

- **NVPTYP: Nonvolatile Program Memory Type**

NVPTYP			Memory
0	0	0	ROM
0	0	1	ROMless or on-chip Flash
1	0	0	SRAM emulating ROM
0	1	0	Embedded Flash Memory
0	1	1	ROM and Embedded Flash Memory NVPSIZ is ROM size NVPSIZ2 is Flash size

- **EXT: Extension Flag**

0 = Chip ID has a single register definition without extension

1 = An extended Chip ID exists.

30.5.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Access: Read-only

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

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

30.5.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Access: Read/Write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor’s TAP controller is driven by the power_on_reset signal.

1 = NTRST of the ARM processor’s TAP controller is held low.

31. Serial Peripheral Interface (SPI)

31.1 Overview

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

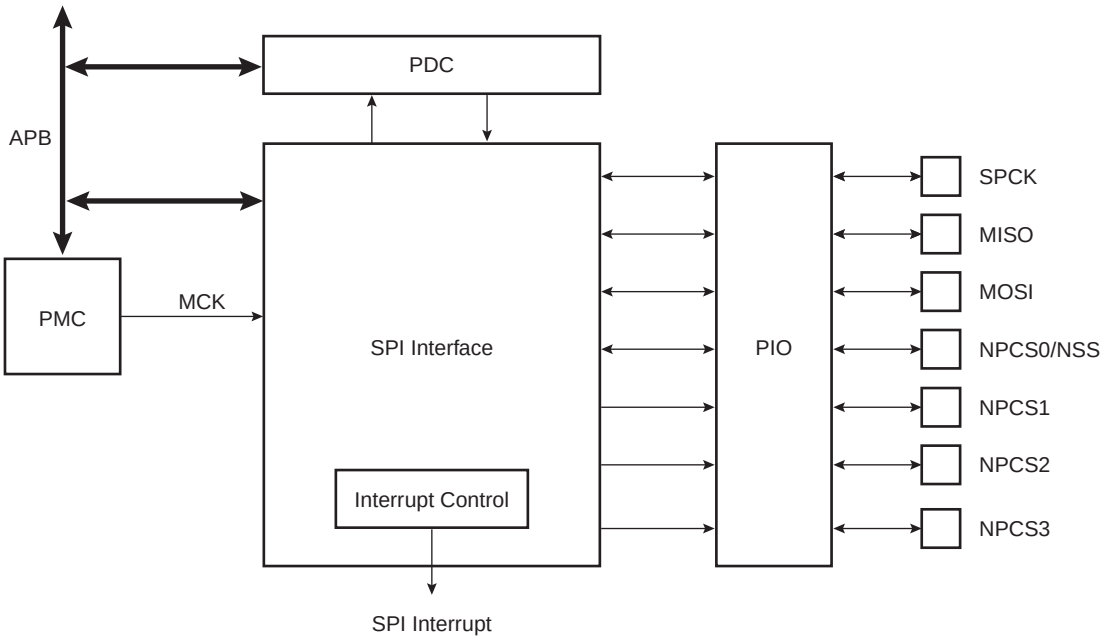
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

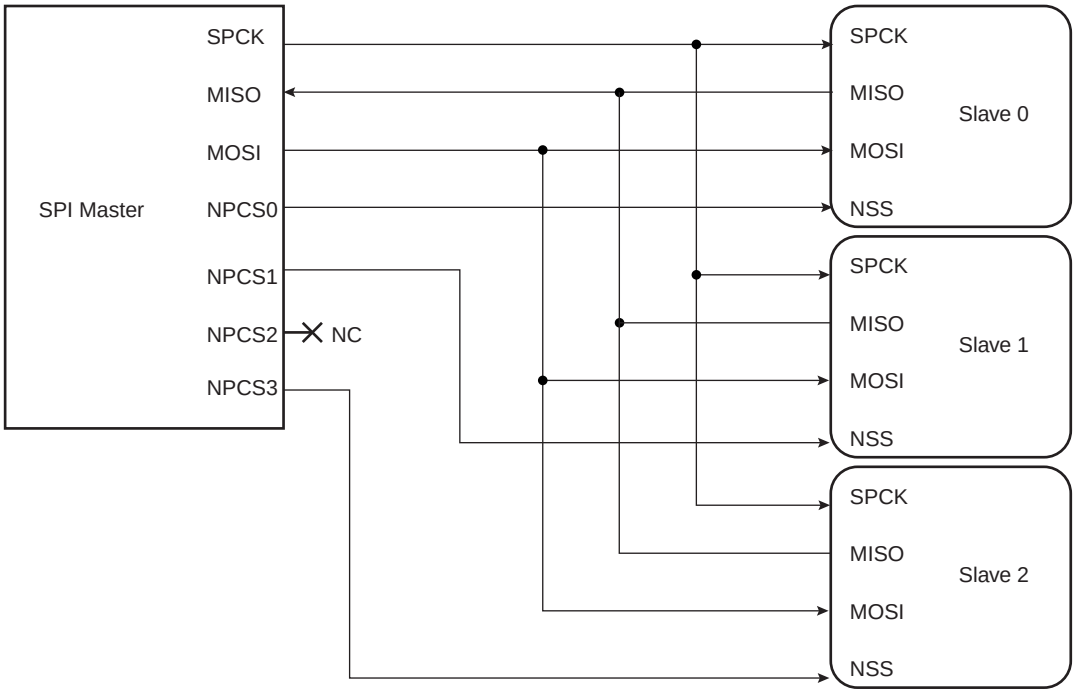
31.2 Block Diagram

Figure 31-1. Block Diagram



31.3 Application Block Diagram

Figure 31-2. Application Block Diagram: Single Master/Multiple Slave Implementation



31.4 Signal Description

Table 31-1. 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

31.5 Product Dependencies

31.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

31.5.2 Power Management

The SPI may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

31.5.3 Interrupt

The SPI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the SPI interrupt requires programming the AIC before configuring the SPI.

31.6 Functional Description

31.6.1 Modes of Operation

The SPI operates in Master Mode or in Slave Mode.

Operation in Master Mode is programmed by writing at 1 the MSTR bit in the Mode Register. The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MSTR bit is written at 0, the SPI operates in Slave Mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in Master Mode.

31.6.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Chip Select Register. The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

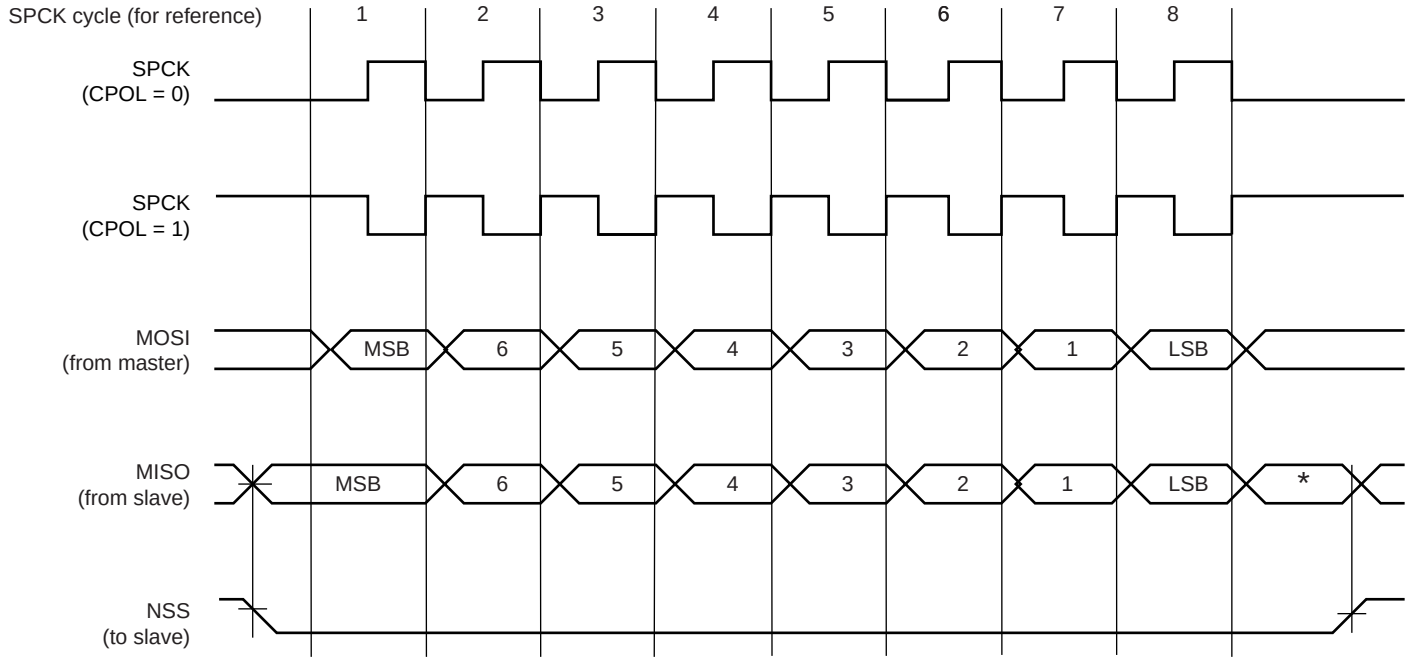
Table 31-2 shows the four modes and corresponding parameter settings.

Table 31-2. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

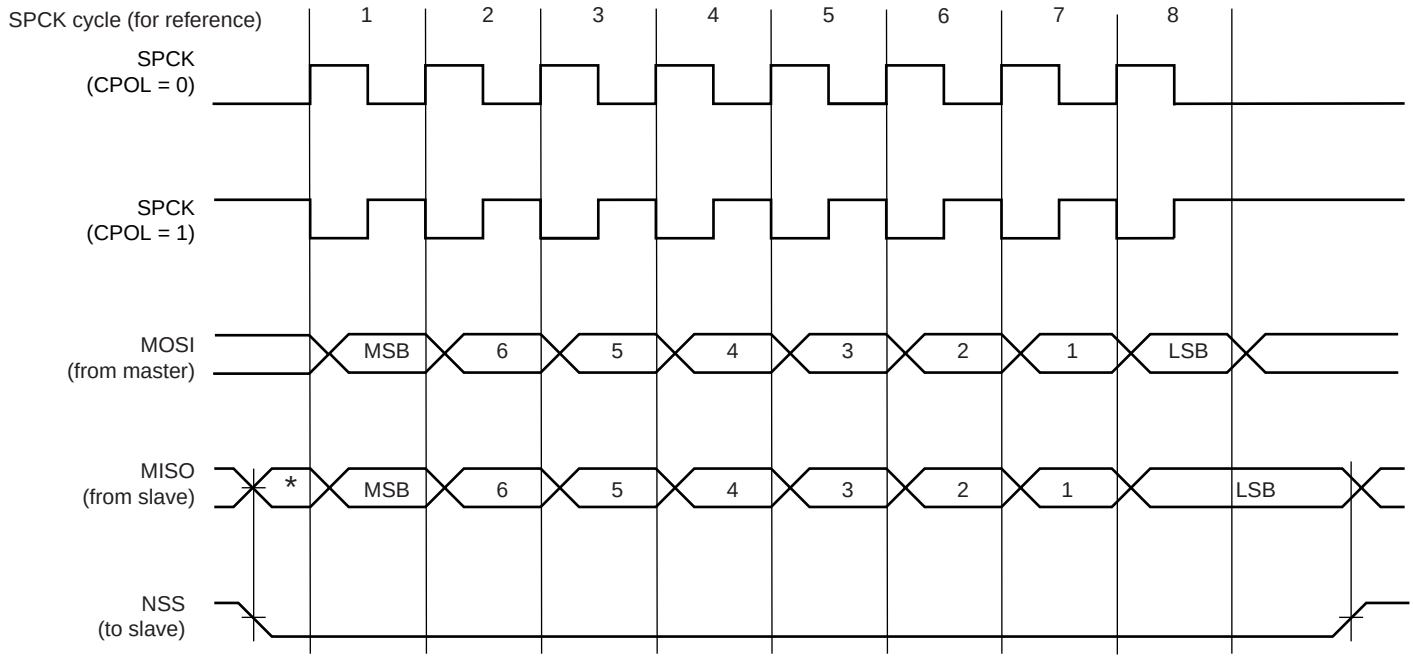
Figure 31-3 and Figure 31-4 show examples of data transfers.

Figure 31-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Figure 31-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

31.6.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writing the TDR, the PCS field must be set in order to select a slave.

If new data is written in SPI_TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to SPI_RDR, the data in SPI_TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in SPI_TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SPI_SR). When new data is written in SPI_TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit PDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

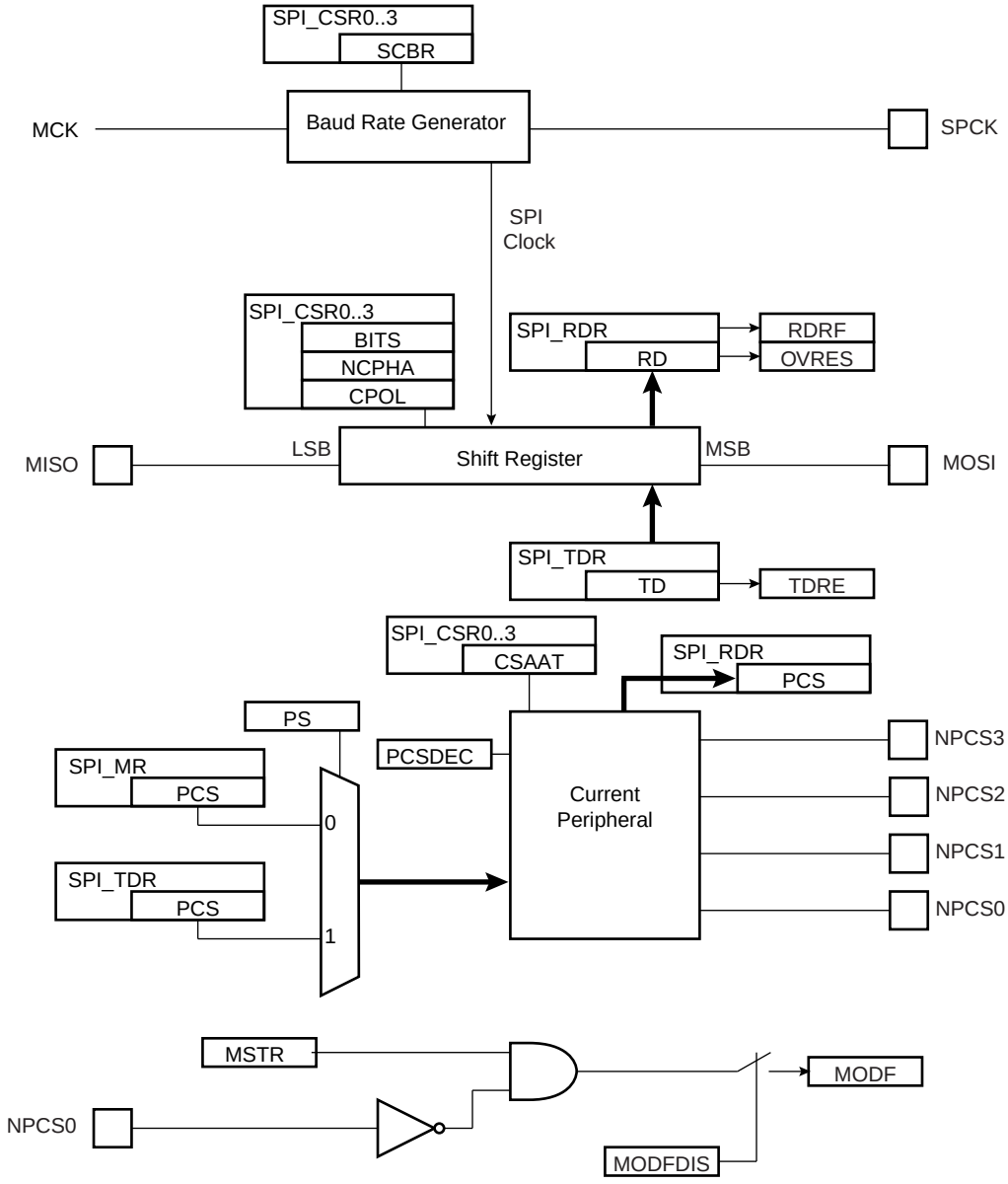
The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 31-5 on page 329 shows a block diagram of the SPI when operating in Master Mode. Figure 31-6 on page 330 shows a flow chart describing how transfers are handled.

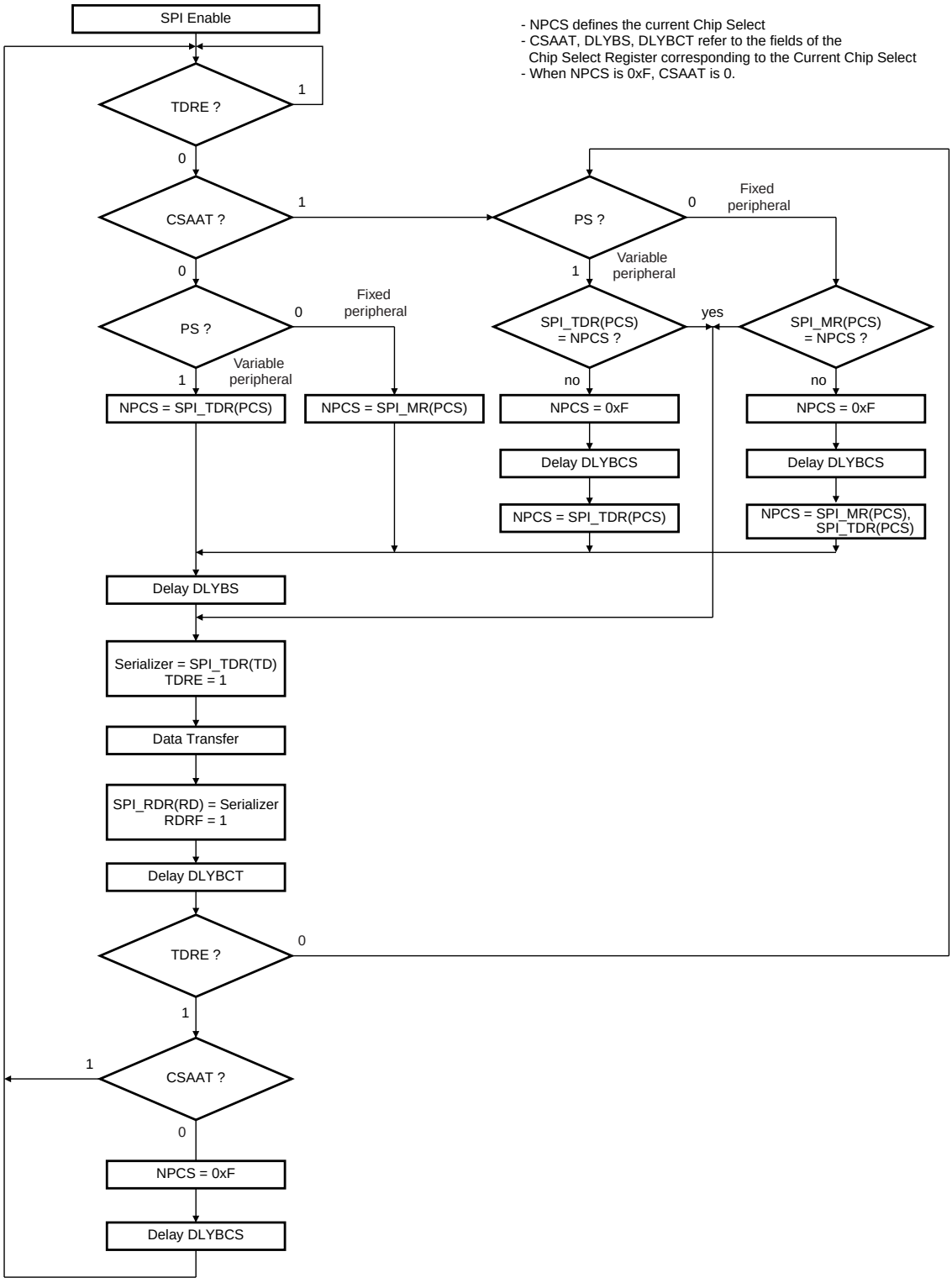
31.6.3.1 Master Mode Block Diagram

Figure 31-5. Master Mode Block Diagram



31.6.3.2 Master Mode Flow Diagram

Figure 31-6. Master Mode Flow Diagram S



31.6.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the Master Clock (MCK), by a value between 1 and 255.

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255.

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field of the Chip Select Registers. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

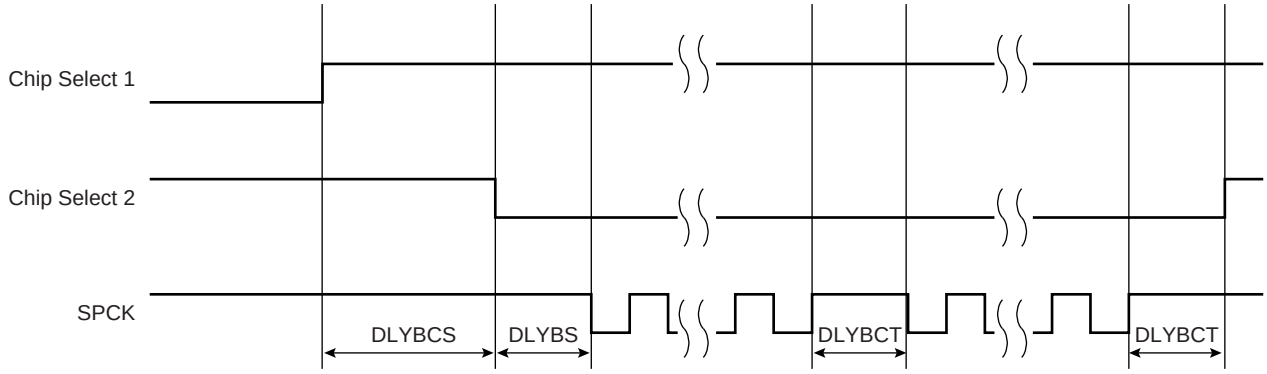
31.6.3.4 Transfer Delays

Figure 31-7 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing the DLYBCS field in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the DLYBCT field. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 31-7. Programmable Delays



31.6.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral

- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Fixed Peripheral Select is activated by writing the PS bit to zero in SPI_MR (Mode Register). In this case, the current peripheral is defined by the PCS field in SPI_MR and the PCS field in the SPI_TDR has no effect.

Variable Peripheral Select is activated by setting PS bit to one. The PCS field in SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the Mode Register. Data written in SPI_TDR is 32 bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the PDC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 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.

31.6.3.6 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

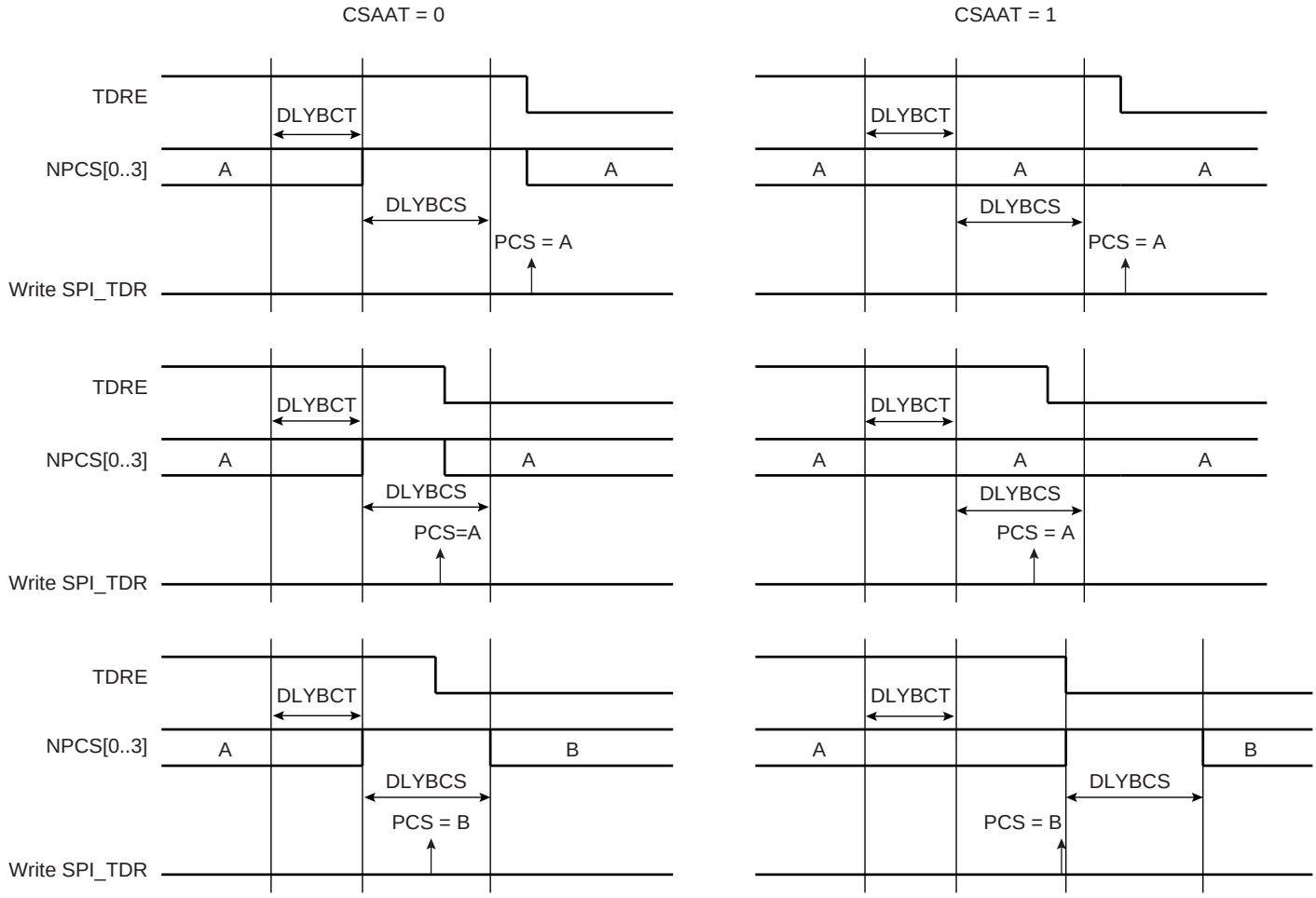
31.6.3.7 Peripheral Deselection

When operating normally, as soon as the transfer of the last data written in SPI_TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

Figure 31-8 shows different peripheral deselection cases and the effect of the CSAAT bit.

Figure 31-8. Peripheral Deselection



31.6.3.8 Mode Fault Detection

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal. NPCS0, MOSI, MISO and SPCK must be configured in open drain through the PIO controller, so that external pull up resistors are needed to guarantee high level.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI is automatically disabled until re-enabled by writing the SPIEN bit in the SPI_CR (Control Register) at 1.

By default, the Mode Fault detection circuitry is enabled. The user can disable Mode Fault detection by setting the MODFDIS bit in the SPI Mode Register (SPI_MR).

31.6.4 SPI Slave Mode

When operating in Slave Mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits

defined by the BITS field of the Chip Select Register 0 (SPI_CSR0). These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits of the SPI_CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select Registers have no effect when the SPI is programmed in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

When all the bits are processed, the received data is transferred in the Receive Data Register and the RDRF bit rises. If RDRF is already high when the data is transferred, the Overrun bit rises and the data transfer to SPI_RDR is aborted.

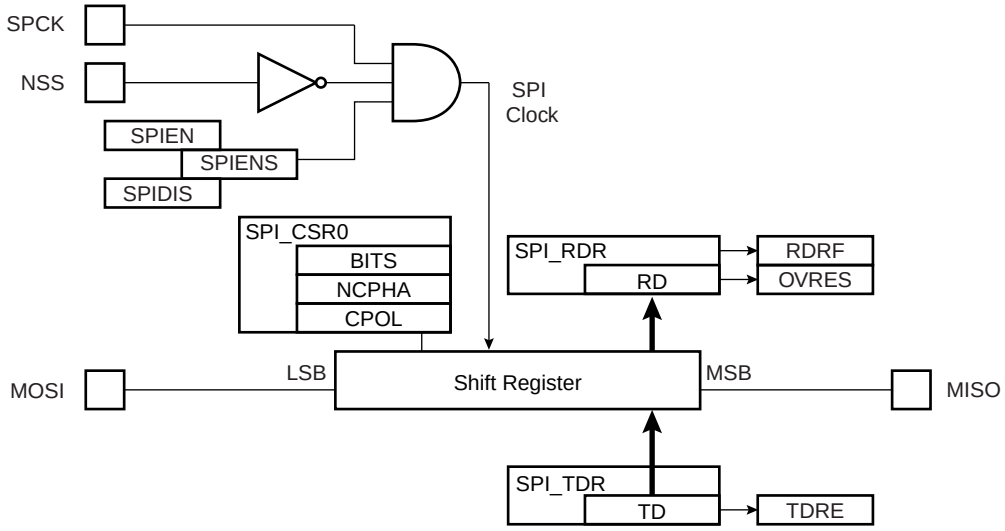
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets at 0.

When a first data is written in SPI_TDR, it is transferred immediately in the Shift Register and the TDRE bit rises. If new data is written, it remains in SPI_TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in SPI_TDR is transferred in the Shift Register and the TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the Transmit Data Register. In case no character is ready to be transmitted, i.e. no character has been written in SPI_TDR since the last load from SPI_TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted.

Figure 31-9 shows a block diagram of the SPI when operating in Slave Mode.

Figure 31-9. Slave Mode Functional Block Diagram



31.7 Serial Peripheral Interface (SPI) User Interface

Table 31-3. SPI Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	---
0x04	Mode Register	SPI_MR	Read/Write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	---
0x10	Status Register	SPI_SR	Read-only	0x000000F0
0x14	Interrupt Enable Register	SPI_IER	Write-only	---
0x18	Interrupt Disable Register	SPI_IDR	Write-only	---
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20 - 0x2C	Reserved			
0x30	Chip Select Register 0	SPI_CSR0	Read/Write	0x0
0x34	Chip Select Register 1	SPI_CSR1	Read/Write	0x0
0x38	Chip Select Register 2	SPI_CSR2	Read/Write	0x0
0x3C	Chip Select Register 3	SPI_CSR3	Read/Write	0x0
0x004C - 0x00F8	Reserved	–	–	–
0x004C - 0x00FC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC			

31.7.1 SPI Control Register

Name: SPI_CR

Access: Write-only

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

- **SPIEN: SPI Enable**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0 = No effect.

1 = Disables the SPI.

As soon as SPIDIS is set, SPI finishes its transfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in slave mode after software reset.

PDC channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

31.7.2 SPI Mode Register

Name: SPI_MR

Access: Read/Write

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

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The Chip Select Registers define the characteristics of the 15 chip selects according to the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled.

LLB controls the local loopback on the data serializer for testing in Master Mode only. (MISO is internally connected on MOSI.)

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = 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:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

31.7.3 SPI Receive Data Register

Name: SPI_RDR

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.

31.7.4 SPI Transmit Data Register

Name: SPI_TDR

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).

31.7.5 SPI Status Register

Name: SPI_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SPIENS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SPI_SR.

1 = A Mode Fault occurred since the last read of the SPI_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI_SR.

1 = An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

1 = The Receive Counter Register has reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

1 = The Transmit Counter Register has reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

- **RXBUFF: RX Buffer Full**

0 = SPI_RCR(1) or SPI_RNCR(1) has a value other than 0.

1 = Both SPI_RCR(1) and SPI_RNCR(1) have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI_TCR(1) or SPI_TNCR(1) has a value other than 0.

1 = Both SPI_TCR(1) and SPI_TNCR(1) have a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in SPI_TDR.

1 = SPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

Note: 1. SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR are physically located in the PDC.

31.7.6 SPI Interrupt Enable Register

Name: SPI_IER

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**
- **NSSR: NSS Rising Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

31.7.7 SPI Interrupt Disable Register

Name: SPI_IDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**
- **MODF: Mode Fault Error Interrupt Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**
- **NSSR: NSS Rising Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

31.7.8 SPI Interrupt Mask Register

Name: SPI_IMR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**
- **NSSR: NSS Rising Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

31.7.9 SPI Chip Select Register

Name: SPI_CSR0... SPI_CSR3

Access: Read/Write

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

- **CPOL: Clock Polarity**

0 = The inactive state value of SPCK is logic level zero.

1 = The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSAAT: Chip Select Active After Transfer**

0 = The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1 = The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

- **BITS: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16

BITS	Bits Per Transfer
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS valid to the first valid SPCK transition.

When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

Otherwise, the following equations determine the delay:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

32. Two Wire Interface (TWI)

32.1 Overview

The Atmel Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and I²C compatible device such as Real Time Clock (RTC), Dot Matrix/Graphic LCD Controllers and Temperature Sensor, to name but a few. The TWI is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported. Arbitration of the bus is performed internally and puts the TWI in slave mode automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Below, Table 32-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I2C compatible device.

Table 32-1. Atmel TWI compatibility with i2C Standard

I2C Standard	Atmel TWI
Standard Mode Speed (100 KHz)	Supported
Fast Mode Speed (400 KHz)	Supported
7 or 10 bits Slave Addressing	Supported
START BYTE(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope control and input filtering (Fast mode)	Not Supported
Clock stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

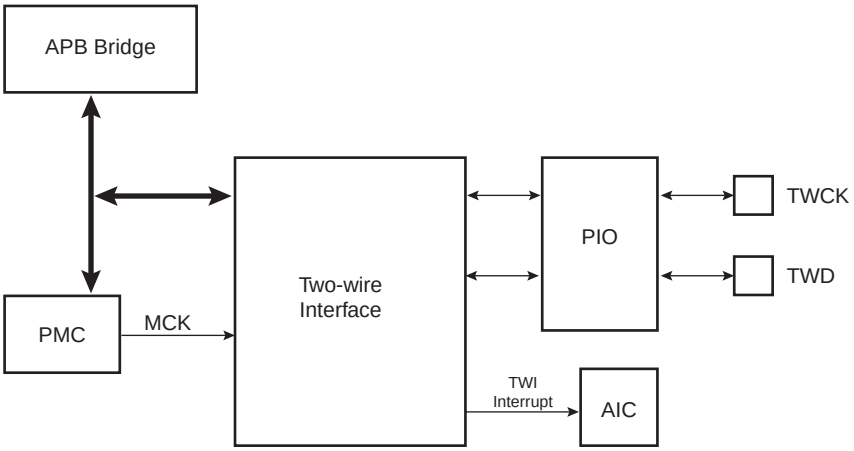
32.2 List of Abbreviations

Table 32-2. 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

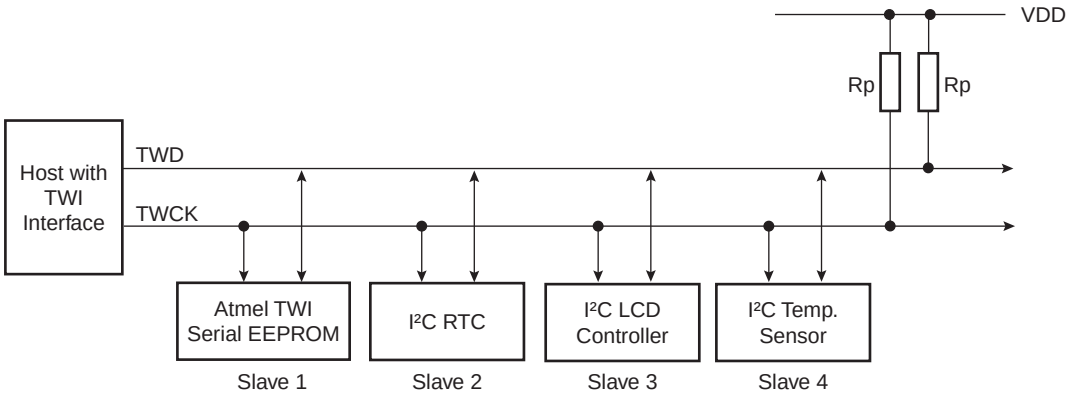
32.3 Block Diagram

Figure 32-1. Block Diagram



32.4 Application Block Diagram

Figure 32-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

32.4.1 I/O Lines Description

Table 32-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

32.5 Product Dependencies

32.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 32-2 on page 350). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following steps:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

32.5.2 Power Management

- Enable the peripheral clock.

The TWI interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TWI clock.

32.5.3 Interrupt

The TWI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). In order to handle interrupts, the AIC must be programmed before configuring the TWI.

32.6 Functional Description

32.6.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 32-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 32-3).

- 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 32-3. START and STOP Conditions

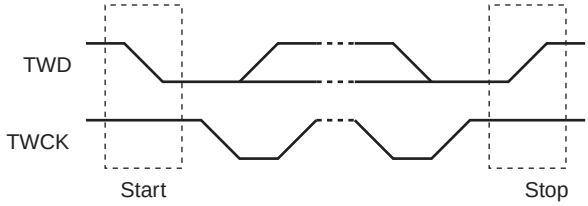
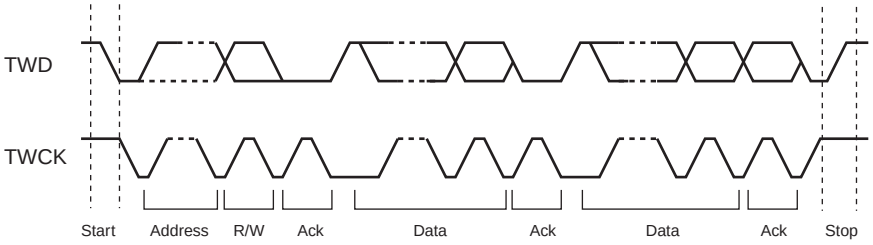


Figure 32-4. Transfer Format



32.6.2 Modes of Operation

The TWI has six modes of operations:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode
- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following chapters.

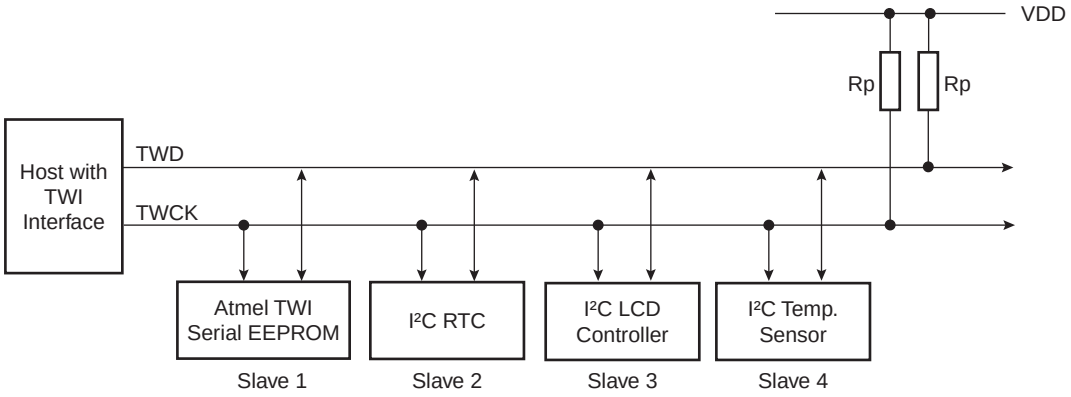
32.7 Master Mode

32.7.1 Definition

The Master is the device which starts a transfer, generates a clock and stops it.

32.7.2 Application Block Diagram

Figure 32-5. Master Mode Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

32.7.3 Programming Master Mode

The following registers have to be programmed before entering Master mode:

1. DADR (+ IADRSZ + IADR if a 10 bit device is addressed): The device address is used to access slave devices in read or write mode.
2. CKDIV + CHDIV + CLDIV: Clock Waveform.
3. SVDIS: Disable the slave mode.
4. MSEN: Enable the master mode.

32.7.4 Master Transmitter Mode

After the master initiates a Start condition when writing into the Transmit Holding Register, TWI_THR, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction, 0 in this case (MREAD = 0 in TWI_MMR).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the Not Acknowledge bit (**NACK)** in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI_IER). If the slave acknowledges the byte, the data written in the TWI_THR, is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR. When no more data is written into the TWI_THR, the master generates a stop condition to end the transfer. The end of the complete transfer is marked by the TWI_TXCOMP bit set to one. See Figure 32-6, Figure 32-7, and Figure 32-8.

Figure 32-6. Master Write with One Data Byte

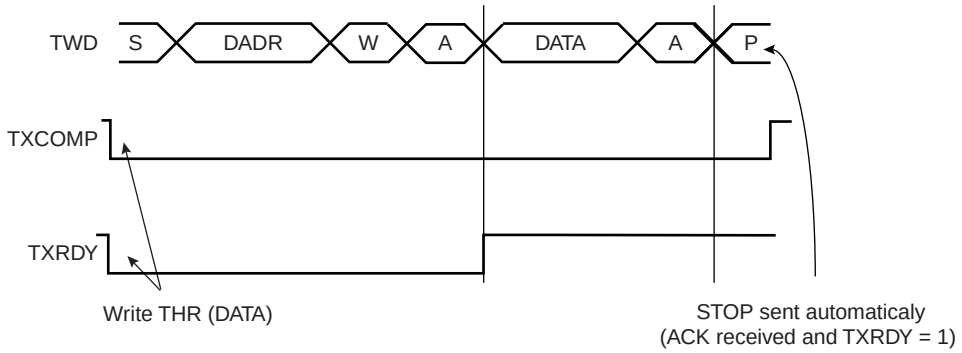


Figure 32-7. Master Write with Multiple Data Byte

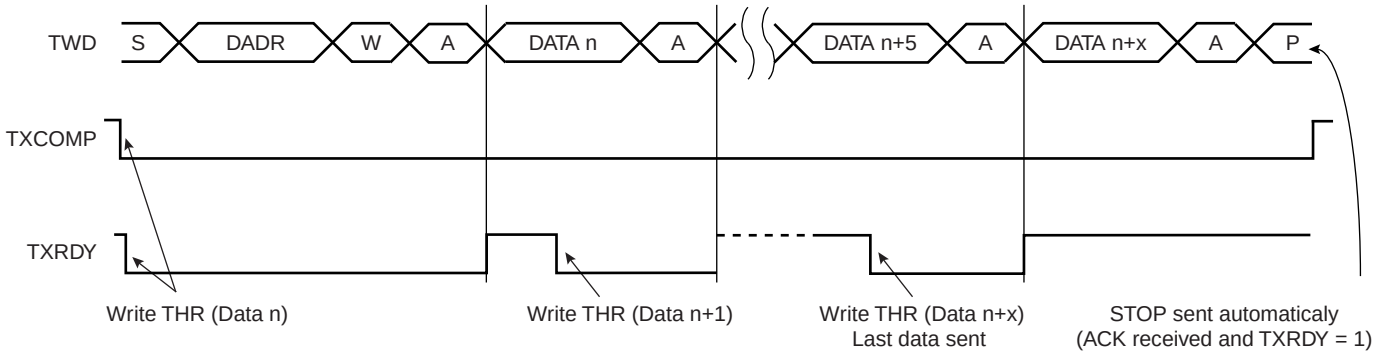
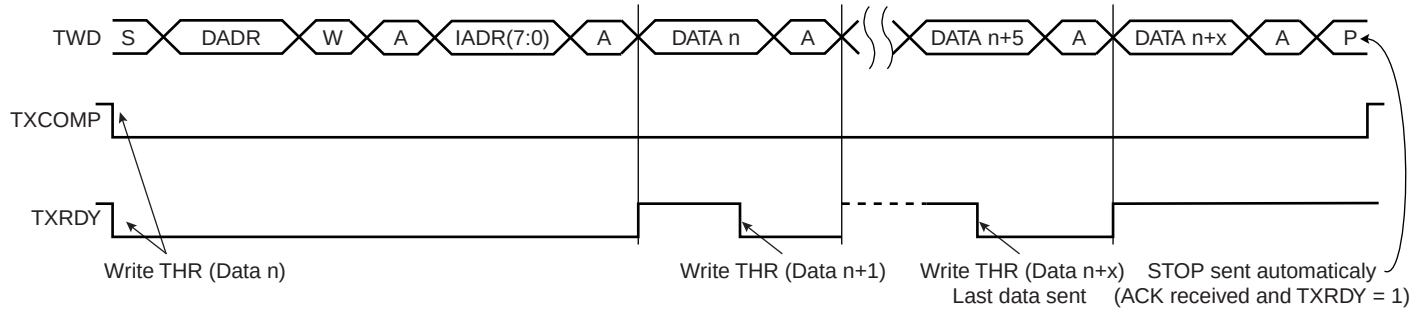


Figure 32-8. Master Write with One Byte Internal Address and Multiple Data Bytes



32.7.5 Master Receiver Mode

The read sequence begins by setting the START bit. After the start condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction, 1 in this case (MREAD = 1 in TWI_MMR). During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the **NACK** bit in the status register if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data, after the stop condition. See Figure 32-9. 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 32-9. 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 32-10. For Internal Address usage see Section 32.7.6.

Figure 32-9. Master Read with One Data Byte

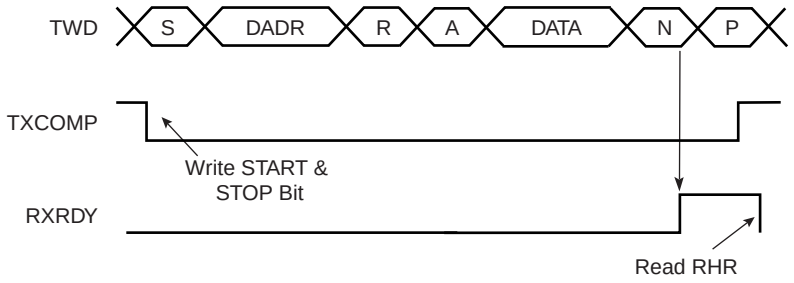
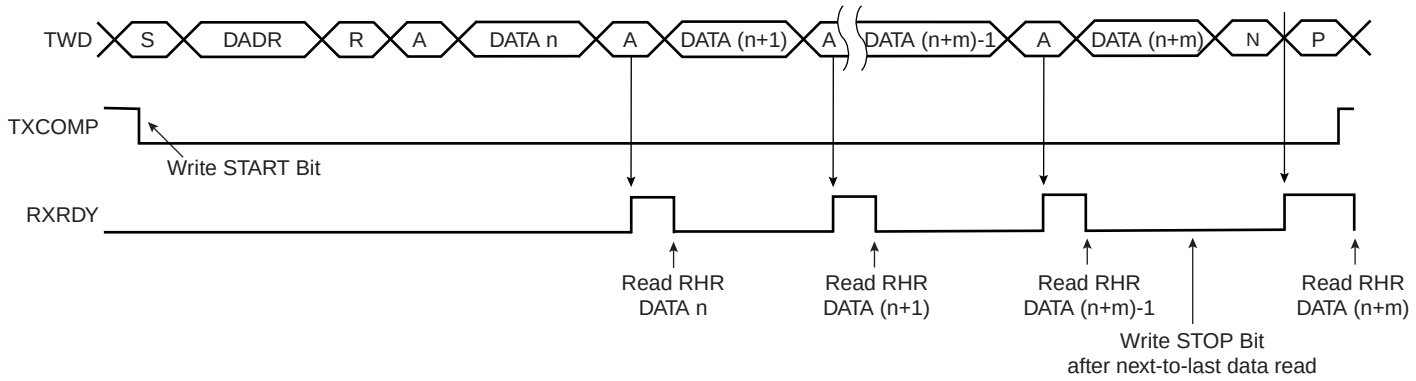


Figure 32-10. Master Read with Multiple Data Bytes



32.7.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

32.7.6.1 7-bit Slave Addressing

When Addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, within a memory page location in a serial memory, for example. When performing read operations with an internal address, the TWI performs a write operation to set the internal address into the slave device, and then switch to Master Receiver mode. Note that the second start condition (after sending the IADR) is sometimes called “repeated start” (Sr) in I2C fully-compatible devices. See Figure 32-12. See Figure 32-11 and Figure 32-13 for Master Write operation with internal address.

The three internal address bytes are configurable through the Master Mode register (TWI_MMR).

If the slave device supports only a 7-bit address, i.e. no internal address, **IADRSZ** must be set to 0.

In the figures below the following abbreviations are used:

- S Start
- Sr Repeated Start
- P Stop
- W Write
- R Read
- A Acknowledge
- N Not Acknowledge
- DADR Device Address
- IADR Internal Address

Figure 32-11. Master Write with One, Two or Three Bytes Internal Address and One Data Byte

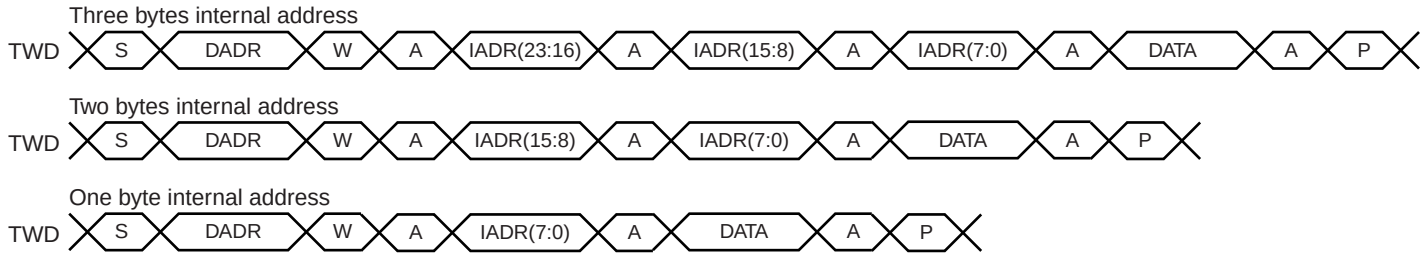
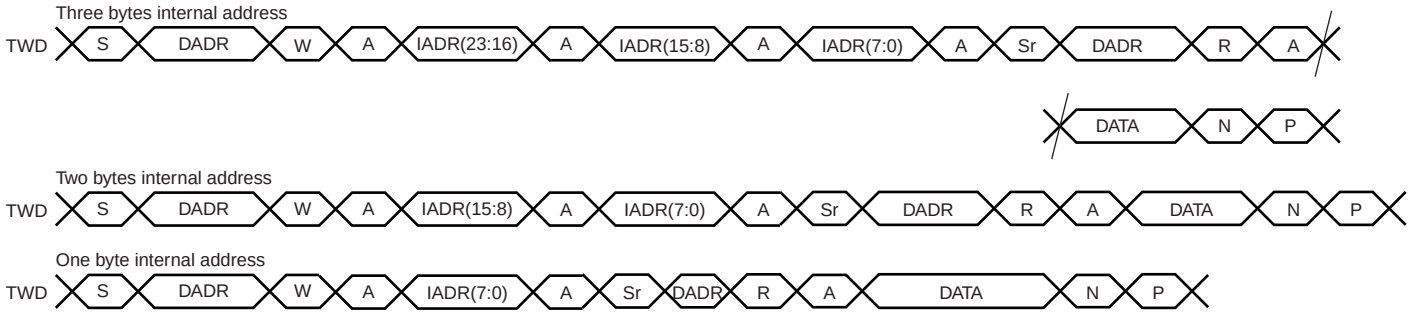


Figure 32-12. Master Read with One, Two or Three Bytes Internal Address and One Data Byte



32.7.6.2 10-bit Slave Addressing

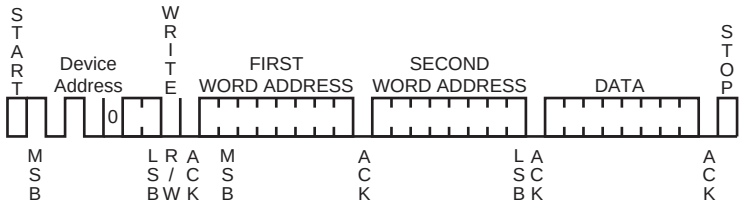
For a slave address higher than 7 bits, the user must configure the address size **(IADRSZ**) and set the other slave address bits in the internal address register (TWI_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 32-13 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 32-13. Internal Address Usage



32.7.7 Read-write Flowcharts

The following flowcharts shown in Figure 32-14, Figure 32-15 on page 359, Figure 32-16 on page 360, Figure 32-17 on page 361, Figure 32-18 on page 362 and Figure 32-19 on page 363 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 32-14. TWI Write Operation with Single Data Byte without Internal Address

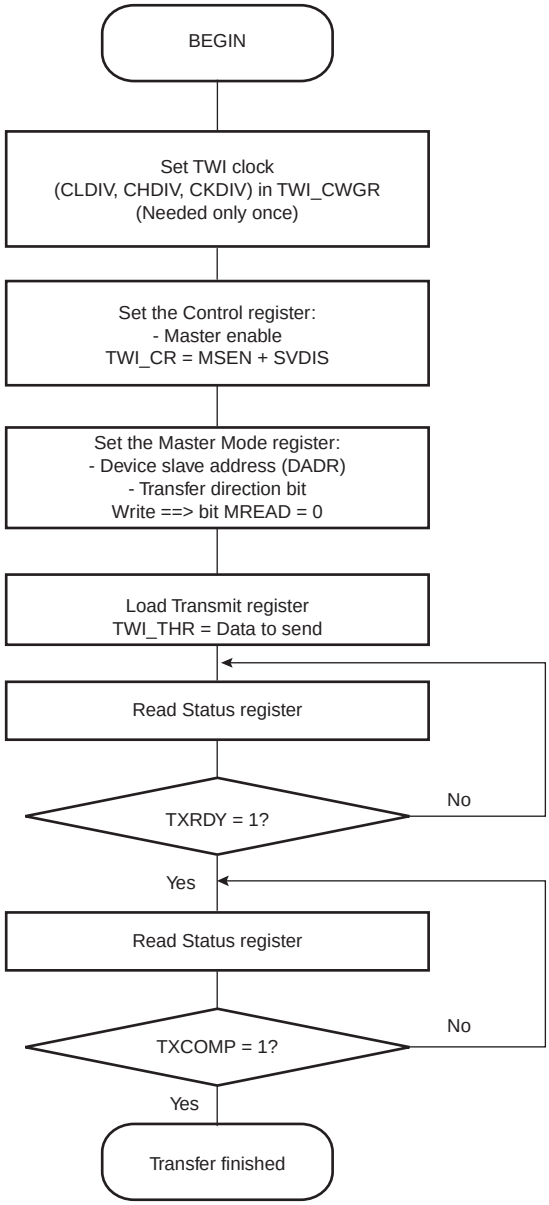


Figure 32-15. TWI Write Operation with Single Data Byte and Internal Address

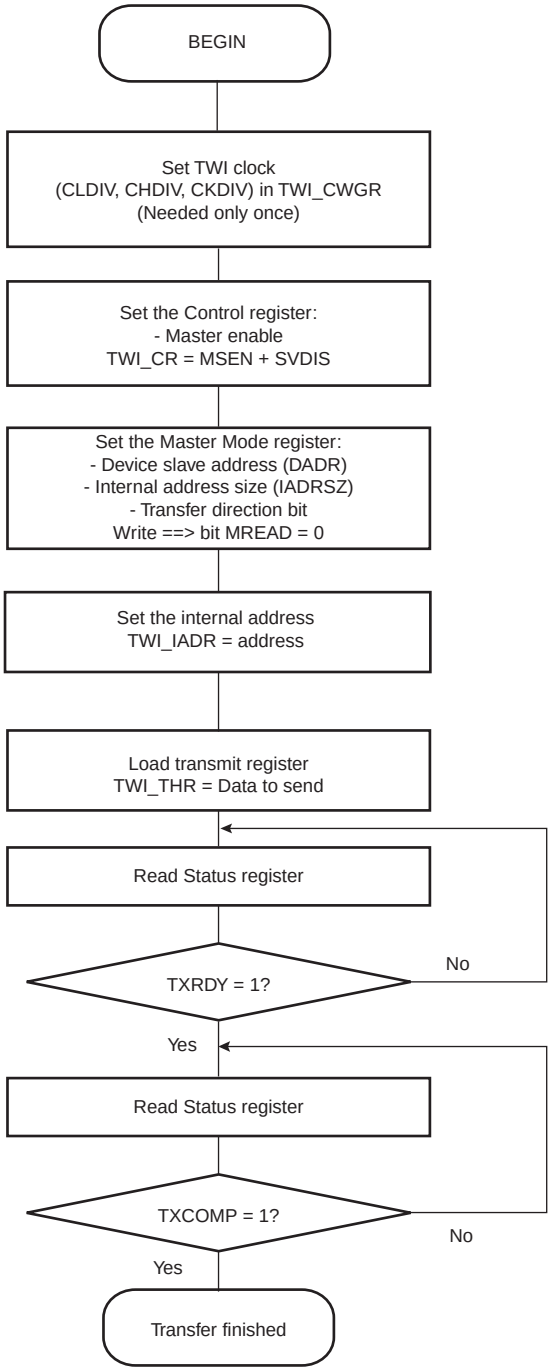


Figure 32-16. TWI Write Operation with Multiple Data Bytes with or without Internal Address

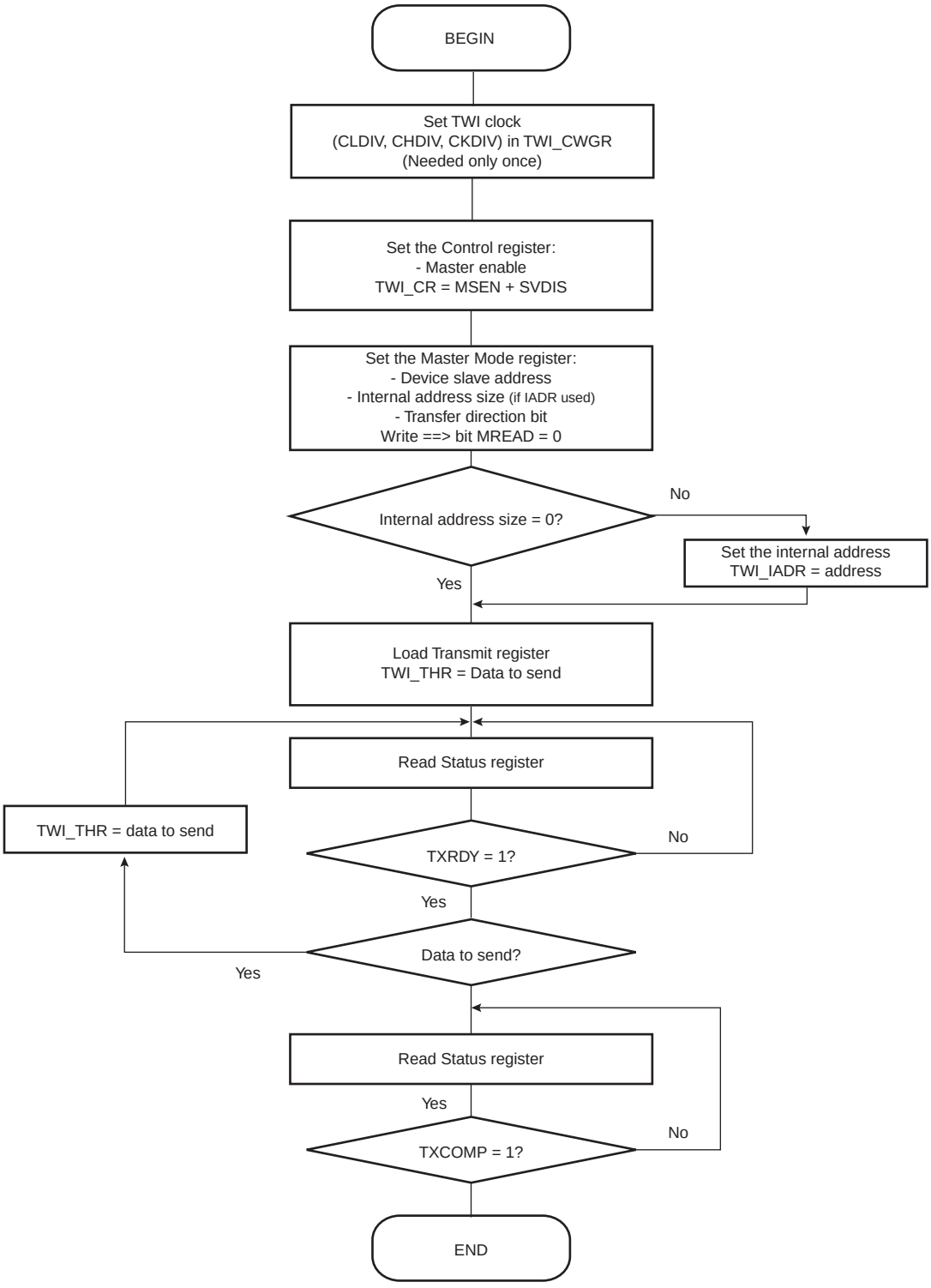


Figure 32-17. TWI Read Operation with Single Data Byte without Internal Address

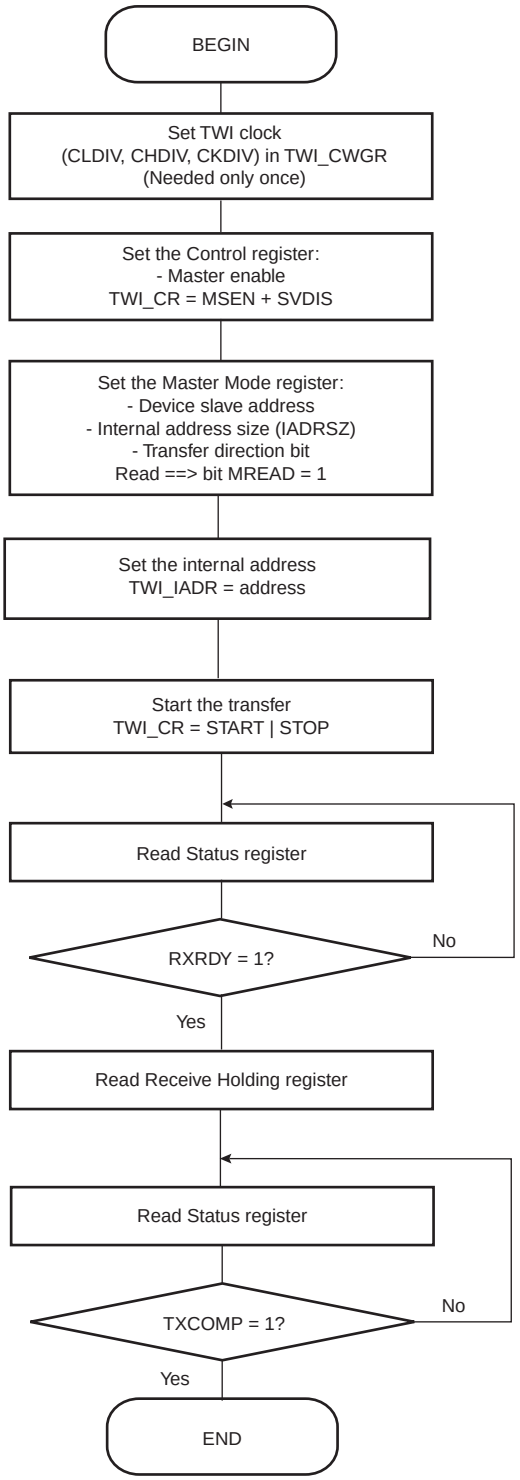


Figure 32-18. TWI Read Operation with Single Data Byte and Internal Address

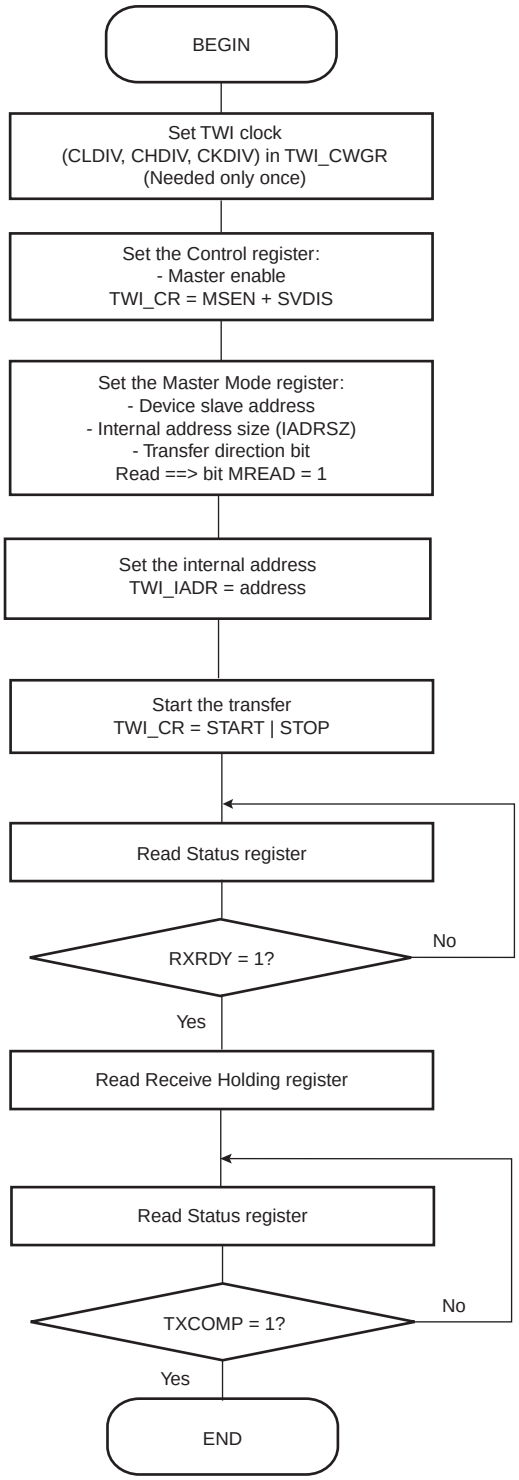
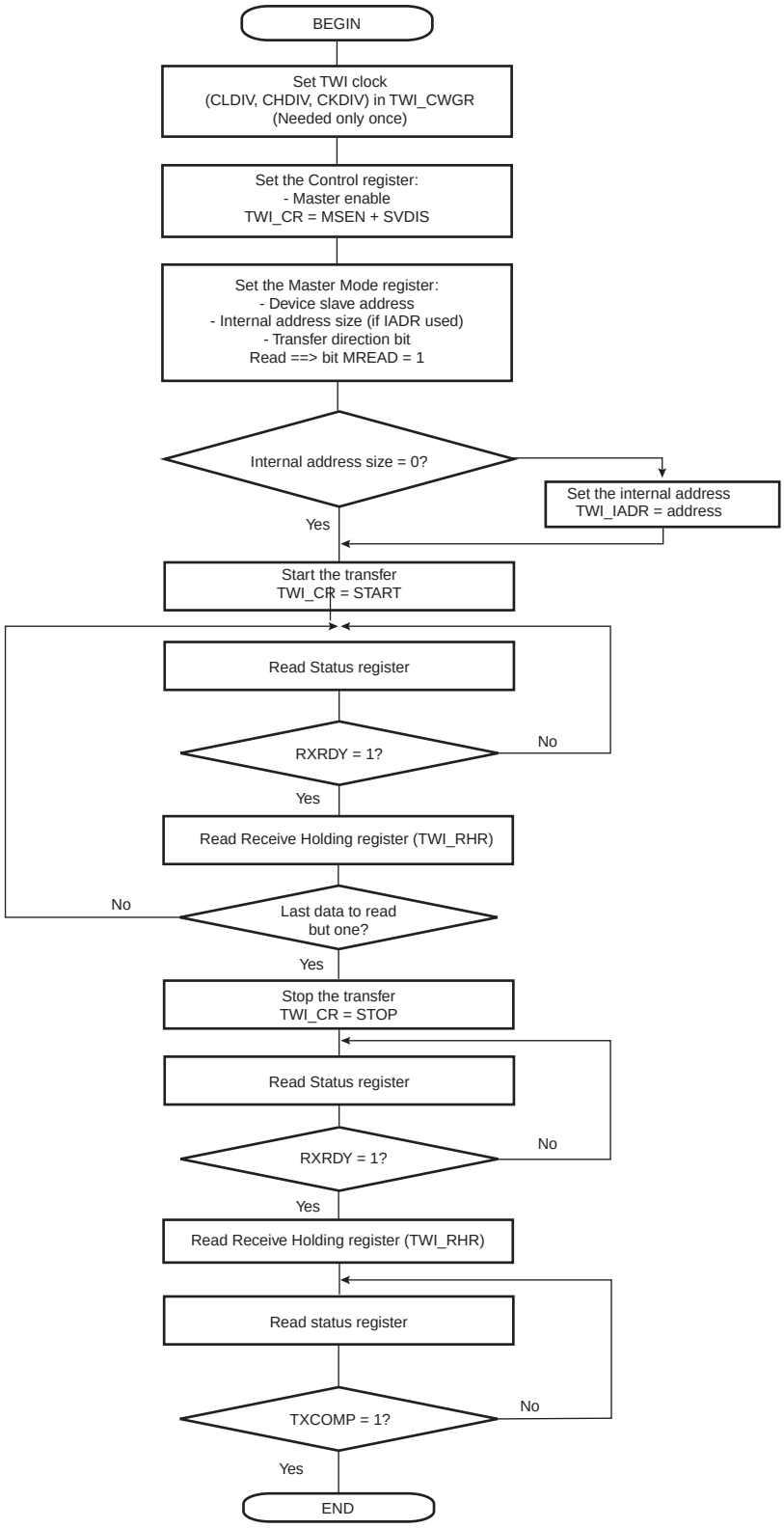


Figure 32-19. TWI Read Operation with Multiple Data Bytes with or without Internal Address



32.8 Multi-master Mode

32.8.1 Definition

More than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master who has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in Figure 32-21 on page 365.

32.8.2 Different Multi-master Modes

Two multi-master modes may be distinguished:

1. TWI is considered as a Master only and will never be addressed.
2. TWI may be either a Master or a Slave and may be addressed.

Note: In both Multi-master modes arbitration is supported.

32.8.2.1 TWI as Master Only

In this mode, TWI is considered as a Master only (MSEN is always at one) and must be driven like a Master with the ARBLST (ARBitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the programmer must reinitiate the data transfer.

If the user starts a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWI automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 32-20 on page 365).

Note: The state of the bus (busy or free) is not indicated in the user interface.

32.8.2.2 TWI as Master or Slave

The automatic reversal from Master to Slave is not supported in case of a lost arbitration.

Then, in the case where TWI may be either a Master or a Slave, the programmer must manage the pseudo Multi-master mode described in the steps below.

1. Program TWI in Slave mode (SADR + MSDIS + SVEN) and perform Slave Access (if TWI is addressed).
2. If TWI has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, TWI scans the bus in order to detect if it is busy or free. When the bus is considered as free, TWI initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWI in Slave mode in the case where the Master that won the arbitration wanted to access the TWI.
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 32-20. Programmer Sends Data While the Bus is Busy

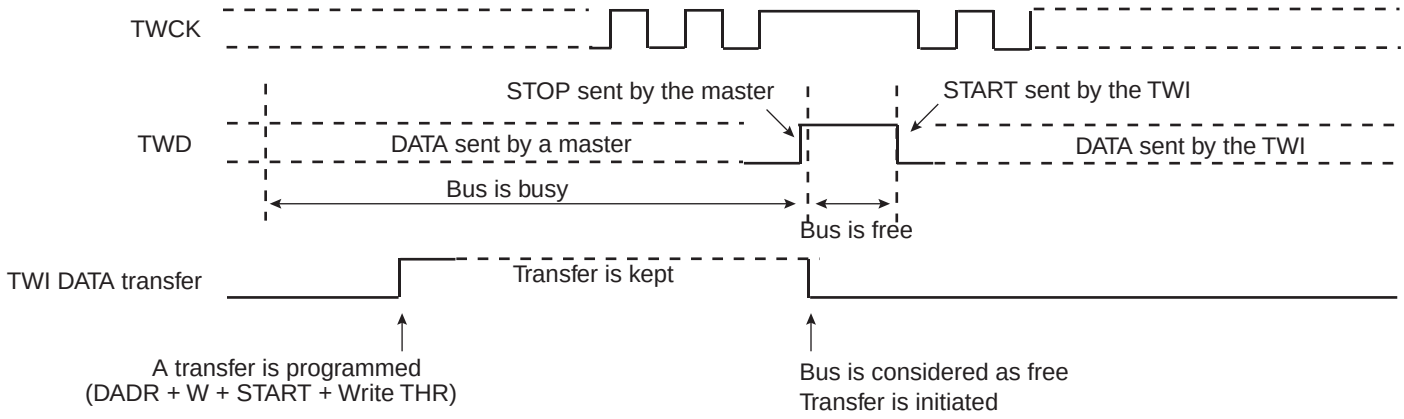
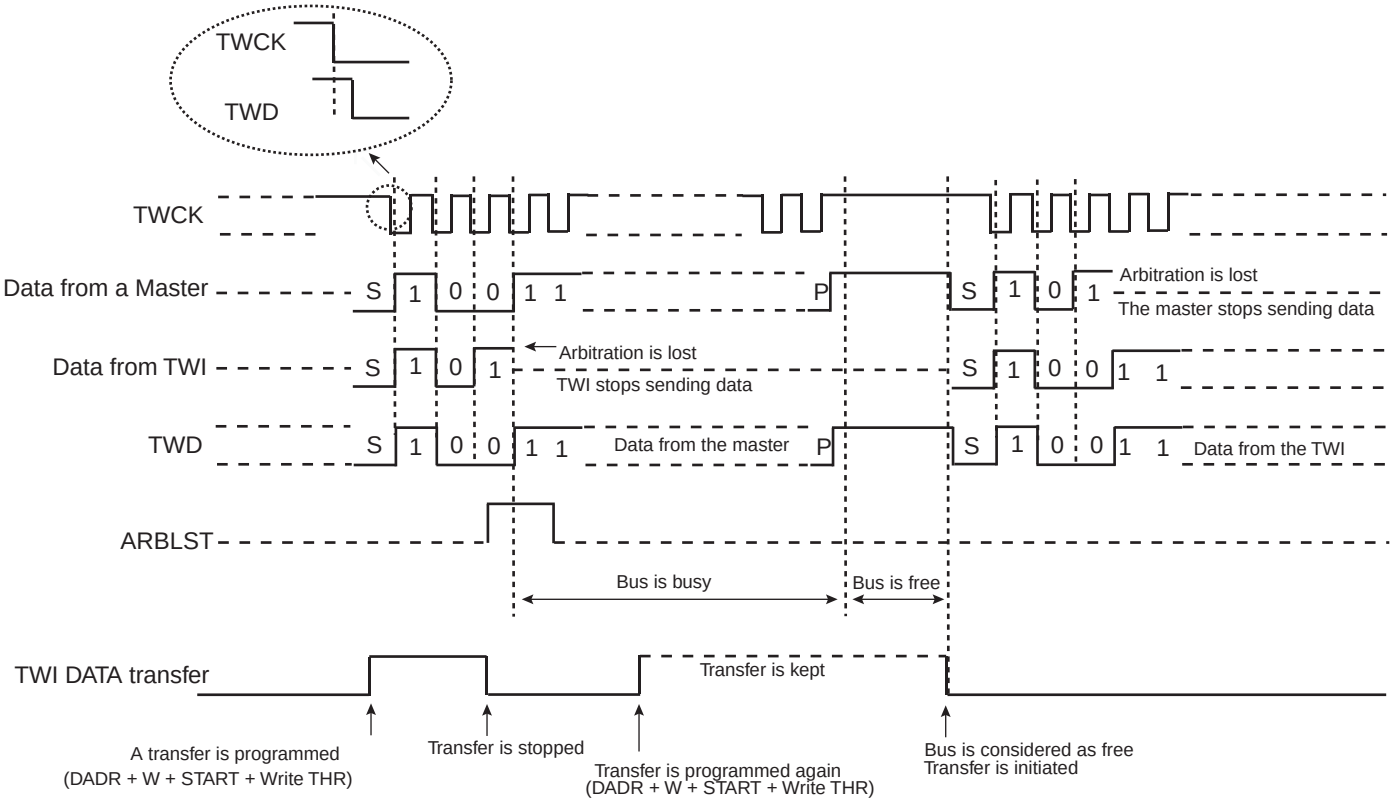
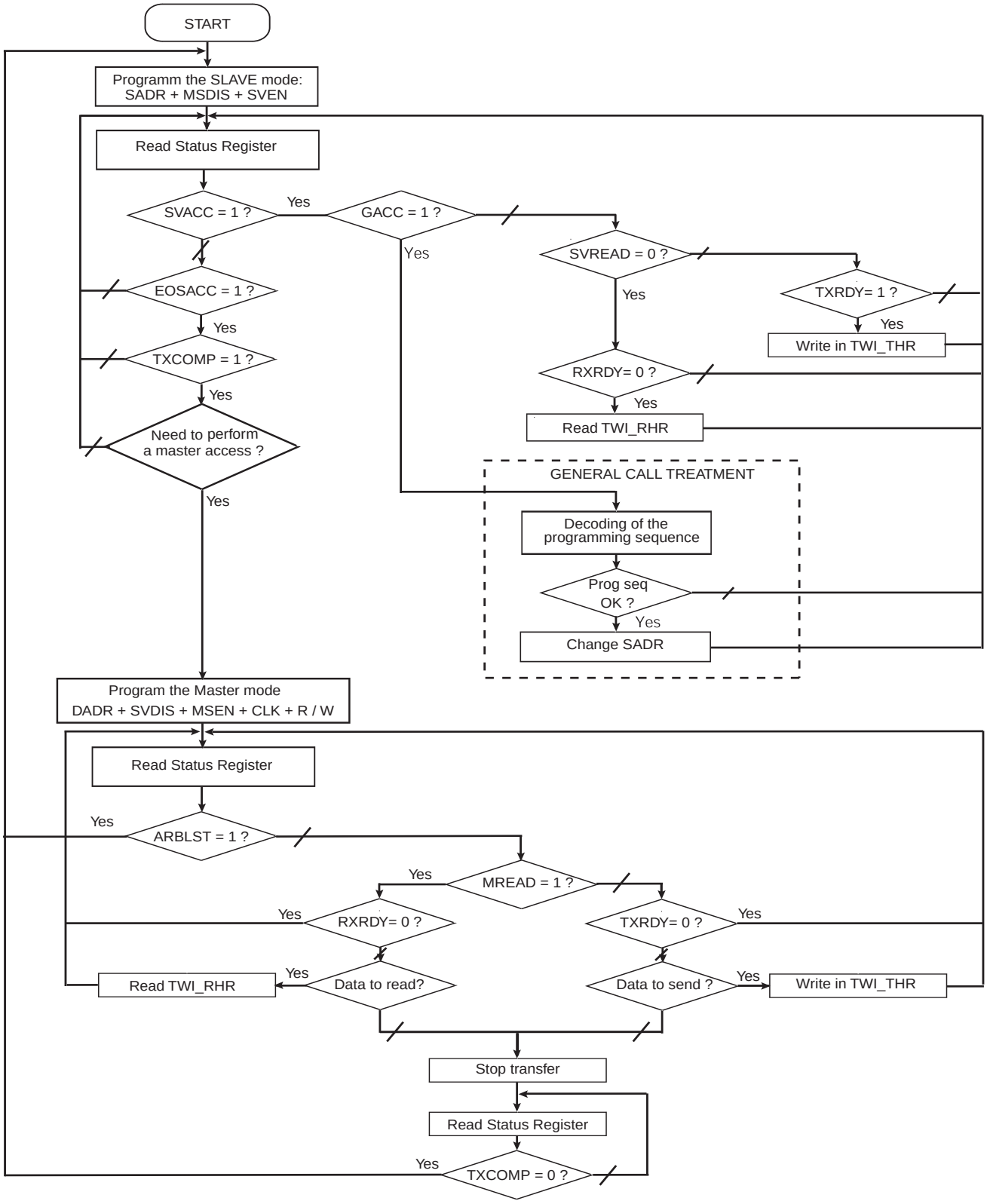


Figure 32-21. Arbitration Cases



The flowchart shown in Figure 32-22 on page 366 gives an example of read and write operations in Multi-master mode.

Figure 32-22. Multi-master Flowchart



32.9 Slave Mode

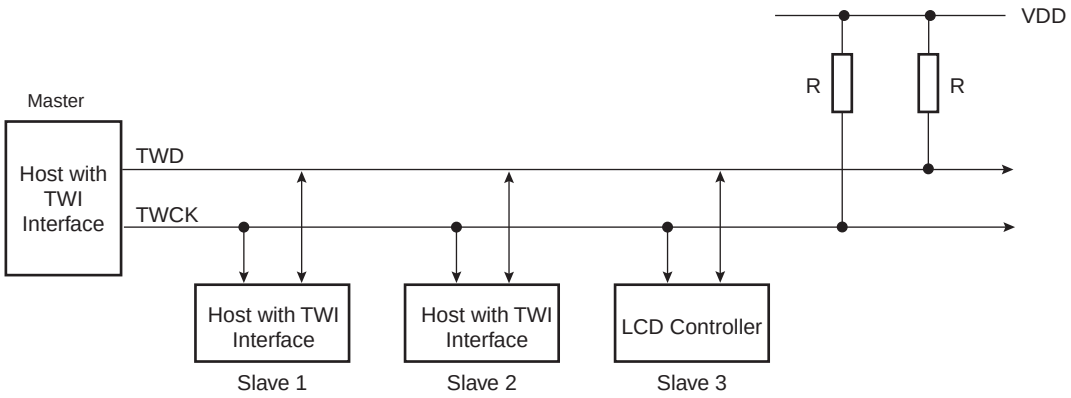
32.9.1 Definition

The Slave Mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED_START and STOP conditions are always provided by the master).

32.9.2 Application Block Diagram

Figure 32-23. Slave Mode Typical Application Block Diagram



32.9.3 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. SADR (TWI_SMR): The slave device address is used in order to be accessed by master devices in read or write mode.
2. MSDIS (TWI_CR): Disable the master mode.
3. SVEN (TWI_CR): Enable the slave mode.

As the device receives the clock, values written in TWI_CWGR are not taken into account.

32.9.4 Receiving Data

After a Start or Repeated Start condition is detected and if the address sent by the Master matches with the Slave address programmed in the SADR (Slave ADdress) field, SVACC (Slave ACCess) flag is set and SVREAD (Slave READ) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a repeated START is detected. When such a condition is detected, EOSACC (End Of Slave ACCess) flag is set.

32.9.4.1 Read Sequence

In the case of a Read sequence (SVREAD is high), TWI transfers data written in the TWI_THR (TWI Transmit Holding Register) until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as data is written in the TWI_THR, TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the shift register is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a repeated START always follows a NACK.

See Figure 32-24 on page 369.

32.9.4.2 *Write Sequence*

In the case of a Write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWI_RHR (TWI Receive Holding Register). RXRDY is reset when reading the TWI_RHR.

TWI continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence TXCOMP flag is set and SVACC reset.

See Figure 32-25 on page 369.

32.9.4.3 *Clock Synchronization Sequence*

In the case where TWI_THR or TWI_RHR is not written/read in time, TWI performs a clock synchronization.

Clock stretching information is given by the SCLWS (Clock Wait state) bit.

See Figure 32-27 on page 371 and Figure 32-28 on page 372.

32.9.4.4 *General Call*

In the case where a GENERAL CALL is performed, GACC (General Call ACCess) flag is set.

After GACC is set, it is up to the programmer to interpret the meaning of the GENERAL CALL and to decode the new address programming sequence.

See Figure 32-26 on page 370.

32.9.5 Data Transfer

32.9.5.1 *Read Operation*

The read mode is defined as a data requirement from the master.

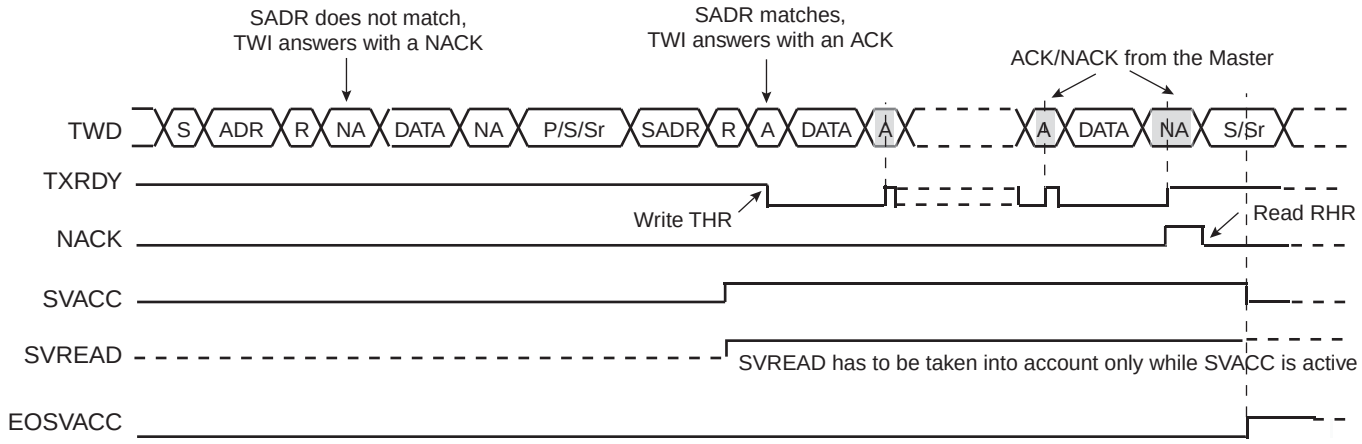
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 32-24 on page 369 describes the write operation.

Figure 32-24. Read Access Ordered by a MASTER



Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. TXRDY is reset when data has been transmitted from TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.

32.9.5.2 Write Operation

The write mode is defined as a data transmission from the master.

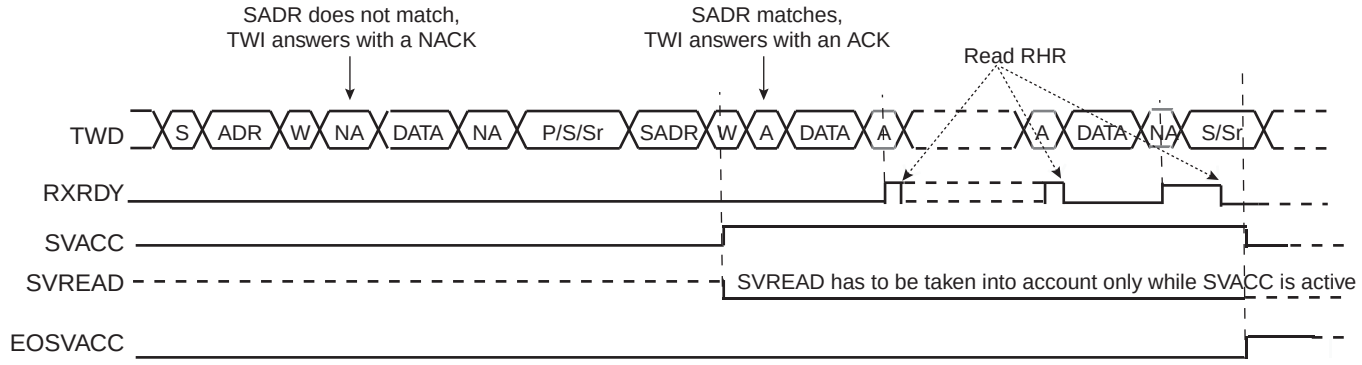
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 32-25 on page 369 describes the Write operation.

Figure 32-25. Write Access Ordered by a Master



Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. RXRDY is set when data has been transmitted from the shift register to the TWI_RHR and reset when this data is read.

32.9.5.3 General Call

The general call is performed in order to change the address of the slave.

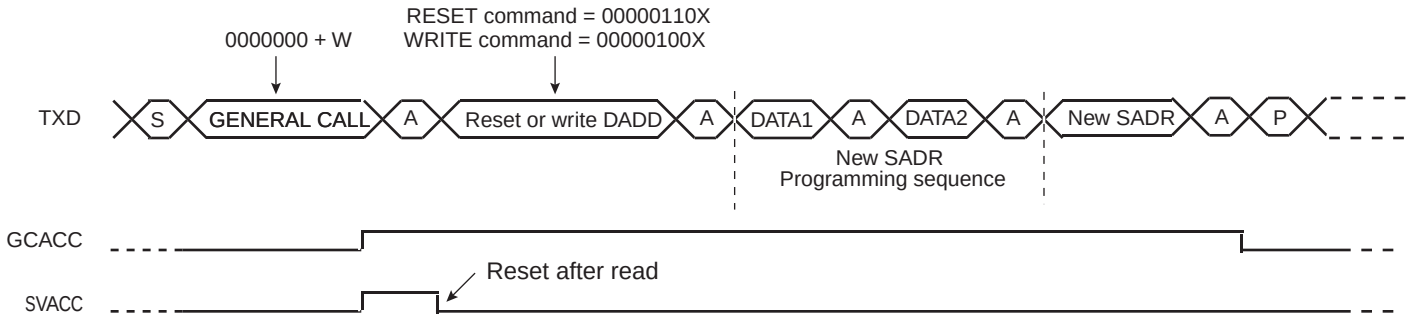
If a GENERAL CALL is detected, GACC is set.

After the detection of General Call, it is up to the programmer to decode the commands which come afterwards.

In case of a WRITE command, the programmer has to decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 32-26 on page 370 describes the General Call access.

Figure 32-26. Master Performs a General Call



Note: This method allows the user to create an own programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

32.9.5.4 Clock Synchronization

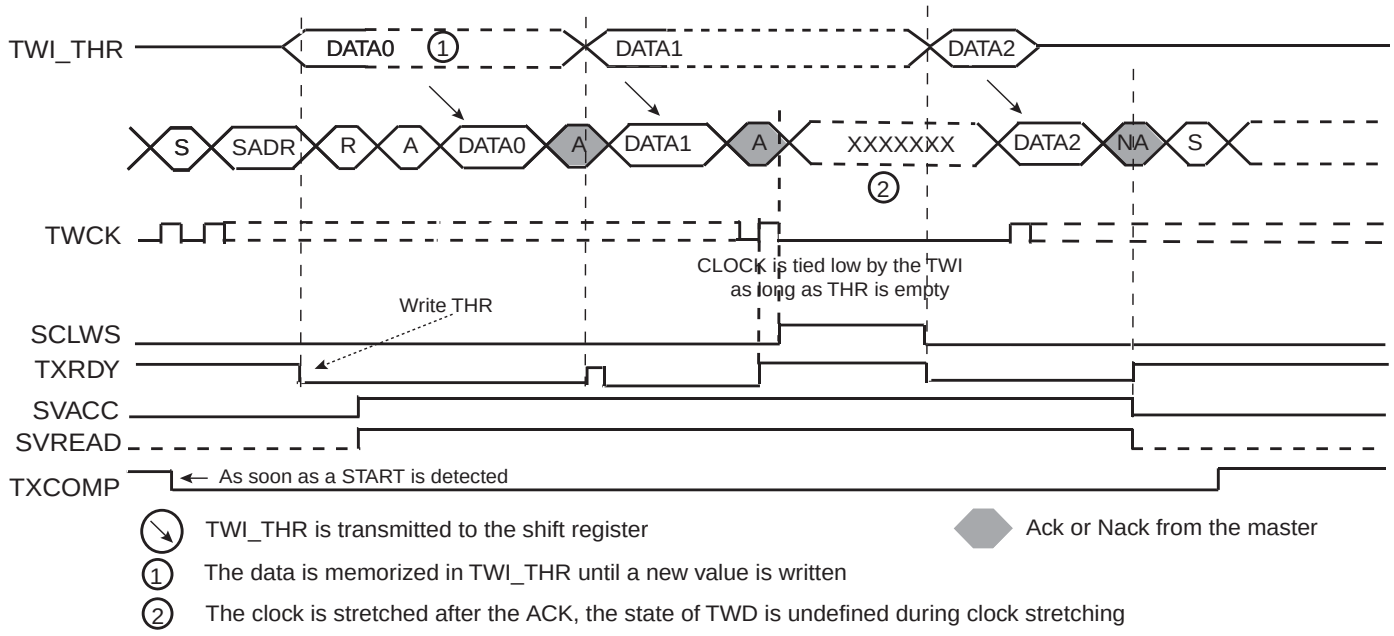
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

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 32-27 on page 371 describes the clock synchronization in Read mode.

Figure 32-27. Clock Synchronization in Read Mode



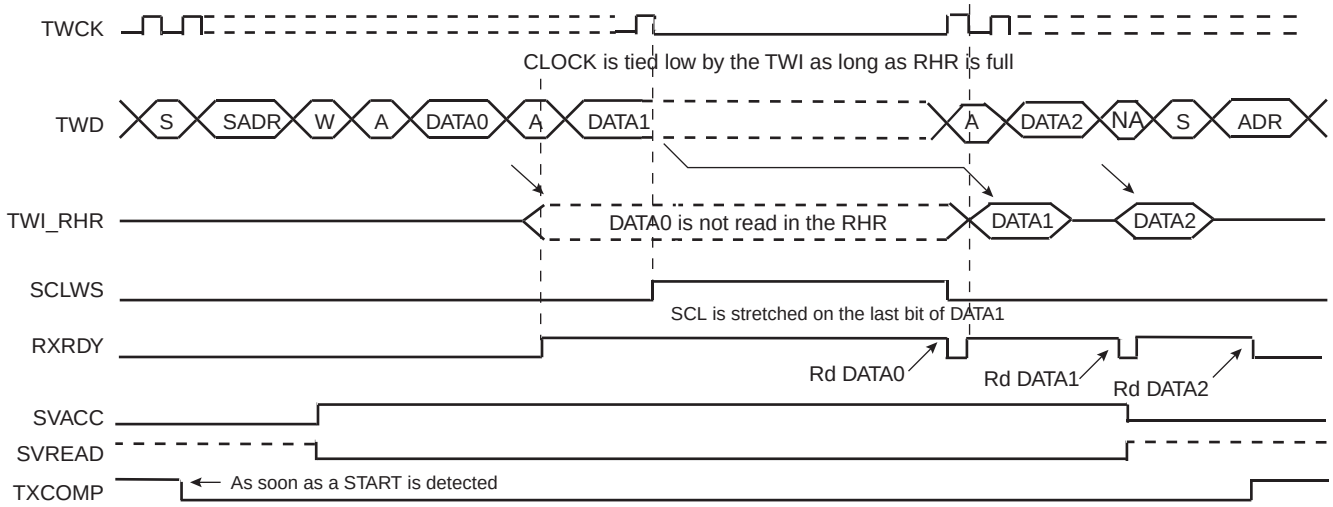
Notes:
1. TXRDY is reset when data has been written in the TWI_TH to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock synchronization mechanism is started.

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 32-28 on page 372 describes the clock synchronization in Read mode.

Figure 32-28. Clock Synchronization in Write Mode



Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

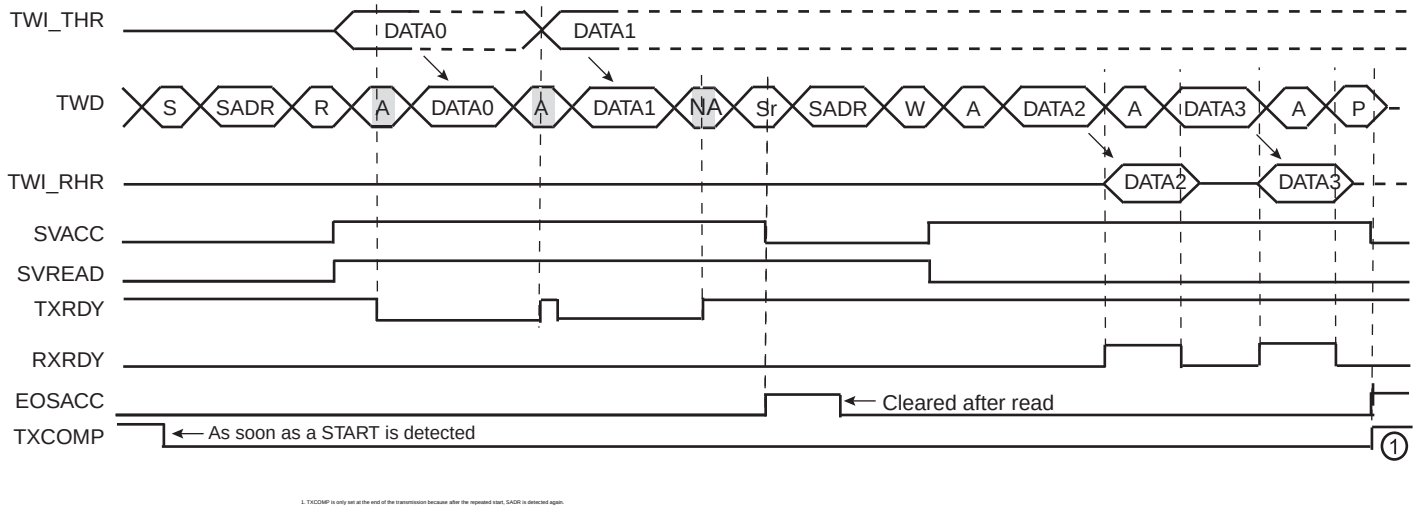
32.9.5.5 Reversal after a Repeated Start

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 32-29 on page 373 describes the repeated start + reversal from Read to Write mode.

Figure 32-29. Repeated Start + Reversal from Read to Write Mode

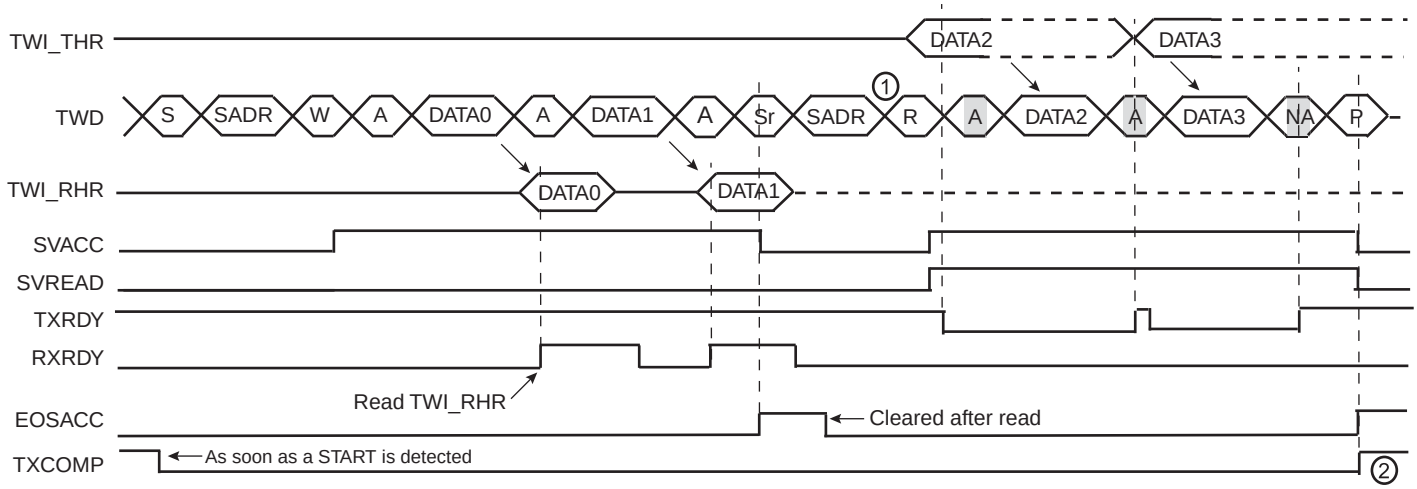


1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command.Figure 32-30 on page 373 describes the repeated start + reversal from Write to Read mode.

Figure 32-30. Repeated Start + Reversal from Write to Read Mode

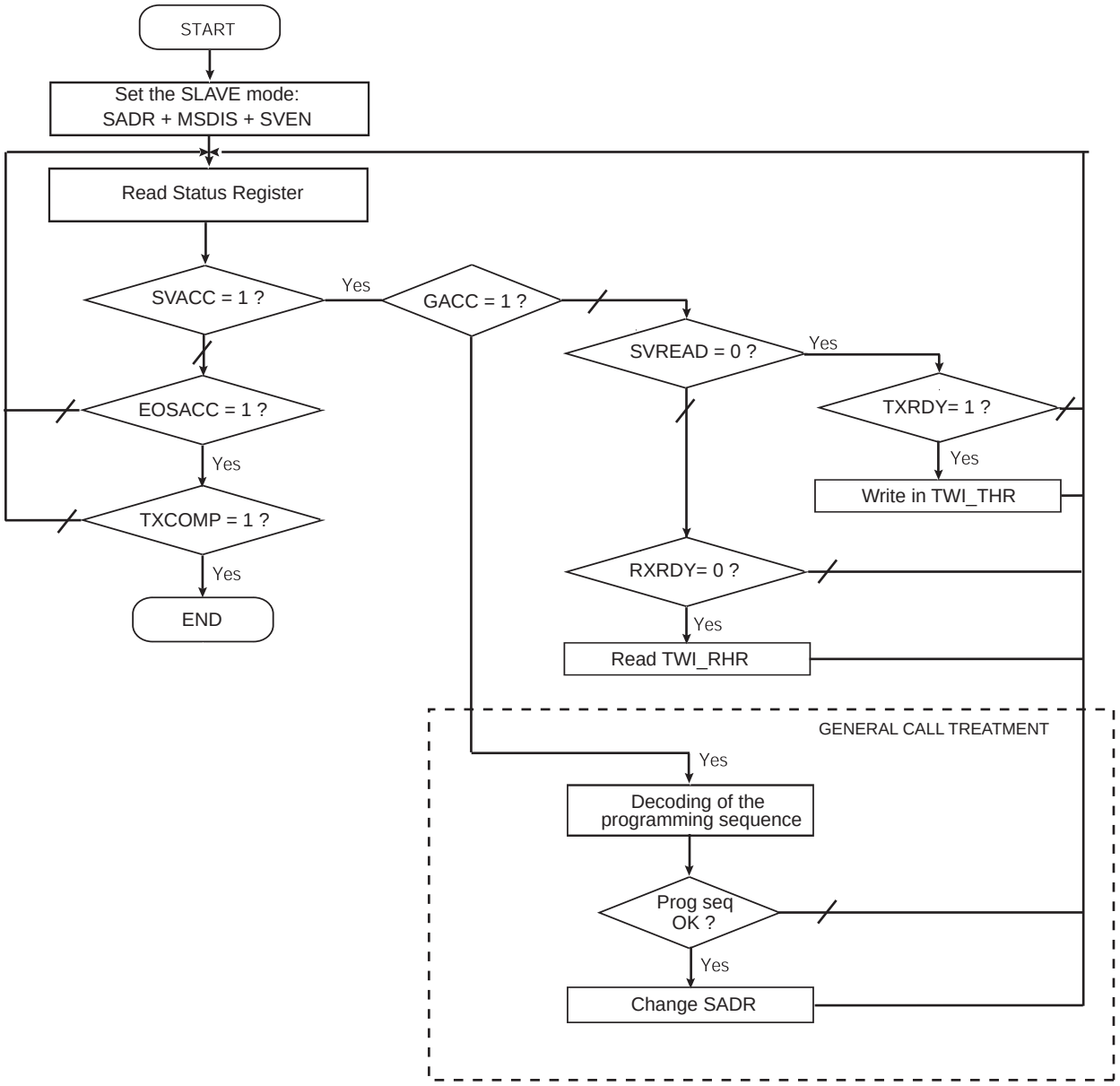


Notes:
1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

32.9.6 Read Write Flowcharts

The flowchart shown in Figure 32-31 on page 374 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 32-31. Read Write Flowchart in Slave Mode



32.10 Two-wire Interface (TWI) User Interface

Table 32-4. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TWI_CR	Write-only	N / A
0x04	Master Mode Register	TWI_MMR	Read-write	0x00000000
0x08	Slave Mode Register	TWI_SMR	Read-write	0x00000000
0x0C	Internal Address Register	TWI_IADR	Read-write	0x00000000
0x10	Clock Waveform Generator Register	TWI_CWGR	Read-write	0x00000000
0x20	Status Register	TWI_SR	Read-only	0x0000F009
0x24	Interrupt Enable Register	TWI_IER	Write-only	N / A
0x28	Interrupt Disable Register	TWI_IDR	Write-only	N / A
0x2C	Interrupt Mask Register	TWI_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWI_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWI_THR	Write-only	0x00000000
0x38 - 0xFC	Reserved	–	–	–

32.10.1 TWI Control Register

Name: TWI_CR

Access: Write-only

Reset Value: 0x00000000

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

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWI_THR).

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read mode.

  – In single data byte master read, the START and STOP must both be set.
  – In multiple data bytes master read, the STOP must be set after the last data received but one.
  – In master read mode, if a NACK bit is received, the STOP is automatically performed.
  – In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

- **MSEN: TWI Master Mode Enabled**

0 = No effect.

1 = If MSDIS = 0, the master mode is enabled.

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWI Master Mode Disabled**

0 = No effect.

1 = The master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWI Slave Mode Enabled**

0 = No effect.

1 = If SVDIS = 0, the slave mode is enabled.

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWI Slave Mode Disabled**

0 = No effect.

1 = The slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

32.10.2 TWI Master Mode Register

Name: TWI_MMR

Access: Read-write

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
–	DADR						

15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **IADRSZ: Internal Device Address Size**

IADRSZ[9:8]		
0	0	No internal device address
0	1	One-byte internal device address
1	0	Two-byte internal device address
1	1	Three-byte internal device address

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in read or write mode. Those bits are only used in Master mode.

32.10.3 TWI Slave Mode Register

Name: TWI_SMR

Access: Read-write

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
–	SADR						

15	14	13	12	11	10	9	8
–	–	–	–	–	–		

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

32.10.4 TWI Internal Address Register

Name: TWI_IADR

Access: Read-write

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
IADR							

15	14	13	12	11	10	9	8
IADR							

7	6	5	4	3	2	1	0
IADR							

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

32.10.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Access: Read-write

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
					CKDIV		

15	14	13	12	11	10	9	8
CHDIV							

7	6	5	4	3	2	1	0
CLDIV							

TWI_CWGR is only used in Master mode.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **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.

32.10.6 TWI Status Register

Name: TWI_SR

Access: Read-only

Reset Value: 0x0000F009

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
				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 32-8 on page 354 and in Figure 32-10 on page 355.

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 32-27 on page 371, Figure 32-28 on page 372, Figure 32-29 on page 373 and Figure 32-30 on page 373.

- **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 32-10 on page 355.

RXRDY behavior in Slave mode can be seen in Figure 32-25 on page 369, Figure 32-28 on page 372, Figure 32-29 on page 373 and Figure 32-30 on page 373.

- **TXRDY: Transmit Holding Register Ready (automatically set / reset)**

TXRDY used in Master mode:

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI_THR register.

1 = As soon as a data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

TXRDY behavior in Master mode can be seen in Figure 32-8 on page 354.

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 32-24 on page 369, Figure 32-27 on page 371, Figure 32-29 on page 373 and Figure 32-30 on page 373.

- **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 32-24 on page 369, Figure 32-25 on page 369, Figure 32-29 on page 373 and Figure 32-30 on page 373.

- **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 32-24 on page 369, Figure 32-25 on page 369, Figure 32-29 on page 373 and Figure 32-30 on page 373.

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0 = No General Call has been detected.

1 = A General Call has been detected. After the detection of General Call, the programmer decoded the commands that follow and the programming sequence.

GACC behavior can be seen in Figure 32-26 on page 370.

- **OVRE: Overrun Error (clear on read)**

This bit is only used in Master mode.

0 = TWI_RHR has not been loaded while RXRDY was set

1 = TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

- **NACK: Not Acknowledged (clear on read)**

NACK used in Master mode:

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0 = Each data byte has been correctly received by the Master.

1 = In read mode, a data byte has not been acknowledged by the Master. When NACK is set the programmer must not fill TWI_THR even if TXRDY is set, because it means that the Master will stop the data transfer or re initiate it.

Note that in Slave Write mode all data is 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 32-27 on page 371 and Figure 32-28 on page 372.

- **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 32-29 on page 373 and Figure 32-30 on page 373

32.10.7 TWI Interrupt Enable Register

Name: TWI_IER

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
				EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ARBLST: Arbitration Lost Interrupt Enable**
- **SCL_WS: Clock Wait State Interrupt Enable**
- **EOSACC: End Of Slave Access Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

32.10.8 TWI Interrupt Disable Register

Name: TWI_IDR

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
				EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Disable**
- **RXRDY: Receive Holding Register Ready Interrupt Disable**
- **TXRDY: Transmit Holding Register Ready Interrupt Disable**
- **SVACC: Slave Access Interrupt Disable**
- **GACC: General Call Access Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **NACK: Not Acknowledge Interrupt Disable**
- **ARBLST: Arbitration Lost Interrupt Disable**
- **SCL_WS: Clock Wait State Interrupt Disable**
- **EOSACC: End Of Slave Access Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

32.10.9 TWI Interrupt Mask Register

Name: TWI_IMR

Access: Read-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
				EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ARBLST: Arbitration Lost Interrupt Mask**
- **SCL_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

32.10.10 TWI Receive Holding Register

Name: TWI_RHR

Access: Read-only

Reset Value: 0x00000000

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

- **RXDATA: Master or Slave Receive Holding Data**

32.10.11 TWI Transmit Holding Register

Name: TWI_THR

Access: Read-write

Reset Value: 0x00000000

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

- **TXDATA: Master or Slave Transmit Holding Data**

33. Universal Synchronous Asynchronous Receiver Transceiver (USART)

33.1 Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

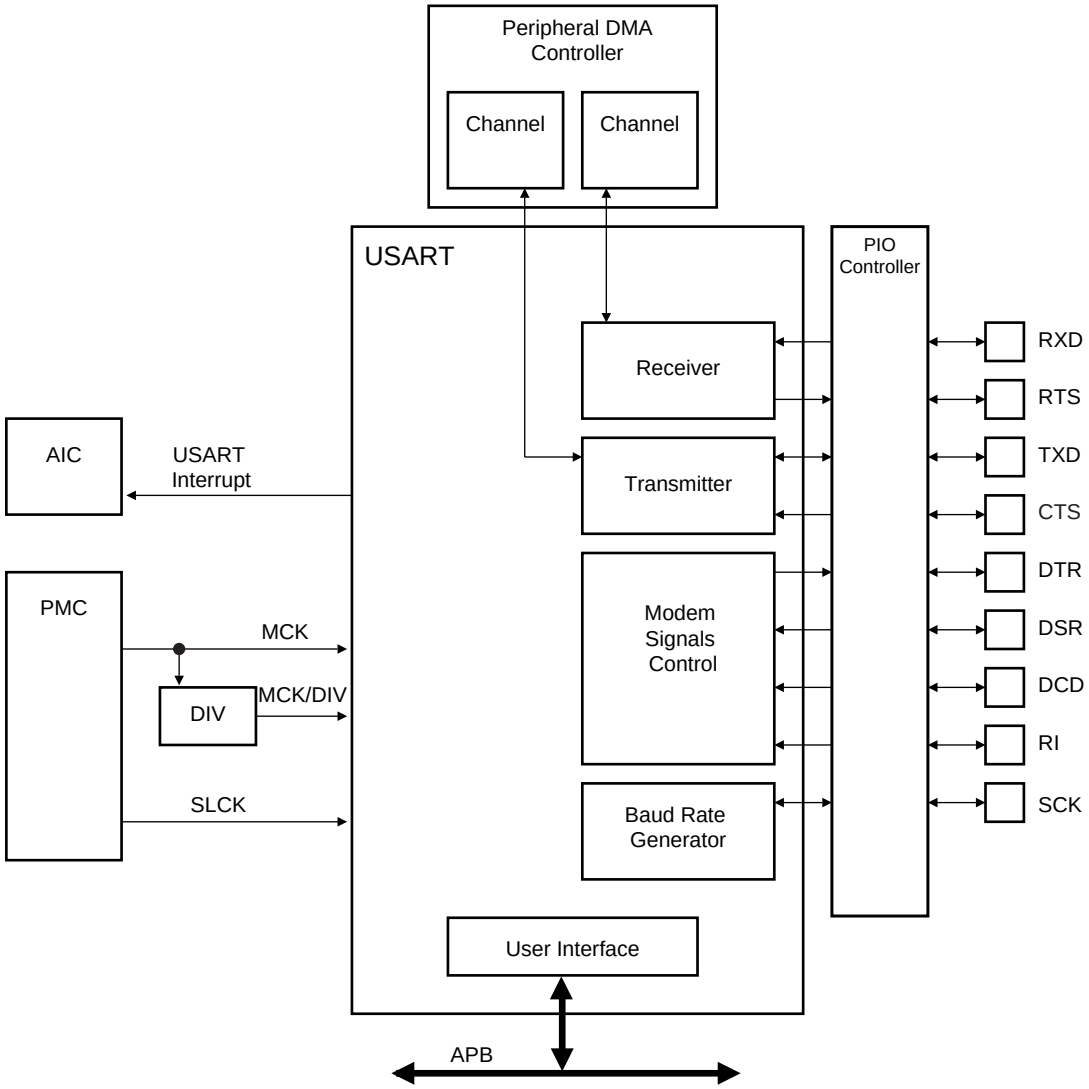
The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 buses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

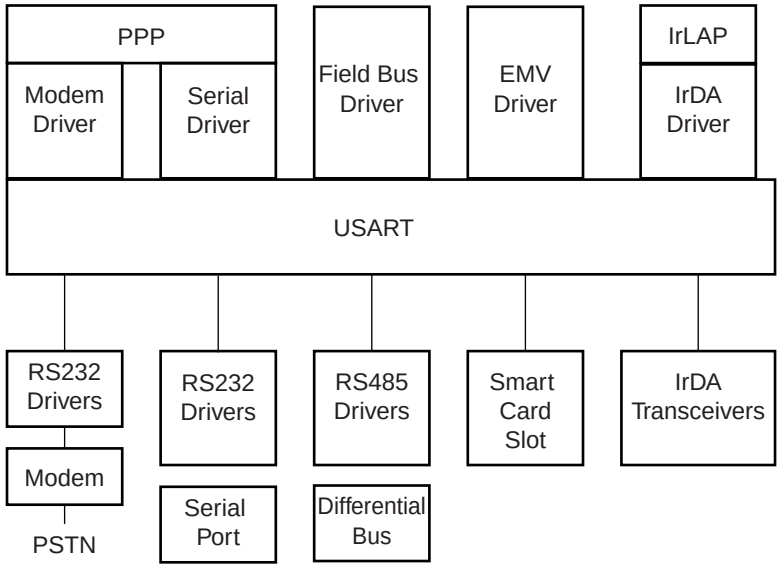
33.2 Block Diagram

Figure 33-1. USART Block Diagram



33.3 Application Block Diagram

Figure 33-2. Application Block Diagram



33.4 I/O Lines Description

Table 33-1. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	
TXD	Transmit Serial Data	I/O	
RXD	Receive Serial Data	Input	
RI	Ring Indicator	Input	Low
DSR	Data Set Ready	Input	Low
DCD	Data Carrier Detect	Input	Low
DTR	Data Terminal Ready	Output	Low
CTS	Clear to Send	Input	Low
RTS	Request to Send	Output	Low

33.5 Product Dependencies

33.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory.

All the pins of the modems may or may not be implemented on the USART. Only USART1 is fully equipped with all the modem signals. On USARTs not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

33.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

33.5.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the AIC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

33.6 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- InfraRed IrDA Modulation and Demodulation
- Test modes
  - Remote loopback, local loopback, automatic echo

33.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

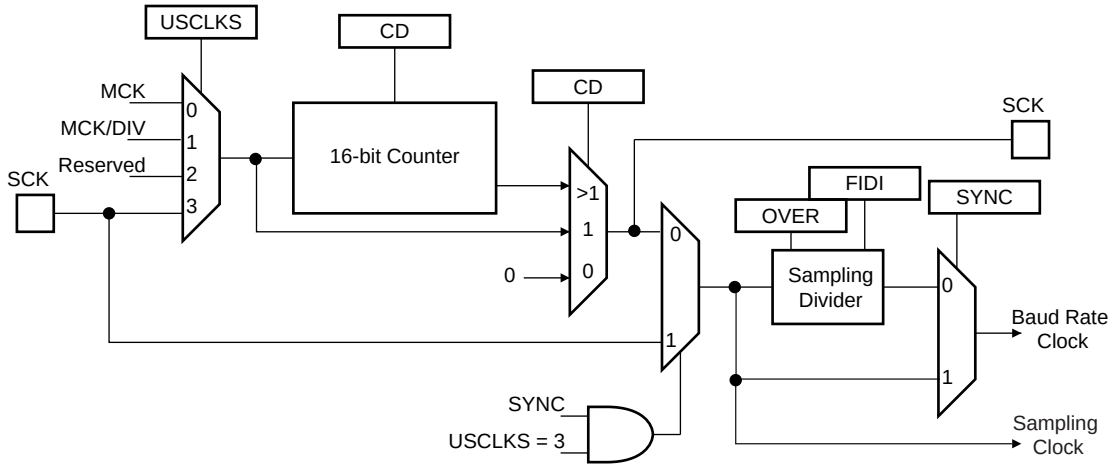
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US_BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.

Figure 33-3. Baud Rate Generator



33.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

33.6.1.2 Baud Rate Calculation Example

Table 33-2 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 33-2. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

The baud rate is calculated with the following formula:

$$BaudRate = MCK / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate} \right)$$

33.6.1.3 Fractional Baud Rate in Asynchronous Mode

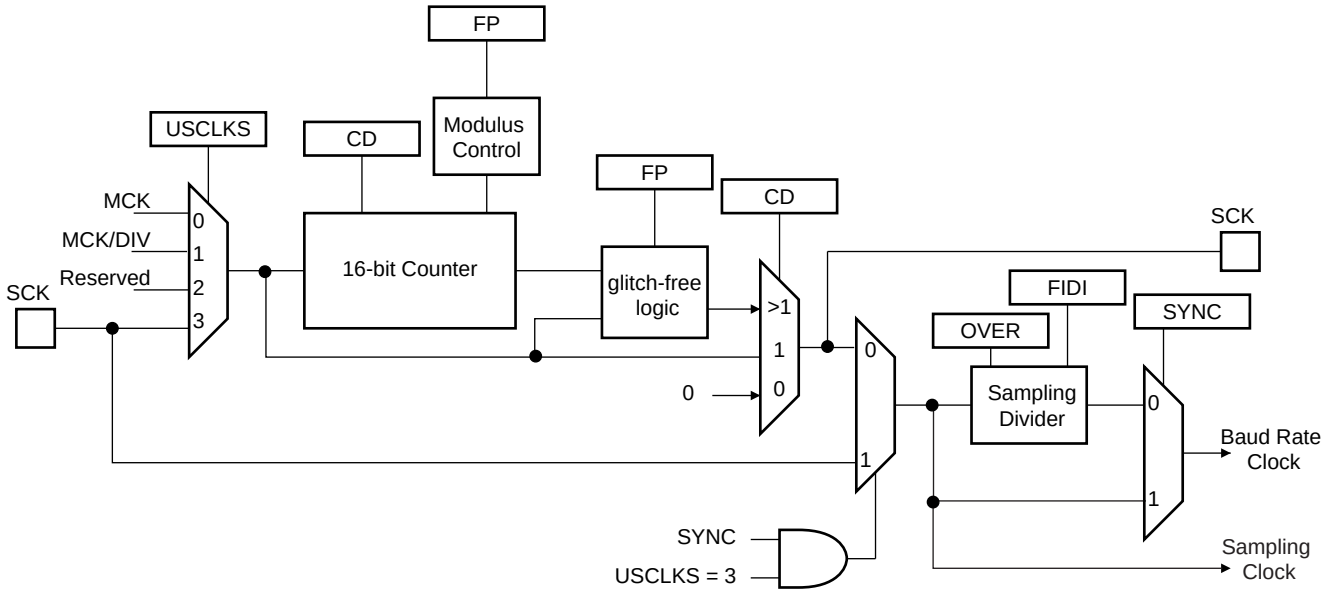
The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain Baud Rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the Baud Rate Generator Register

(US_BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2 - Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

Figure 33-4. Fractional Baud Rate Generator



33.6.1.4 *Baud Rate in Synchronous Mode*

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

33.6.1.5 *Baud Rate in ISO 7816 Mode*

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 33-3.

Table 33-3. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 33-4.

Table 33-4. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 33-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 33-5. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

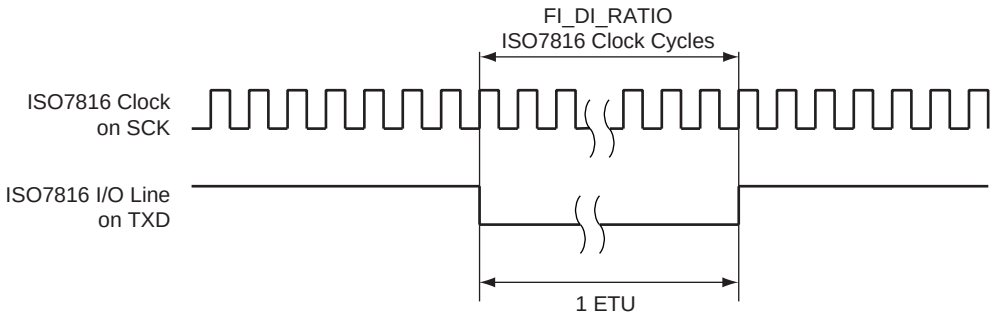
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 33-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 33-5. Elementary Time Unit (ETU)



33.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (US_CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (US_CR). The reset commands have the same effect as a hardware reset on the corresponding logic. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (US_THR). If a timeguard is programmed, it is handled normally.

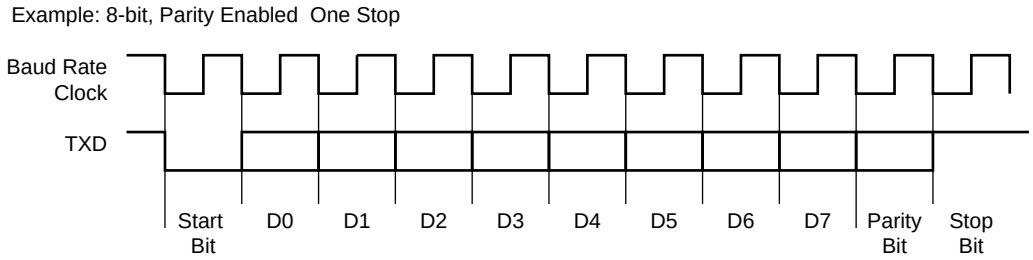
33.6.3 Synchronous and Asynchronous Modes

33.6.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (US_MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in US_MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in US_MR. The 1.5 stop bit is supported in asynchronous mode only.

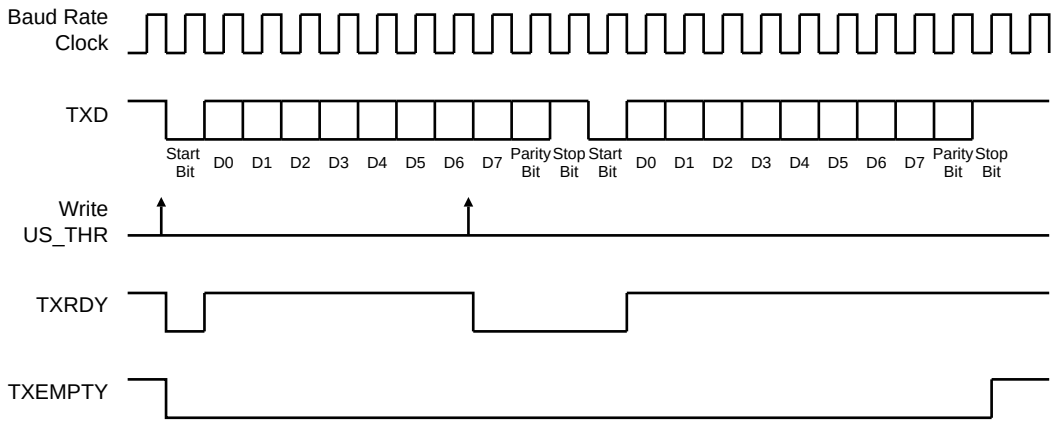
Figure 33-6. Character Transmit



The characters are sent by writing in the Transmit Holding Register (US_THR). The transmitter reports two status bits in the Channel Status Register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift Register of the transmitter and US_THR becomes empty, thus TXRDY raises.

Both TXRDY and TXEMPTY bits are low since the transmitter is disabled. Writing a character in US_THR while TXRDY is active has no effect and the written character is lost.

Figure 33-7. Transmitter Status



33.6.3.2 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver over-samples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US_MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the

transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 33-8 and Figure 33-9 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 33-8. Asynchronous Start Detection

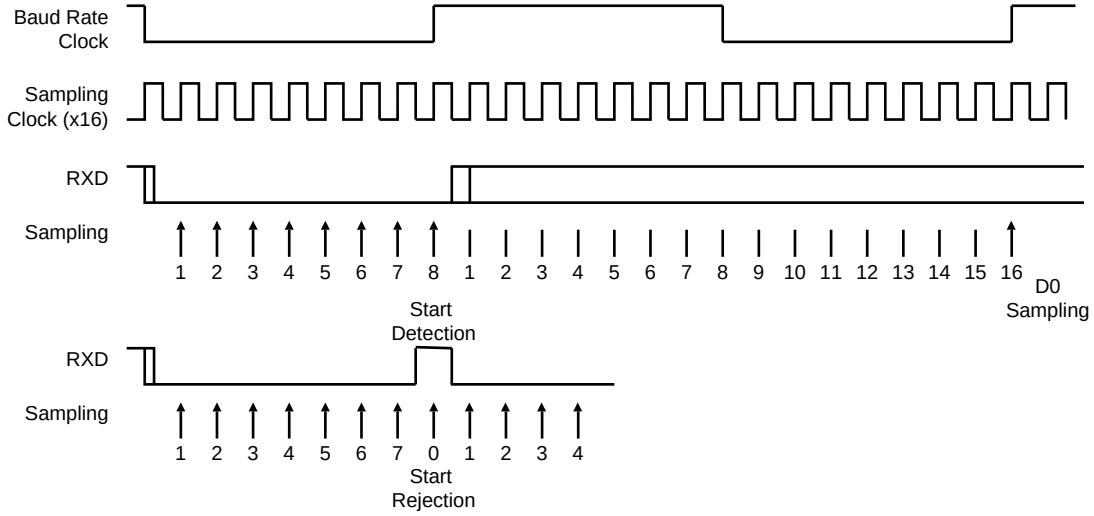
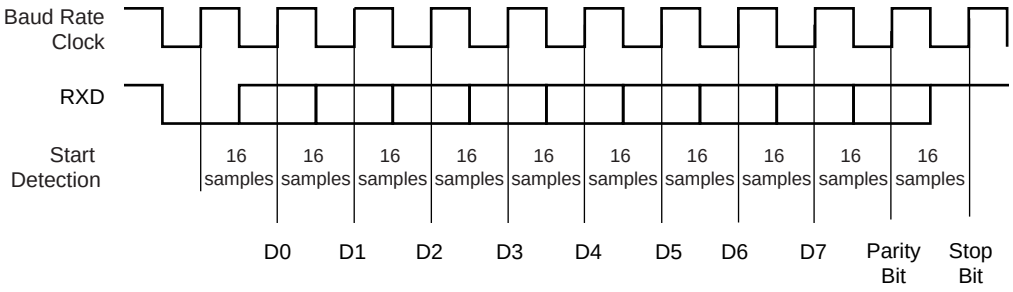


Figure 33-9. Asynchronous Character Reception

Example: 8-bit, Parity Enabled



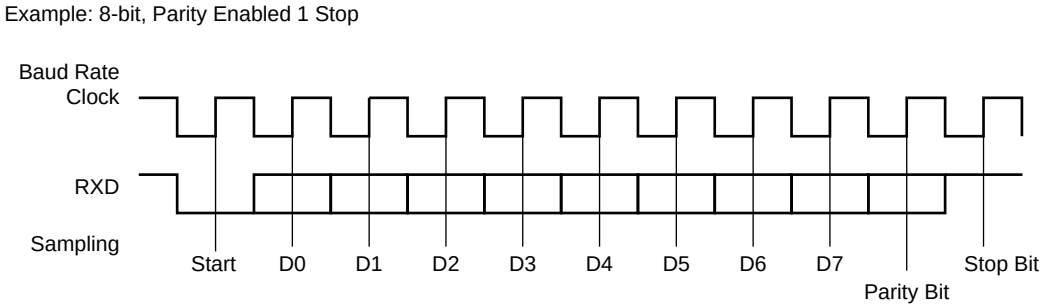
33.6.3.3 Synchronous Receiver

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 33-10 illustrates a character reception in synchronous mode.

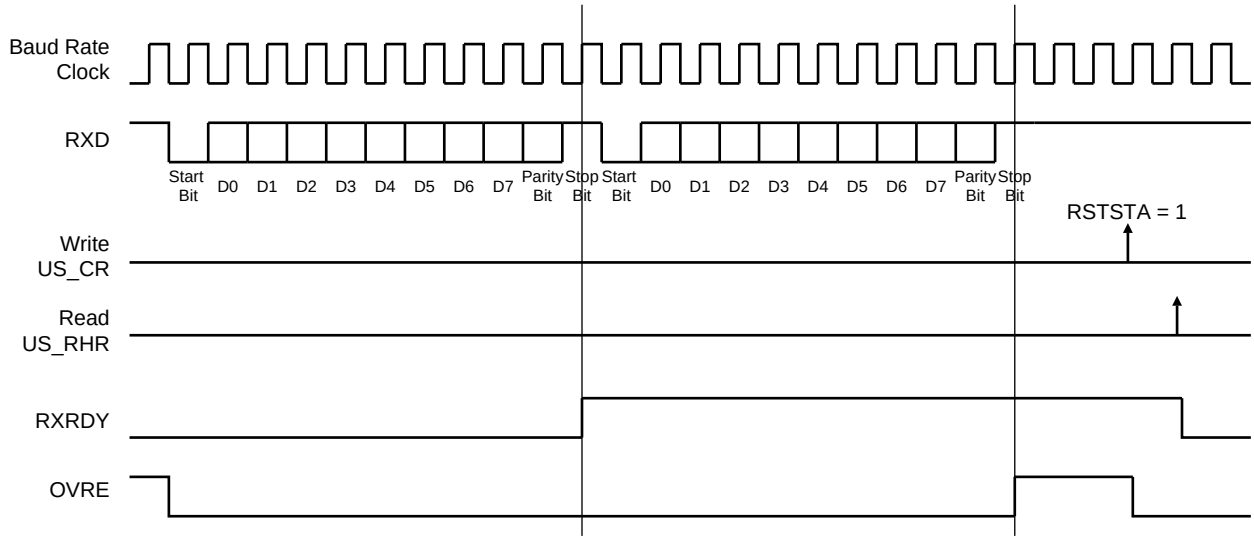
Figure 33-10. Synchronous Mode Character Reception



33.6.3.4 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit at 1.

Figure 33-11. Receiver Status



33.6.3.5 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see “Multidrop Mode” on page 403. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

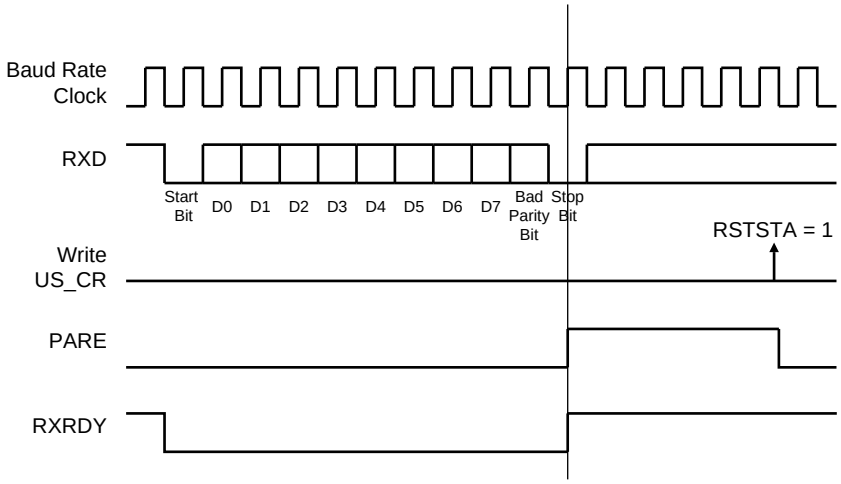
Table 33-6 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 33-6. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (US_CSR). The PARE bit can be cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. Figure 33-12 illustrates the parity bit status setting and clearing.

Figure 33-12. Parity Error



33.6.3.6 *Multidrop Mode*

If the PAR field in the Mode Register (US_MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to US_CR. In this case, the next byte written to US_THR is transmitted as an address. Any character written in US_THR without having written the command SENDA is transmitted normally with the parity at 0.

33.6.3.7 *Transmitter Timeguard*

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (US_TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 33-13, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 33-13. Timeguard Operations

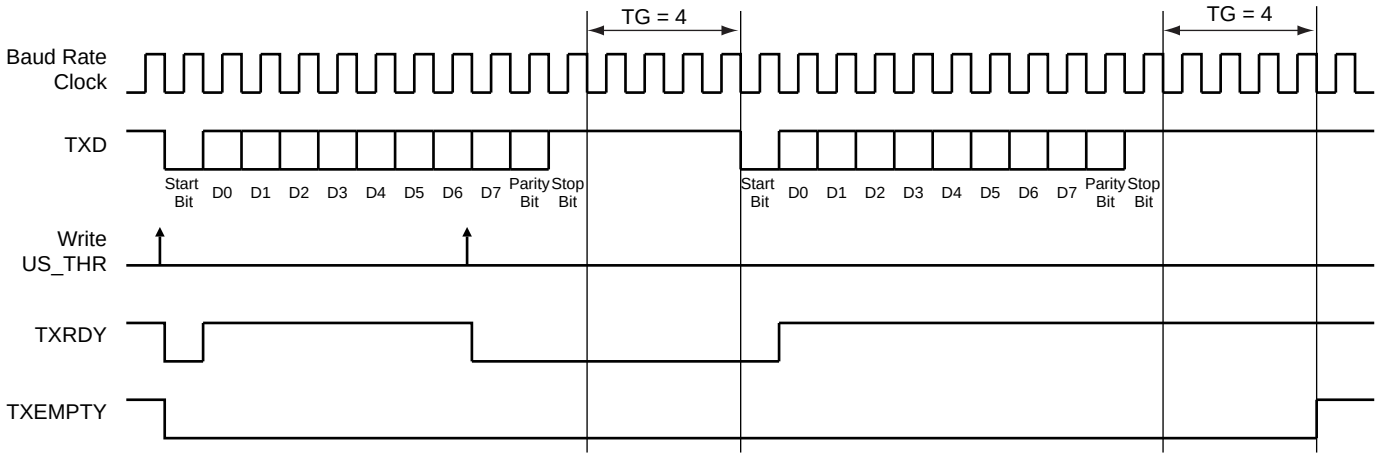


Table 33-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 33-7. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

33.6.3.8 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US_RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing the Control Register (US_CR) with the STTTO (Start Time-out) bit at 1. In this case, the idle state

on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.

- Obtain an interrupt while no character is received. This is performed by writing US_CR with the RETTO (Reload and Start Time-out) bit at 1. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 33-14 shows the block diagram of the Receiver Time-out feature.

Figure 33-14. Receiver Time-out Block Diagram

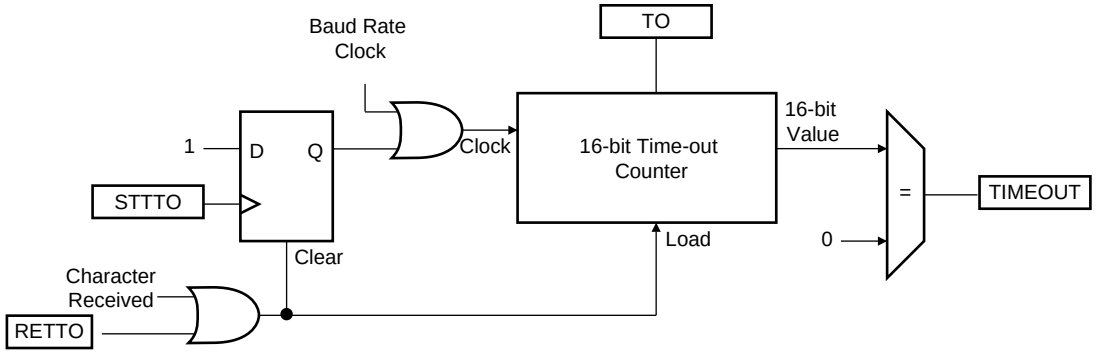


Table 33-8 gives the maximum time-out period for some standard baud rates.

Table 33-8. Maximum Time-out Period

Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962

Table 33-8. Maximum Time-out Period (Continued)

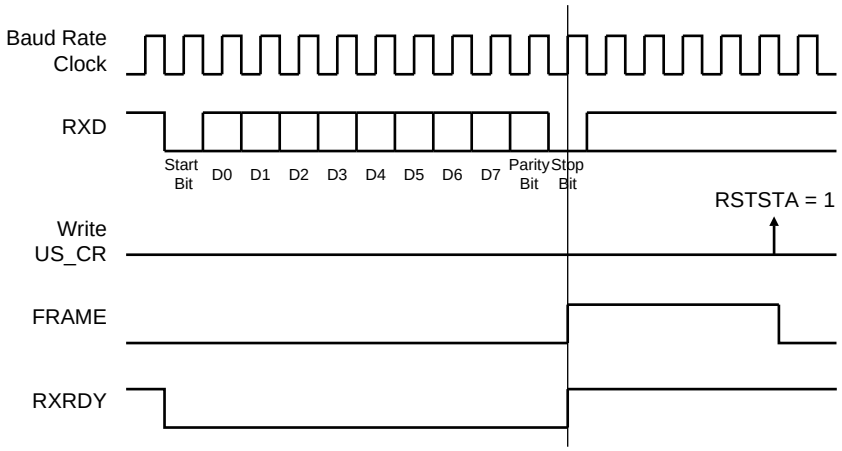
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

33.6.3.9 *Framing Error*

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (US_CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1.

Figure 33-15. Framing Error Status



33.6.3.10 *Transmit Break*

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US_CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing US_CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in US_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

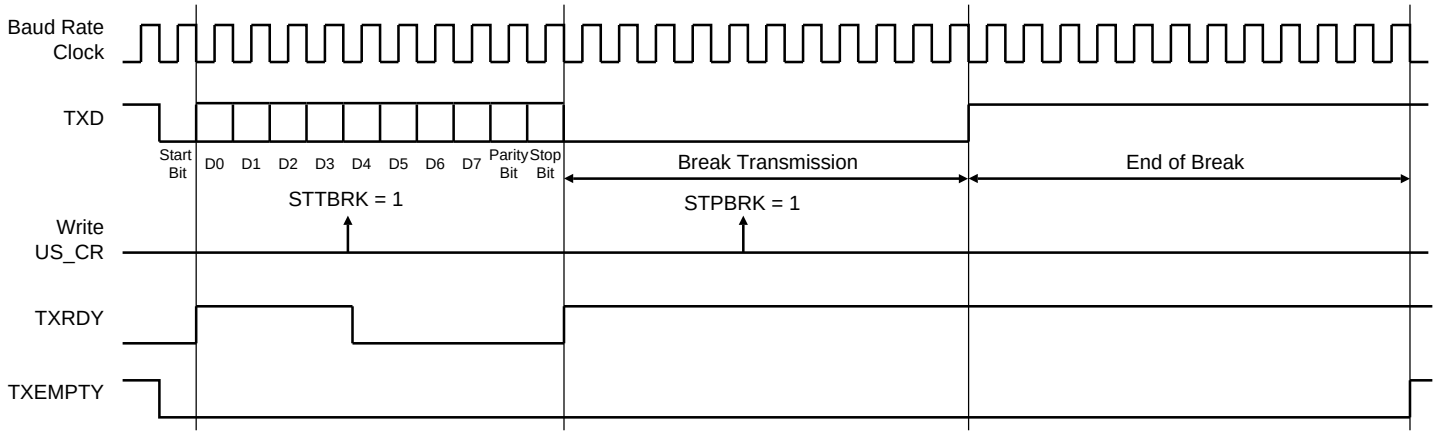
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 33-16 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 33-16. Break Transmission



33.6.3.11 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

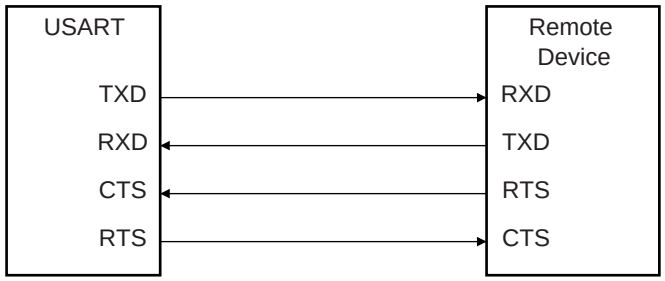
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

33.6.3.12 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 33-17.

Figure 33-17. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 33-18 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 33-18. Receiver Behavior when Operating with Hardware Handshaking

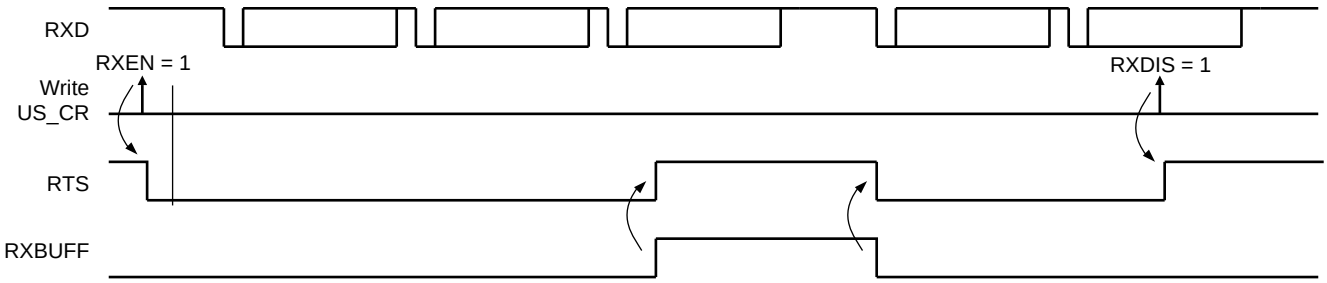
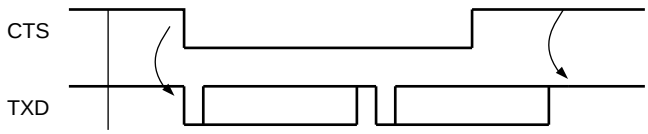


Figure 33-19 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 33-19. Transmitter Behavior when Operating with Hardware Handshaking



33.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

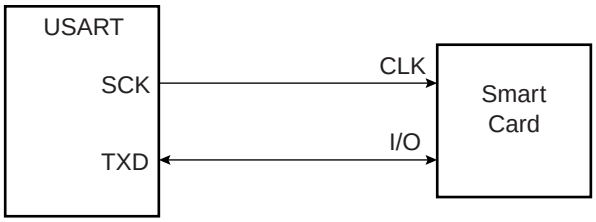
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

33.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 393).

The USART connects to a smart card as shown in Figure 33-20. 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 33-20. 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 421 and "PAR: Parity Type" on page 422.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US_THR) or after reading it in the Receive Holding Register (US_RHR).

33.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 33-21.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 33-22. 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 33-21. T = 0 Protocol without Parity Error

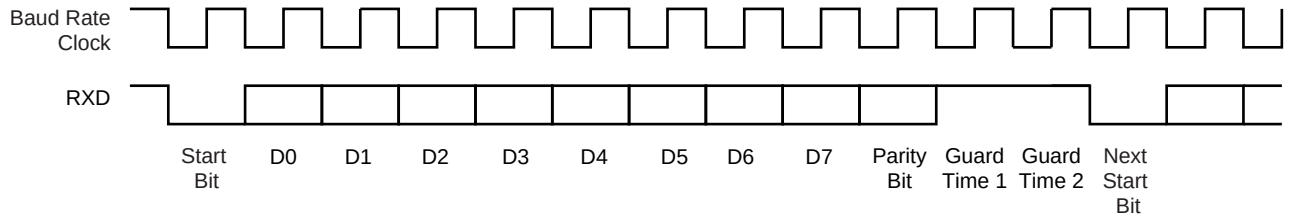
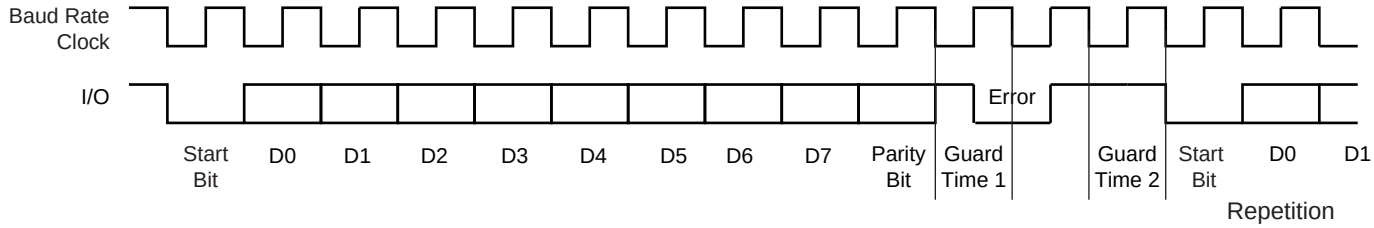


Figure 33-22. T = 0 Protocol with Parity Error



33.6.4.3 Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

33.6.4.4 Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US_MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (US_SR). The INACK bit can be cleared by writing the Control Register (US_CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

33.6.4.5 Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (US_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (US_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US_CSR can be cleared by writing the Control Register with the RSIT bit at 1.

33.6.4.6 Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US_MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

33.6.4.7 Protocol T = 1

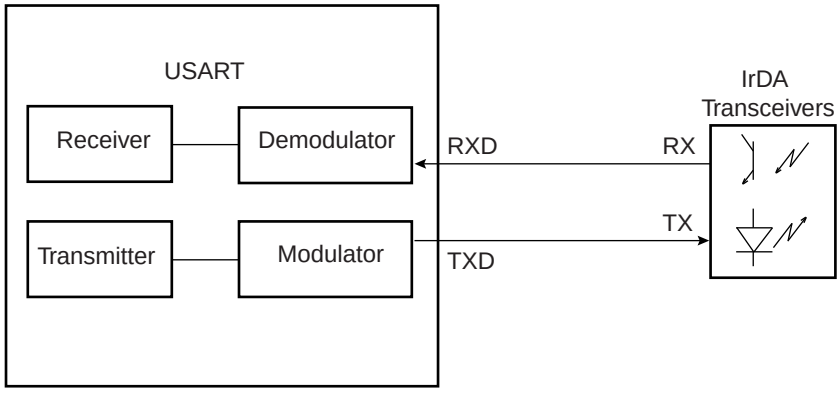
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

33.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 33-23. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the USART_MODE field in the Mode Register (US_MR) to the value 0x8. The IrDA Filter Register (US_IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

Figure 33-23. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

33.6.5.1 IrDA Modulation

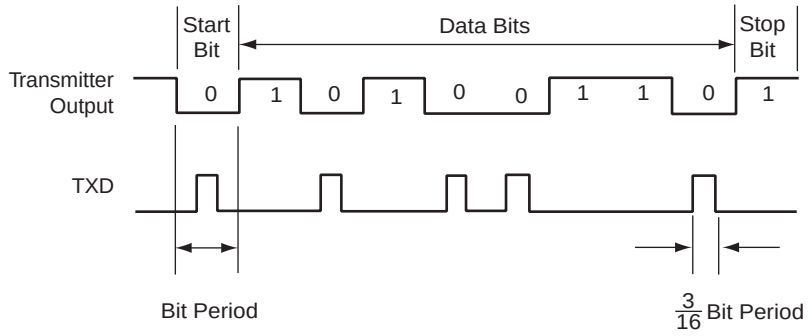
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 33-9.

Table 33-9. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 33-24 shows an example of character transmission.

Figure 33-24. IrDA Modulation



33.6.5.2 IrDA Baud Rate

Table 33-10 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 33-10. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88

Table 33-10. IrDA Baud Rate Error (Continued)

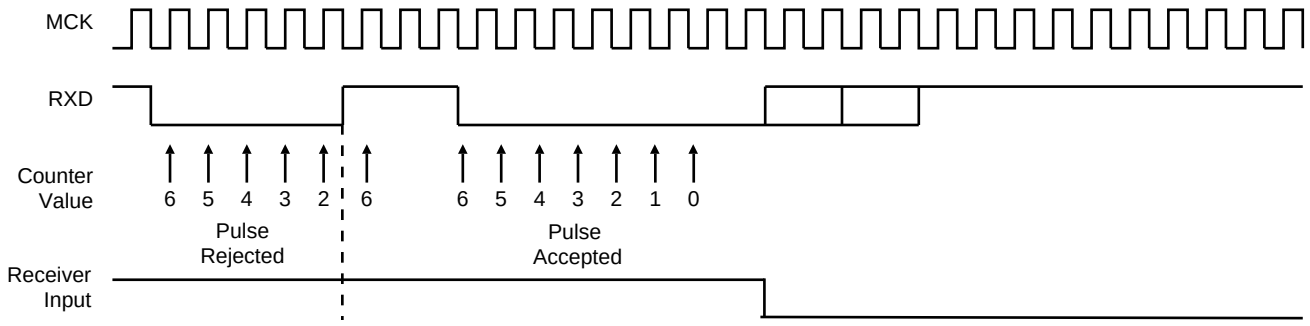
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

33.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 33-25 illustrates the operations of the IrDA demodulator.

Figure 33-25. IrDA Demodulator Operations

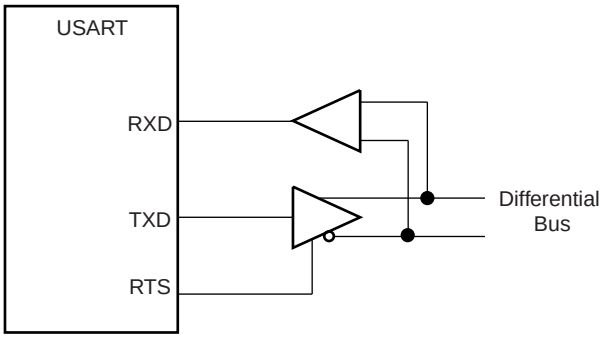


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

33.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 33-26.

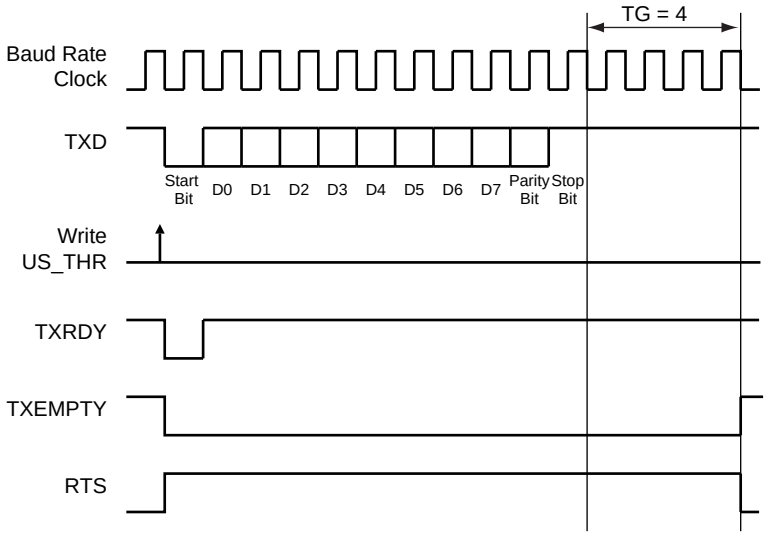
Figure 33-26. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the USART_MODE field in the Mode Register (US_MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 33-27 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 33-27. Example of RTS Drive with Timeguard



33.6.7 Modem Mode

The USART features modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in modem mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x3. While operating in modem mode the USART behaves as though in asynchronous mode and all the parameter configurations are available.

Table 33-11 gives the correspondence of the USART signals with modem connection standards.

Table 33-11. Circuit References

USART Pin	V24	CCITT	Direction
TXD	2	103	From terminal to modem
RTS	4	105	From terminal to modem
DTR	20	108.2	From terminal to modem
RXD	3	104	From modem to terminal
CTS	5	106	From terminal to modem
DSR	6	107	From terminal to modem
DCD	8	109	From terminal to modem
RI	22	125	From terminal to modem

The control of the DTR output pin is performed by writing the Control Register (US_CR) with the DTRDIS and DTREN bits respectively at 1. The disable command forces the corresponding pin to its inactive level, i.e. high. The enable command forces the corresponding pin to its active level, i.e. low. RTS output pin is automatically controlled in this mode

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in the Channel Status Register (US_CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when US_CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

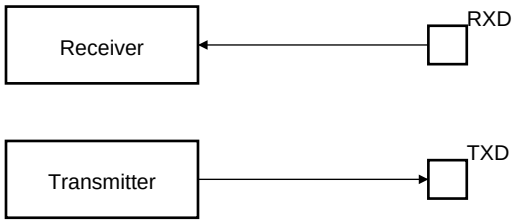
33.6.8 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

33.6.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

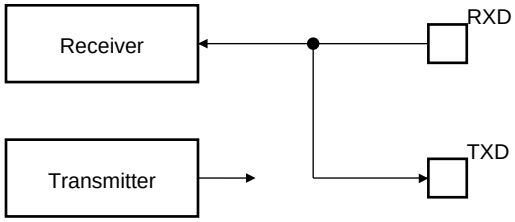
Figure 33-28. Normal Mode Configuration



33.6.8.2 *Automatic Echo Mode*

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 33-29. 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.

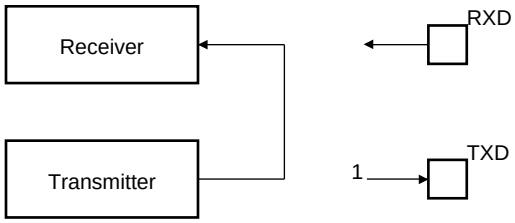
Figure 33-29. Automatic Echo Mode Configuration



33.6.8.3 *Local Loopback Mode*

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 33-30. 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.

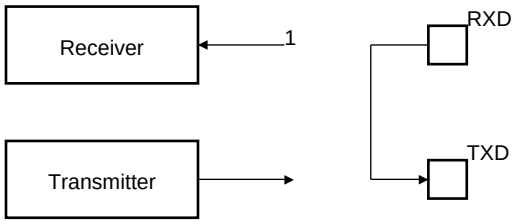
Figure 33-30. Local Loopback Mode Configuration



33.6.8.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 33-31. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 33-31. Remote Loopback Mode Configuration



33.7 USART User Interface

Table 33-12. USART Memory Map

Offset	Register	Name	Access	Reset State
0x0000	Control Register	US_CR	Write-only	–
0x0004	Mode Register	US_MR	Read/Write	–
0x0008	Interrupt Enable Register	US_IER	Write-only	–
0x000C	Interrupt Disable Register	US_IDR	Write-only	–
0x0010	Interrupt Mask Register	US_IMR	Read-only	0x0
0x0014	Channel Status Register	US_CSR	Read-only	–
0x0018	Receiver Holding Register	US_RHR	Read-only	0x0
0x001C	Transmitter Holding Register	US_THR	Write-only	–
0x0020	Baud Rate Generator Register	US_BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	US_RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	US_TTGR	Read/Write	0x0
0x2C - 0x3C	Reserved	–	–	–
0x0040	FI DI Ratio Register	US_FIDI	Read/Write	0x174
0x0044	Number of Errors Register	US_NER	Read-only	–
0x0048	Reserved	–	–	–
0x004C	IrDA Filter Register	US_IF	Read/Write	0x0
0x5C - 0xFC	Reserved	–	–	–
0x100 - 0x128	Reserved for PDC Registers	–	–	–

33.7.1 USART Control Register

Name: US_CR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	DTRDIS	DTREN
15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, and RXBRK in US_CSR.

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0: No effect.

1: Starts waiting for a character before clocking the time-out counter. Resets the status bit TIMEOUT in US_CSR.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop Mode only, the next character written to the US_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITERATION in US_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in US_CSR.

- **RETTO: Rearm Time-out**

0: No effect

1: Restart Time-out

- **DTREN: Data Terminal Ready Enable**

0: No effect.

1: Drives the pin DTR at 0.

- **DTRDIS: Data Terminal Ready Disable**

0: No effect.

1: Drives the pin DTR to 1.

- **RTSEN: Request to Send Enable**

0: No effect.

1: Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0: No effect.

1: Drives the pin RTS to 1.

33.7.2 USART Mode Register

Name: US_MR

Access: Read/Write

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

23	22	21	20	19	18	17	16
–	–	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

- **USART_MODE**

USART_MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	0	1	1	Modem
0	1	0	0	IS07816 Protocol: T = 0
0	1	0	1	Reserved
0	1	1	0	IS07816 Protocol: T = 1
0	1	1	1	Reserved
1	0	0	0	IrDA
1	1	x	x	Reserved

- **USCLKS: Clock Selection**

USCLKS		Selected Clock
0	0	MCK
0	1	MCK/DIV (DIV = 8)
1	0	Reserved
1	1	SCK

- **CHRL: Character Length.**

CHRL		Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	x	No parity
1	1	x	Multidrop mode

- **NBSTOP: Number of Stop Bits**

NBSTOP		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input..
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16x Oversampling.

1: 8x Oversampling.

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **MAX_ITERATION**

Defines the maximum number of iterations in mode ISO7816, protocol T= 0.

- **FILTER: Infrared Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

33.7.3 USART Interrupt Enable Register

Name: US_IER

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **ENDRX: End of Receive Transfer Interrupt Enable**
- **ENDTX: End of Transmit Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITERATION: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Enable**
- **RXBUFF: Buffer Full Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **RIIC: Ring Indicator Input Change Enable**
- **DSRIC: Data Set Ready Input Change Enable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**

33.7.4 USART Interrupt Disable Register

Name: US_IDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Transfer Interrupt Disable**
- **ENDTX: End of Transmit Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITERATION: Iteration Interrupt Disable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **RIIC: Ring Indicator Input Change Disable**
- **DSRIC: Data Set Ready Input Change Disable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**

33.7.5 USART Interrupt Mask Register

Name: US_IMR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **ENDRX: End of Receive Transfer Interrupt Mask**
- **ENDTX: End of Transmit Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITERATION: Iteration Interrupt Mask**
- **TXBUFE: Buffer Empty Interrupt Mask**
- **RXBUFF: Buffer Full Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **RIIC: Ring Indicator Input Change Mask**
- **DSRIC: Data Set Ready Input Change Mask**
- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**

33.7.6 USART Channel Status Register

Name: US_CSR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break**

0: No Break received or End of Break detected since the last RSTSTA.

1: Break Received or End of Break detected since the last RSTSTA.

- **ENDRX: End of Receiver Transfer**

0: The End of Transfer signal from the Receive PDC channel is inactive.

1: The End of Transfer signal from the Receive PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0: The End of Transfer signal from the Transmit PDC channel is inactive.

1: The End of Transfer signal from the Transmit PDC channel is active.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out**

0: There has not been a time-out since the last Start Time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command (STTTO in US_CR).

- **TXEMPTY: Transmitter Empty**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **ITERATION: Max number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSIT.

1: Maximum number of repetitions has been reached since the last RSIT.

- **TXBUFE: Transmission Buffer Empty**

0: The signal Buffer Empty from the Transmit PDC channel is inactive.

1: The signal Buffer Empty from the Transmit PDC channel is active.

- **RXBUFF: Reception Buffer Full**

0: The signal Buffer Full from the Receive PDC channel is inactive.

1: The signal Buffer Full from the Receive PDC channel is active.

- **NACK: Non Acknowledge**

0: No Non Acknowledge has not been detected since the last RSTNACK.

1: At least one Non Acknowledge has been detected since the last RSTNACK.

- **RIIC: Ring Indicator Input Change Flag**

0: No input change has been detected on the RI pin since the last read of US_CSR.

1: At least one input change has been detected on the RI pin since the last read of US_CSR.

- **DSRIC: Data Set Ready Input Change Flag**

0: No input change has been detected on the DSR pin since the last read of US_CSR.

1: At least one input change has been detected on the DSR pin since the last read of US_CSR.

- **DCDIC: Data Carrier Detect Input Change Flag**

0: No input change has been detected on the DCD pin since the last read of US_CSR.

1: At least one input change has been detected on the DCD pin since the last read of US_CSR.

- **CTSIC: Clear to Send Input Change Flag**

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

- **RI: Image of RI Input**

0: RI is at 0.

1: RI is at 1.

- **DSR: Image of DSR Input**

0: DSR is at 0

1: DSR is at 1.

- **DCD: Image of DCD Input**

0: DCD is at 0.

1: DCD is at 1.

- **CTS: Image of CTS Input**

0: CTS is at 0.

1: CTS is at 1.

33.7.7 USART Receive Holding Register

Name: US_RHR

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.

33.7.8 USART Transmit Holding Register

Name: US_THR

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.

33.7.9 USART Baud Rate Generator Register

Name: US_BRGR

Access: Read/Write

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

- **CD: Clock Divider**

CD	**USART_MODE ≠ ISO7816**			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	**OVER = 1**		
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/16/CD	Baud Rate = Selected Clock/8/CD	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/CD/FI_DI_RATIO

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baudrate resolution, defined by FP x 1/8.

33.7.10 USART Receiver Time-out Register

Name: US_RTOR

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

33.7.11 USART Transmitter Timeguard Register

Name: US_TTGR

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TG							

- **TG: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

33.7.12 USART FI DI RATIO Register

Name: US_FIDI

Access: Read/Write

Reset Value : 0x174

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

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

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

7	6	5	4	3	2	1	0
FI_DI_RATIO							

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI_DI_RATIO.

33.7.13 USART Number of Errors Register

Name: US_NER

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

33.7.14 USART IrDA FILTER Register

Name: US_IF

Access: Read/Write

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

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

34. Parallel Input Output Controller (PIO)

34.1 Overview

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

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

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

34.2 Block Diagram

Figure 34-1. Block Diagram

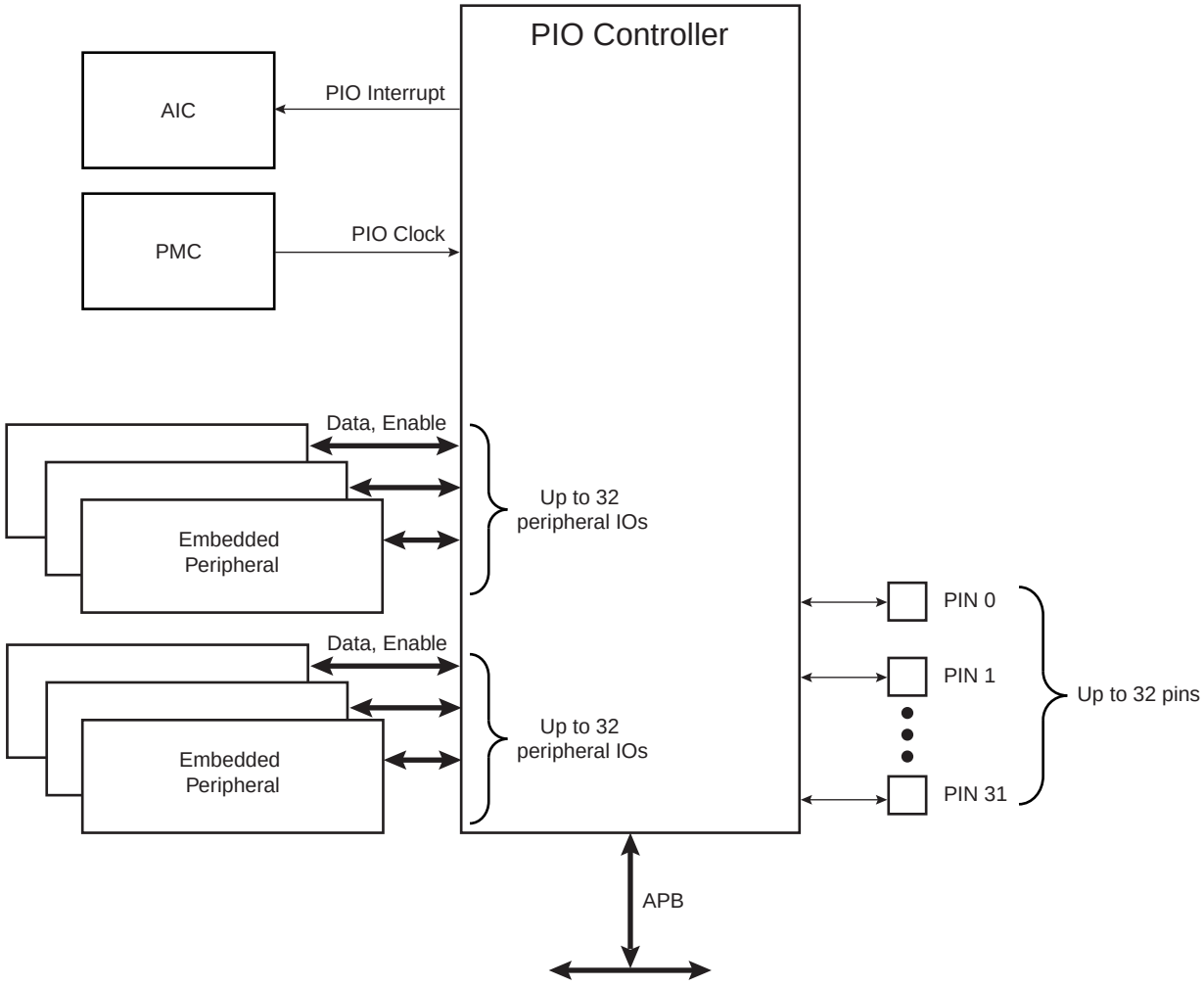
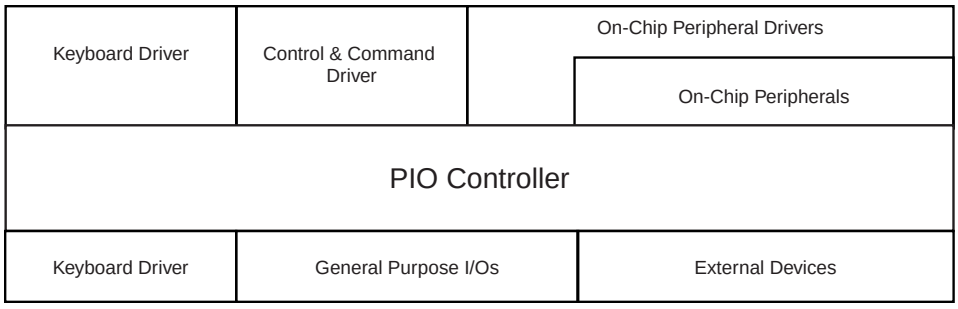


Figure 34-2. Application Block Diagram



34.3 Product Dependencies

34.3.1 Pin Multiplexing

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

34.3.2 External Interrupt Lines

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

34.3.3 Power Management

The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

34.3.4 Interrupt Generation

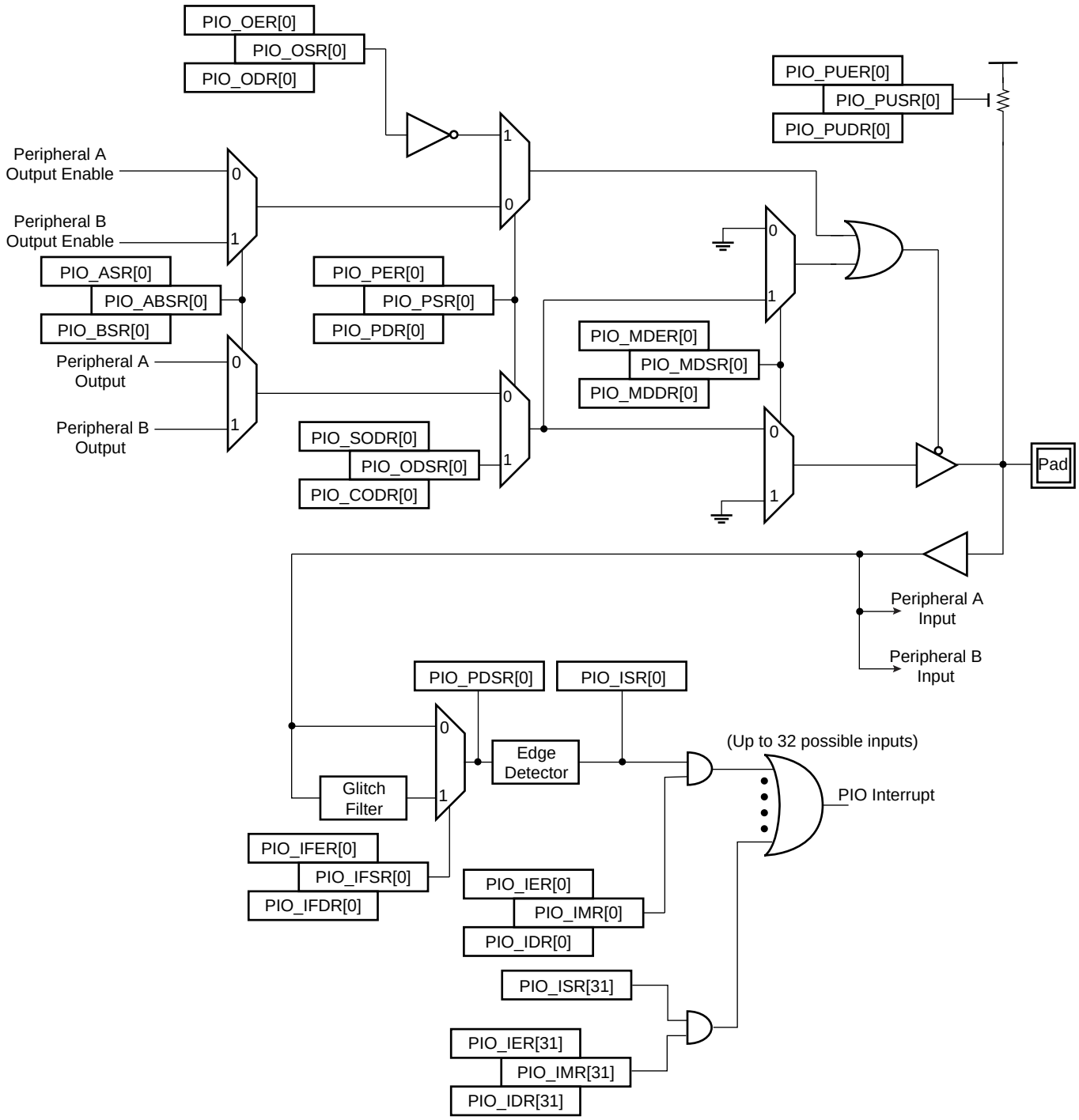
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

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

34.4 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 34-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 34-3. I/O Line Control Logic



34.4.1 Pull-up Resistor Control

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

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e. PIO_PUSR resets at the value 0x0.

34.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

34.4.3 Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO_ASR (A Select Register) and PIO_BSR (Select B Register). PIO_ABSR (AB Select Status Register) indicates which peripheral line is currently selected. For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

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

After reset, PIO_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO_ASR and PIO_BSR manages PIO_ABSR regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the corresponding peripheral selection register (PIO_ASR or PIO_BSR) in addition to a write in PIO_PDR.

34.4.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e. the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register).

The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

34.4.5 Synchronous Data Output

Controlling all parallel busses using several PIOs requires two successive write operations in the PIO_SODR and PIO_CODR registers. This may lead to unexpected transient values. The PIO controller offers a direct control of PIO outputs by single write access to PIO_ODSR (Output Data Status Register). Only bits unmasked by PIO_OWSR (Output Write Status Register) are written. The mask bits in the PIO_OWSR are set by writing to PIO_OWER (Output Write Enable Register) and cleared by writing to PIO_OWDR (Output Write Disable Register).

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

34.4.6 Multi Drive Control (Open Drain)

Each I/O can be independently programmed in Open Drain by using the Multi Drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

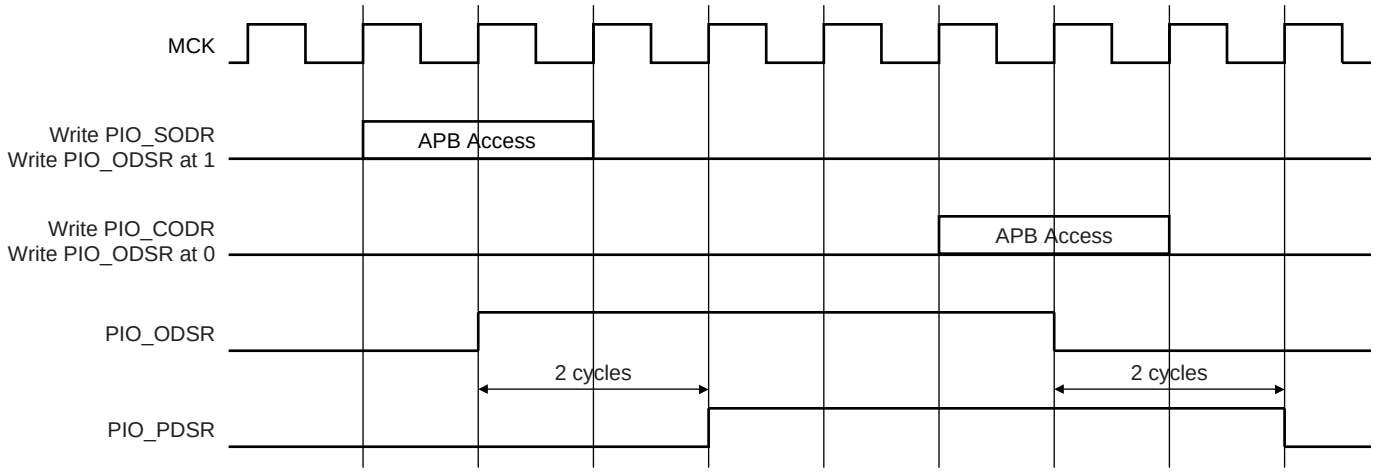
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates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

After reset, the Multi Drive feature is disabled on all pins, i.e. PIO_MDSR resets at value 0x0.

34.4.7 Output Line Timings

Figure 34-4 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 34-4 also shows when the feedback in PIO_PDSR is available.

Figure 34-4. Output Line Timings



34.4.8 Inputs

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

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

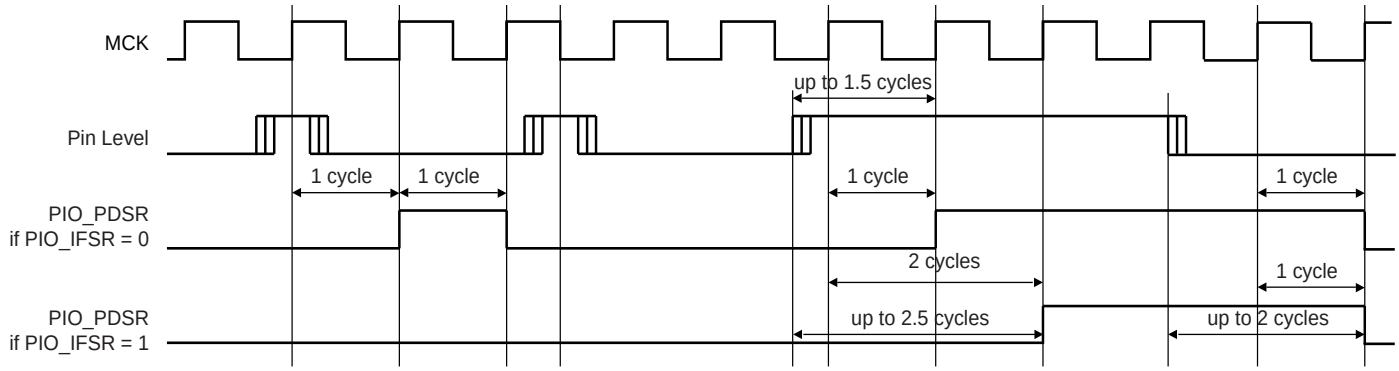
34.4.9 Input Glitch Filtering

Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations between 1/2 Master Clock cycle and 1 Master Clock cycle the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible it must exceed 1 Master Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Master Clock cycle. The filter introduces one Master Clock cycle latency if the pin level change occurs before a rising edge. However, this latency does not appear if the pin level change occurs before a falling edge. This is illustrated in Figure 34-5.

The glitch filters are controlled by the register set; PIO_IFER (Input Filter Enable Register), PIO_IFDR (Input Filter Disable Register) and PIO_IFSR (Input Filter Status Register). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch filters require that the PIO Controller clock is enabled.

Figure 34-5. Input Glitch Filter Timing



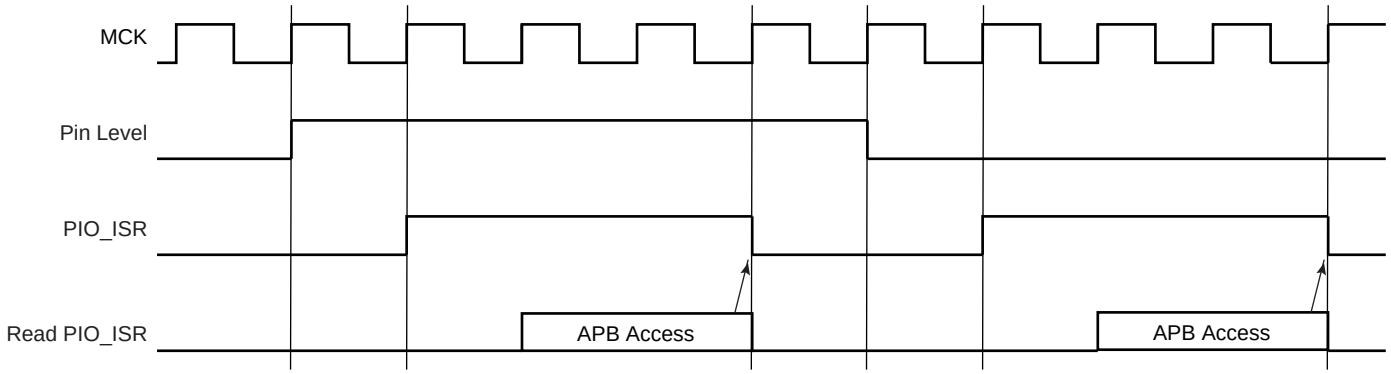
34.4.10 Input Change Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an input change on an I/O line. The Input Change Interrupt is controlled by writing PIO_IER (Interrupt Enable Register) and PIO_IDR (Interrupt Disable Register), which respectively enable and disable the input change interrupt by setting and clearing the corresponding bit in PIO_IMR (Interrupt Mask Register). As Input change detection is possible only by comparing two successive samplings of the input of the I/O line, the PIO Controller clock must be enabled. The Input Change Interrupt is available, regardless of the configuration of the I/O line, i.e. configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

When an input change is detected on an I/O line, the corresponding bit in PIO_ISR (Interrupt Status Register) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the thirty-two channels are ORed-wired together to generate a single interrupt signal to the Advanced Interrupt Controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled.

Figure 34-6. Input Change Interrupt Timings



34.5 I/O Lines Programming Example

The programing example as shown in Table 34-1 below is used to define the following configuration.

- 4-bit output port on I/O lines 0 to 3, (should be written in a single write operation), open-drain, with pull-up resistor

- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions, no pull-up resistor
- I/O line 24 to 27 assigned to peripheral A with Input Change Interrupt and pull-up resistor

Table 34-1. Programming Example

Register	Value to be Written
PIO_PER	0x0000 FFFF
PIO_PDR	0x0FFF 0000
PIO_OER	0x0000 00FF
PIO_ODR	0x0FFF FF00
PIO_IFER	0x0000 0F00
PIO_IFDR	0x0FFF F0FF
PIO_SODR	0x0000 0000
PIO_CODR	0x0FFF FFFF
PIO_IER	0x0F00 0F00
PIO_IDR	0x00FF F0FF
PIO_MDER	0x0000 000F
PIO_MDDR	0x0FFF FFF0
PIO_PUDR	0x00F0 00F0
PIO_PUER	0x0F0F FF0F
PIO_ASR	0x0F0F 0000
PIO_BSR	0x00F0 0000
PIO_OWER	0x0000 000F
PIO_OWDR	0x0FFF FFF0

34.6 PIO User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 34-2. PIO Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000
0x001C	Reserved			
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x0000 0000
0x002C	Reserved			
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or(2) Read/Write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved			
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	0x00000000
0x006C	Reserved			

Table 34-2. PIO Register Mapping (Continued)

Offset	Register	Name	Access	Reset Value
0x0070	Peripheral A Select Register[5]	PIO_ASR	Write-only	–
0x0074	Peripheral B Select Register[5]	PIO_BSR	Write-only	–
0x0078	AB Status Register[5]	PIO_ABSR	Read-only	0x00000000
0x007C to 0x009C	Reserved			
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved			

Notes:
1. Reset value of PIO_PSR depends on the product implementation.
2. PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.

34.6.1 PIO Controller PIO Enable Register

Name: PIO_PER

Access: Write-only

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

- **P0-P31: PIO Enable**

0 = No effect.

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

34.6.2 PIO Controller PIO Disable Register

Name: PIO_PDR

Access: Write-only

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

- **P0-P31: PIO Disable**

0 = No effect.

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

34.6.3 PIO Controller PIO Status Register

Name: PIO_PSR

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).

34.6.4 PIO Controller Output Enable Register

Name: PIO_OER

Access: Write-only

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

- **P0-P31: Output Enable**

0 = No effect.

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

34.6.5 PIO Controller Output Disable Register

Name: PIO_ODR

Access: Write-only

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

- **P0-P31: Output Disable**

0 = No effect.

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

34.6.6 PIO Controller Output Status Register

Name: PIO_OSR

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.

34.6.7 PIO Controller Input Filter Enable Register

Name: PIO_IFER

Access: Write-only

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

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

0 = No effect.

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

34.6.8 PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Access: Write-only

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

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

0 = No effect.

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

34.6.9 PIO Controller Input Filter Status Register

Name: PIO_IFSR

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.

34.6.10 PIO Controller Set Output Data Register

Name: PIO_SODR

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.

34.6.11 PIO Controller Clear Output Data Register

Name: PIO_CODR

Access: Write-only

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

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

0 = No effect.

1 = Clears the data to be driven on the I/O line.

34.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

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.

34.6.13 PIO Controller Pin Data Status Register

Name: PIO_PDSR

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.

34.6.14 PIO Controller Interrupt Enable Register

Name: PIO_IER

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.

34.6.15 PIO Controller Interrupt Disable Register

Name: PIO_IDR

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.

34.6.16 PIO Controller Interrupt Mask Register

Name: PIO_IMR

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.

34.6.17 PIO Controller Interrupt Status Register

Name: PIO_ISR

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.

34.6.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Access: Write-only

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

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

0 = No effect.

1 = Enables Multi Drive on the I/O line.

34.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Access: Write-only

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

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

0 = No effect.

1 = Disables Multi Drive on the I/O line.

34.6.20 PIO Multi-driver Status Register

Name: PIO_MDSR

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.

34.6.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Access: Write-only

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

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

0 = No effect.

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

34.6.22 PIO Pull Up Enable Register

Name: PIO_PUER

Access: Write-only

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

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

0 = No effect.

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

34.6.23 PIO Pull Up Status Register

Name: PIO_PUSR

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.

34.6.24 PIO Peripheral A Select Register

Name: PIO_ASR

Access: Write-only

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

- **P0-P31: Peripheral A Select.**

0 = No effect.

1 = Assigns the I/O line to the Peripheral A function.

34.6.25 PIO Peripheral B Select Register

Name: PIO_BSR

Access: Write-only

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

- **P0-P31: Peripheral B Select.**

0 = No effect.

1 = Assigns the I/O line to the peripheral B function.

34.6.26 PIO Peripheral A B Status Register

Name: PIO_ABSR

Access: Read-only

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

- **P0-P31: Peripheral A B Status.**

0 = The I/O line is assigned to the Peripheral A.

1 = The I/O line is assigned to the Peripheral B.

34.6.27 PIO Output Write Enable Register

Name: PIO_OWER

Access: Write-only

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

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

0 = No effect.

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

34.6.28 PIO Output Write Disable Register

Name: PIO_OWDR

Access: Write-only

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

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

0 = No effect.

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

34.6.29 PIO Output Write Status Register

Name: PIO_OWSR

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.

35. Synchronous Serial Controller (SSC)

35.1 Description

The Atmel Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

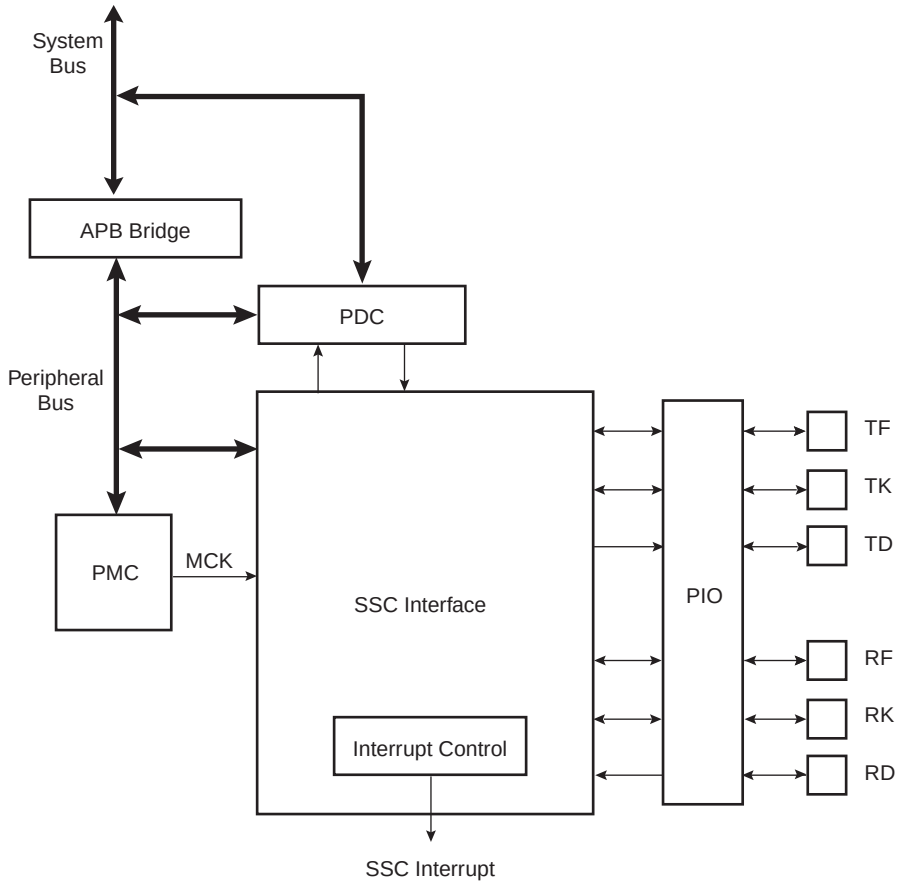
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

- CODEC's in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

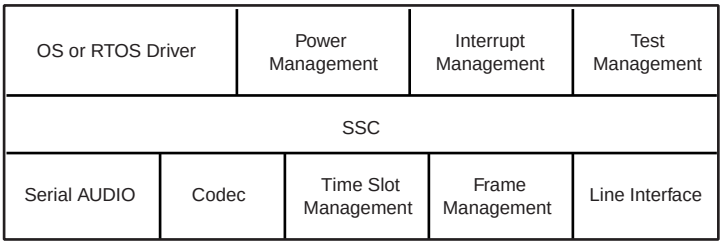
35.2 Block Diagram

Figure 35-1. Block Diagram



35.3 Application Block Diagram

Figure 35-2. Application Block Diagram



35.4 Pin Name List

Table 35-1. I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

35.5 Product Dependencies

35.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

35.5.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

35.5.3 Interrupt

The SSC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the SSC.

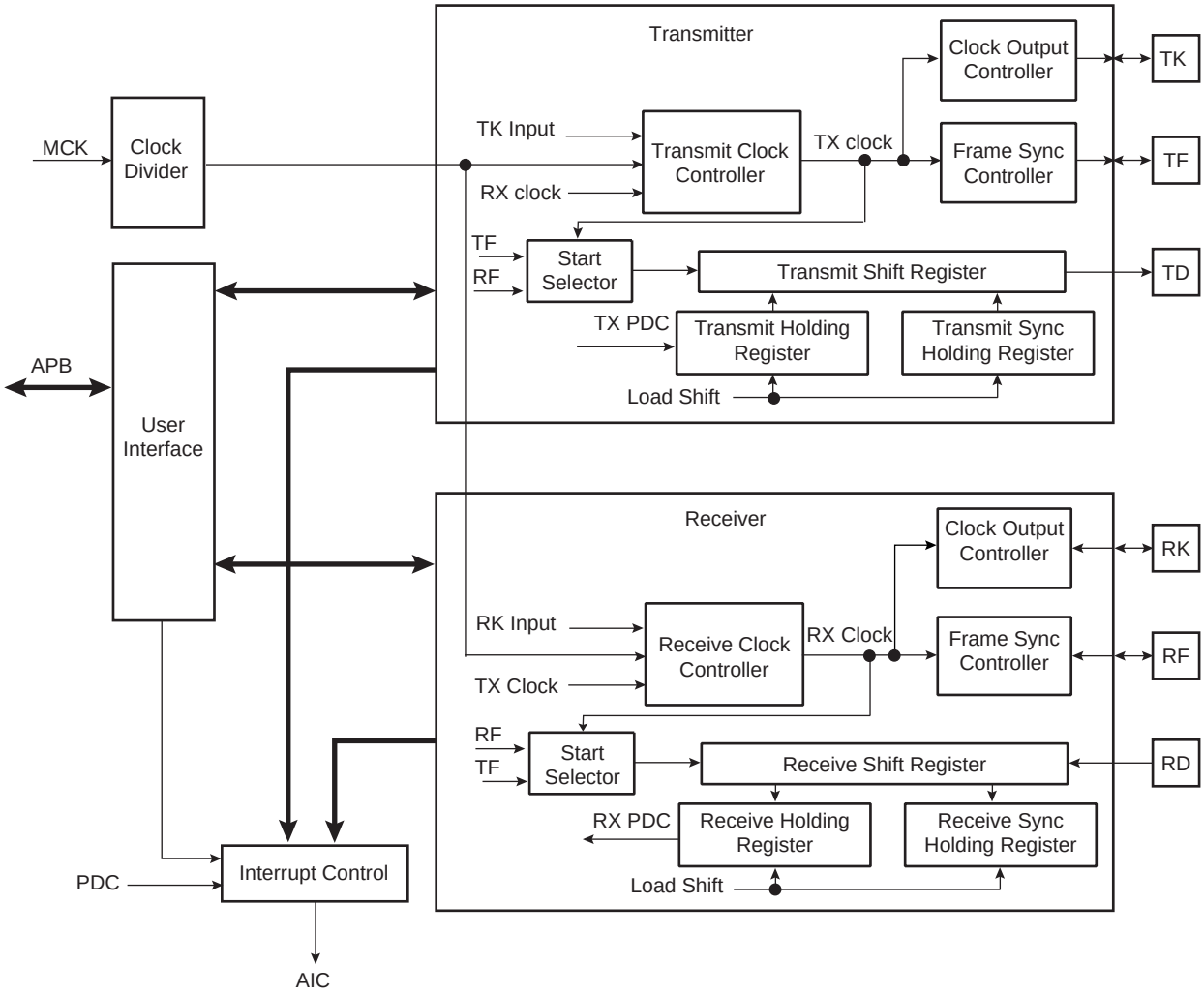
All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

35.6 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2.

Figure 35-3. SSC Functional Block Diagram



35.6.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

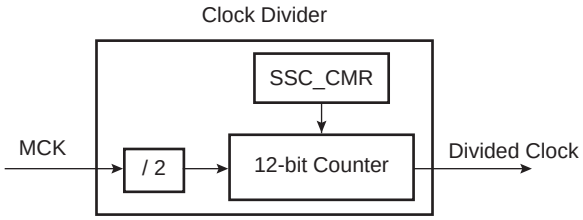
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

35.6.1.1 Clock Divider

Figure 35-4. Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register SSC_CMR, allowing a Master Clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of Master Clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the Master Clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

Figure 35-5. Divided Clock Generation

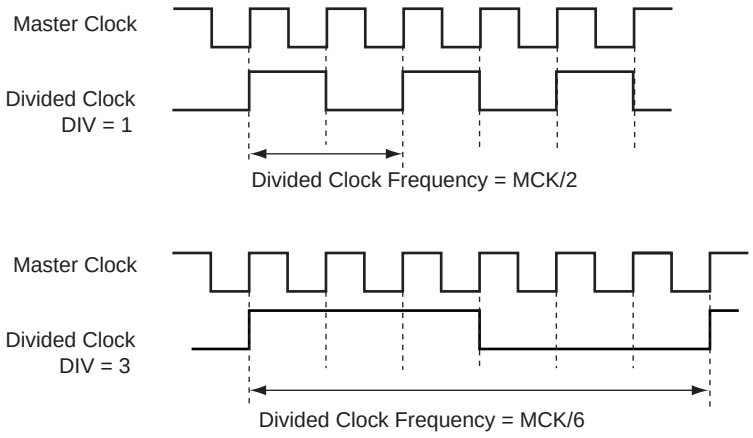


Table 35-2.

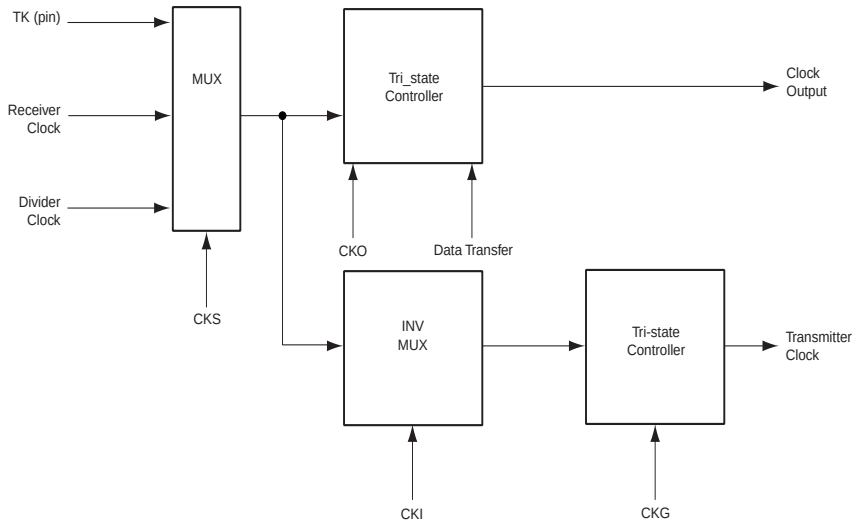
Maximum	Minimum
MCK / 2	MCK / 8190

35.6.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in SSC_TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in SSC_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_TCMR register. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR register to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

Figure 35-6. Transmitter Clock Management

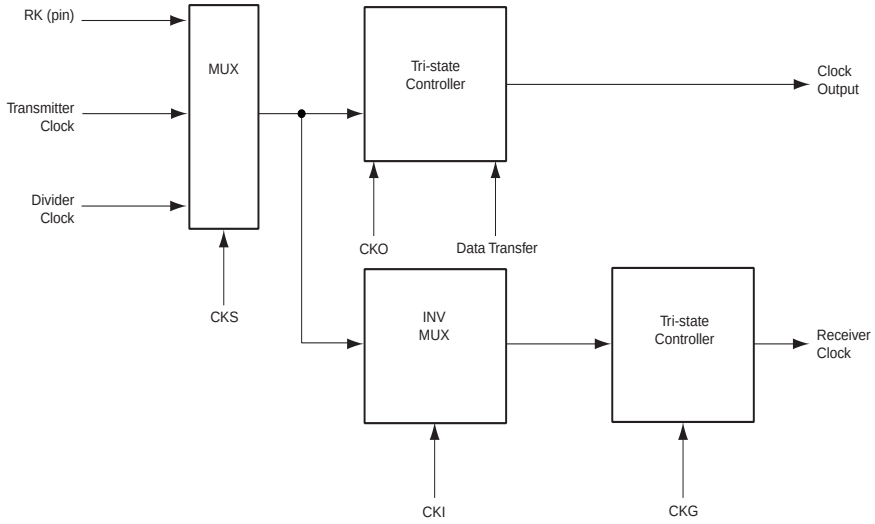


35.6.1.3 *Receiver Clock Management*

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_RCMR register. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR register to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 35-7. Receiver Clock Management



35.6.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Master Clock divided by 2 if Receiver Frame Synchro is input
- Master Clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

- Master Clock divided by 6 if Transmit Frame Synchro is input
- Master Clock divided by 2 if Transmit Frame Synchro is output

35.6.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

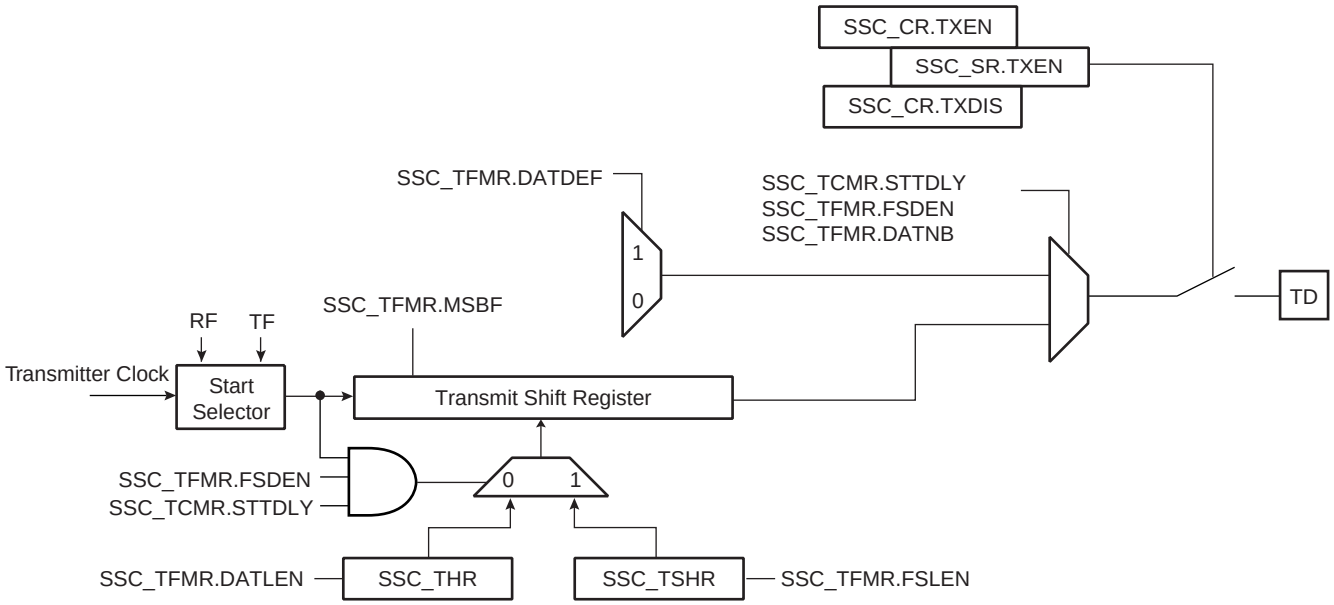
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See “Start” on page 471.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See “Frame Sync” on page 473.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC_TCMR. Data is written by the application to the SSC_THR register then transferred to the shift register according to the data format selected.

When both the SSC_THR and the transmit shift register are empty, the status flag TXEMPTY is set in SSC_SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SSC_SR and additional data can be loaded in the holding register.

Figure 35-8. Transmitter Block Diagram



35.6.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

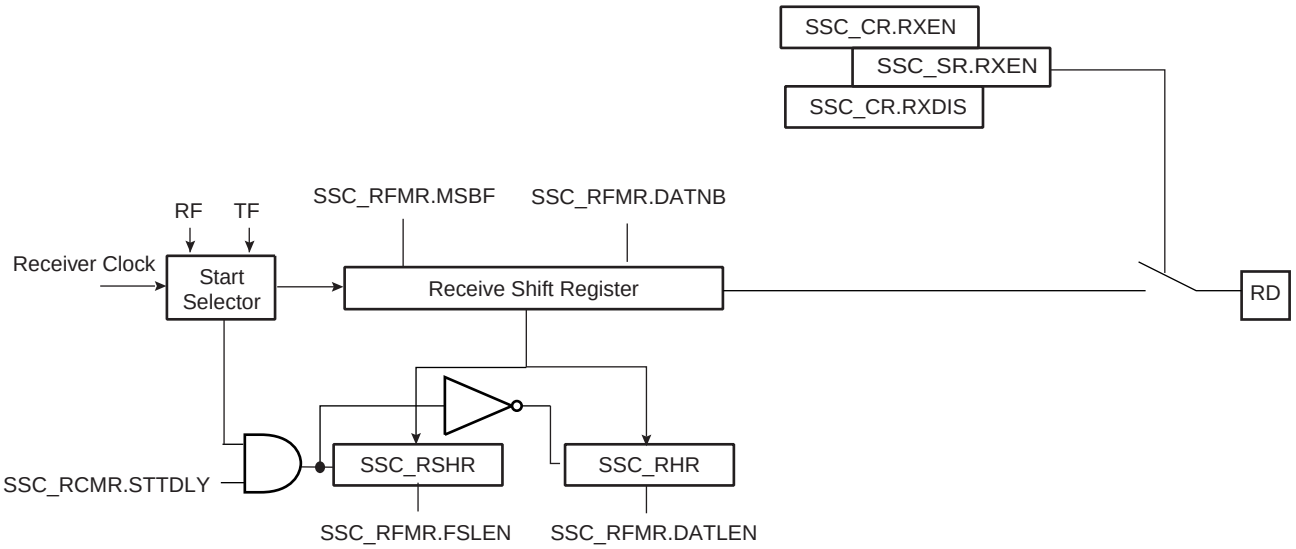
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See “Start” on page 471.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See “Frame Sync” on page 473.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR register, the status flag OVERUN is set in SSC_SR and the receiver shift register is transferred in the RHR register.

Figure 35-9. Receiver Block Diagram



35.6.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (TFMR/RFMR).

Figure 35-10. Transmit Start Mode

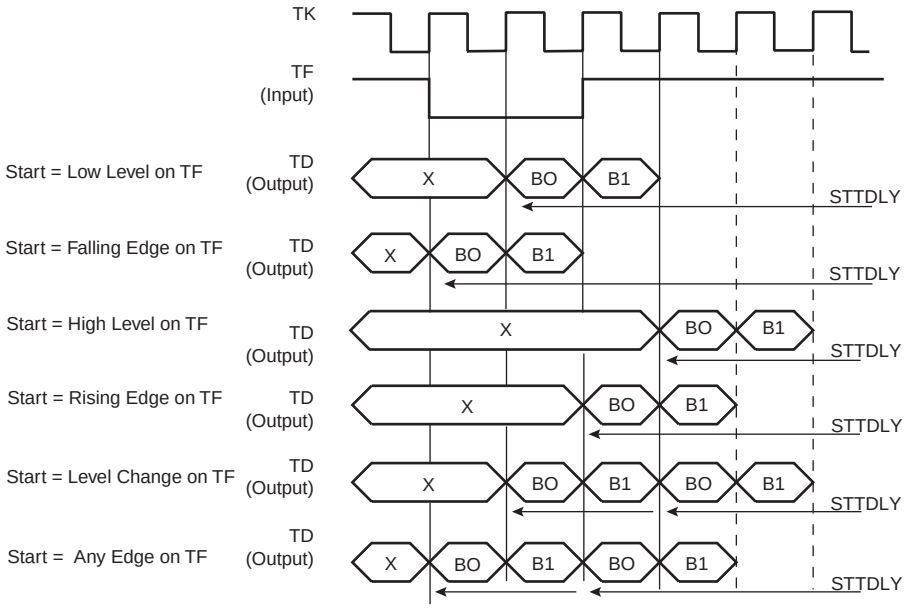
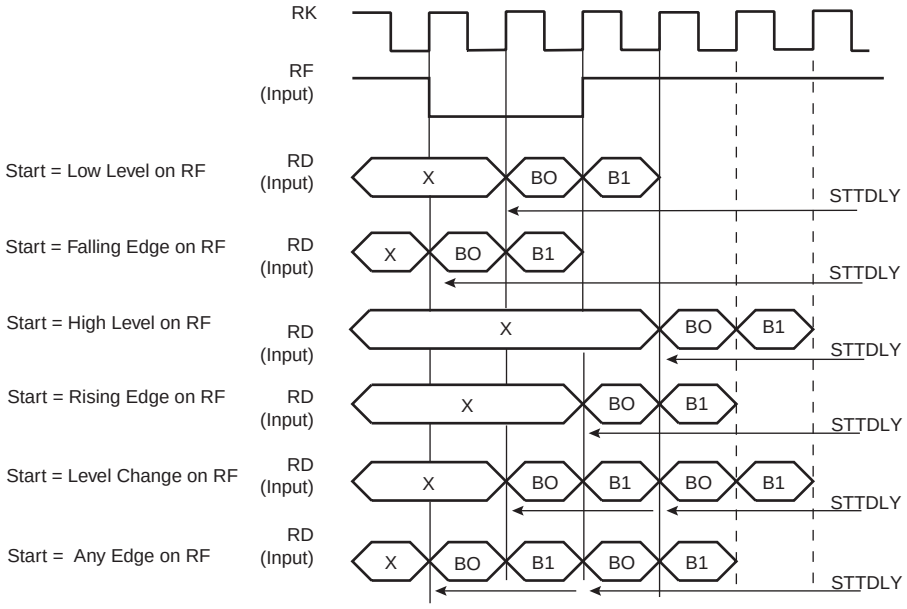


Figure 35-11. Receive Pulse/Edge Start Modes



35.6.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 16 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

35.6.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC_RFMR/SSC_TFMR and has a maximum value of 16.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

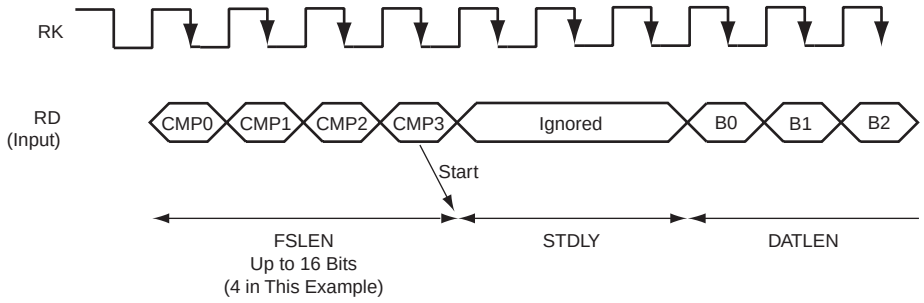
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

35.6.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC_RFMR/SSC_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC_SR) on frame synchro edge detection (signals RF/TF).

35.6.6 Receive Compare Modes

Figure 35-12. Receive Compare Modes



35.6.6.1 *Compare Functions*

Length of the comparison patterns (Compare 0, Compare 1) and thus the number of bits they are compared to is defined by FSLEN, but with a maximum value of 16 bits. Comparison is always done by comparing the last bits received with the comparison pattern. Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last bits received at the Compare 0 pattern contained in the Compare 0 Register (SSC_RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the bit (STOP) in SSC_RCMR.

35.6.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC_TFMR) and the Receiver Frame Mode Register (SSC_RFMR). In either case, the user can independently select:

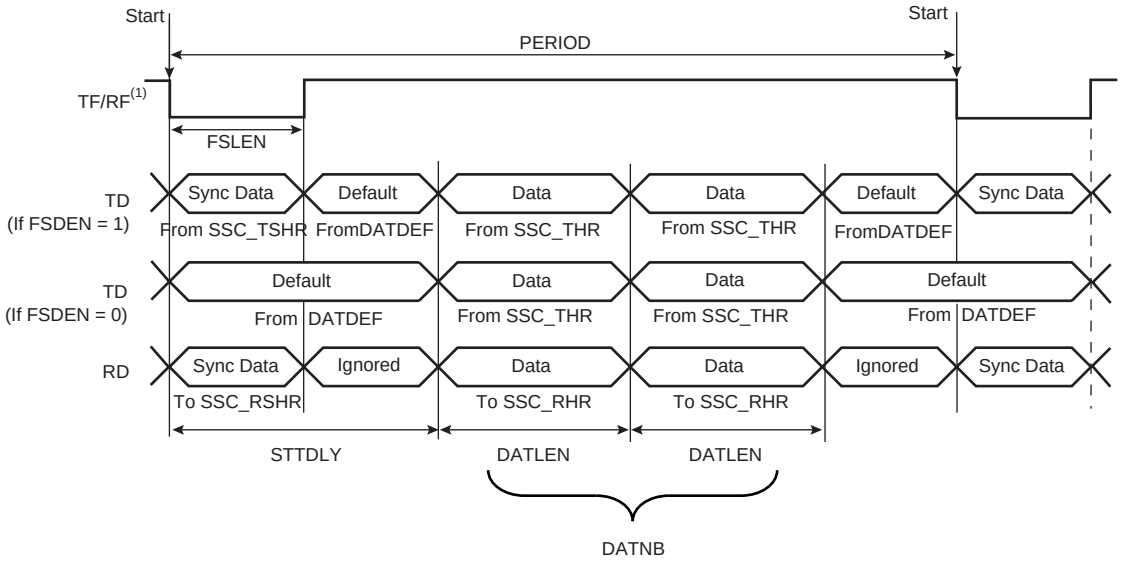
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start event (FSLEN)
- the bit sense: most or lowest significant bit first (MSBF)

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC_TFMR.

Table 35-3. Data Frame Registers

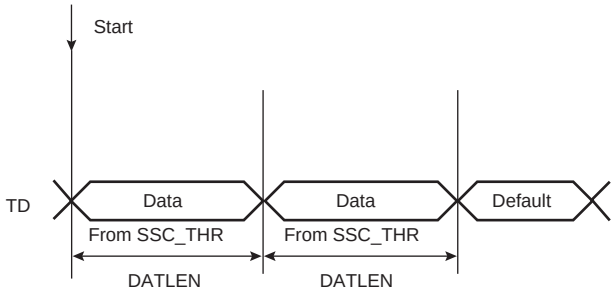
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data default value ended
SSC_TFMR		FSDEN		Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

Figure 35-13. Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

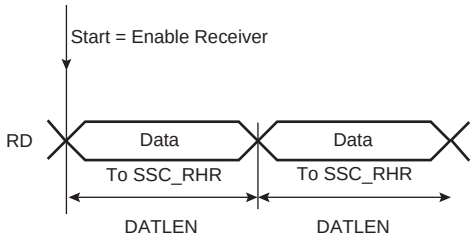
Figure 35-14. Transmit Frame Format in Continuous Mode



Start: 1. TXEMPTY set to 1
2. Write into the SSC_THR

Note: 1. STTDLY is set to 0. In this example, SSC_THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 35-15. Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

35.6.8 Loop Mode

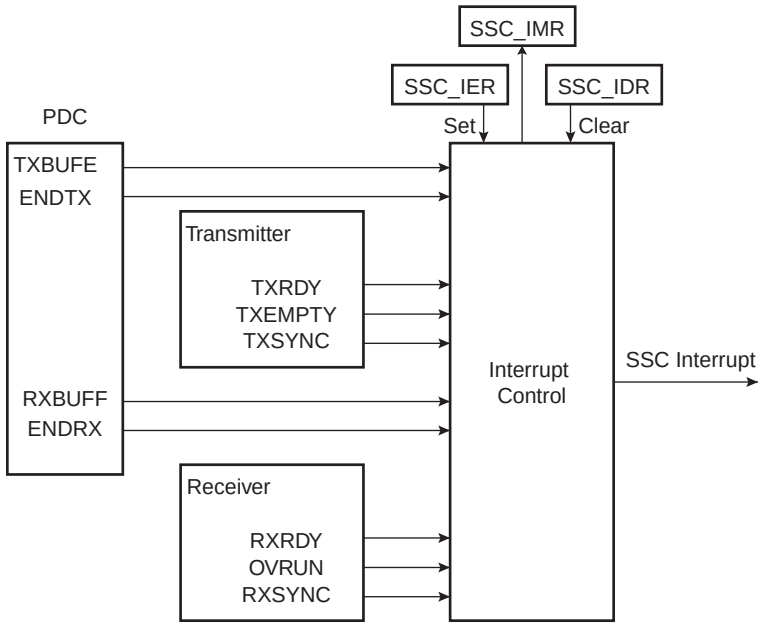
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

35.6.9 Interrupt

Most bits in SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing SSC_IER (Interrupt Enable Register) and SSC_IDR (Interrupt Disable Register) These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in SSC_IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the AIC.

Figure 35-16. Interrupt Block Diagram



35.7 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 35-17. Audio Application Block Diagram

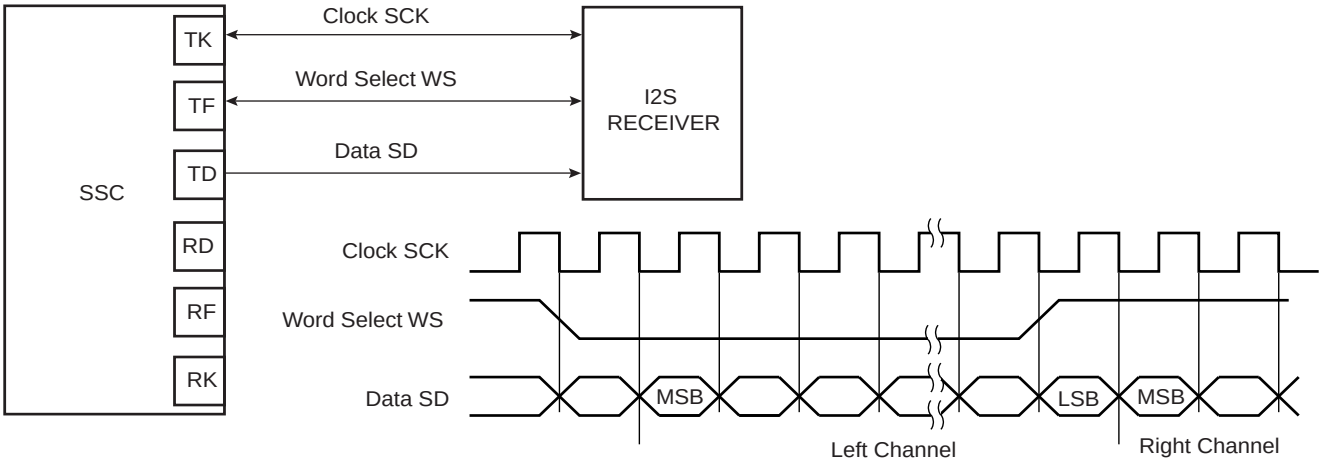


Figure 35-18. Codec Application Block Diagram

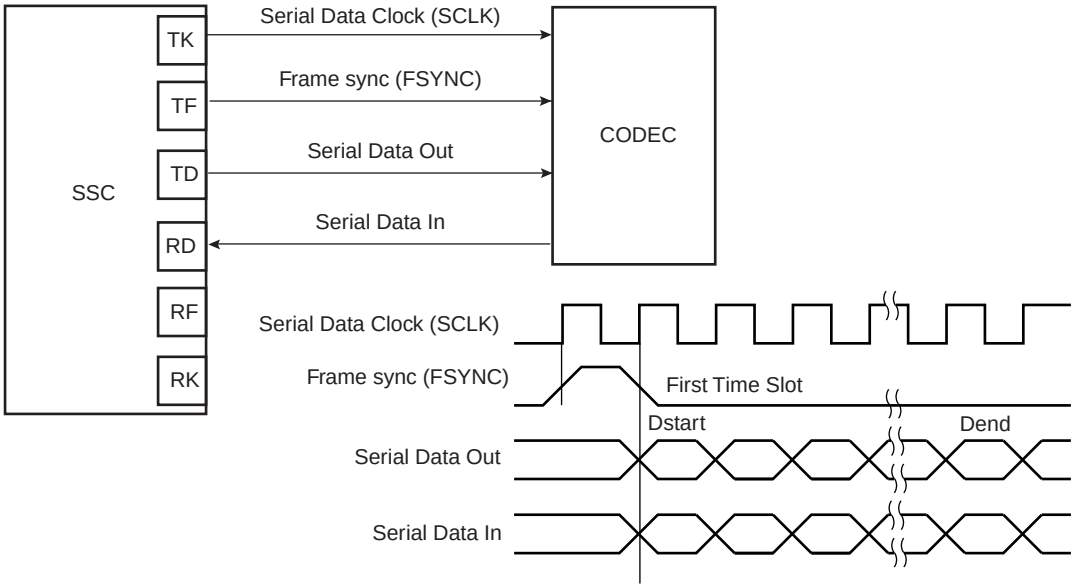
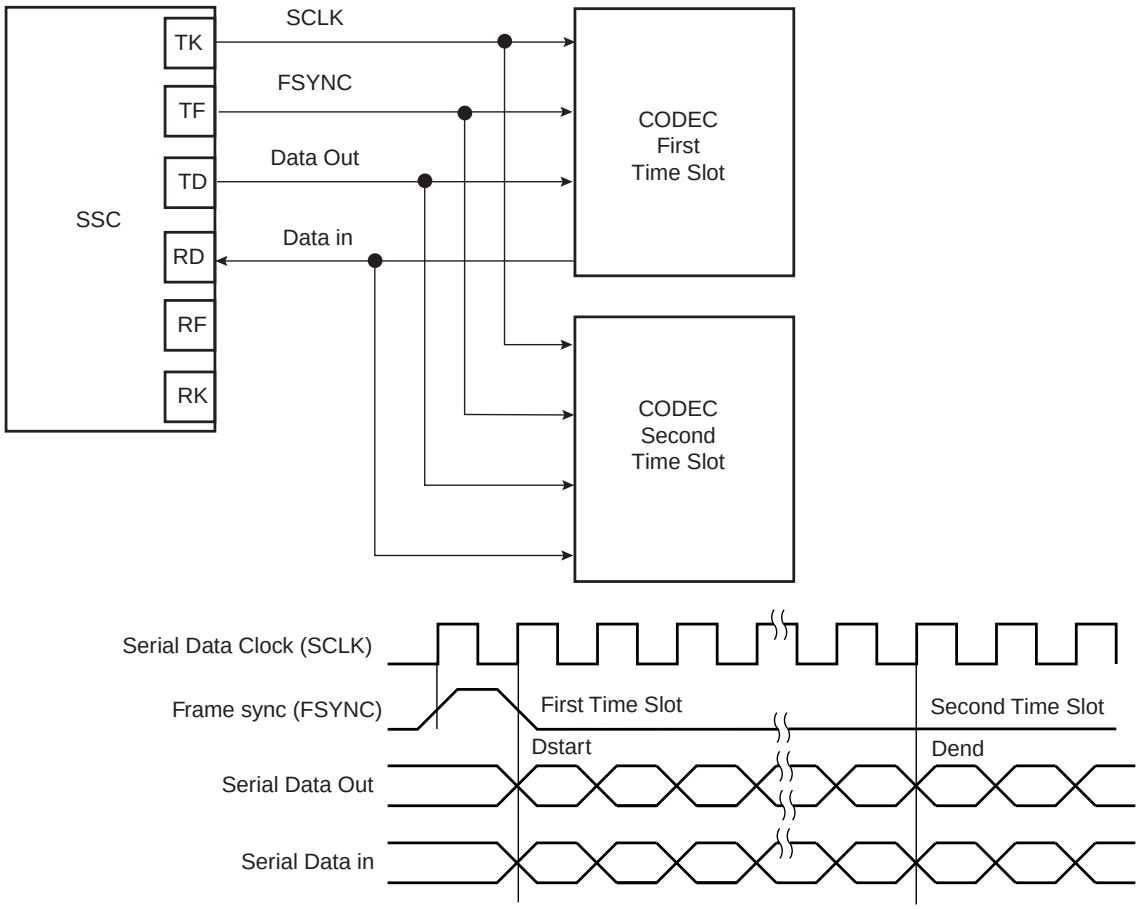


Figure 35-19. Time Slot Application Block Diagram



35.8 Synchronous Serial Controller (SSC) User Interface

Table 35-4. Register Mapping

Offset	Register	Register Name	Access	Reset
0x0	Control Register	SSC_CR	Write	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8	Reserved	–	–	–
0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read	0x0
0x24	Transmit Holding Register	SSC_THR	Write	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write	–
0x48	Interrupt Disable Register	SSC_IDR	Write	–
0x4C	Interrupt Mask Register	SSC_IMR	Read	0x0
0x50-0xFC	Reserved	–	–	–
0x100- 0x124	Reserved for Peripheral Data Controller (PDC)	–	–	–

35.8.1 SSC Control Register

Name: SSC_CR

Access: Write-only

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

- **RXEN: Receive Enable**

0: No effect.

1: Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0: No effect.

1: Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC_CR.

35.8.2 SSC Clock Mode Register

Name: SSC_CMR

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

- **DIV: Clock Divider**

0: The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is MCK/2. The minimum bit rate is MCK/2 x 4095 = MCK/8190.

35.8.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Access: Read/Write

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

- **CKS: Receive Clock Selection**

CKS	Selected Receive Clock
0x0	Divided Clock
0x1	TK Clock signal
0x2	RK pin
0x3	Reserved

- **CKO: Receive Clock Output Mode Selection**

CKO	Receive Clock Output Mode	RK pin
0x0	None	Input-only
0x1	Continuous Receive Clock	Output
0x2	Receive Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Receive Clock Inversion**

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

- **CKG: Receive Clock Gating Selection**

CKG	Receive Clock Gating
0x0	None, continuous clock
0x1	Receive Clock enabled only if RF Low
0x2	Receive Clock enabled only if RF High
0x3	Reserved

- **START: Receive Start Selection**

START	Receive Start
0x0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
0x1	Transmit start
0x2	Detection of a low level on RF signal
0x3	Detection of a high level on RF signal
0x4	Detection of a falling edge on RF signal
0x5	Detection of a rising edge on RF signal
0x6	Detection of any level change on RF signal
0x7	Detection of any edge on RF signal
0x8	Compare 0
0x9-0xF	Reserved

- **STOP: Receive Stop Selection**

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

- **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

- **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD+1) Receive Clock.

35.8.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Access: Read/Write

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

23	22	21	20	19	18	17	16
–	FSOS			FSLEN			

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

7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Receiver. If DATLEN is lower or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **LOOP: Loop Mode**

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Receive Frame Sync Length**

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + 1 Receive Clock periods.

- **FSOS: Receive Frame Sync Output Selection**

FSOS	Selected Receive Frame Sync Signal	RF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

35.8.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Access: Read/Write

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

- **CKS: Transmit Clock Selection**

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

- **CKO: Transmit Clock Output Mode Selection**

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2	Transmit Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Transmit Clock Inversion**

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

- **CKG: Transmit Clock Gating Selection**

CKG	Transmit Clock Gating
0x0	None, continuous clock
0x1	Transmit Clock enabled only if TF Low
0x2	Transmit Clock enabled only if TF High
0x3	Reserved

- **START: Transmit Start Selection**

START	Transmit Start
0x0	Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
0x1	Receive start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8 - 0xF	Reserved

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each 2 x (PERIOD+1) Transmit Clock.

35.8.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Access: Read/Write

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

23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			

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

7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Transmit. If DATLEN is lower or equal to 7, data transfers are bytes, if DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per frame**

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB +1).

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

This field is used with FSLEN_EXT to determine the pulse length of the Transmit Frame Sync signal.

Pulse length is equal to FSLEN + 1 Transmit Clock periods.

- **FSOS: Transmit Frame Sync Output Selection**

FSOS	Selected Transmit Frame Sync Signal	TF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSDEN: Frame Sync Data Enable**

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

35.8.7 SSC Receive Holding Register

Name: SSC_RHR

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.

35.8.8 SSC Transmit Holding Register

Name: SSC_THR

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.

35.8.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

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**

35.8.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

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**

35.8.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CP0							
7	6	5	4	3	2	1	0
CP0							

- **CP0: Receive Compare Data 0**

35.8.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CP1							
7	6	5	4	3	2	1	0
CP1							

- **CP1: Receive Compare Data 1**

35.8.13 SSC Status Register

Name: SSC_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready**

0: Data has been loaded in SSC_THR and is waiting to be loaded in the Transmit Shift Register (TSR).

1: SSC_THR is empty.

- **TXEMPTY: Transmit Empty**

0: Data remains in SSC_THR or is currently transmitted from TSR.

1: Last data written in SSC_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

- **ENDTX: End of Transmission**

0: The register SSC_TCR has not reached 0 since the last write in SSC_TCR or SSC_TNCR.

1: The register SSC_TCR has reached 0 since the last write in SSC_TCR or SSC_TNCR.

- **TXBUFE: Transmit Buffer Empty**

0: SSC_TCR or SSC_TNCR have a value other than 0.

1: Both SSC_TCR and SSC_TNCR have a value of 0.

- **RXRDY: Receive Ready**

0: SSC_RHR is empty.

1: Data has been received and loaded in SSC_RHR.

- **OVRUN: Receive Overrun**

0: No data has been loaded in SSC_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC_RHR while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception**

0: Data is written on the Receive Counter Register or Receive Next Counter Register.

1: End of PDC transfer when Receive Counter Register has arrived at zero.

- **RXBUFF: Receive Buffer Full**

0: SSC_RCR or SSC_RNCR have a value other than 0.

1: Both SSC_RCR and SSC_RNCR have a value of 0.

- **CP0: Compare 0**

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

- **CP1: Compare 1**

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.

- **TXSYN: Transmit Sync**

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

- **RXSYN: Receive Sync**

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0: Transmit is disabled.

1: Transmit is enabled.

- **RXEN: Receive Enable**

0: Receive is disabled.

1: Receive is enabled.

35.8.14 SSC Interrupt Enable Register

Name: SSC_IER

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0: No effect.

1: Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Enable**

0: No effect.

1: Enables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Buffer Empty Interrupt

- **RXRDY: Receive Ready Interrupt Enable**

0: No effect.

1: Enables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Enable**

0: No effect.

1: Enables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Enable**

0: No effect.

1: Enables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0: No effect.

1: Enables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0: No effect.

1: Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0: No effect.

1: Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Enables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Enables the Rx Sync Interrupt.

35.8.15 SSC Interrupt Disable Register

Name: SSC_IDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Disable**

0: No effect.

1: Disables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Disable**

0: No effect.

1: Disables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Buffer Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Disable**

0: No effect.

1: Disables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Disable**

0: No effect.

1: Disables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Disable**

0: No effect.

1: Disables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Disable**

0: No effect.

1: Disables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Disable**

0: No effect.

1: Disables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Disable**

0: No effect.

1: Disables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Disables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Disables the Rx Sync Interrupt.

35.8.16 SSC Interrupt Mask Register

Name: SSC_IMR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Mask**

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

- **TXEMPTY: Transmit Empty Interrupt Mask**

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

- **ENDTX: End of Transmission Interrupt Mask**

0: The End of Transmission Interrupt is disabled.

1: The End of Transmission Interrupt is enabled.

- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0: The Transmit Buffer Empty Interrupt is disabled.

1: The Transmit Buffer Empty Interrupt is enabled.

- **RXRDY: Receive Ready Interrupt Mask**

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

- **OVRUN: Receive Overrun Interrupt Mask**

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

- **ENDRX: End of Reception Interrupt Mask**

0: The End of Reception Interrupt is disabled.

1: The End of Reception Interrupt is enabled.

- **RXBUFF: Receive Buffer Full Interrupt Mask**

0: The Receive Buffer Full Interrupt is disabled.

1: The Receive Buffer Full Interrupt is enabled.

- **CP0: Compare 0 Interrupt Mask**

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

- **CP1: Compare 1 Interrupt Mask**

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.

- **TXSYN: Tx Sync Interrupt Mask**

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

- **RXSYN: Rx Sync Interrupt Mask**

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

36. Timer/Counter (TC)

36.1 Overview

The Timer Counter (TC) includes three identical 16-bit Timer Counter channels.

Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The Timer Counter block has two global registers which act upon all three TC channels.

The Block Control Register allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register defines the external clock inputs for each channel, allowing them to be chained.

Table gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2.

Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	MCK/1024

36.2 Block Diagram

Figure 36-1. Timer/Counter Block Diagram

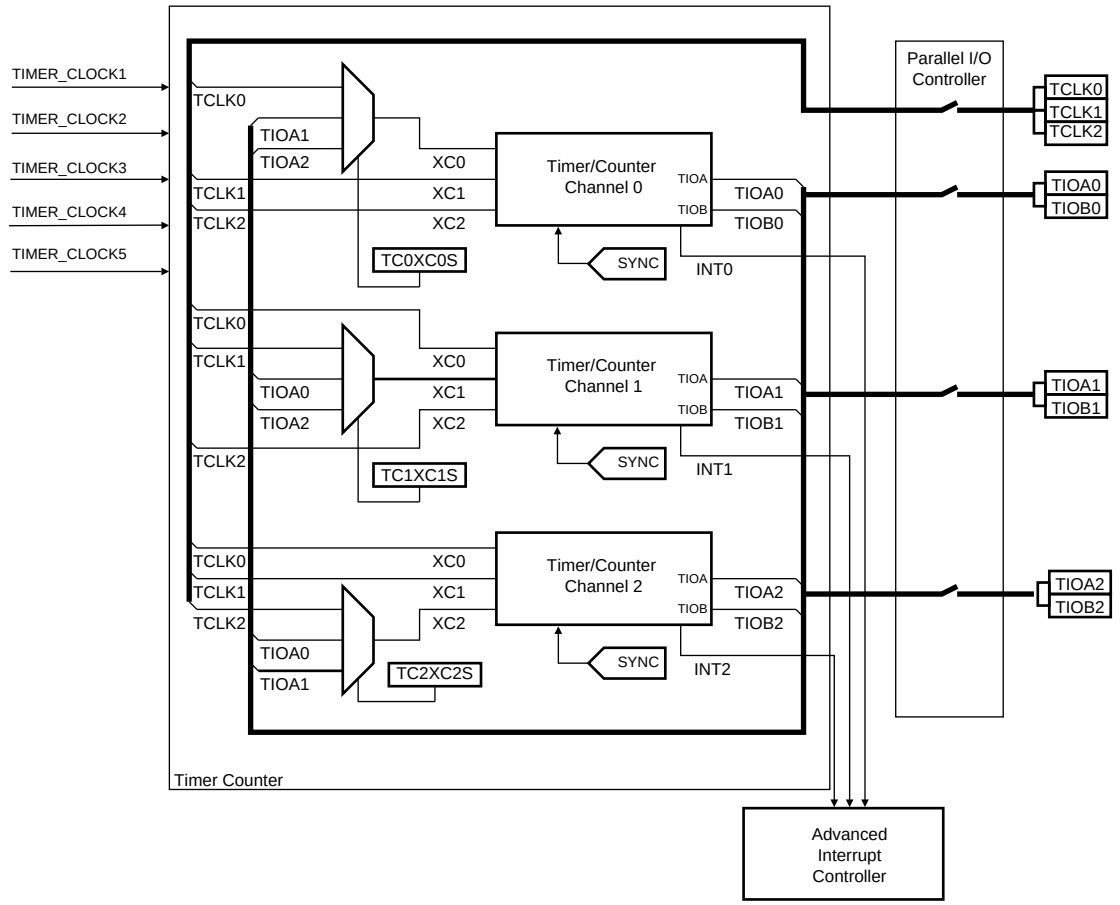


Table 36-1. Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer/Counter Input Waveform Mode: Timer/Counter Output
	TIOB	Capture Mode: Timer/Counter Input Waveform Mode: Timer/Counter Input/output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

36.3 Pin Name List

Table 36-2. TC pin list

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

36.4 Product Dependencies

36.4.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

36.4.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer/Counter clock.

36.4.3 Interrupt

The TC has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the TC interrupt requires programming the AIC before configuring the TC.

36.5 Functional Description

36.5.1 TC Description

The three channels of the Timer/Counter are independent and identical in operation. The registers for channel programming are listed in Table 36-4 on page 517.

36.5.1.1 16-bit Counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the COVFS bit in TC_SR (Status Register) is set.

The current value of the counter is accessible in real time by reading the Counter Value Register, TC_CV. The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

36.5.1.2 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 configurable I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC_BMR (Block Mode). See Figure 36-2 on page 505.

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5
- External clock signals: XC0, XC1 or XC2

This selection is made by the TCCLKS bits in the TC Channel Mode Register.

The selected clock can be inverted with the CLKI bit in TC_CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2). See Figure 36-3 on page 505.

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the master clock period. The external clock frequency must be at least 2.5 times lower than the master clock.

Figure 36-2. Clock Chaining Selection

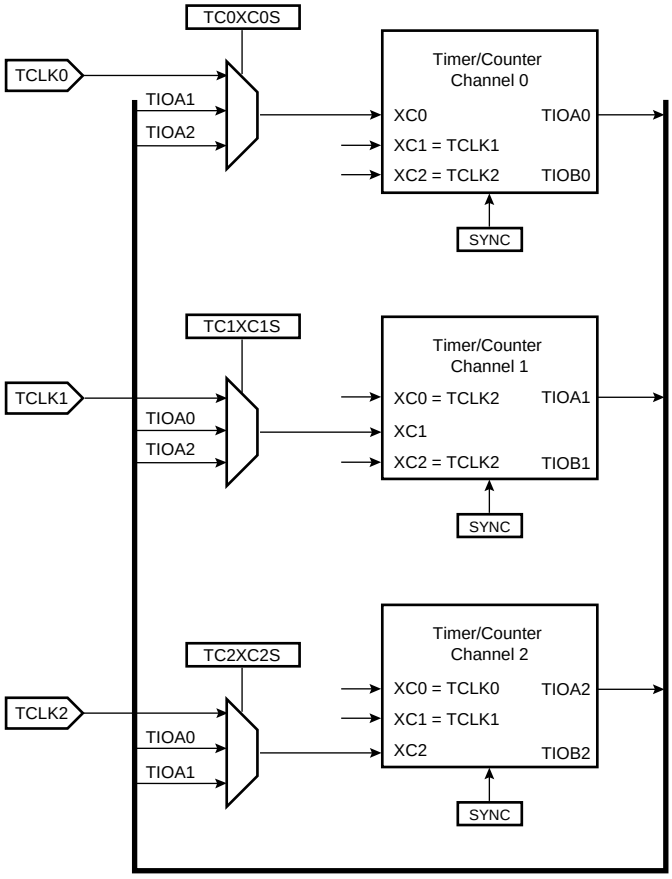
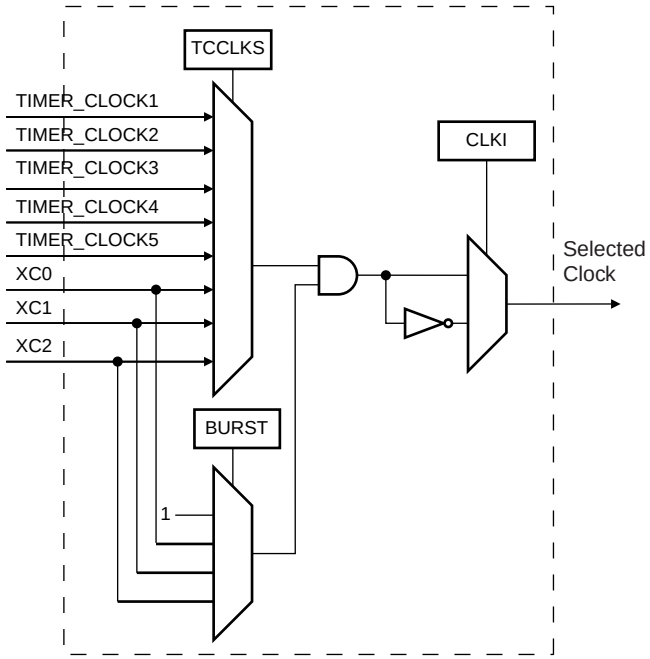


Figure 36-3. Clock Selection

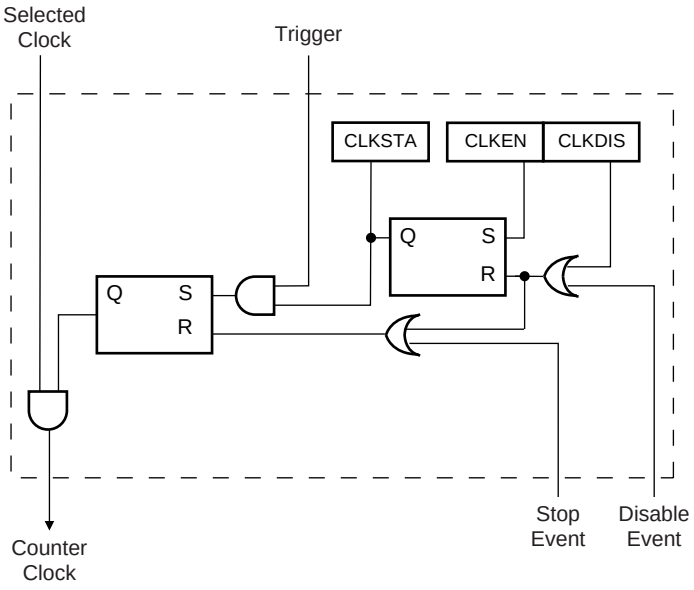


36.5.1.3 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 36-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the Control Register. In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in TC_CMR. In Waveform Mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the Control Register can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the Status Register.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture Mode (LDBSTOP = 1 in TC_CMR) or a RC compare event in Waveform Mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands have effect only if the clock is enabled.

Figure 36-4. Clock Control



36.5.1.4 TC Operating Modes

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register.

In Capture Mode, TIOA and TIOB are configured as inputs.

In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

36.5.1.5 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TC_CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in TC_CMR.

The channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting ENETRG in TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the master clock period in order to be detected.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

36.5.2 Capture Operating Mode

This mode is entered by clearing the WAVE parameter in TC_CMR (Channel Mode Register).

Capture Mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 36-5 shows the configuration of the TC channel when programmed in Capture Mode.

36.5.2.1 Capture Registers A and B

Registers A and B (RA and RB) are used as capture registers. This means that they can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The LDRA parameter in TC_CMR defines the TIOA edge for the loading of register A, and the LDRB parameter defines the TIOA edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

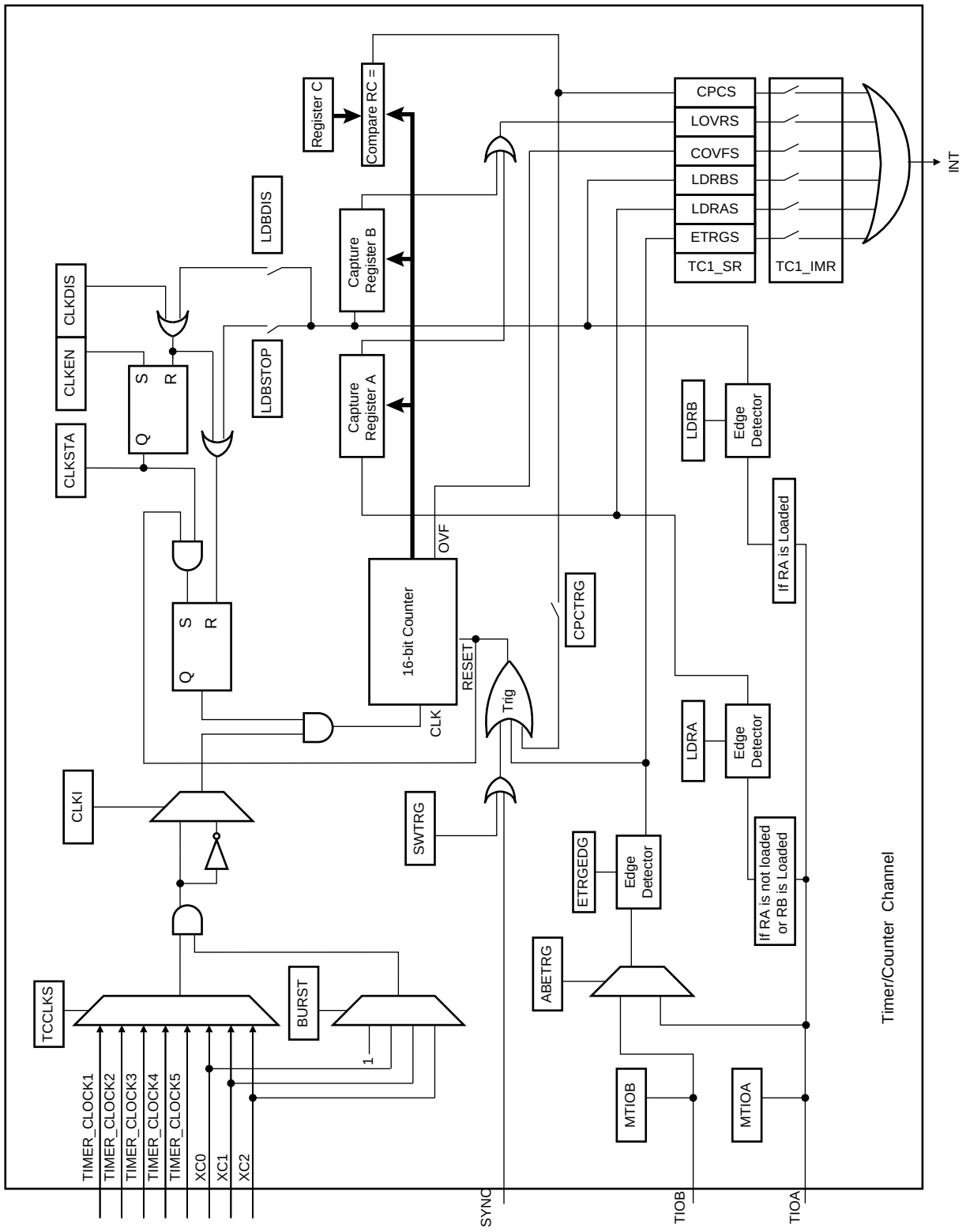
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

36.5.2.2 Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The ABETRG bit in TC_CMR selects TIOA or TIOB input signal as an external trigger. The ETRGEDG parameter defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Figure 36-5. Capture Mode



36.5.3 Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in TC_CMR (Channel Mode Register).

In Waveform Operating Mode the TC channel generates 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 36-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

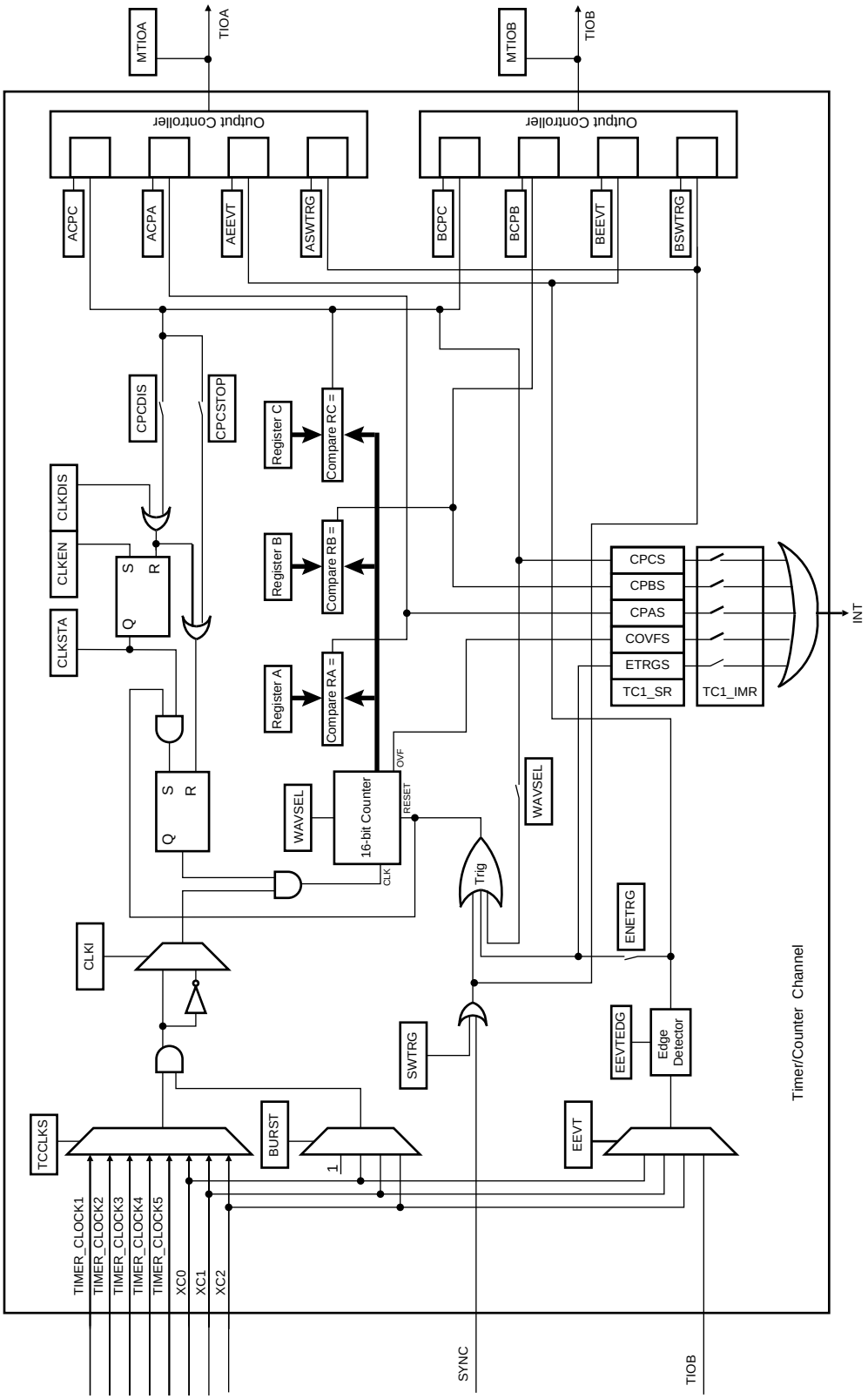
36.5.3.1 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Figure 36-6. Waveform Mode



36.5.3.2 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 36-7.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 36-8.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 36-7. WAVSEL= 00 without trigger

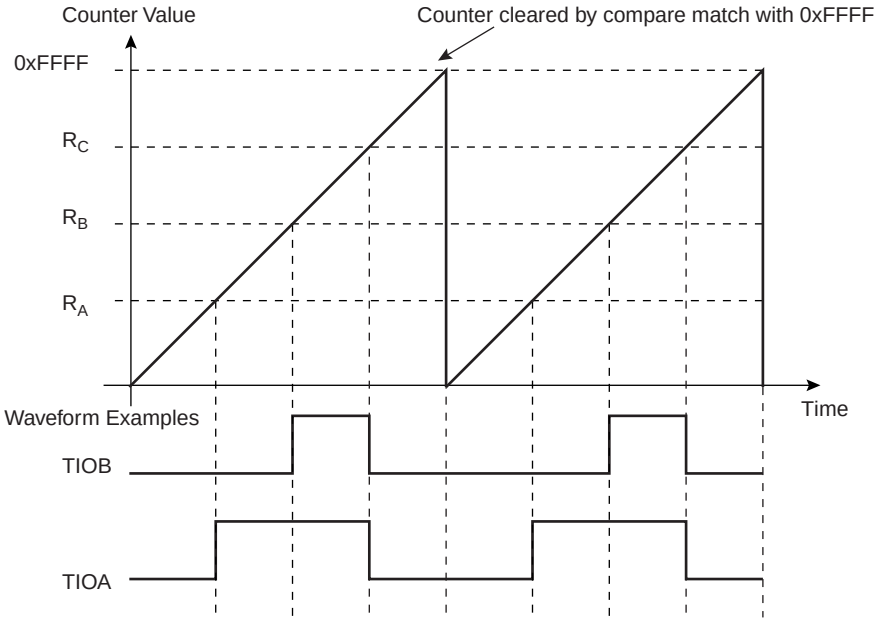
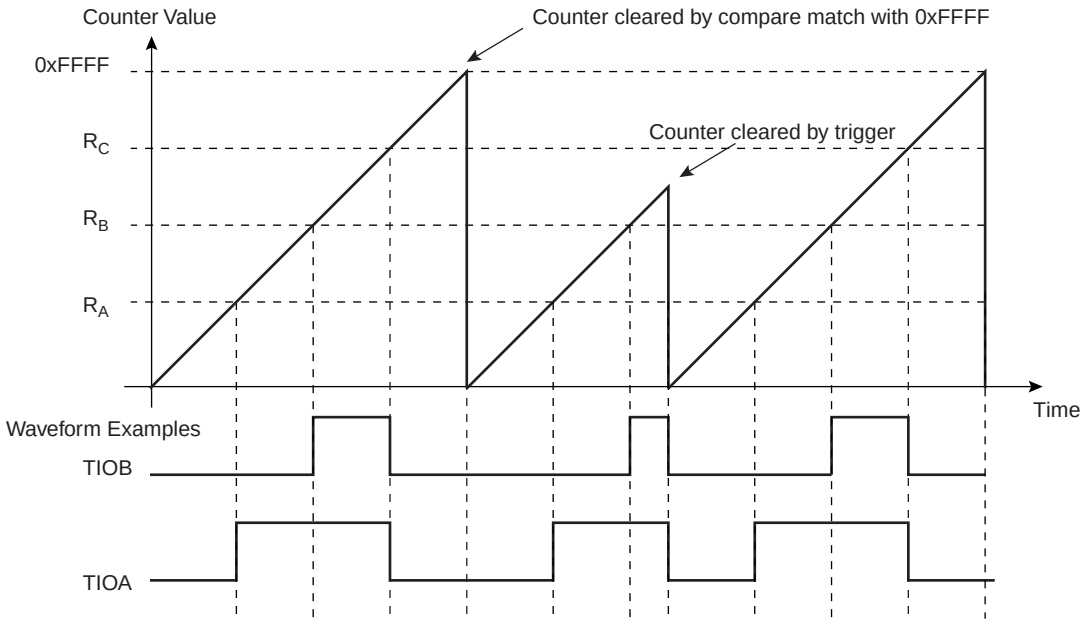


Figure 36-8. WAVSEL= 00 with trigger



36.5.3.3 *WAVSEL = 10*

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 36-9.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 36-10.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 36-9. WAVSEL = 10 Without Trigger

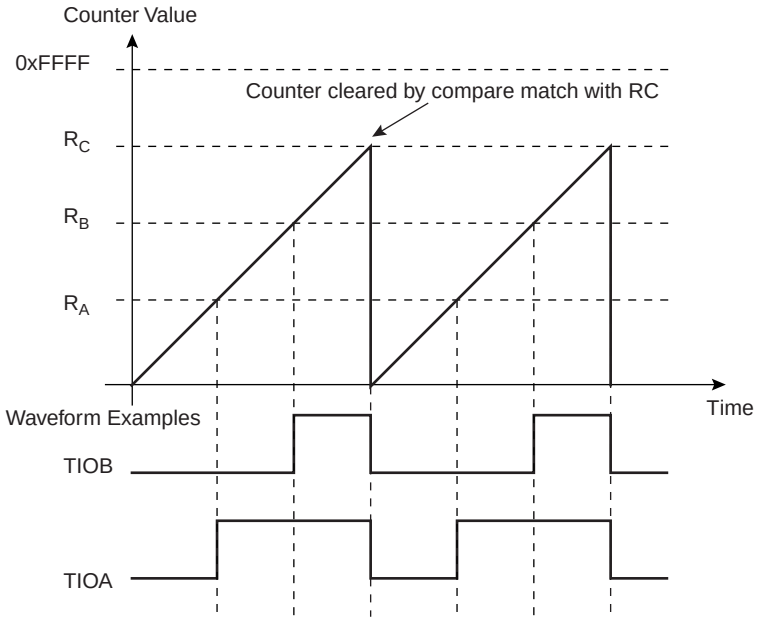
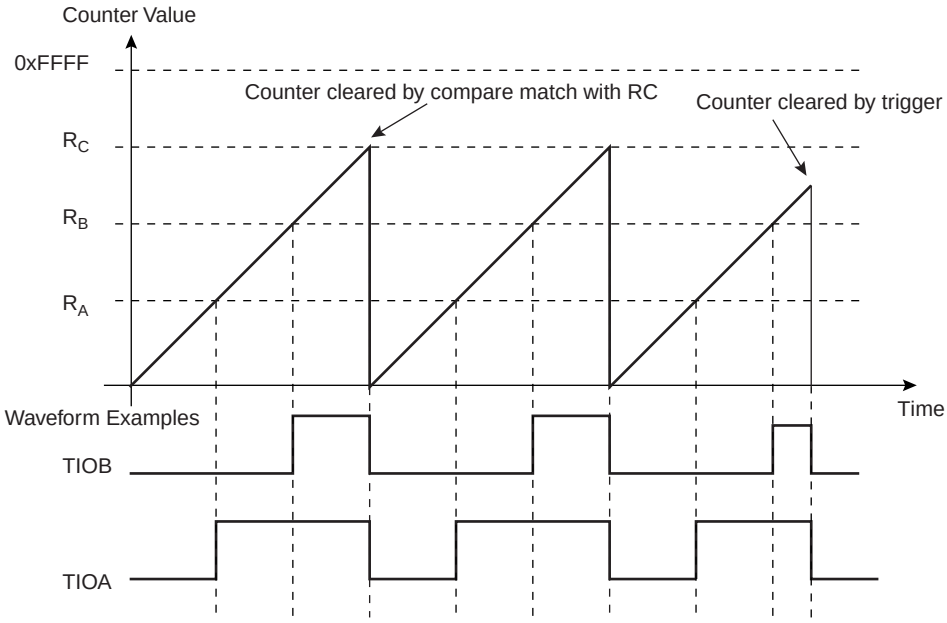


Figure 36-10. WAVSEL = 10 With Trigger



36.5.3.4 WAVSEL = 01

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 36-11.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 36-12.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 36-11. WAVSEL = 01 Without Trigger

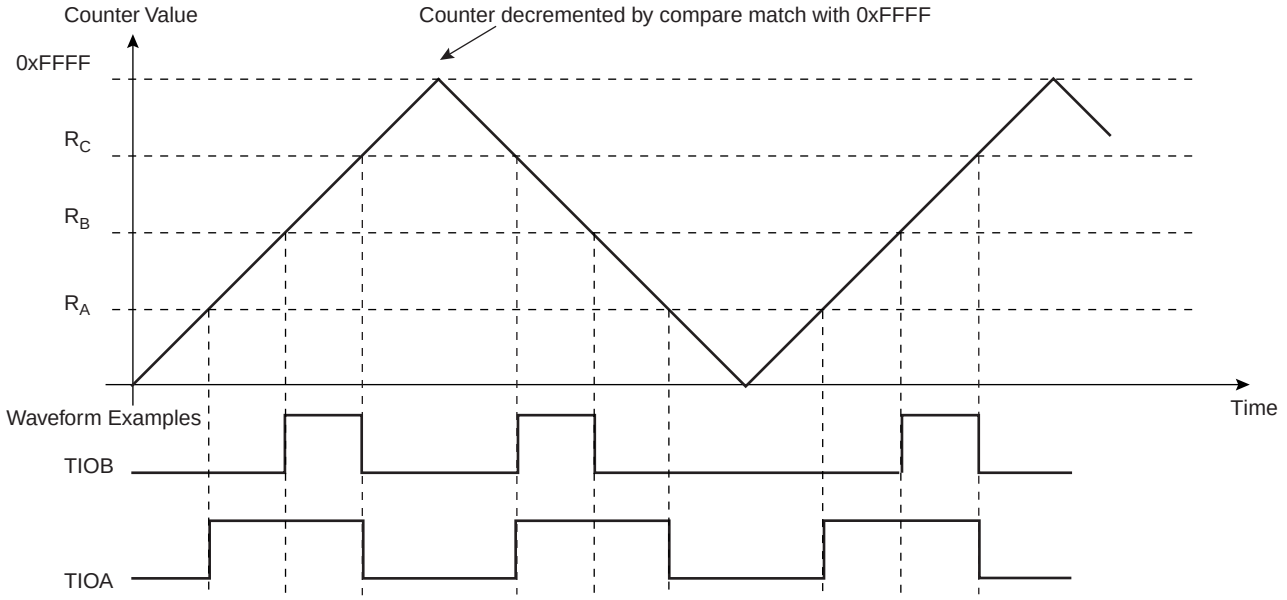
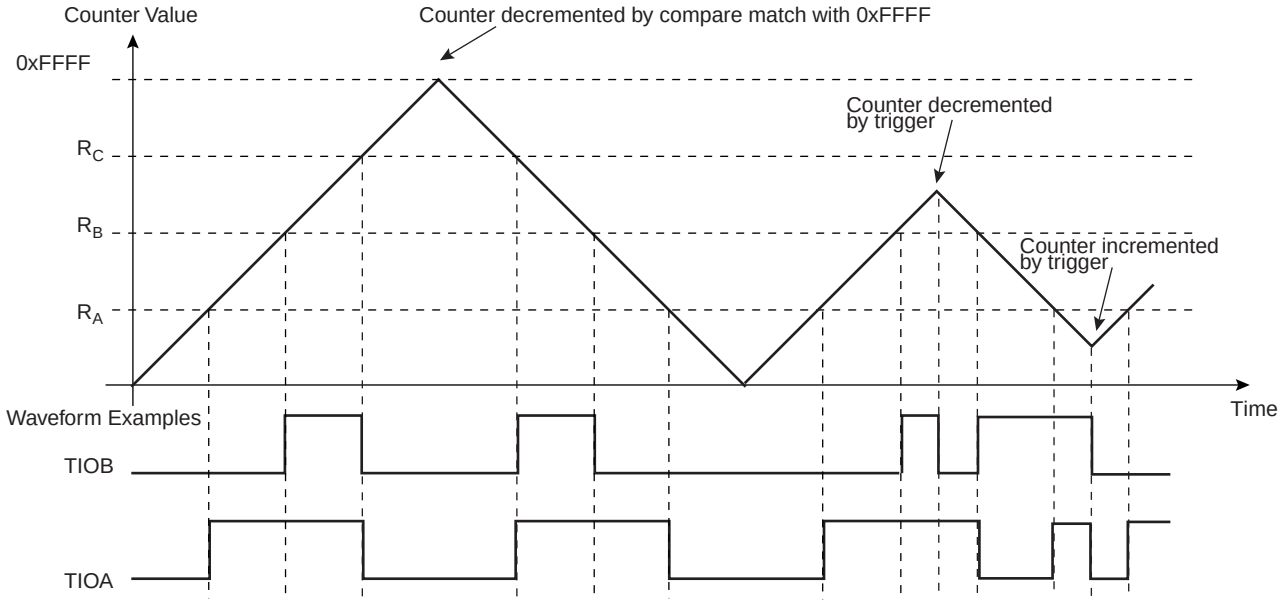


Figure 36-12. WAVSEL = 01 With Trigger



36.5.3.5 WAVSEL = 11

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 36-13.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 36-14.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 36-13. WAVSEL = 11 Without Trigger

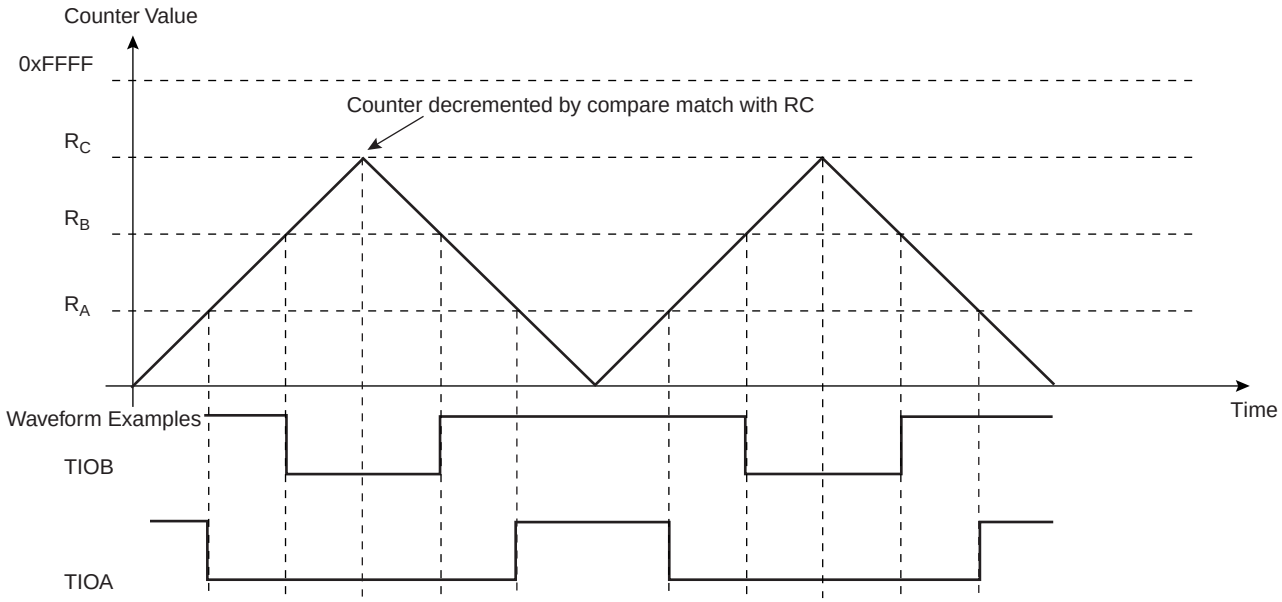
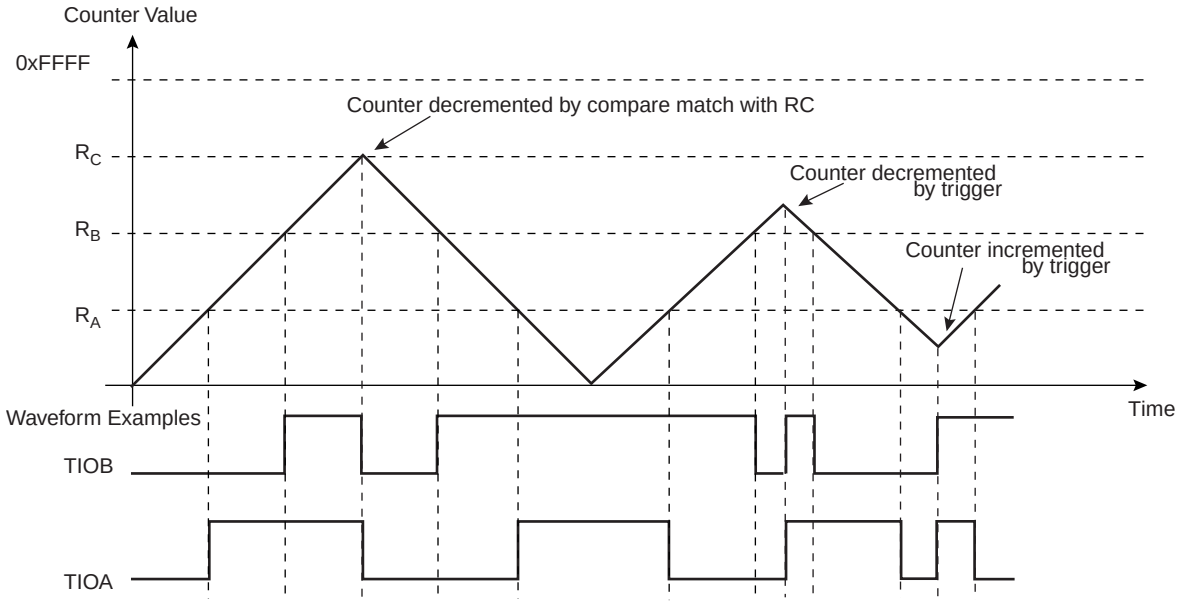


Figure 36-14. WAVSEL = 11 With Trigger



36.5.3.6 External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The EEVT parameter in TC_CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in TC_CMR.

As in Capture Mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

36.5.3.7 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

36.6 Timer/Counter (TC) User Interface

36.6.1 Global Register Mapping

Table 36-3. Timer/Counter (TC) Global Register Map

Offset	Channel/Register	Name	Access	Reset Value
0x00	TC Channel 0	See Table 36-4		
0x40	TC Channel 1	See Table 36-4		
0x80	TC Channel 2	See Table 36-4		
0xC0	TC Block Control Register	TC_BCR	Write-only	–
0xC4	TC Block Mode Register	TC_BMR	Read/Write	0

TC_BCR (Block Control Register) and TC_BMR (Block Mode Register) control the whole TC block. TC channels are controlled by the registers listed in Table 36-4. The offset of each of the channel registers in Table 36-4 is in relation to the offset of the corresponding channel as mentioned in Table 36-4.

36.6.2 Channel Memory Mapping

Table 36-4. TC Channel Memory Map

Offset	Register	Name	Access	Reset Value
0x00	Channel Control Register	TC_CCR	Write-only	–
0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x08	Reserved			–
0x0C	Reserved			–
0x10	Counter Value	TC_CV	Read-only	0
0x14	Register A	TC_RA	Read/Write[1]	0
0x18	Register B	TC_RB	Read/Write[1]	0
0x1C	Register C	TC_RC	Read/Write	0
0x20	Status Register	TC_SR	Read-only	0
0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xFC	Reserved	–	–	–

Note: 1. Read-only if WAVE = 0

36.6.3 TC Block Control Register

Name: TC_BCR

Access: Write-only

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

- **SYNC: Synchro Command**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

36.6.4 TC Block Mode Register

Name: TC_BMR

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TC2XC2S		TCXC1S		TC0XC0S	

- **TC0XC0S: External Clock Signal 0 Selection**

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

36.6.5 TC Channel Control Register

Name: TC_CCR

Access: Write-only

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

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

1 = A software trigger is performed: the counter is reset and the clock is started.

36.6.6 TC Channel Mode Register: Capture Mode

Name: TC_CMR

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 = 0	CPCTRG	–	–	–	ABETRG	ETRGEDG	
7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **LDBSTOP: Counter Clock Stopped with RB Loading**

0 = Counter clock is not stopped when RB loading occurs.

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

0 = Counter clock is not disabled when RB loading occurs.

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

ETRGEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

- **WAVE**

0 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

LDRA		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

- **LDRB: RB Loading Selection**

LDRB		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

36.6.7 TC Channel Mode Register: Waveform Mode

Name: TC_CMR

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 = 1	WAVSEL		ENETRG	EEVT		EEVTEDG	
7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **CPCSTOP: Counter Clock Stopped with RC Compare**

0 = Counter clock is not stopped when counter reaches RC.

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

0 = Counter clock is not disabled when counter reaches RC.

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

EEVTEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **EEVT: External Event Selection**

EEVT		Signal selected as external event	TIOB Direction
0	0	TIOB	input[(1)]
0	1	XC0	output
1	0	XC1	output
1	1	XC2	output

Note: 1. *If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms and subsequently no IRQs*.

- **ENETRG: External Event Trigger Enable**

0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.

1 = The external event resets the counter and starts the counter clock.

- **WAVSEL: Waveform Selection**

WAVSEL		Effect
0	0	UP mode without automatic trigger on RC Compare
1	0	UP mode with automatic trigger on RC Compare
0	1	UPDOWN mode without automatic trigger on RC Compare
1	1	UPDOWN mode with automatic trigger on RC Compare

- **WAVE = 1**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

ACPA		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **ACPC: RC Compare Effect on TIOA**

ACPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **AEEVT: External Event Effect on TIOA**

AEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **ASWTRG: Software Trigger Effect on TIOA**

ASWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPB: RB Compare Effect on TIOB**

BCPB		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPC: RC Compare Effect on TIOB**

BCPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BEEVT: External Event Effect on TIOB**

BEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BSWTRG: Software Trigger Effect on TIOB**

BSWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

36.6.8 TC Counter Value Register

Name: TC_CV

Access: Read-only

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

- **CV: Counter Value**

CV contains the counter value in real time.

36.6.9 TC Register A

Name: TC_RA

Access: Read-only if WAVE = 0, Read/Write if WAVE = 1

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

- **RA: Register A**

RA contains the Register A value in real time.

36.6.10 TC Register B

Name: TC_RB

Access: Read-only if WAVE = 0, Read/Write if WAVE = 1

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

- **RB: Register B**

RB contains the Register B value in real time.

36.6.11 TC Register C

Name: TC_RC

Access: Read/Write

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

- **RC: Register C**

RC contains the Register C value in real time.

36.6.12 TC Status Register

Name: TC_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow Status**

0 = No counter overflow has occurred since the last read of the Status Register.

1 = A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status**

0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.

- **CPAS: RA Compare Status**

0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPBS: RB Compare Status**

0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPCS: RC Compare Status**

0 = RC Compare has not occurred since the last read of the Status Register.

1 = RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status**

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- **LDRBS: RB Loading Status**

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- **ETRGS: External Trigger Status**

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

36.6.13 TC Interrupt Enable Register

Name: TC_IER

Access: Write-only

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

- **COVFS: Counter Overflow**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.

36.6.14 TC Interrupt Disable Register

Name: TC_IDR

Access: Write-only

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

- **COVFS: Counter Overflow**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

36.6.15 TC Interrupt Mask Register

Name: TC_IMR

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.

37. Pulse WIdth Modulation Controller (PWM)

37.1 Overview

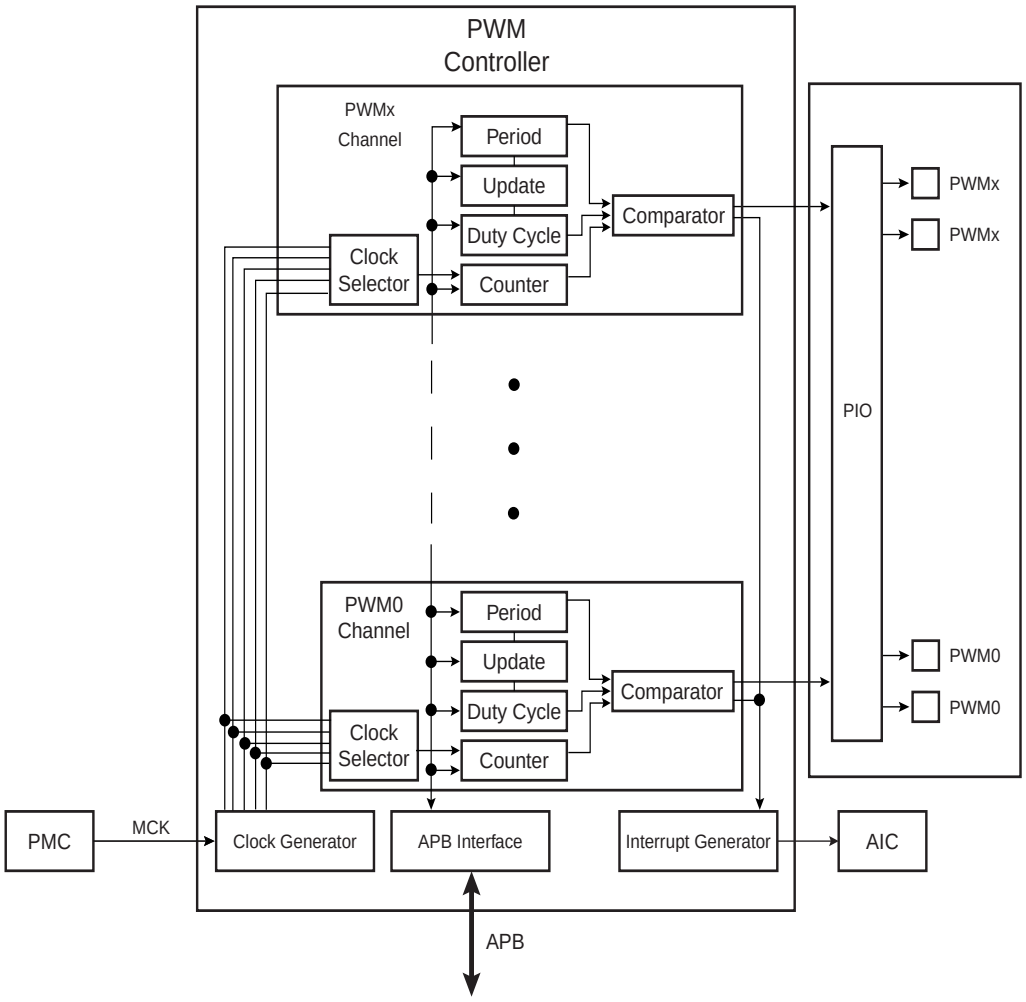
The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM macrocell master clock.

All PWM macrocell accesses are made through APB mapped registers.

Channels can be synchronized, to generate non overlapped waveforms. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period or the duty-cycle.

37.2 Block Diagram

Figure 37-1. Pulse Width Modulation Controller Block Diagram



37.3 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 37-1. I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

37.4 Product Dependencies

37.4.1 I/O Lines

The pins used for interfacing the PWM may be multiplexed with PIO lines. The programmer must first program the PIO controller to assign the desired PWM pins to their peripheral function. If I/O lines of the PWM are not used by the application, they can be used for other purposes by the PIO controller.

All of the PWM outputs may or may not be enabled. If an application requires only four channels, then only four PIO lines will be assigned to PWM outputs.

37.4.2 Power Management

The PWM is not continuously clocked. The programmer must first enable the PWM clock in the Power Management Controller (PMC) before using the PWM. However, if the application does not require PWM operations, the PWM clock can be stopped when not needed and be restarted later. In this case, the PWM will resume its operations where it left off.

Configuring the PWM does not require the PWM clock to be enabled.

37.4.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the PWM interrupt requires the AIC to be programmed first. Note that it is not recommended to use the PWM interrupt line in edge sensitive mode.

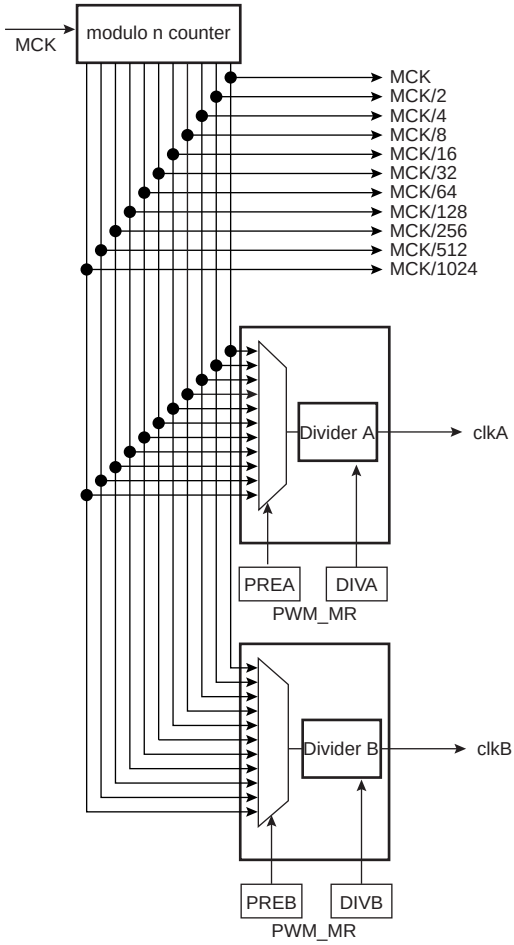
37.5 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

- Clocked by the system clock, MCK, the clock generator module provides 13 clocks.
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

37.5.1 PWM Clock Generator

Figure 37-2. Functional View of the Clock Generator Block Diagram



Caution: Before using the PWM macrocell, the programmer must first enable the PWM clock in the Power Management Controller (PMC).

The PWM macrocell master clock, MCK, is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.

The clock generator is divided in three blocks:

- a modulo n counter which provides 11 clocks: F_{MCK}, $F_{MCK}/2$, $F_{MCK}/4$, $F_{MCK}/8$, $F_{MCK}/16$, $F_{MCK}/32$, $F_{MCK}/64$, $F_{MCK}/128$, $F_{MCK}/256$, $F_{MCK}/512$, $F_{MCK}/1024$

– two linear dividers (1, 1/2, 1/3, ... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Mode register (PWM_MR). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value in the PWM Mode register (PWM_MR).

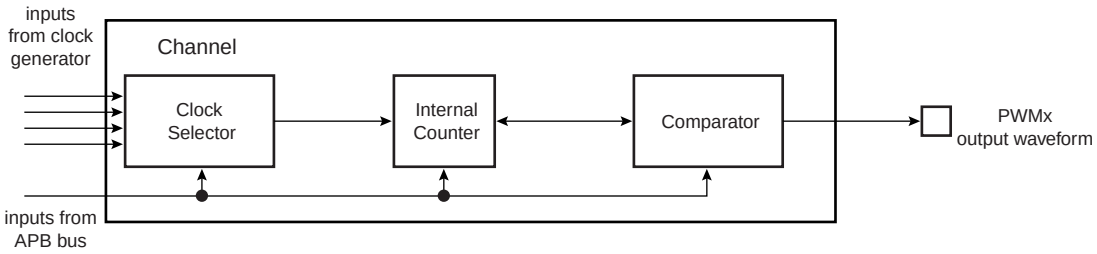
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

At reset, all clocks provided by the modulo n counter are turned off except clock “clk”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

37.5.2 PWM Channel

37.5.2.1 Block Diagram

Figure 37-3. Functional View of the Channel Block Diagram



Each of the 4 channels is composed of three blocks:

- A clock selector which selects one of the clocks provided by the clock generator described in Section 37.5.1 “PWM Clock Generator” on page 537.
- An internal counter clocked by the output of the clock selector. This internal counter is incremented or decremented according to the channel configuration and comparators events. The size of the internal counter is 16 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

37.5.2.2 Waveform Properties

The different properties of output waveforms are:

- the ***internal clock selection***. The internal channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the PWM_CMRx register. This field is reset at 0.
- the ***waveform period***. This channel parameter is defined in the CPRD field of the PWM_CPRDx register.
  - If the waveform is left aligned, then the output waveform period depends on the counter source clock and can be calculated:
  By using the Master Clock (MCK) divided by an X given prescaler value
  (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula

will be:

$$\frac{(X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
By using the Master Clock (MCK) divided by an X given prescaler value
(with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

- the ***waveform duty cycle***. This channel parameter is defined in the CDTY field of the PWM_CDTYx register.
  If the waveform is left aligned then:

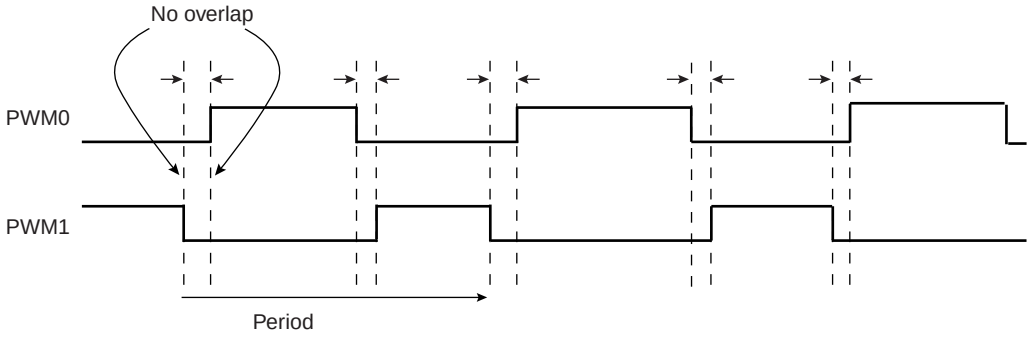
$$\text{duty cycle} = \frac{(period - 1 / \text{fchannel\_x\_clock} \times CDTY)}{period}$$

  If the waveform is center aligned, then:

$$\text{duty cycle} = \frac{((period / 2) - 1 / \text{fchannel\_x\_clock} \times CDTY))}{(period / 2)}$$

- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL field of the PWM_CMRx register. By default the signal starts by a low level.
- the ***waveform alignment***. The output waveform can be left or center aligned. Center aligned waveforms can be used to generate non overlapped waveforms. This property is defined in the CALG field of the PWM_CMRx register. The default mode is left aligned.

Figure 37-4. Non Overlapped Center Aligned Waveforms



Note: 1. See Figure 37-5 on page 541 for a detailed description of center aligned waveforms.

When center aligned, the internal channel counter increases up to CPRD and.decreases down to 0. This ends the period.

When left aligned, the internal channel counter increases up to CPRD and is reset. This ends the period.

Thus, for the same CPRD value, the period for a center aligned channel is twice the period for a left aligned channel.

Waveforms are fixed at 0 when:

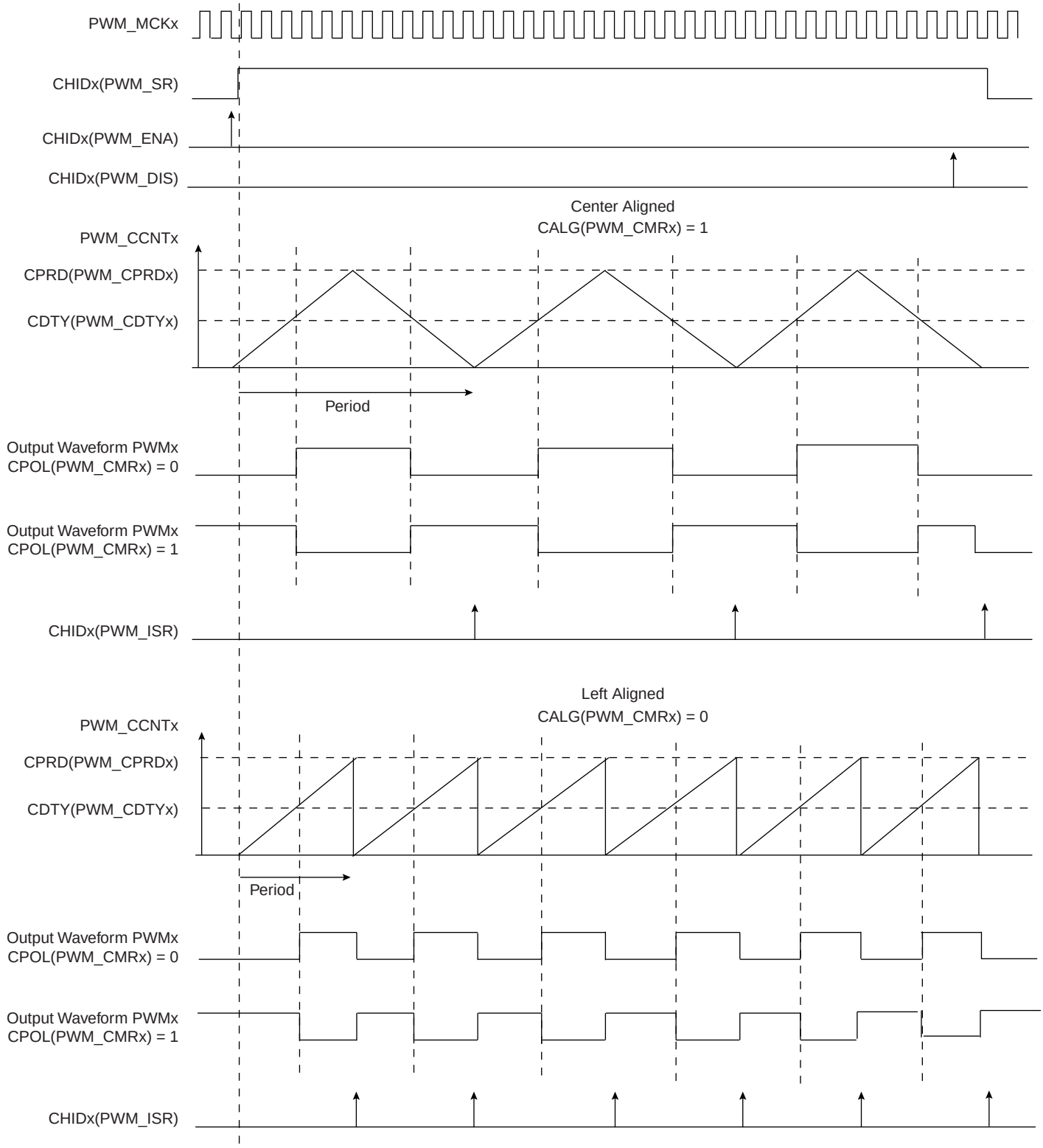
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

Waveforms are fixed at 1 (once the channel is enabled) when:

- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1

The waveform polarity must be set before enabling the channel. This immediately affects the channel output level. Changes on channel polarity are not taken into account while the channel is enabled.

Figure 37-5. Waveform Properties



37.5.3 PWM Controller Operations

37.5.3.1 Initialization

Before enabling the output channel, this channel must have been configured by the software application:

- Configuration of the clock generator if DIVA and DIVB are required
- Selection of the clock for each channel (CPRE field in the PWM_CMRx register)
- Configuration of the waveform alignment for each channel (CALG field in the PWM_CMRx register)
- Configuration of the period for each channel (CPRD in the PWM_CPRDx register). Writing in PWM_CPRDx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CUPDx Register to update PWM_CPRDx as explained below.
- Configuration of the duty cycle for each channel (CDTY in the PWM_CDTYx register). Writing in PWM_CDTYx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CUPDx Register to update PWM_CDTYx as explained below.
- Configuration of the output waveform polarity for each channel (CPOL in the PWM_CMRx register)
- Enable Interrupts (Writing CHIDx in the PWM_IER register)
- Enable the PWM channel (Writing CHIDx in the PWM_ENA register)

It is possible to synchronize different channels by enabling them at the same time by means of writing simultaneously several CHIDx bits in the PWM_ENA register.

- In such a situation, all channels may have the same clock selector configuration and the same period specified.

37.5.3.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the Period Register (PWM_CPRDx) and the Duty Cycle Register (PWM_CDTYx) can help the user in choosing. The event number written in the Period Register gives the PWM accuracy. The Duty Cycle quantum cannot be lower than *1/PWM_CPRDx* value. The higher the value of PWM_CPRDx, the greater the PWM accuracy.

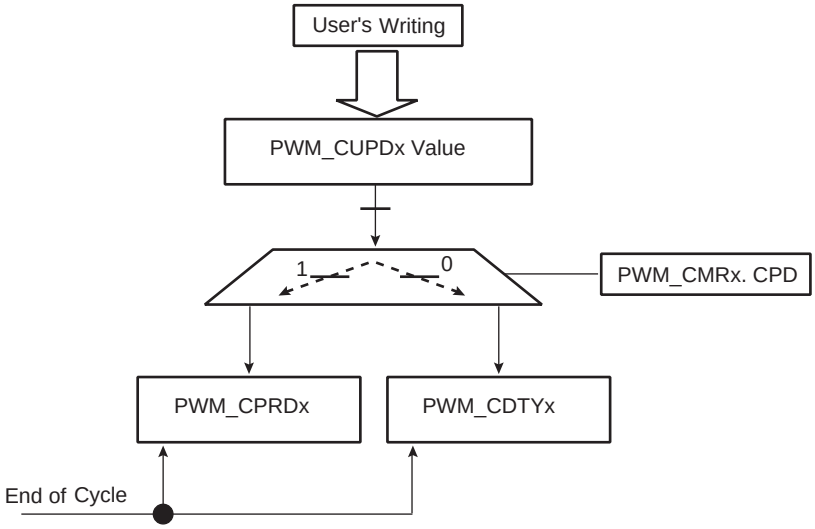
For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value between 1 up to 14 in PWM_CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

37.5.3.3 Changing the Duty Cycle or the Period

It is possible to modulate the output waveform duty cycle or period.

To prevent unexpected output waveform, the user must use the update register (PWM_CUPDx) to change waveform parameters while the channel is still enabled. The user can write a new period value or duty cycle value in the update register (PWM_CUPDx). This register holds the new value until the end of the current cycle and updates the value for the next cycle. Depending on the CPD field in the PWM_CMRx register, PWM_CUPDx either updates PWM_CPRDx or PWM_CDTYx. Note that even if the update register is used, the period must not be smaller than the duty cycle.

Figure 37-6. Synchronized Period or Duty Cycle Update



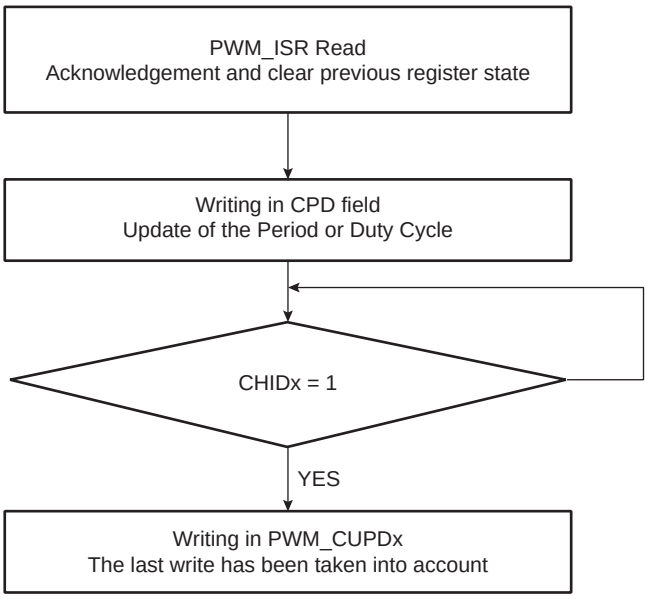
To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 37-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

Figure 37-7. Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

37.5.3.4 Interrupts

Depending on the interrupt mask in the PWM_IMR register, an interrupt is generated at the end of the corresponding channel period. The interrupt remains active until a read operation in the PWM_ISR register occurs.

A channel interrupt is enabled by setting the corresponding bit in the PWM_IER register. A channel interrupt is disabled by setting the corresponding bit in the PWM_IDR register.

37.6 Pulse Width Modulation (PWM) Controller User Interface

Table 37-2. PWM Controller Registers

Offset	Register	Name	Access	Peripheral Reset Value
0x00	PWM Mode Register	PWM_MR	Read/Write	0
0x04	PWM Enable Register	PWM_ENA	Write-only	-
0x08	PWM Disable Register	PWM_DIS	Write-only	-
0x0C	PWM Status Register	PWM_SR	Read-only	0
0x10	PWM Interrupt Enable Register	PWM_IER	Write-only	-
0x14	PWM Interrupt Disable Register	PWM_IDR	Write-only	-
0x18	PWM Interrupt Mask Register	PWM_IMR	Read-only	0
0x1C	PWM Interrupt Status Register	PWM_ISR	Read-only	0
0x4C - 0xFC	Reserved	–	–	–
0x100 - 0x1FC	Reserved			
0x200	Channel 0 Mode Register	PWM_CMR0	Read/Write	0x0
0x204	Channel 0 Duty Cycle Register	PWM_CDTY0	Read/Write	0x0
0x208	Channel 0 Period Register	PWM_CPRD0	Read/Write	0x0
0x20C	Channel 0 Counter Register	PWM_CCNT0	Read-only	0x0
0x210	Channel 0 Update Register	PWM_CUPD0	Write-only	-
...	Reserved			
0x220	Channel 1 Mode Register	PWM_CMR1	Read/Write	0x0
0x224	Channel 1 Duty Cycle Register	PWM_CDTY1	Read/Write	0x0
0x228	Channel 1 Period Register	PWM_CPRD1	Read/Write	0x0
0x22C	Channel 1 Counter Register	PWM_CCNT1	Read-only	0x0
0x230	Channel 1 Update Register	PWM_CUPD1	Write-only	-
...	...	...	...	...

37.6.1 PWM Mode Register

Name: PWM_MR

Access: Read/Write

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

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

DIVA, DIVB	CLKA, CLKB
0	CLKA, CLKB clock is turned off
1	CLKA, CLKB clock is clock selected by PREA, PREB
2-255	CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor.

- **PREA, PREB**

PREA, PREB				Divider Input Clock
0	0	0	0	MCK.
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
Other				Reserved

37.6.2 PWM Enable Register

Name: PWM_ENA

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Enable PWM output for channel x.

37.6.3 PWM Disable Register

Name: PWM_DIS

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Disable PWM output for channel x.

37.6.4 PWM Status Register

Name: PWM_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

37.6.5 PWM Interrupt Enable Register

Name: PWM_IER

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Enable interrupt for PWM channel x.

37.6.6 PWM Interrupt Disable Register

Name: PWM_IDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Disable interrupt for PWM channel x.

37.6.7 PWM Interrupt Mask Register

Name: PWM_IMR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

37.6.8 PWM Interrupt Status Register

Name: PWM_ISR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No new channel period has been achieved since the last read of the PWM_ISR register.

1 = At least one new channel period has been achieved since the last read of the PWM_ISR register.

Note: Reading PWM_ISR automatically clears CHIDx flags.

37.6.9 PWM Channel Mode Register

Name: PWM_CMRx

Access: Read/Write

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

- **CPRE: Channel Pre-scaler**

CPRE				Channel Pre-scaler
0	0	0	0	MCK
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
1	0	1	1	CLKA
1	1	0	0	CLKB
Other				Reserved

- **CALG: Channel Alignment**

0 = The period is left aligned.

1 = The period is center aligned.

- **CPOL: Channel Polarity**

0 = The output waveform starts at a low level.

1 = The output waveform starts at a high level.

- **CPD: Channel Update Period**

0 = Writing to the PWM_CUPDx will modify the duty cycle at the next period start event.

1 = Writing to the PWM_CUPDx will modify the period at the next period start event.

37.6.10 PWM Channel Duty Cycle Register

Name: PWM_CDTYx

Access: Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CDTY: Channel Duty Cycle**

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (PWM_CPRx).

37.6.11 PWM Channel Period Register

Name: PWM_CPRDx

Access: Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

37.6.12 PWM Channel Counter Register

Name: PWM_CCNTx

Access: Read-only

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

- **CNT: Channel Counter Register**

Internal counter value. This register is reset when:

the channel is enabled (writing CHIDx in the PWM_ENA register).

the counter reaches CPRD value defined in the PWM_CPRDx register if the waveform is left aligned.

37.6.13 PWM Channel Update Register

Name: PWM_CUPDx

Access: Write-only

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

This register acts as a double buffer for the period or the duty cycle. This prevents an unexpected waveform when modifying the waveform period or duty-cycle.

Only the first 16 bits (internal channel counter size) are significant.

CPD (PWM_CMRx Register)	
0	The duty-cycle (CDTC in the PWM_CDRx register) is updated with the CUPD value at the beginning of the next period.
1	The period (CPRD in the PWM_CPRx register) is updated with the CUPD value at the beginning of the next period.

38. USB Device Port (UDP)

38.1 Overview

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1M bytes/s) by working with endpoints with two banks of DPR.

Table 38-1. USB Endpoint Description

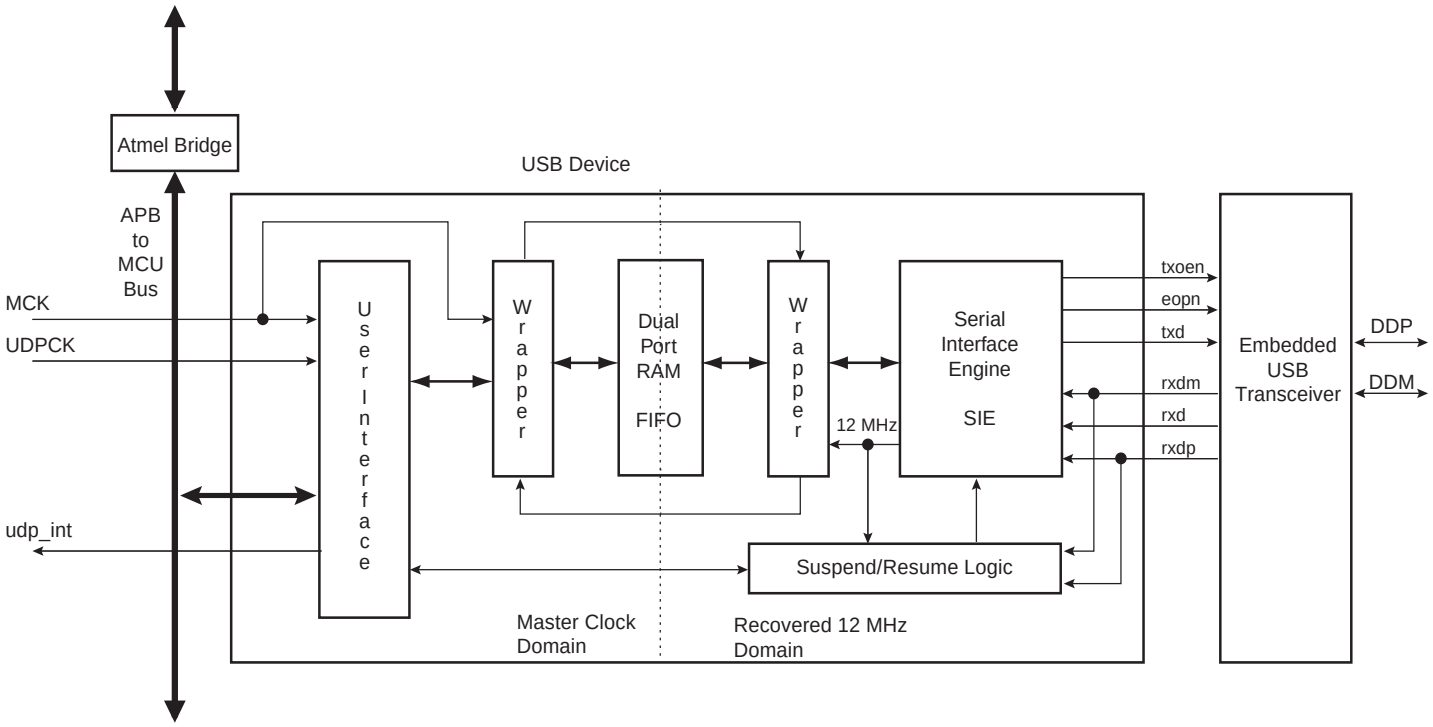
Endpoint Number	Mnemonic	Dual-Bank[1]	Max. Endpoint Size	Endpoint Type
0	EP0	No	8	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
2	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	64	Bulk/Iso/Interrupt
5	EP5	Yes	64	Bulk/Iso/Interrupt
6	EP6	Yes	64	Bulk/Iso/Interrupt
7	EP7	Yes	64	Bulk/Iso/Interrupt

Note: 1. The Dual-Bank function provides two banks for an endpoint. This feature is used for ping-pong mode.

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wake up to the USB host controller.

38.2 Block Diagram

Figure 38-1. Block Diagram



Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the Master Clock domain (MCK) and a 48 MHz clock (UDPCK) used by the 12 MHz domain.

A USB 2.0 full-speed pad is embedded and controlled by the Serial Interface Engine (SIE).

The signal external_resume is optional. It allows the UDP peripheral to wake up once in system mode. The host is then notified that the device asks for a resume. This optional feature must also be negotiated with the host during the enumeration.

38.2.1 Signal Description

Table 38-2. Signal Names

Signal Name	Description	Type
UDPCK	48 MHz clock	input
MCK	Master clock	input
udp_int	Interrupt line connected to the Advanced Interrupt Controller (AIC)	input
DDP	USB D+ line	I/O
DDM	USB D- line	I/O

38.3 Product Dependencies

For further details on the USB Device hardware implementation, see the specific Product Properties document.

The USB physical transceiver is integrated into the product. The bidirectional differential signals DDP and DDM are available from the product boundary.

38.3.1 I/O Lines

DDP and DDM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

38.3.2 Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL with an accuracy of ± 0.25%.

Thus, the USB device receives two clocks from the Power Management Controller (PMC): the master clock, MCK, used to drive the peripheral user interface, and the UDPCK, used to interface with the bus USB signals (recovered 12 MHz domain).

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

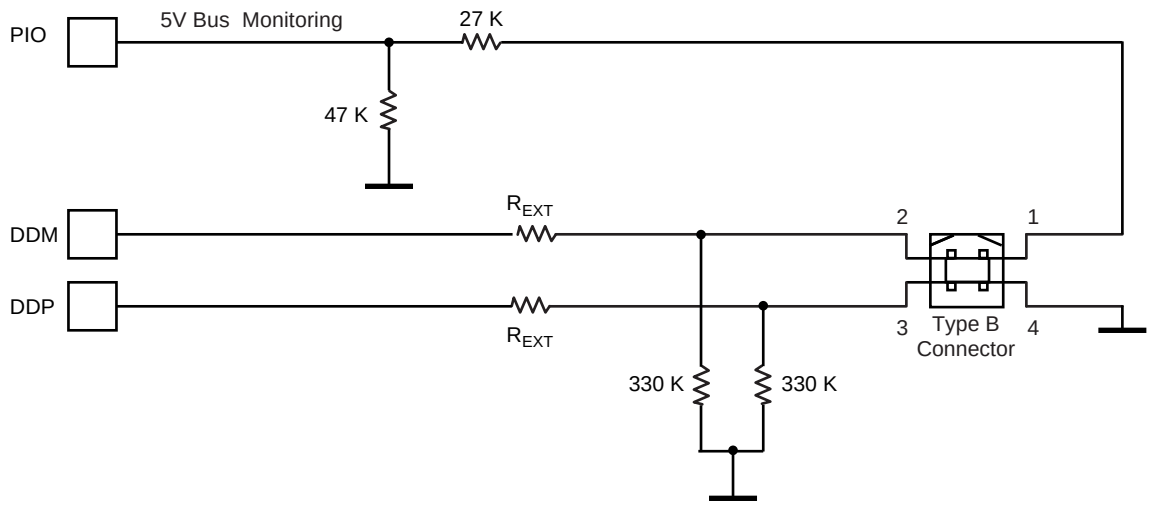
38.3.3 Interrupt

The USB device interface has an interrupt line connected to the Interrupt Controller.

Handling the USB device interrupt requires programming the Interrupt Controller before configuring the UDP.

38.4 Typical Connection

Figure 38-2. Board Schematic to Interface Device Peripheral



38.4.1 USB Device Transceiver

The USB device transceiver is embedded in the product. A few discrete components are required as follows:

- the application detects all device states as defined in chapter 9 of the USB specification;
  - VBUS monitoring
- to reduce power consumption the host is disconnected
- for line termination.

38.4.2 VBUS Monitoring

VBUS monitoring is required to detect host connection. VBUS monitoring is done using a standard PIO with internal pullup disabled. When the host is switched off, it should be considered as a disconnect, the pullup must be disabled in order to prevent powering the host through the pullup resistor.

When the host is disconnected and the transceiver is enabled, then DDP and DDM are floating. This may lead to over consumption. A solution is to connect 330 KΩ pulldowns on DDP and DDM. These pulldowns do not alter DDP and DDM signal integrity.

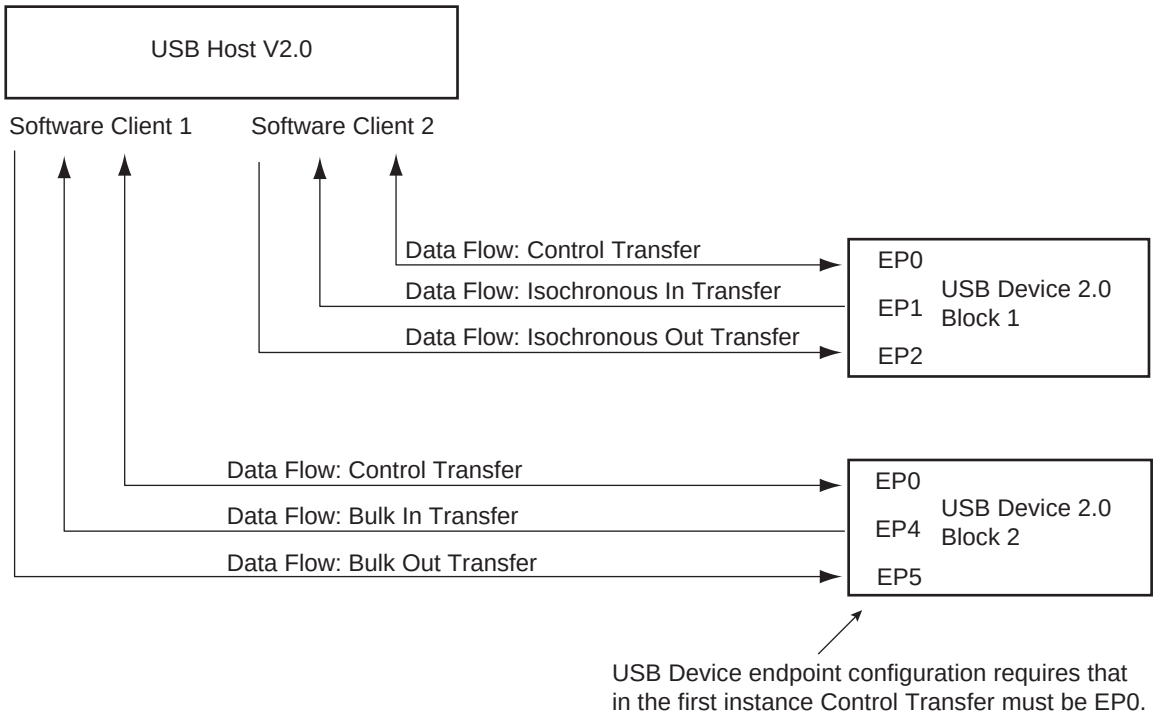
A termination serial resistor must be connected to DDP and DDM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

38.5 Functional Description

38.5.1 USB V2.0 Full-speed Introduction

The USB V2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with a USB device through a set of communication flows.

Figure 38-3. Example of USB V2.0 Full-speed Communication Control



The Control Transfer endpoint EP0 is always used when a USB device is first configured (USB v. 2.0 specifications).

38.5.1.1 USB V2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

Table 38-3. USB Communication Flow

Transfer	Direction	Bandwidth	Supported Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	64	Yes	No
Interrupt	Unidirectional	Not guaranteed	≤64	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8, 16, 32, 64	Yes	Yes

38.5.1.2 USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are three kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction

38.5.1.3 USB Transfer Event Definitions

As indicated below, transfers are sequential events carried out on the USB bus.

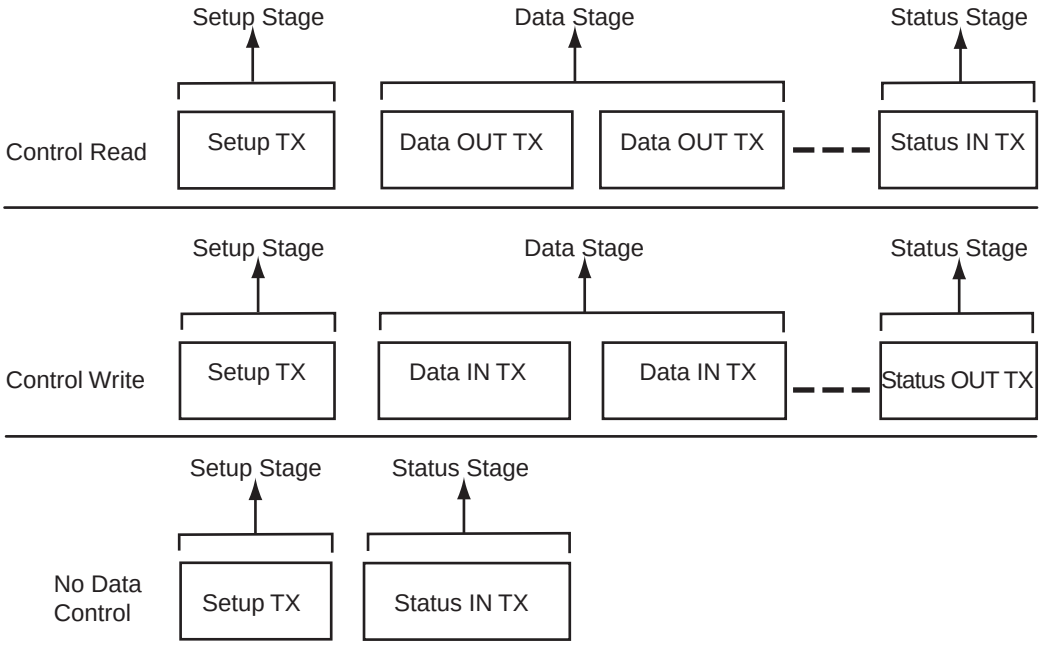
Table 38-4. USB Transfer Events

Control Transfers(1) (3)	• Setup transaction > Data IN transactions > Status OUT transaction • Setup transaction > Data OUT transactions > Status IN transaction • Setup transaction > Status IN transaction
Interrupt IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Interrupt OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction
Isochronous IN Transfer(2) (device toward host)	• Data IN transaction > Data IN transaction
Isochronous OUT Transfer(2) (host toward device)	• Data OUT transaction > Data OUT transaction
Bulk IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Bulk OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction

Notes:
1. Control transfer must use endpoints with no ping-pong attributes.
2. Isochronous transfers must use endpoints with ping-pong attributes.
3. Control transfers can be aborted using a stall handshake.

A status transaction is a special type of host-to-device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

Figure 38-4. Control Read and Write Sequences



Notes: 1. During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0,* for more information on the protocol layer.

2. During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

38.5.2 Handling Transactions with USB V2.0 Device Peripheral

38.5.2.1 Setup Transaction

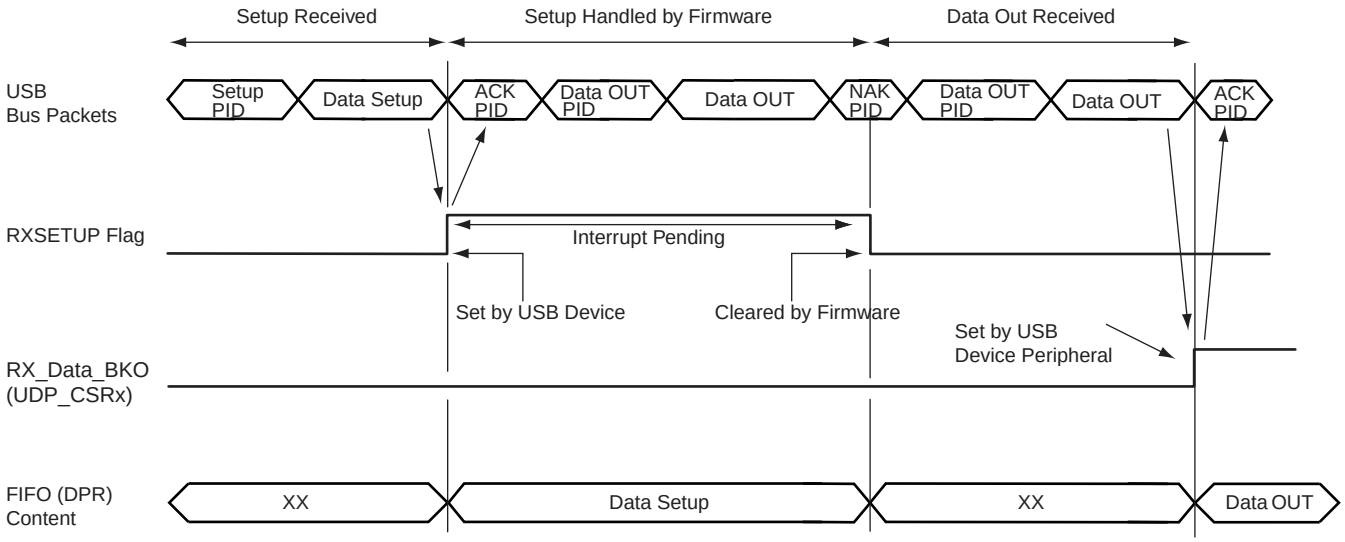
Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet
- RXSETUP is set in the UDP_CSRx register
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the UDP_CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

Figure 38-5. Setup Transaction Followed by a Data OUT Transaction



38.5.2.2 *Data IN Transaction*

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

38.5.2.3 *Using Endpoints Without Ping-pong Attributes*

To perform a Data IN transaction using a non ping-pong endpoint:

1. The application checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's UDP_CSRx register (TXPKTRDY must be cleared).
2. The application writes the first packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx register,
3. The application notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
4. The application is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx register has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.
5. The microcontroller writes the second packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx register,
6. The microcontroller notifies the USB peripheral it has finished by setting the TXPK-TRDY in the endpoint's UDP_CSRx register.
7. The application clears the TXCOMP in the endpoint's UDP_CSRx.

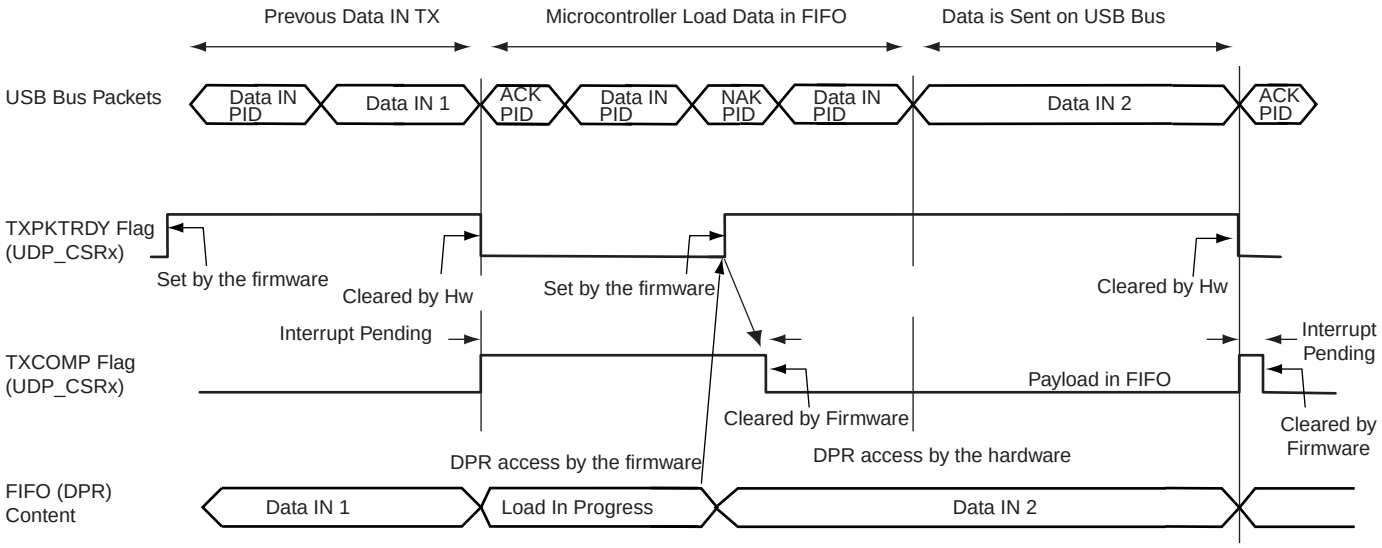
After the last packet has been sent, the application must clear TXCOMP once this has been set.

TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

Note: Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the Data IN protocol layer.

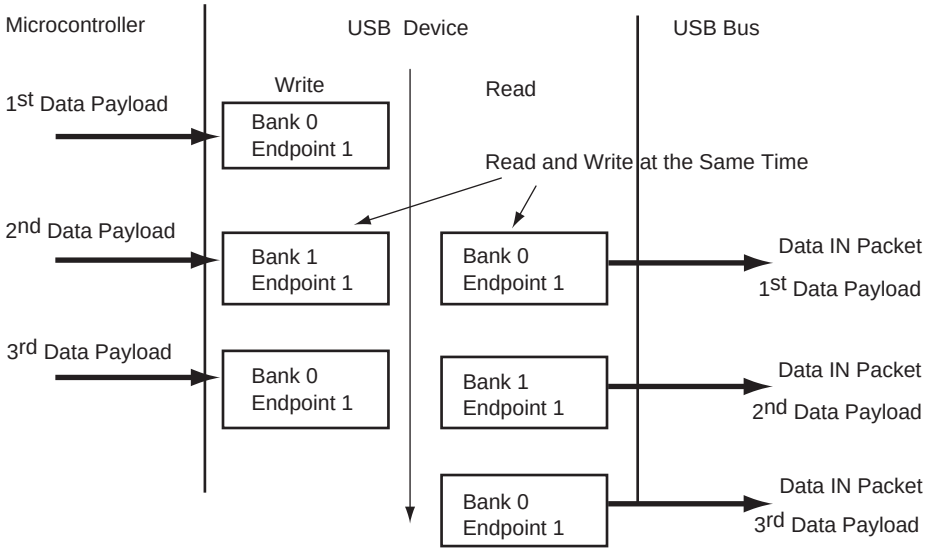
Figure 38-6. Data IN Transfer for Non Ping-pong Endpoint



38.5.2.4 Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. This also allows handling the maximum bandwidth defined in the USB specification during bulk transfer. To be able to guarantee a constant or the maximum bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

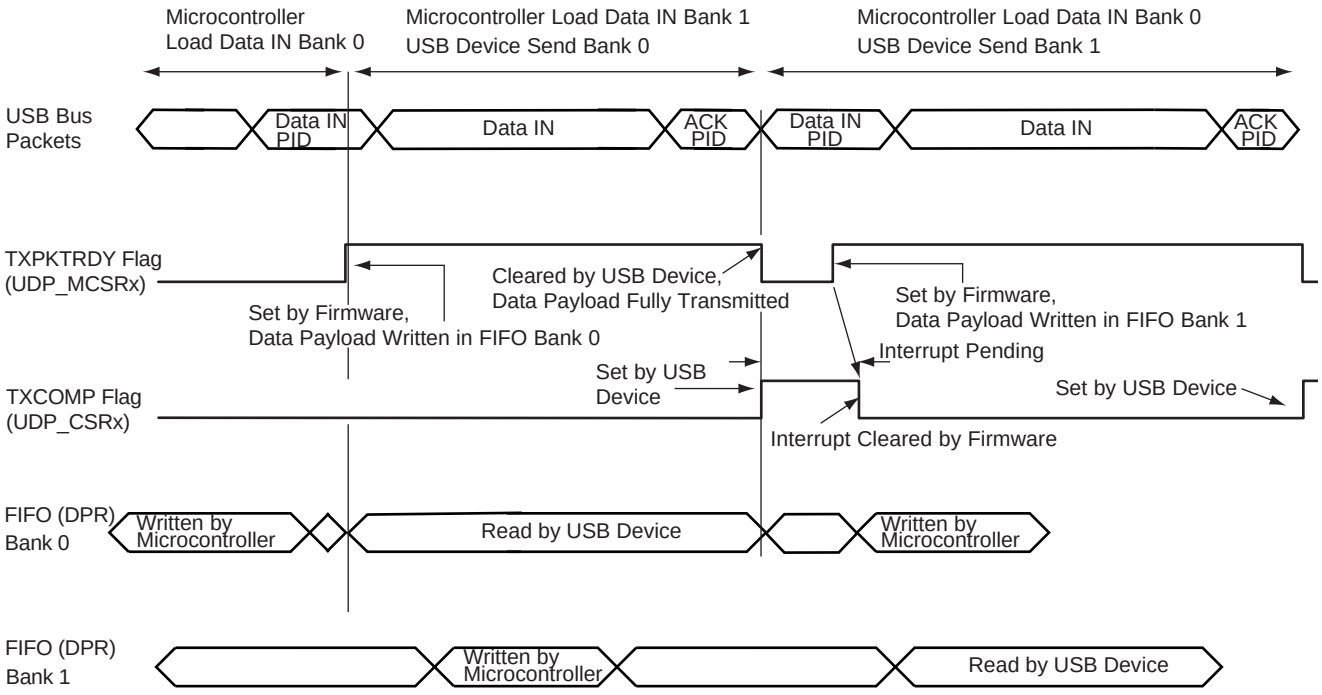
Figure 38-7. Bank Swapping Data IN Transfer for Ping-pong Endpoints



When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's UDP_CSRx register.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's UDP_FDRx register.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1), writing zero or more byte values in the endpoint's UDP_FDRx register.
5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx register is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent, raising TXPKTRDY in the endpoint's UDP_CSRx register.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

Figure 38-8. Data IN Transfer for Ping-pong Endpoint



Warning: There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX_COMP to set TX_PKTRDY. If the delay between receiving TX_COMP is set and TX_PKTRDY is set too long, some Data IN packets may be NACKed, reducing the bandwidth.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

38.5.2.5 Data OUT Transaction

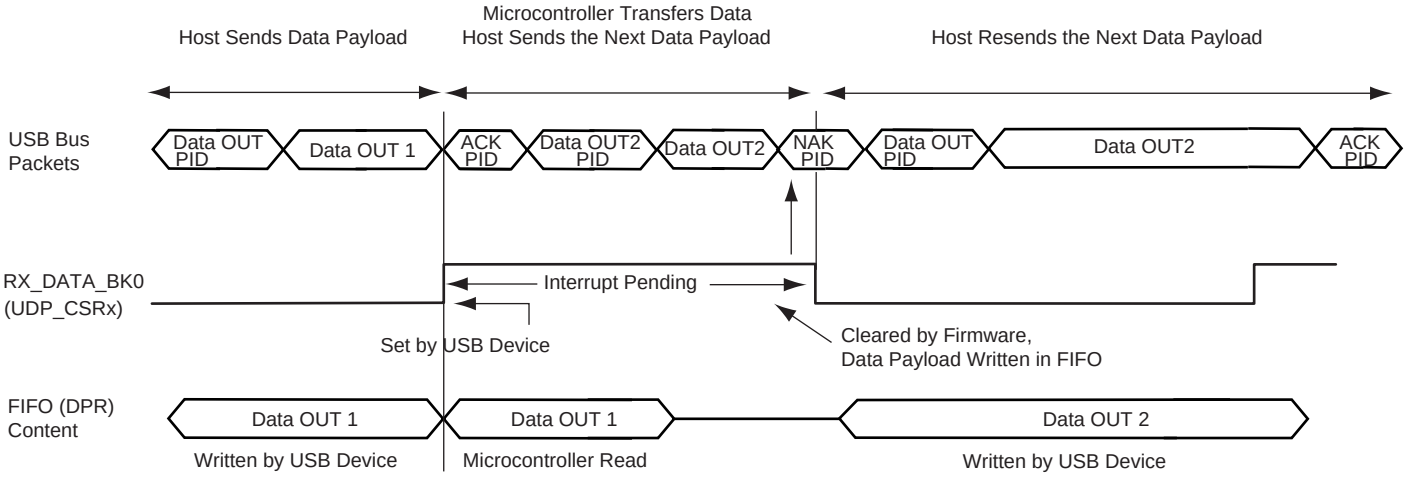
Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

38.5.2.6 Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data is written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX_DATA_BK0 in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx register.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's UDP_FDRx register.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx register.
7. A new Data OUT packet can be accepted by the USB device.

Figure 38-9. Data OUT Transfer for Non Ping-pong Endpoints

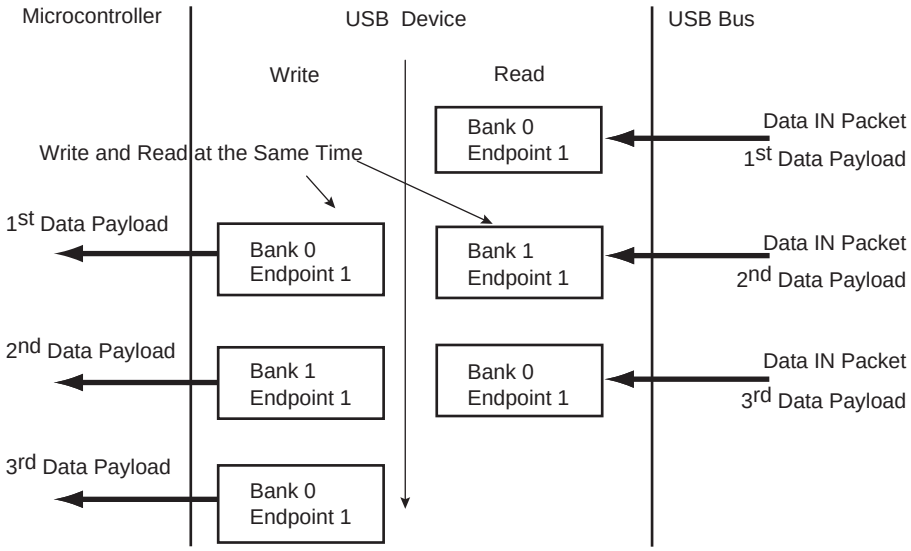


An interrupt is pending while the flag RX_DATA_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory can not be done after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

38.5.2.7 *Using Endpoints With Ping-pong Attributes*

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 38-10. Bank Swapping in Data OUT Transfers for Ping-pong Endpoints

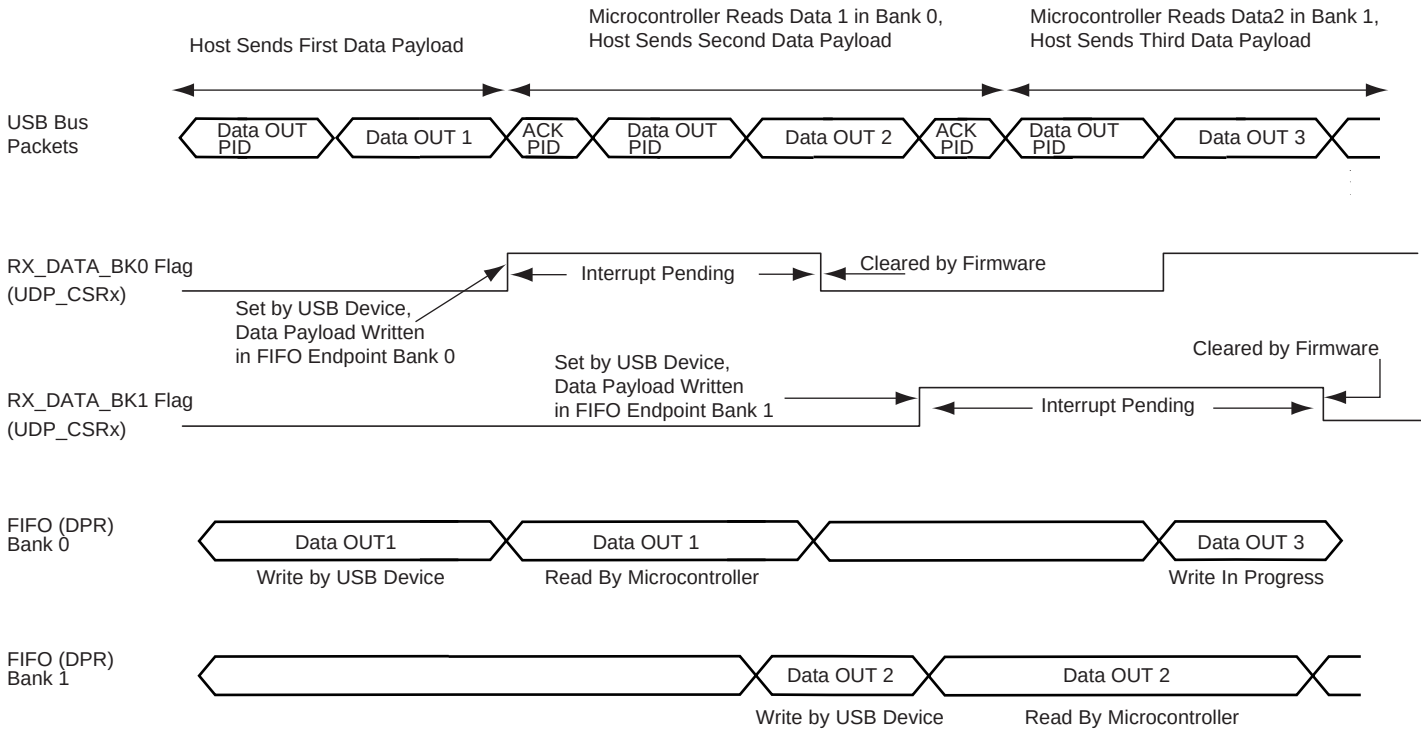


When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx register.
6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's UDP_FDRx register.
7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx register.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX_DATA_BK1 set in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK1 is set.

10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's UDP_FDRx register.
11. The microcontroller notifies the USB device it has finished the transfer by clearing RX_DATA_BK1 in the endpoint's UDP_CSRx register.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 0.

Figure 38-11. Data OUT Transfer for Ping-pong Endpoint



Note: An interrupt is pending while the RX_DATA_BK0 or RX_DATA_BK1 flag is set.

Warning: When RX_DATA_BK0 and RX_DATA_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX_DATA_BK0 then RX_DATA_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

38.5.2.8 Stall Handshake

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0.*)

- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

1. The microcontroller sets the FORCESTALL flag in the UDP_CSRx endpoint's register.
2. The host receives the stall packet.
3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

Figure 38-12. Stall Handshake (Data IN Transfer)

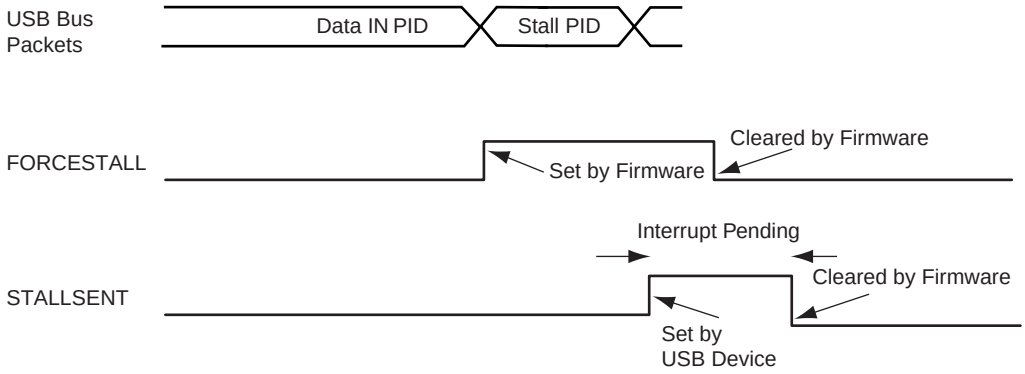
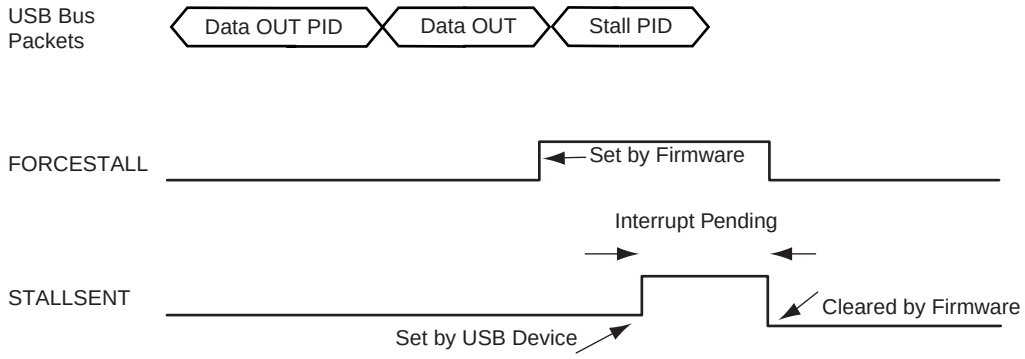


Figure 38-13. Stall Handshake (Data OUT Transfer)



38.5.2.9 Transmit Data Cancellation

Some endpoints have dual-banks whereas some endpoints have only one bank. The procedure to cancel transmission data held in these banks is described below.

To see the organization of dual-bank availability refer to Table 38-1 ”USB Endpoint Description”.

38.5.2.10 Endpoints **Without** Dual-Banks

There are two possibilities: In one case, TXPKTRDY field in UDP_CSR has already been set. In the other instance, TXPKTRDY is not set.

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See Section 38.6.9 ”UDP Reset Endpoint Register”.)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY so that no packet is ready to be sent
  - Reset the endpoint to clear the FIFO (pointers). (See Section 38.6.9 ”UDP Reset Endpoint Register”.)

38.5.2.11 Endpoints **With** Dual-Banks

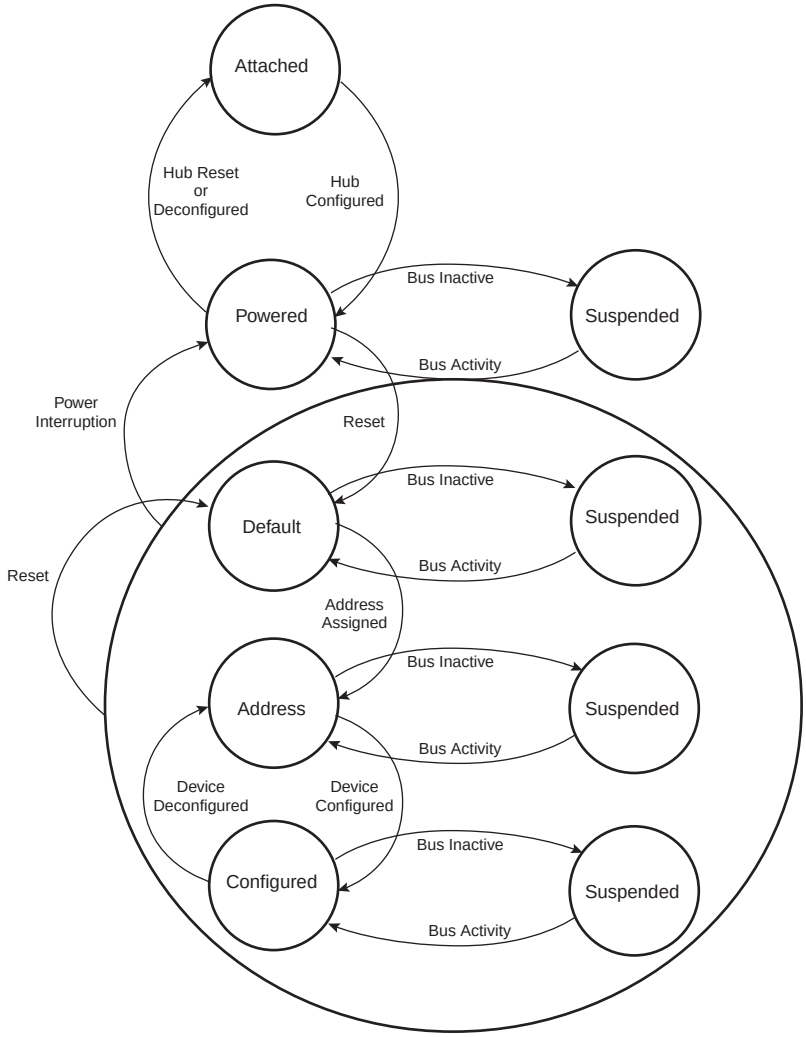
There are two possibilities: In one case, TXPKTRDY field in UDP_CSR has already been set. In the other instance, TXPKTRDY is not set.

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See Section 38.6.9 ”UDP Reset Endpoint Register”.)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY and read it back until actually read at 0.
  - Set TXPKTRDY and read it back until actually read at 1.
  - Clear TXPKTRDY so that no packet is ready to be sent.
  - Reset the endpoint to clear the FIFO (pointers). (See Section 38.6.9 ”UDP Reset Endpoint Register”.)

38.5.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 38-14. USB Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices may not consume more than 500 µA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake up feature is not mandatory for all devices and must be negotiated with the host.

38.5.3.1 Not Powered State

Self powered devices can detect 5V VBUS using a PIO as described in the typical connection section. When the device is not connected to a host, device power consumption can be reduced by disabling MCK for the UDP, disabling UDPCK and disabling the transceiver. DDP and DDM lines are pulled down by 330 KΩ resistors.

38.5.3.2 Entering Attached State

When no device is connected, the USB DDP and DDM signals are tied to GND by 15 KΩ pull-down resistors integrated in the hub downstream ports. When a device is attached to a hub downstream port, the device connects a 1.5 KΩ pull-up resistor on DDP. The USB bus line goes into IDLE state, DDP is pulled up by the device 1.5 KΩ resistor to 3.3V and DDM is pulled down by the 15 KΩ resistor of the host.

To enable integrated pull-up, the PUON bit in the UDP_TXVC register must be set.

Warning: To write to the UDP_TXVC register, MCK clock must be enabled on the UDP. This is done in the Power Management Controller.

After pullup connection, the device enters the powered state. In this state, the UDPCK and MCK must be enabled in the Power Management Controller. The transceiver can remain disabled.

38.5.3.3 From Powered State to Default State

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmaskable flag ENDBUSRES is set in the register UDP_ISR and an interrupt is triggered.

Once the ENDBUSRES interrupt has been triggered, the device enters Default State. In this state, the UDP software must:

- Enable the default endpoint, setting the EPEDS flag in the UDP_CSR[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the UDP_IER register. The enumeration then begins by a control transfer.
- Configure the interrupt mask register which has been reset by the USB reset detection
- Enable the transceiver clearing the TXVDIS flag in the UDP_TXVC register.

In this state UDPCK and MCK must be enabled.

Warning: Each time an ENDBUSRES interrupt is triggered, the Interrupt Mask Register and UDP_CSR registers have been reset.

38.5.3.4 From Default State to Address State

After a set address standard device request, the USB host peripheral enters the address state.

Warning: Before the device enters in address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP_CSR[0] register has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the UDP_GLB_STAT register, sets its new address, and sets the FEN bit in the UDP_FADDR register.

38.5.3.5 From Address State to Configured State

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP_CSRx registers and, optionally, enabling corresponding interrupts in the UDP_IER register.

38.5.3.6 Entering in Suspend State

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP_ISR register is set. This triggers an interrupt if the corresponding bit is set in the UDP_IMR register.This flag is cleared by writing to the UDP_ICR register. Then the device enters Suspend Mode.

In this state bus powered devices must drain less than 500uA from the 5V VBUS. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The USB device peripheral clocks can be switched off. Resume event is asynchronously detected. MCK and UDPCK can be switched off in the Power Management controller and the USB transceiver can be disabled by setting the TXVDIS field in the UDP_TXVC register.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. Switching off MCK for the UDP peripheral must be one of the last operations after writing to the UDP_TXVC and acknowledging the RXSUSP.

38.5.3.7 Receiving a Host Resume

In suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks are disabled (however the pullup shall not be removed).

Once the resume is detected on the bus, the WAKEUP signal in the UDP_ISR is set. It may generate an interrupt if the corresponding bit in the UDP_IMR register is set. This interrupt may be used to wake up the core, enable PLL and main oscillators and configure clocks.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. MCK for the UDP must be enabled before clearing the WAKEUP bit in the UDP_ICR register and clearing TXVDIS in the UDP_TXVC register.

38.5.3.8 Sending a Device Remote Wakeup

In Suspend state it is possible to wake up the host sending an external resume.

- The device must wait at least 5 ms after being entered in suspend before sending an external resume.
- The device has 10 ms from the moment it starts to drain current and it forces a K state to resume the host.
- The device must force a K state from 1 to 15 ms to resume the host

Before sending a K state to the host, MCK, UDPCK and the transceiver must be enabled. Then to enable the remote wakeup feature, the RMWUPE bit in the UDP_GLB_STAT register must be enabled. To force the K state on the line, a transition of the ESR bit from 0 to 1 has to be done in the UDP_GLB_STAT register. This transition must be accomplished by first writing a 0 in the ESR bit and then writing a 1.

The K state is automatically generated and released according to the USB 2.0 specification.

38.6 USB Device Port (UDP) User Interface

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers, including the UDP_TXVC register.

Table 38-5. Register Mapping

Offset	Register	Name	Access	Reset
0x000	Frame Number Register	UDP_FRM_NUM	Read-only	0x0000_0000
0x004	Global State Register	UDP_GLB_STAT	Read-write	0x0000_0010
0x008	Function Address Register	UDP_FADDR	Read-write	0x0000_0100
0x00C	Reserved	–	–	–
0x010	Interrupt Enable Register	UDP_IER	Write-only	
0x014	Interrupt Disable Register	UDP_IDR	Write-only	
0x018	Interrupt Mask Register	UDP_IMR	Read-only	0x0000_1200
0x01C	Interrupt Status Register	UDP_ISR	Read-only	–(1)
0x020	Interrupt Clear Register	UDP_ICR	Write-only	
0x024	Reserved	–	–	–
0x028	Reset Endpoint Register	UDP_RST_EP	Read-write	0x0000_0000
0x02C	Reserved	–	–	–
0x030 + 0x4 * (ept_num - 1)	Endpoint Control and Status Register	UDP_CSR	Read-write	0x0000_0000
0x050 + 0x4 * (ept_num - 1)	Endpoint FIFO Data Register	UDP_FDR	Read-write	0x0000_0000
0x070	Reserved	–	–	–
0x074	Transceiver Control Register	UDP_TXVC(2)	Read-write	0x0000_0100
0x078 - 0xFC	Reserved	–	–	–

Notes: 1. Reset values are not defined for UDP_ISR.
2. See Warning above the "Register Mapping" on this page.

38.6.1 UDP Frame Number Register

Name: UDP_FRM_NUM

Access: Read-only

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

- **FRM_NUM[10:0]: Frame Number as Defined in the Packet Field Formats**

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame.

Value Updated at the SOF_EOP (Start of Frame End of Packet).

- **FRM_ERR: Frame Error**

This bit is set at SOF_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF_PID.

- **FRM_OK: Frame OK**

This bit is set at SOF_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM_OK is bit 4 of FRM_NUM_H and FRM_ERR is bit 3 of FRM_NUM_L.

38.6.2 UDP Global State Register

Name: UDP_GLB_STAT

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
–	–	–	RMWUPE	RSMINPR	ESR	CONFG	FADDEN

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

- **FADDEN: Function Address Enable**

Read:

0 = Device is not in address state.

1 = Device is in address state.

Write:

0 = No effect, only a reset can bring back a device to the default state.

1 = Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_FADDR register must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **CONFG: Configured**

Read:

0 = Device is not in configured state.

1 = Device is in configured state.

Write:

0 = Sets device in a non configured state

1 = Sets device in configured state.

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **ESR: Enable Send Resume**

0 = Mandatory value prior to starting any Remote Wake Up procedure.

1 = Starts the Remote Wake Up procedure if this bit value was 0 and if RMWUPE is enabled.

- **RMWUPE: Remote Wake Up Enable**

0 = The Remote Wake Up feature of the device is disabled.

1 = The Remote Wake Up feature of the device is enabled.

38.6.3 UDP Function Address Register

Name: UDP_FADDR

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
–	–	–	–	–	–	–	FEN
7	6	5	4	3	2	1	0
–	FADD						

- **FADD[6:0]: Function Address Value**

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information. After power up or reset, the function address value is set to 0.

- **FEN: Function Enable**

Read:

0 = Function endpoint disabled.

1 = Function endpoint enabled.

Write:

0 = Disables function endpoint.

1 = Default value.

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

38.6.4 UDP Interrupt Enable Register

Name: UDP_IER

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
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Enable Endpoint 0 Interrupt**
- **EP1INT: Enable Endpoint 1 Interrupt**
- **EP2INT: Enable Endpoint 2Interrupt**
- **EP3INT: Enable Endpoint 3 Interrupt**
- **EP4INT: Enable Endpoint 4 Interrupt**
- **EP5INT: Enable Endpoint 5 Interrupt**
- **EP6INT: Enable Endpoint 6 Interrupt**
- **EP7INT: Enable Endpoint 7 Interrupt**

0 = No effect.

1 = Enables corresponding Endpoint Interrupt.

- **RXSUSP: Enable UDP Suspend Interrupt**

0 = No effect.

1 = Enables UDP Suspend Interrupt.

- **RXRSM: Enable UDP Resume Interrupt**

0 = No effect.

1 = Enables UDP Resume Interrupt.

- **SOFINT: Enable Start Of Frame Interrupt**

0 = No effect.

1 = Enables Start Of Frame Interrupt.

- **WAKEUP: Enable UDP bus Wakeup Interrupt**

0 = No effect.

1 = Enables USB bus Interrupt.

38.6.5 UDP Interrupt Disable Register

Name: UDP_IDR

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
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Disable Endpoint 0 Interrupt**
- **EP1INT: Disable Endpoint 1 Interrupt**
- **EP2INT: Disable Endpoint 2 Interrupt**
- **EP3INT: Disable Endpoint 3 Interrupt**
- **EP4INT: Disable Endpoint 4 Interrupt**
- **EP5INT: Disable Endpoint 5 Interrupt**
- **EP6INT: Disable Endpoint 6 Interrupt**
- **EP7INT: Disable Endpoint 7 Interrupt**

0 = No effect.

1 = Disables corresponding Endpoint Interrupt.

- **RXSUSP: Disable UDP Suspend Interrupt**

0 = No effect.

1 = Disables UDP Suspend Interrupt.

- **RXRSM: Disable UDP Resume Interrupt**

0 = No effect.

1 = Disables UDP Resume Interrupt.

- **SOFINT: Disable Start Of Frame Interrupt**

0 = No effect.

1 = Disables Start Of Frame Interrupt

- **WAKEUP: Disable USB Bus Interrupt**

0 = No effect.

1 = Disables USB Bus Wakeup Interrupt.

38.6.6 UDP Interrupt Mask Register

Name: UDP_IMR

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
–	–	WAKEUP	BIT12	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Mask Endpoint 0 Interrupt**
- **EP1INT: Mask Endpoint 1 Interrupt**
- **EP2INT: Mask Endpoint 2 Interrupt**
- **EP3INT: Mask Endpoint 3 Interrupt**
- **EP4INT: Mask Endpoint 4 Interrupt**
- **EP5INT: Mask Endpoint 5 Interrupt**
- **EP6INT: Mask Endpoint 6 Interrupt**
- **EP7INT: Mask Endpoint 7 Interrupt**

0 = Corresponding Endpoint Interrupt is disabled.

1 = Corresponding Endpoint Interrupt is enabled.

- **RXSUSP: Mask UDP Suspend Interrupt**

0 = UDP Suspend Interrupt is disabled.

1 = UDP Suspend Interrupt is enabled.

- **RXRSM: Mask UDP Resume Interrupt.**

0 = UDP Resume Interrupt is disabled.

1 = UDP Resume Interrupt is enabled.

- **SOFINT: Mask Start Of Frame Interrupt**

0 = Start of Frame Interrupt is disabled.

1 = Start of Frame Interrupt is enabled.

- **BIT12: UDP_IMR Bit 12**

Bit 12 of UDP_IMR cannot be masked and is always read at 1.

- **WAKEUP: USB Bus WAKEUP Interrupt**

0 = USB Bus Wakeup Interrupt is disabled.

1 = USB Bus Wakeup Interrupt is enabled.

Note: When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register UDP_IMR is enabled.

38.6.7 UDP Interrupt Status Register

Name: UDP_ISR

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
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Endpoint 0 Interrupt Status**
- **EP1INT: Endpoint 1 Interrupt Status**
- **EP2INT: Endpoint 2 Interrupt Status**
- **EP3INT: Endpoint 3 Interrupt Status**
- **EP4INT: Endpoint 4 Interrupt Status**
- **EP5INT: Endpoint 5 Interrupt Status**
- **EP6INT: Endpoint 6 Interrupt Status**
- **EP7INT: Endpoint 7Interrupt Status**

0 = No Endpoint0 Interrupt pending.

1 = Endpoint0 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_CSR0 bit.

- **RXSUSP: UDP Suspend Interrupt Status**

0 = No UDP Suspend Interrupt pending.

1 = UDP Suspend Interrupt has been raised.

The USB device sets this bit when it detects no activity for 3ms. The USB device enters Suspend mode.

- **RXRSM: UDP Resume Interrupt Status**

0 = No UDP Resume Interrupt pending.

1 =UDP Resume Interrupt has been raised.

The USB device sets this bit when a UDP resume signal is detected at its port.

After reset, the state of this bit is undefined, the application must clear this bit by setting the RXRSM flag in the UDP_ICR register.

- **SOFINT: Start of Frame Interrupt Status**

0 = No Start of Frame Interrupt pending.

1 = Start of Frame Interrupt has been raised.

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

- **ENDBUSRES: End of BUS Reset Interrupt Status**

0 = No End of Bus Reset Interrupt pending.

1 = End of Bus Reset Interrupt has been raised.

This interrupt is raised at the end of a UDP reset sequence. The USB device must prepare to receive requests on the end-point 0. The host starts the enumeration, then performs the configuration.

- **WAKEUP: UDP Resume Interrupt Status**

0 = No Wakeup Interrupt pending.

1 = A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

After reset the state of this bit is undefined, the application must clear this bit by setting the WAKEUP flag in the UDP_ICR register.

38.6.8 UDP Interrupt Clear Register

Name: UDP_ICR

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
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **RXSUSP: Clear UDP Suspend Interrupt**

0 = No effect.

1 = Clears UDP Suspend Interrupt.

- **RXRSM: Clear UDP Resume Interrupt**

0 = No effect.

1 = Clears UDP Resume Interrupt.

- **SOFINT: Clear Start Of Frame Interrupt**

0 = No effect.

1 = Clears Start Of Frame Interrupt.

- **ENDBUSRES: Clear End of Bus Reset Interrupt**

0 = No effect.

1 = Clears End of Bus Reset Interrupt.

- **WAKEUP: Clear Wakeup Interrupt**

0 = No effect.

1 = Clears Wakeup Interrupt.

38.6.9 UDP Reset Endpoint Register

Name: UDP_RST_EP

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
EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**
- **EP6: Reset Endpoint 6**
- **EP7: Reset Endpoint 7**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the register UDP_CSRx.It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_CSRx flags.

0 = No reset.

1 = Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_CSRx register.

Resetting the endpoint is a two-step operation:

1. Set the corresponding EPx field.
2. Clear the corresponding EPx field.

38.6.10 UDP Endpoint Control and Status Register

Name: UDP_CSRx [x = 0..Y]

Access: Read-write

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

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_BK1	FORCE STALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_BK0	TXCOMP

WARNING: Due to synchronization between MCK and UDPCK, the software application must wait for the end of the write operation before executing another write by polling the bits which must be set/cleared.

```
/// Bitmap for all status bits in CSR that are not effected by a value 1.
#define REG_NO_EFFECT_1_ALL   AT91C_UDP_RX_DATA_BK0\
                            | AT91C_UDP_RX_DATA_BK1\
                            | AT91C_UDP_STALLSENT\
                            | AT91C_UDP_RXSETUP\
                            | AT91C_UDP_TXCOMP

/// Sets the specified bit(s) in the UDP_CSR register.
/// \param endpoint The endpoint number of the CSR to process.
/// \param flags The bitmap to set to 1.
#define SET_CSR(endpoint, flags) \
   { \
       volatile unsigned int reg; \
       reg = AT91C_BASE_UDP->UDP_CSR[endpoint] ; \
       reg |= REG_NO_EFFECT_1_ALL; \
       reg |= (flags); \
       AT91C_BASE_UDP->UDP_CSR[endpoint] = reg; \
       while ( (AT91C_BASE_UDP->UDP_CSR[endpoint] & (flags)) != (flags)); \
   }

/// Clears the specified bit(s) in the UDP_CSR register.
/// \param endpoint The endpoint number of the CSR to process.
/// \param flags The bitmap to clear to 0.
#define CLEAR_CSR(endpoint, flags) \
   { \
       volatile unsigned int reg; \
       reg = AT91C_BASE_UDP->UDP_CSR[endpoint]; \
       reg |= REG_NO_EFFECT_1_ALL; \
```

```
    reg &= ~(flags); \
    AT91C_BASE_UDP->UDP_CSR[endpoint] = reg; \
    while ( (AT91C_BASE_UDP->UDP_CSR[endpoint] & (flags)) == (flags)); \
}
```

Note: In a preemptive environment, set or clear the flag and wait for a time of 1 UDPCK clock cycle and 1peripheral clock cycle. However, RX_DATA_BK0, TXPKTRDY, RX_DATA_BK1 require wait times of 3 UDPCK clock cycles and 5 peripheral clock cycles before accessing DPR.

- **TXCOMP: Generates an IN Packet with Data Previously Written in the DPR**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Clear the flag, clear the interrupt.

1 = No effect.

Read (Set by the USB peripheral):

0 = Data IN transaction has not been acknowledged by the Host.

1 = Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

- **RX_DATA_BK0: Receive Data Bank 0**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1 = To leave the read value unchanged.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 0.

1 = A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_FDRx register. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

- **RXSETUP: Received Setup**

This flag generates an interrupt while it is set to one.

Read:

0 = No setup packet available.

1 = A setup data packet has been sent by the host and is available in the FIFO.

Write:

0 = Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1 = No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_FDRx register to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

- **STALLSENT: Stall Sent (Control, Bulk Interrupt Endpoints)/ISOERROR (Isochronous Endpoints)**

This flag generates an interrupt while it is set to one.

STALLSENT: This ends a STALL handshake.

Read:

0 = The host has not acknowledged a STALL.

1 = Host has acknowledged the stall.

Write:

0 = Resets the STALLSENT flag, clears the interrupt.

1 = No effect.

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

ISOERROR: A CRC error has been detected in an isochronous transfer.

Read:

0 = No error in the previous isochronous transfer.

1 = CRC error has been detected, data available in the FIFO are corrupted.

Write:

0 = Resets the ISOERROR flag, clears the interrupt.

1 = No effect.

- **TXPKTRDY: Transmit Packet Ready**

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0 = There is no data to send.

1 = The data is waiting to be sent upon reception of token IN.

Write:

0 = Can be used in the procedure to cancel transmission data. (See, Section 38.5.2.9 “Transmit Data Cancellation” on page 569)

1 = A new data payload has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_FDRx register. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

- **FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)**

Read:

0 = Normal state.

1 = Stall state.

Write:

0 = Return to normal state.

1 = Send STALL to the host.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this bit indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: This bit notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

- **RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notifies USB device that data have been read in the FIFO’s Bank 1.

1 = To leave the read value unchanged.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 1.

1 = A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_FDRx register. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

- **DIR: Transfer Direction (only available for control endpoints)**

Read-write

0 = Allows Data OUT transactions in the control data stage.

1 = Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

- **EPTYPE[2:0]: Endpoint Type**

Read-Write

Value	Name	Description
000	CTRL	Control
001	ISO_OUT	Isochronous OUT
101	ISO_IN	Isochronous IN
010	BULK_OUT	Bulk OUT
110	BULK_IN	Bulk IN
011	INT_OUT	Interrupt OUT
111	INT_IN	Interrupt IN

- **DTGLE: Data Toggle**

Read-only

0 = Identifies DATA0 packet.

1 = Identifies DATA1 packet.

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

- **EPEDS: Endpoint Enable Disable**

Read:

0 = Endpoint disabled.

1 = Endpoint enabled.

Write:

0 = Disables endpoint.

1 = Enables endpoint.

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset, all endpoints are configured as control endpoints (zero).

- **RXBYTECNT[10:0]: Number of Bytes Available in the FIFO**

Read-only

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_FDRx register.

38.6.11 UDP FIFO Data Register

Name: UDP_FDRx [x = 0..Y]

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
FIFO_DATA							

- **FIFO_DATA[7:0]: FIFO Data Value**

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding UDP_CSRx register is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information.

38.6.12 UDP Transceiver Control Register

Name: UDP_TXVC

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
–	–	–	–	–	–	PUON	TXVDIS
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

- **TXVDIS: Transceiver Disable**

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS field.

To enable the transceiver, TXVDIS must be cleared.

- **PUON: Pullup On**

0: The 1.5KΩ integrated pullup on DDP is disconnected.

1: The 1.5 KΩ integrated pullup on DDP is connected.

NOTE: If the USB pullup is not connected on DDP, the user should not write in any UDP register other than the UDP_TXVC register. This is because if DDP and DDM are floating at 0, or pulled down, then SE0 is received by the device with the consequence of a USB Reset.

39. Analog-to-Digital Converter (ADC)

39.1 Overview

The ADC is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates an 8-to-1 analog multiplexer, making possible the analog-to-digital conversions of 8 analog lines. The conversions extend from 0V to ADVREF.

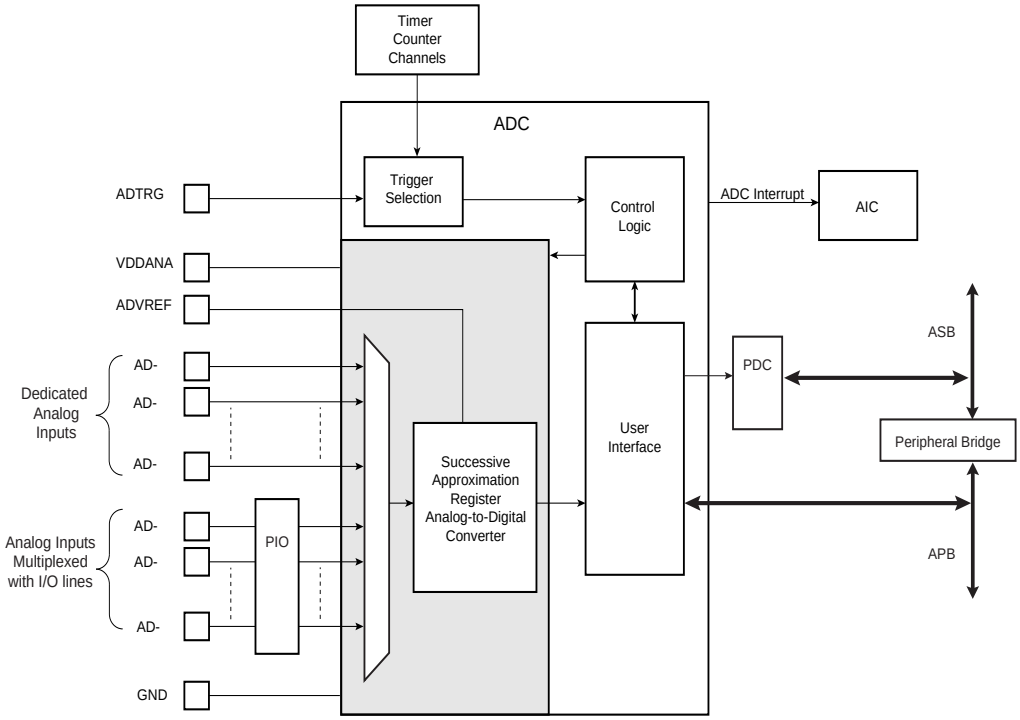
The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a PDC channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as Startup Time and Sample & Hold Time.

39.2 Block Diagram

Figure 39-1. Analog-to-Digital Converter Block Diagram



39.3 Signal Description

Table 39-1. ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
ADVREF	Reference voltage
AD0 - AD7	Analog input channels
ADTRG	External trigger

39.4 Product Dependencies

39.4.1 Power Management

The ADC is automatically clocked after the first conversion in Normal Mode. In Sleep Mode, the ADC clock is automatically stopped after each conversion. As the logic is small and the ADC cell can be put into Sleep Mode, the Power Management Controller has no effect on the ADC behavior.

39.4.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the ADC interrupt requires the AIC to be programmed first.

39.4.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing the register ADC_CHER. By default, after reset, the PIO line is configured as input with its pull-up enabled and the ADC input is connected to the GND.

39.4.4 I/O Lines

The pin ADTRG may be shared with other peripheral functions through the PIO Controller. In this case, the PIO Controller should be set accordingly to assign the pin ADTRG to the ADC function.

39.4.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be non-connected.

39.4.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the DC Characteristics section.

39.5 Functional Description

39.5.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires Sample and Hold Clock cycles as defined in the field SHTIM of the “ADC Mode Register” on page 606 and 10 ADC Clock cycles. The ADC Clock frequency is selected in the PRESCAL field of the Mode Register (ADC_MR).

The ADC clock range is between MCK/2, if PRESCAL is 0, and MCK/128, if PRESCAL is set to 63 (0x3F). PRESCAL must be programmed in order to provide an ADC clock frequency according to the parameters given in the Product definition section.

39.5.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage pin ADVREF. Analog inputs between these voltages convert to values based on a linear conversion.

39.5.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the bit LOWRES in the ADC Mode Register (ADC_MR). By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the bit LOWRES, the ADC switches in the lowest resolution and the conversion results can be read in the eight lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding ADC_CDR register and of the LDATA field in the ADC_LCDR register read 0.

Moreover, when a PDC channel is connected to the ADC, 10-bit resolution sets the transfer request sizes to 16-bit. Setting the bit LOWRES automatically switches to 8-bit data transfers. In this case, the destination buffers are optimized.

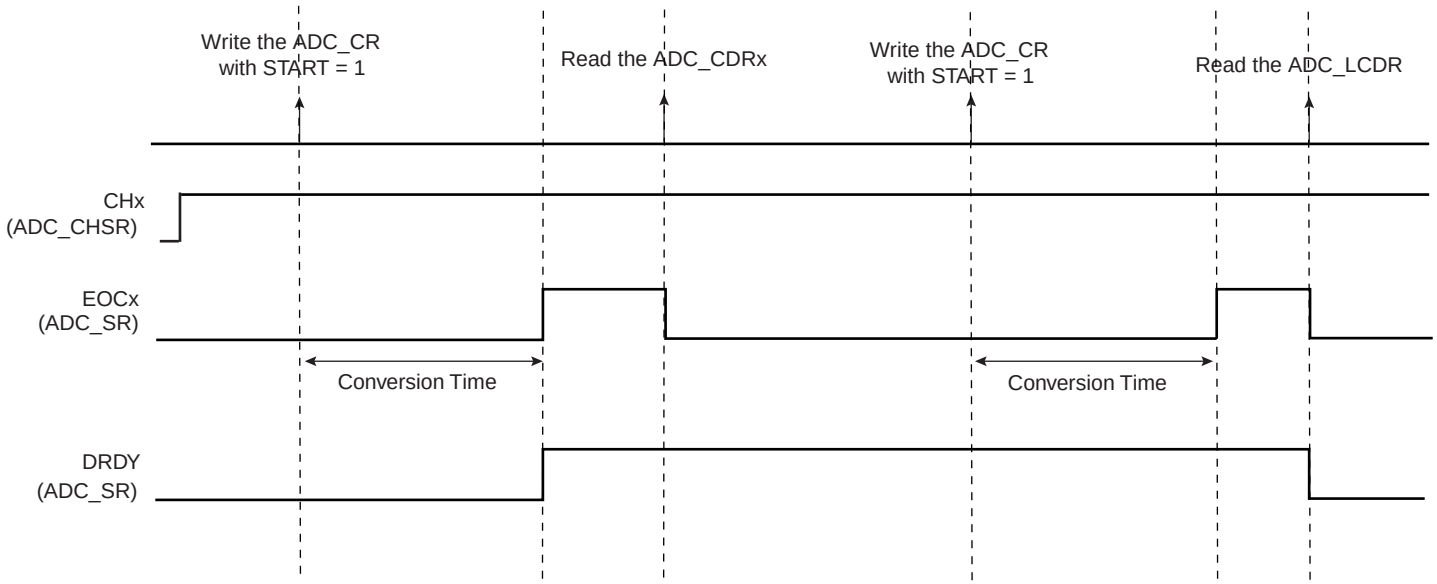
39.5.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the Channel Data Register (ADC_CDR) of the current channel and in the ADC Last Converted Data Register (ADC_LCDR).

The channel EOC bit in the Status Register (ADC_SR) is set and the DRDY is set. In the case of a connected PDC channel, DRDY rising triggers a data transfer request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDR registers clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit and the EOC bit corresponding to the last converted channel.

Figure 39-2. EOCx and DRDY Flag Behavior

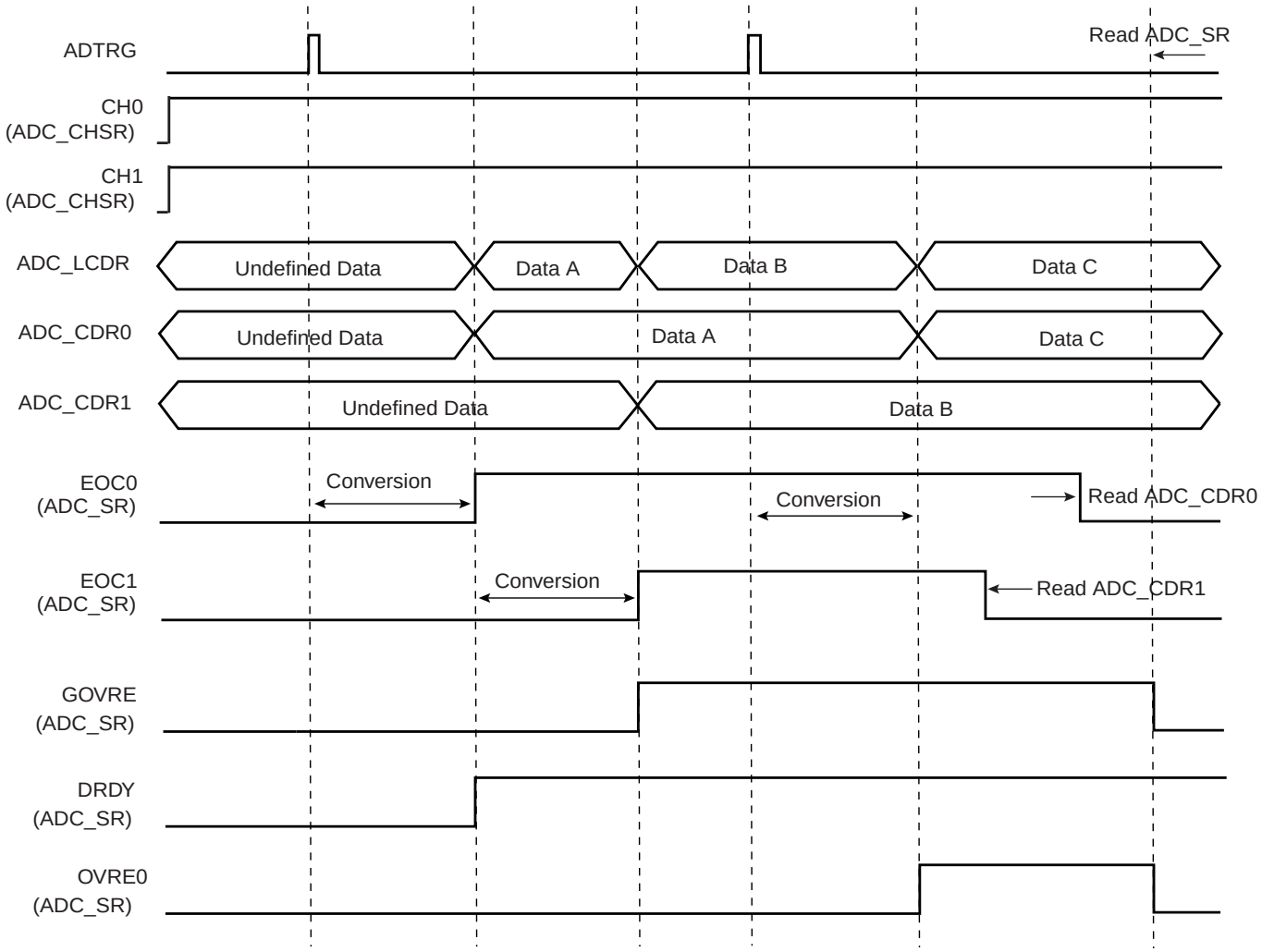


If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the Status Register (ADC_SR).

In the same way, new data converted when DRDY is high sets the bit GOVRE (General Overrun Error) in ADC_SR.

The OVRE and GOVRE flags are automatically cleared when ADC_SR is read.

Figure 39-3. GOVRE and OVREx Flag Behavior



Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

39.5.5 Conversion Triggers

Conversions of the active analog channels are started with a software or a hardware trigger. The software trigger is provided by writing the Control Register (ADC_CR) with the bit START at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, or the external trigger input of the ADC (ADTRG). The hardware trigger is selected with the field TRGSEL in the Mode Register (ADC_MR). The selected hardware trigger is enabled with the bit TRGEN in the Mode Register (ADC_MR).

If a hardware trigger is selected, the start of a conversion is detected at each rising edge of the selected signal. If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform Mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a PDC, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

Warning: Enabling hardware triggers does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can be initiated either by the hardware or the software trigger.

39.5.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by setting the bit SLEEP in the Mode Register ADC_MR.

The SLEEP mode is automatically managed by a conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using a Timer/Counter output. The periodic acquisition of several samples can be processed automatically without any intervention of the processor thanks to the PDC.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

39.5.7 ADC Timings

Each ADC has its own minimal Startup Time that is programmed through the field STARTUP in the Mode Register ADC_MR.

In the same way, a minimal Sample and Hold Time is necessary for the ADC to guarantee the best converted final value between two channels selection. This time has to be programmed through the SHTIM bitfield in the Mode Register ADC_MR.

Warning: No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the SHTIM field. See the section ADC Characteristics in the product datasheet.

39.6 Analog-to-digital Converter (ADC) User Interface

Table 39-2. ADC Register Mapping

Offset	Register	Name	Access	Reset State
0x00	Control Register	ADC_CR	Write-only	–
0x04	Mode Register	ADC_MR	Read/Write	0x00000000
0x08	Reserved	–	–	–
0x0C	Reserved	–	–	–
0x10	Channel Enable Register	ADC_CHER	Write-only	–
0x14	Channel Disable Register	ADC_CHDR	Write-only	–
0x18	Channel Status Register	ADC_CHSR	Read-only	0x00000000
0x1C	Status Register	ADC_SR	Read-only	0x000C0000
0x20	Last Converted Data Register	ADC_LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	ADC_IER	Write-only	–
0x28	Interrupt Disable Register	ADC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x30	Channel Data Register 0	ADC_CDR0	Read-only	0x00000000
0x34	Channel Data Register 1	ADC_CDR1	Read-only	0x00000000
...	...	...	...	...
0x4C	Channel Data Register 7	ADC_CDR7	Read-only	0x00000000
0x50 - 0xFC	Reserved	–	–	–

39.6.1 ADC Control Register

Name: ADC_CR

Access: Write-only

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

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

39.6.2 ADC Mode Register

Name: ADC_MR

Access: Read/Write

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

- **TRGEN: Trigger Enable**

TRGEN	Selected TRGEN
0	Hardware triggers are disabled. Starting a conversion is only possible by software.
1	Hardware trigger selected by TRGSEL field is enabled.

- **TRGSEL: Trigger Selection**

TRGSEL			Selected TRGSEL
0	0	0	TIOA Ouput of the Timer Counter Channel 0
0	0	1	TIOA Ouput of the Timer Counter Channel 1
0	1	0	TIOA Ouput of the Timer Counter Channel 2
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	External trigger
1	1	1	Reserved

- **LOWRES: Resolution**

LOWRES	Selected Resolution
0	10-bit resolution
1	8-bit resolution

- **SLEEP: Sleep Mode**

SLEEP	Selected Mode
0	Normal Mode
1	Sleep Mode

- **PRESCAL: Prescaler Rate Selection**
  ADCClock = MCK / ((PRESCAL+1) * 2)

- **STARTUP: Start Up Time**
  Startup Time = (STARTUP+1) * 8 / ADCClock

- **SHTIM: Sample & Hold Time**
  Sample & Hold Time = SHTIM / ADCClock

39.6.3 ADC Channel Enable Register

Name: ADC_CHER

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

39.6.4 ADC Channel Disable Register

Name: ADC_CHDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

39.6.5 ADC Channel Status Register

Name: ADC_CHSR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

39.6.6 ADC Status Register

Name: ADC_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished.

1 = Corresponding analog channel is enabled and conversion is complete.

- **OVREx: Overrun Error x**

0 = No overrun error on the corresponding channel since the last read of ADC_SR.

1 = There has been an overrun error on the corresponding channel since the last read of ADC_SR.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_SR.

1 = At least one General Overrun Error has occurred since the last read of ADC_SR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in ADC_RCR or ADC_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in ADC_RCR or ADC_RNCR.

- **RXBUFF: RX Buffer Full**

0 = ADC_RCR or ADC_RNCR have a value other than 0.

1 = Both ADC_RCR and ADC_RNCR have a value of 0.

39.6.7 ADC Last Converted Data Register

Name: ADC_LCDR

Access: Read-only

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

- **LDATA: Last Data Converted**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

39.6.8 ADC Interrupt Enable Register

Name: ADC_IER

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Enable x**
- **OVREx: Overrun Error Interrupt Enable x**
- **DRDY: Data Ready Interrupt Enable**
- **GOVRE: General Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

39.6.9 ADC Interrupt Disable Register

Name: ADC_IDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Disable x**
- **OVREx: Overrun Error Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

39.6.10 ADC Interrupt Mask Register

Name: ADC_IMR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Mask x**
- **OVREx: Overrun Error Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

39.6.11 ADC Channel Data Register

Name: ADC_CDRx

Access: Read-only

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

- **DATA: Converted Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. The Convert Data Register (CDR) is only loaded if the corresponding analog channel is enabled.

40. SAM7SE512/256/32 Electrical Characteristics

40.1 Absolute Maximum Ratings

Table 40-1. Absolute Maximum Ratings*

Operating Temperature (Industrial).....-40· C to + 85· C
Storage Temperature.....-60°C to + 150°C
Voltage on Input Pins with Respect to Ground.....-0.3V to + 5.5V
Maximum Operating Voltage (VDDCORE, and VDDPLL).....2.0V
Maximum Operating Voltage (VDDIO, VDDIN and VDDFLASH).....4.0V
Total DC Output Current on all I/O lines 128-lead LQFP/144-ball LFBGA.....200 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.

40.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified.

Table 40-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{VDDCORE}$	DC Supply Core		1.65		1.95	V
V_{VDDPLL}	DC Supply PLL		1.65		1.95	V
V_{VDDIO}	DC Supply I/Os	3.3V domain	3.0		3.6	V
V_{VDDIO}	DC Supply I/Os	1.8V domain	1.65		1.95	
$V_{VDDFLASH}$	DC Supply Flash		3.0		3.6	V
V_{IL}	Input Low-level Voltage	V_{VDDIO} from 3.0V to 3.6V	-0.3		0.8	V
		V_{VDDIO} from 1.65V to 1.95V	-0.3		0.3 x V_{VDDIO}	V
V_{IH}	Input High-level Voltage	V_{VDDIO} from 3.0V to 3.6V	2.0		V_{VDDIO} +0.3V	V
		V_{VDDIO} from 1.65V to 1.95V	0.7 x V_{VDDIO}		V_{VDDIO} +0.3V	V
V_{Hys}	Hysteresis Voltage	V_{VDDIO} from 3.0V to 3.6V	0.4		0.7	V
		V_{VDDIO} from 1.65V to 1.95V	0.3		0.6	V
V_{OL}	Output Low-level Voltage	I_O max, V_{VDDIO} from 3.0V to 3.6V			0.4	V
		I_O max, V_{VDDIO} from 1.65V to 1.95V			0.25 x V_{VDDIO}	
V_{OH}	Output High-level Voltage	I_O max, V_{VDDIO} from 3.0V to 3.6V	V_{VDDIO} -0.4			V
		I_O max, V_{VDDIO} from 1.65V to 1.95V	0.75 x V_{VDDIO}			
I_{LEAK}	Input Leakage Current	PA0-PA3, Pull-up resistors disabled (Typ: T_A = 25°C, Max: T_A = 85°C)		40	400	nA
		Other PIOs, Pull-up resistors disabled (Typ: T_A = 25°C, Max: T_A = 85°C)		20	200	nA
R_{PULLUP}	Pull-up Resistor	PA0-PA31, PB0-PB31,PC0-PC23, V_{VDDIO} from 3.0V to 3.6V	70	130	190	kΩ
		PA0-PA31, PB0-PB31,PC0-PC23, V_{VDDIO} from 1.65V to 1.95V	95	147	320	kΩ
$R_{PULLDOWN}$	Pull-down Resistor, (TST, ERASE, JTAGSEL)	V_{VDDIO} from 3.0V to 3.6V, Pins connected to V_{VDDIO}	8	15	28	kΩ
C_{IN}	Input Capacitance				14	pF

Table 40-2. DC Characteristics (Continued)

Symbol	Parameter	Conditions		Min	Typ	Max	Units
I_{SC}	Static Current (SAM7SE512/256)	On $V_{VDDCORE}$ = 1.85V, MCK = 500Hz	T_A = 25°C		12	60	µA
		All inputs driven at 1 (including TMS, TDI, TCK, NRST) Flash in standby mode All peripherals off	T_A = 85°C		40	300	
I_{SC}	Static Current (SAM7SE32)	On $V_{VDDCORE}$ = 1.85V, MCK = 500Hz	T_A = 25°C		10	40	µA
		All inputs driven at 1 (including TMS, TDI, TCK, NRST) Flash in standby mode All peripherals off	T_A = 85°C		20	150	
I_O	Output Current	PA0-PA3, V_{VDDIO} from 3.0V to 3.6V				16	mA
		PA0-PA3, V_{VDDIO} from 1.65V to 1.95V				8	mA
		PA4-PA31, PB0-PB31, PC0-PC23 and NRST, V_{VDDIO} from 3.0V to 3.6V				8	mA
		PA4-PA31, PB0-PB31, PC0-PC23 and NRST, V_{VDDIO} from 1.65V to 1.95V				4	mA

Table 40-3. 1.8V Voltage Regulator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{VDDIN}	Supply Voltage		3.0	3.3	3.6	V
V_{VDDOUT}	Output Voltage	I_O = 20 mA	1.81	1.85	1.89	V
I_{VDDIN}	Current consumption	After startup, no load		90		µA
		After startup, Idle mode, no load		10	25	µA
T_{START}	Startup Time	C_{load} = 2.2 µF, after V_{DDIN} > 2.7V			150	µS
I_O	Maximum DC Output Current	V_{DDIN} = 3.3V			100	mA
I_O	Maximum DC Output Current	V_{DDIN} = 3.3V, in Idle Mode			1	mA

Table 40-4. Brownout Detector Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{BOT18-}	VDDCORE Threshold Level		1.65	1.68	1.71	V
V_{HYST18}	VDDCORE Hysteresis	$V_{HYST18} = V_{BOT18+} - V_{BOT18-}$		50	65	mV
V_{BOT33-}	VDDFLASH Threshold Level		2.70	2.80	2.90	V
V_{HYST33}	VDDFLASH Hysteresis	$V_{HYST33} = V_{BOT33+} - V_{BOT33-}$		70	120	mV

Table 40-4. Brownout Detector Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{DD}	Current Consumption	BOD on (GPNVM0 bit active)		24	30	µA
		BOD off (GPNVM0 bit inactive)			1	µA
T_{START}	Startup Time			100	200	µs

Table 40-5. DC Flash Characteristics SAM7SE32

Symbol	Parameter	Conditions	Min	Max	Units
I_{SB}	Standby current	@25°C onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 3 25	 µA µA
		@85°C onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 5 125	 µA µA
I_{CC}	Active current	Random Read @ 30MHz onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 3.4 0.4	 mA mA
		Write onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 400 2.2	 µA mA

Table 40-6. DC Flash Characteristics SAM7SE512/256

Symbol	Parameter	Conditions	Min	Max	Units
I_{SB}	Standby current	@25°C onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 10 40	 µA µA
		@85°C onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 20 120	 µA µA
I_{CC}	Active current	Random Read @ 30MHz (one bank for SAM7SE512) onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 4.5 0.8	 mA mA
		Write (one bank for SAM7SE512) onto VDDCORE = 1.8V onto VDDFLASH = 3.3V		 400 5.5	 µA mA

40.3 Power Consumption

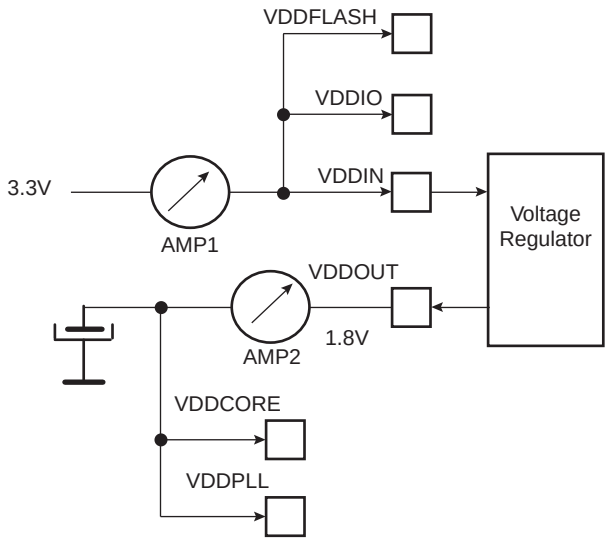
- Typical power consumption of PLLs, Slow Clock and Main Oscillator.
- Power consumption of power supply in two different modes: Active and ultra Low-power.
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

40.3.1 Power Consumption Versus Modes

The values in Table 40-7and Table 40-8 on page 620 are measured values of the power consumption with operating conditions as follows:

- $V_{DDIO} = V_{DDIN} = V_{DDFLASH} = 3.3V$
- $V_{DDCORE} = V_{DDPLL} = 1.85V$
- $T_A = 25° C$
- There is no consumption on the I/Os of the device

Figure 40-1. Measure Schematics:



The figures shown below in Table 40-7 represent the power consumption typically measured on the power supplies..

Table 40-7. Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Active (SAM7SE512/256/32)	Voltage regulator is on. Brown Out Detector is activated. Flash is read. ARM Core clock is 48 MHz. Analog-to-Digital Converter activated. All peripheral clocks activated. USB transceiver enabled. onto AMP1 onto AMP2	 31 29	mA
Ultra Low Power[2] (SAM7SE512/256/32)	Voltage regulator is in Low-power mode. Brown Out Detector is de-activated. Flash is in standby mode.[1] ARM Core in idle mode. MCK @ 500 Hz. Analog-to-Digital Converter de-activated. All peripheral clocks de-activated. USB transceiver disabled. DDM and DDP pins must be left floating. onto AMP1 onto AMP2	 26 12	µA

Notes: 1. “Flash is in standby mode”, means the Flash is not accessed at all.

2. Low power consumption figures stated above cannot be guaranteed when accessing the Flash in Ultra Low Power mode. In order to meet given low power consumption figures, it is recommended to either stop the processor or jump to SRAM.

40.3.2 Peripheral Power Consumption in Active Mode

Table 40-8. Power Consumption on V_{DDCORE}[1]

Peripheral	Consumption (Typ)	Unit
PIO Controller	12	µA/MHz
USART	30	
UDP	24	
PWM	15	
TWI	6	
SPI	18	
SSC	35	
Timer Counter Channels	7	
ARM7TDMI	170	
System Peripherals (SAM7SE512/256/32)	265	

Note: 1. Note: V_{DDCORE} = 1.85V, T_A = 25° C

40.4 Crystal Oscillators Characteristics

40.4.1 RC Oscillator Characteristics

Table 40-9. RC Oscillator Characteristics

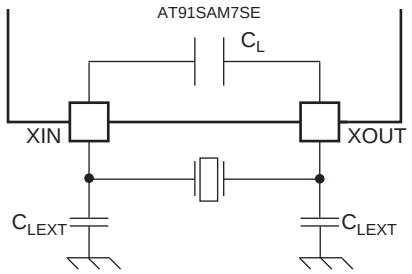
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPRC})$	RC Oscillator Frequency	V_{DDPLL} = 1.65V	22	32	42	kHz
	Duty Cycle		45	50	55	%
t_{ST}	Startup Time	V_{DDPLL} = 1.65V			75	µs
I_{OSC}	Current Consumption	After Startup Time			1.9	µA

40.4.2 Main Oscillator Characteristics

Table 40-10. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		3	16	20	MHz
C_{L1}, C_{L2}	SAM7SE512/256 Internal Load Capacitance ($C_{L1} = C_{L2}$)	Integrated Load Capacitance ((XIN or XOUT))	34	40	46	pF
C_{L1}, C_{L2}	SAM7SE32 Internal Load Capacitance ($C_{L1} = C_{L2}$)	Integrated Load Capacitance (XIN or XOUT)	18	22	26	pF
C_L (6)	SAM7SE512/256 Equivalent Load Capacitance	Integrated Load Capacitance (XIN and XOUT in series)	17	20	23	pF
C_L (6)	SAM7SE32 Equivalent Load Capacitance	Integrated Load Capacitance (XIN and XOUT in series)	9	11	13	pF
	Duty Cycle		30	50	70	%
t_{ST}	Startup Time	V_{DDPLL} = 1.2 to 2V C_S = 3 pF(1) $1/(t_{CPMAIN})$ = 3 MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 16 MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 20 MHz			14.5 1.4 1	ms
I_{DDST}	Standby Current Consumption	Standby mode			1	µA
P_{ON}	Drive level	@3 MHz @8 MHz @16 MHz @20 MHz			15 30 50 50	µW
$I_{DD\ ON}$	Current dissipation	@3 MHz (2) @8 MHz (3) @16 MHz (4) @20 MHz (5)		150 150 300 400	250 250 450 550	µA
C_{LEXT} (6)	Maximum external capacitor on XIN and XOUT				10	pF

Notes:
1. C_S is the shunt capacitance.
2. R_S = 100-200 Ω; C_{SHUNT} = 2.0 - 2.5 pF; C_M = 2 – 1.5 fF (typ, worst case) using 1 K ohm serial resistor on xout.
3. R_S = 50-100 Ω; C_{SHUNT} = 2.0 - 2.5 pF; C_M = 4 - 3 fF (typ, worst case).
4. R_S = 25-50 Ω; C_{SHUNT} = 2.5 - 3.0 pF; C_M = 7 -5 fF (typ, worst case).
5. R_S = 20-50 Ω; C_{SHUNT} = 3.2 - 4.0 pF; C_M = 10 - 8 fF (typ, worst case).
6. C_L and C_{LEXT} ∅



40.4.3 Crystal Characteristics

Table 40-11. Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Fundamental @3 MHz Fundamental @8 MHz Fundamental @16 MHz Fundamental @20 MHz			200 100 80 50	Ω
C_M	Motional capacitance				8	fF
C_{SHUNT}	Shunt capacitance				7	pF

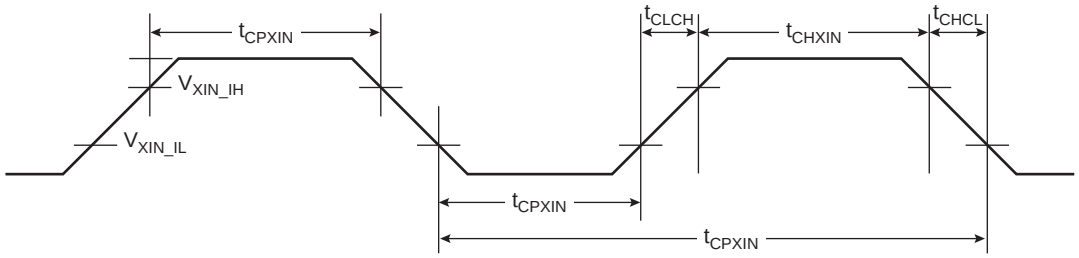
40.4.4 XIN Clock Characteristics

Table 40-12. XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency	(1)		50.0	MHz
t_{CPXIN}	XIN Clock Period	(1)	20.0		ns
t_{CHXIN}	XIN Clock High Half-period	(1)	8.0		ns
t_{CLXIN}	XIN Clock Low Half-period	(1)	8.0		ns
t_{CLCH}	Rise Time	(1)		400	ns
t_{CHCL}	Fall Time	(1)		400	ns
C_{IN}	XIN Input Capacitance (SAM7SE512/256)	(1)		46	pF
C_{IN}	XIN Input Capacitance (SAM7SE32)	(1)		26	pF
R_{IN}	XIN Pull-down Resistor	(1)		500	kΩ
V_{XIN_IL}	V_{XIN} Input Low-level Voltage	(1)	-0.3	0.3 x V_{DDPLL}	V
V_{XIN_IH}	V_{XIN} Input High-level Voltage	(1)	0.7 x V_{DDPLL}	1.95	V
I_{DDBP}	Bypass Current Consumption	(1)		15	µW/MHz

Note: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR_MOR register, see the Clock Generator Main Oscillator Register.

Figure 40-2. XIN Clock Timing



40.5 PLL Characteristics

Table 40-13. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{OUT}	Output Frequency	Field OUT of CKGR_PLL is: 00	80		160	MHz
		Field OUT of CKGR_PLL is: 10	150		220	MHz
F_{IN}	Input Frequency		1		32	MHz
I_{PLL}	Current Consumption	Active mode			4	mA
		Standby mode			1	µA

Note: Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

40.6 USB Transceiver Characteristics

40.6.1 Electrical Characteristics

Table 40-14. Electrical Parameters

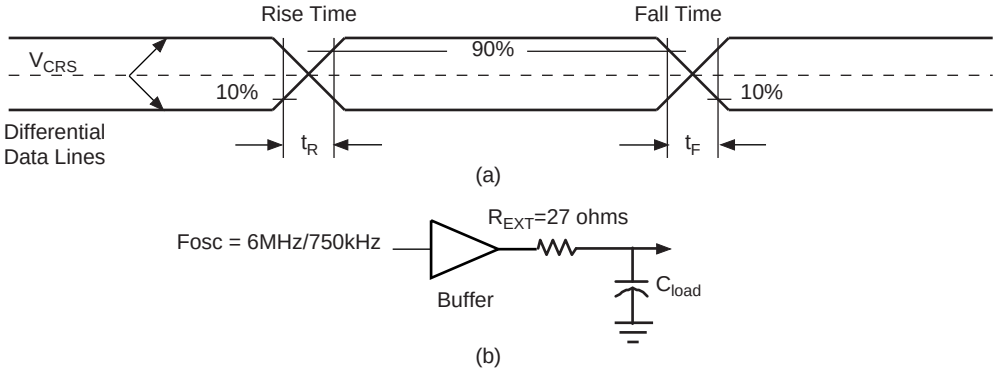
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
		Input Levels				
V_{IL}	Low Level				0.8	V
V_{IH}	High Level		2.0			V
V_{DI}	Differential Input Sensitivity	|(D+) - (D-)|	0.2			V
V_{CM}	Differential Input Common Mode Range		0.8		2.5	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line			9.18	pF
I	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	-10		+10	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		27		Ω
		Output Levels				
V_{OL}	Low Level Output	Measured with R_L of 1.425 kOhm tied to 3.6V	0.0		0.3	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kOhm tied to GND	2.8		3.6	V
V_{CRS}	Output Signal Crossover Voltage	Measure conditions described in Figure 40-3	1.3		2.0	V
		Consumption				
I_{VDDIO}	Current Consumption	Transceiver enabled in input mode DDP=1 and DDM=0		105	200	µA
$I_{VDDCORE}$	Current Consumption			80	150	µA
		Pull-up Resistor				
R_{PUI}	Bus Pull-up Resistor on Upstream Port (idle bus)		0.900		1.575	kΩ
R_{PUA}	Bus Pull-up Resistor on Upstream Port (upstream port receiving)		1.425		3.090	kΩ

40.6.2 Switching Characteristics

Table 40-15. In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 50 pF	4		20	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 50 pF	4		20	ns
t_{FRFM}	Rise/Fall time Matching		90		111.11	%

Figure 40-3. USB Data Signal Rise and Fall Times



40.7 ADC Characteristics

Table 40-16. Channel Conversion Time and ADC Clock

Parameter	Conditions	Min	Typ	Max	Units
ADC Clock Frequency	10-bit resolution mode			5	MHz
	8-bit resolution mode			8	
Startup Time	Return from Idle Mode			20	µs
Track and Hold Acquisition Time		600			ns
Conversion Time	ADC Clock = 5 MHz			2	µs
	ADC Clock = 8 MHz			1.25	
Throughput Rate	ADC Clock = 5 MHz			384(1)	kSPS
	ADC Clock = 8 MHz			533(2)	

Notes: 1. Corresponds to 13 clock cycles at 5 MHz: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

2. Corresponds to 15 clock cycles at 8 MHz: 5 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

Table 40-17. External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Units
ADVREF Input Voltage Range		2.6		V_{DDIN}	V
	8-bit resolution mode	2.5			
ADVREF Average Current	On 13 samples with ADC Clock = 5 MHz		200	250	µA
Current Consumption on VDDIN			0.55	1	mA

Table 40-18. Analog Inputs

Parameter	Min	Typ	Max	Units
Input Voltage Range	0		V_{ADVREF}	
Input Leakage Current			1	µA
Input Capacitance		12	14	pF

The user can drive ADC input with impedance up to:

- $Z_{OUT} \leq$ (SHTIM -470) x 10 in 8-bit resolution mode
- $Z_{OUT} \leq$ (SHTIM -589) x 7.69 in 10-bit resolution mode

with SHTIM (Sample and Hold Time register) expressed in ns and Z_{OUT} expressed in ohms.

Table 40-19. Transfer Characteristics

Parameter	Conditions	Min	Typ	Max	Units
Resolution			10		Bit
Integral Non-linearity				±2	LSB
Differential Non-linearity	No missing code			±1	LSB
Offset Error				±2	LSB
Gain Error				±2	LSB
Absolute Accuracy				±4	LSB

For more information on data converter terminology, please refer to the application note: Data Converter Terminology, Atmel lit° 6022.

40.8 AC Characteristics

40.8.1 Master Clock Characteristics

Table 40-20. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPMCK})$	Master Clock Frequency	V_{DDCORE} = 1.8V		55	MHz
$1/(t_{CPMCK})$	Master Clock Frequency	V_{DDCORE} = 1.65V		48	MHz

40.8.2 I/O Characteristics

Criteria used to define the maximum frequency of the I/Os:

- output duty cycle (30%-70%)
- minimum output swing: 100mV to VDDIO - 100mV
- Addition of rising and falling time inferior to 75% of the period

Table 40-21. I/O Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
$FreqMax_{IO1}$	Pin Group 1 (1) frequency	Load: 30 pF(4)		48.2	MHz
		Load: 30 pF(5)		25	MHz
$PulseminH_{IO1}$	Pin Group 1 (1) High Level Pulse Width	Load: 30 pF(4)	20		ns
		Load: 30 pF(5)	40		
$PulseminL_{IO1}$	Pin Group 1 (1) Low Level Pulse Width	Load: 30 pF(4)	20		ns
		Load: 30 pF(5)	40		
$FreqMax_{IO2}$	Pin Group 2 (2) frequency	Load: 40 pF(4)		25	MHz
		Load: 40 pF(5)		16	MHz
$PulseminH_{IO2}$	Pin Group 2 (2) High Level Pulse Width	Load: 40 pF(4)	20		ns
		Load: 40 pF(5)	31		ns
$PulseminL_{IO2}$	Pin Group 2 (2) Low Level Pulse Width	Load: 40 pF(4)	20		ns
		Load: 40 pF(5)	31		ns
$FreqMax_{IO3}$	Pin Group 3 (3)frequency	Load: 40 pF(4)		30	MHz
		Load: 40 pF(5)		20	MHz
$PulseminH_{IO3}$	Pin Group 3 (3) High Level Pulse Width	Load: 40 pF(4)	16.6		ns
		Load: 40 pF(5)	31		ns
$PulseminL_{IO3}$	Pin Group 3 (3) Low Level Pulse Width	Load: 40 pF(4)	16.6		ns
		Load: 40 pF(5)	31		ns

Notes: 1. Pin Group 1 = SDCK
2. Pin Group 2 = PA4 to PA31, PB0 to PB31 and PC0-PC23

3. Pin Group 3 = PA0 to PA3
4. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40 pF
5. V_{VDDIO} from 1.65V to 1.95V, maximum external capacitor = 40 pF

40.8.3 SPI Characteristics

Figure 40-4. SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)

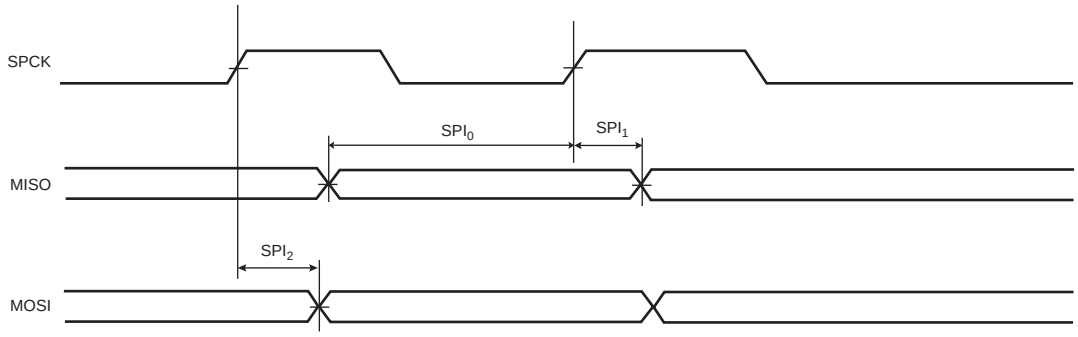


Figure 40-5. SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)

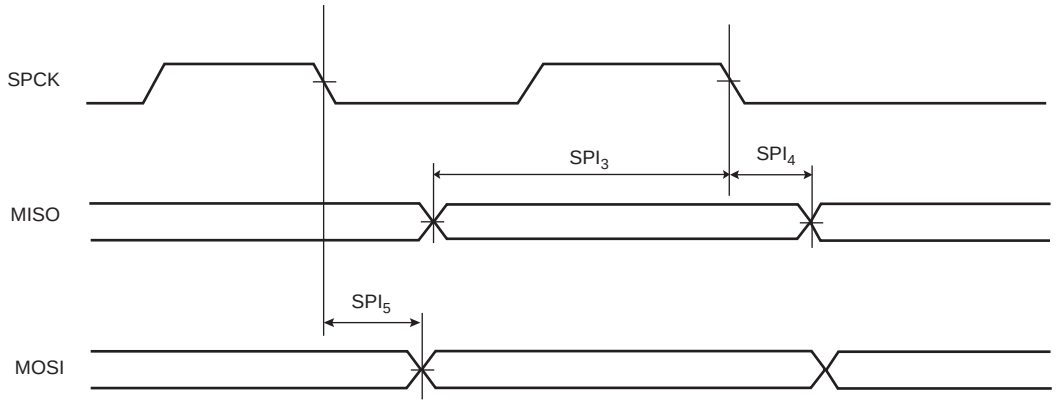


Figure 40-6. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

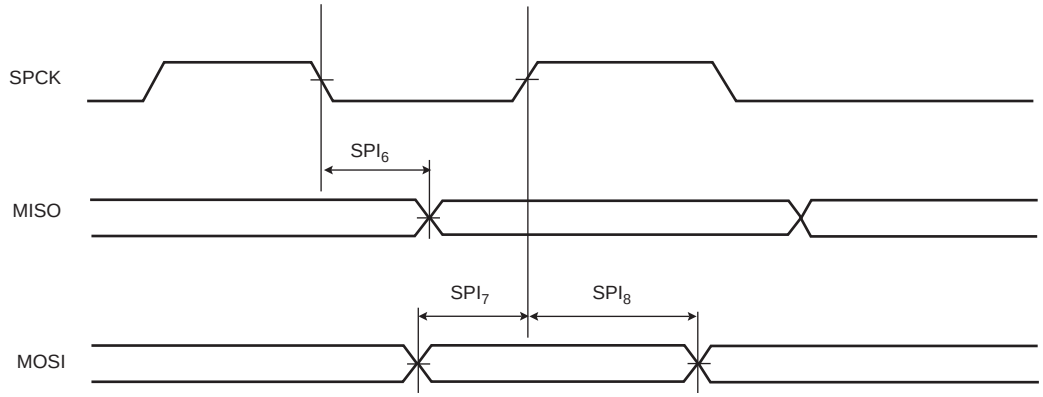


Figure 40-7. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

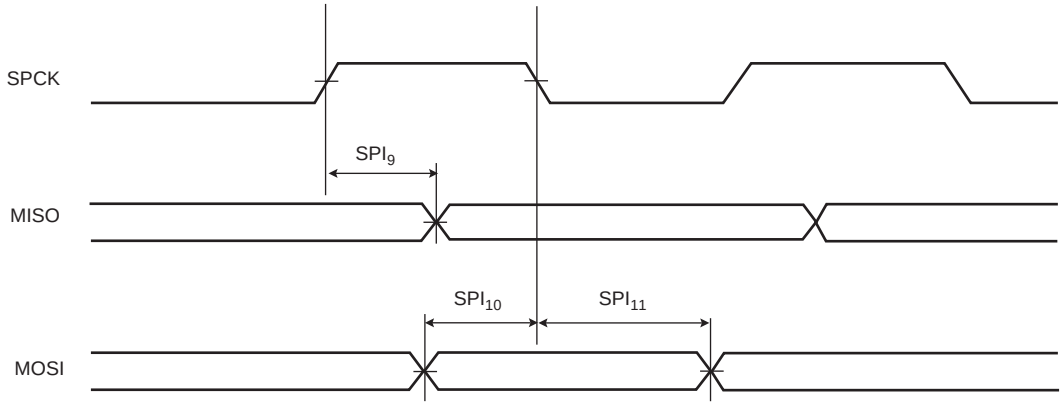


Table 40-22. SAM7SE512/256 SPI Timings

Symbol	Parameter	Conditions	Min	Max	Units
SPI_0	MISO Setup time before SPCK rises (master)	3.3V domain[1]	26 + (t_{CPMCK})/2[3]		ns
		1.8V domain[2]	34 + (t_{CPMCK})/2[3]		ns
SPI_1	MISO Hold time after SPCK rises (master)	3.3V domain[1]	0		ns
		1.8V domain[2]	0		ns
SPI_2	SPCK rising to MOSI Delay (master)	3.3V domain[1]		7	ns
		1.8V domain[2]		10	ns
SPI_3	MISO Setup time before SPCK falls (master)	3.3V domain[1]	26 + (t_{CPMCK})/2[3]		ns
		1.8V domain[2]	34 + (t_{CPMCK})/2[3]		ns
SPI_4	MISO Hold time after SPCK falls (master)	3.3V domain[1]	0		ns
		1.8V domain[2]	0		ns
SPI_5	SPCK falling to MOSI Delay (master)	3.3V domain[1]		7	ns
		1.8V domain[2]		10	ns
SPI_6	SPCK falling to MISO Delay (slave)	3.3V domain[1]		22.5	ns
		1.8V domain[2]		30.5	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	3.3V domain[1]	1		ns
		1.8V domain[2]	2.5		ns
SPI_8	MOSI Hold time after SPCK rises (slave)	3.3V domain[1]	2		ns
		1.8V domain[2]	2		ns
SPI_9	SPCK rising to MISO Delay (slave)	3.3V domain[1]		23	ns
		1.8V domain[2]		28	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	3.3V domain[1]	1		ns
		1.8V domain[2]	1		
SPI_{11}	MOSI Hold time after SPCK falls (slave)	3.3V domain[1]	2		ns
		1.8V domain[2]	2		ns

Notes: 1. 3.3V domain: V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40 pF.
2. 1.8V domain: V_{VDDIO} from 1.65V to 1.95V, maximum external capacitor = 20 pF.
3. t_{CPMCK}: Master Clock period in ns.

SAM7SE32 SPI Timings

Symbol	Parameter	Conditions	Min	Max	Units
SPI_0	MISO Setup time before SPCK rises (master)	3.3V domain[1]	26 + (t_{CPMCK})/2[3]		ns
		1.8V domain[2]	45 + (t_{CPMCK})/2[3]		ns
SPI_1	MISO Hold time after SPCK rises (master)	3.3V domain[1]	0		ns
		1.8V domain[2]	0		ns
SPI_2	SPCK rising to MOSI Delay (master)	3.3V domain[1]		4	ns
		1.8V domain[2]		12	ns
SPI_3	MISO Setup time before SPCK falls (master)	3.3V domain[1]	26 + (t_{CPMCK})/2[3]		ns
		1.8V domain[2]	34 + (t_{CPMCK})/2[3]		ns
SPI_4	MISO Hold time after SPCK falls (master)	3.3V domain[1]	0		ns
		1.8V domain[2]	0		ns
SPI_5	SPCK falling to MOSI Delay (master)	3.3V domain[1]		4	ns
		1.8V domain[2]		6	ns
SPI_6	SPCK falling to MISO Delay (slave)	3.3V domain[1]		23.7	ns
		1.8V domain[2]		42	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	3.3V domain[1]	1		ns
		1.8V domain[2]	1		ns
SPI_8	MOSI Hold time after SPCK rises (slave)	3.3V domain[1]	3		ns
		1.8V domain[2]	3		ns
SPI_9	SPCK rising to MISO Delay (slave)	3.3V domain[1]		24	ns
		1.8V domain[2]		40	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	3.3V domain[1]	1		ns
		1.8V domain[2]	1		
SPI_{11}	MOSI Hold time after SPCK falls (slave)	3.3V domain[1]	3		ns
		1.8V domain[2]	3		ns

Notes:
1. 3.3V domain: V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40 pF.
2. 1.8V domain: V_{VDDIO} from 1.65V to 1.95V, maximum external capacitor = 20 pF.
3. t_{CPMCK}: Master Clock period in ns.

Note that in SPI master mode the ATSAM7SE512/256/32 does not sample the data (MISO) on the opposite edge where data clocks out (MOSI) but the same edge is used as shown in Figure 40-4 and Figure 40-5.

40.8.4 SMC Signals

These timings are given for a maximum 10 pF load on SDCK and a maximum 50 pF load on the databus.

Table 40-23. SAM7SE512/256 General-purpose SMC Signals

Symbol	Parameter	Conditions	Min	Max	Units
SMC_7	NCS Minimum Pulse Width (Address to Chip Select Setup)	3.3V domain	(n + 1) x t_{CPMCK} - 2.5 (1)		ns
		1.8V domain	(n + 1) x t_{CPMCK} - 3.0 (1)		ns
SMC_8	NWAIT Minimum Pulse Width		t_{CPMCK}		ns

Note: 1. n = Number of standard Wait States inserted.

Table 40-24. SAM7SE32 General-purpose SMC Signals

Symbol	Parameter	Conditions	Min	Max	Units
SMC_7	NCS Minimum Pulse Width (Address to Chip Select Setup)	3.3V domain	(n + 1) x t_{CPMCK} - 2.5 (1)		ns
		1.8V domain	(n + 1) x t_{CPMCK} - 5.0 (1)		ns
SMC_8	NWAIT Minimum Pulse Width		t_{CPMCK}		ns

Note: 1. n = Number of standard Wait States inserted.

.

Table 40-25. SAM7SE512/256 SMC Write Signals

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{15}	NWR High to NUB Change (3)	3.3V domain	7.0		ns
		1.8V domain	9.5		ns
SMC_{16}	NWR High to NLB/A0 Change (3)	3.3V domain	7.5		ns
		1.8V domain	10		ns
SMC_{17}	NWR High to A1 - A22 Change(3)	3.3V domain	8		ns
		1.8V domain	8.5		ns
SMC_{18}	NWR High to Chip Select Inactive (3)	3.3V domain	7.0		ns
		1.8V domain	9.0		ns
SMC_{19}	Data Out Valid before NWR High (No Wait States) (3)	3.3V domain	0.5 * t_{CPMCK} - 0.5		ns
		1.8V domain	0.5 * t_{CPMCK} - 1		ns
SMC_{20}	Data Out Valid before NWR High (Wait States) (3)	3.3V domain	n x t_{CPMCK} - 0.5(1)		ns
		1.8V domain	n x t_{CPMCK} - 1(1)		ns
SMC_{21}	Data Out Valid after NWR High (No Wait States)(3))	3.3V domain	0.5 * t_{CPMCK} - 5.7		ns
		1.8V domain	0.5 * t_{CPMCK} - 8		ns
SMC_{22}	Data Out Valid after NWR High (Wait States without Hold Cycles) (3)	3.3V domain	0.5 * t_{CPMCK} - 5.2		ns
		1.8V domain	0.5 * t_{CPMCK} - 8		ns
SMC_{23}	Data Out Valid after NWR High (Wait States with Hold Cycles) (3)	3.3V domain	h x t_{CPMCK} - 5.7 (2)		ns
		1.8V domain	h x t_{CPMCK} - 8.0 (2)		ns

Table 40-25. SAM7SE512/256 SMC Write Signals (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{26}	NWR Minimum Pulse Width (No Wait States) (3)	3.3V domain	0.5 * t_{CPMCK} - 1		ns
		1.8V domain	0.5 * t_{CPMCK} - 1.5		ns
SMC_{27}	NWR Minimum Pulse Width (Wait States) (3)	3.3V domain	n x t_{CPMCK} - 1.5(1)		ns
		1.8V domain	n x t_{CPMCK} - 1.5(1)		ns

Notes: 1. n = Number of standard Wait States inserted.
2. h = Number of Hold Cycles inserted.
3. Not applicable when Address to Chip Select Setup Cycles are inserted.

.

Table 40-26. SAM7SE32 SMC Write Signals

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{15}	NWR High to NUB Change (3)	3.3V domain	6.0		ns
		1.8V domain	9.0		ns
SMC_{16}	NWR High to NLB/A0 Change (3)	3.3V domain	6.0		ns
		1.8V domain	9.0		ns
SMC_{17}	NWR High to A1 - A22 Change(3)	3.3V domain	6.0		ns
		1.8V domain	9.0		ns
SMC_{18}	NWR High to Chip Select Inactive (3)	3.3V domain	5.5		ns
		1.8V domain	9.0		ns
SMC_{19}	Data Out Valid before NWR High (No Wait States) (3)	3.3V domain	0.5 * t_{CPMCK} - 3.5		ns
		1.8V domain	0.5 * t_{CPMCK} - 6.0		ns
SMC_{20}	Data Out Valid before NWR High (Wait States) (3)	3.3V domain	n x t_{CPMCK} - 3.5(1)		ns
		1.8V domain	n x t_{CPMCK} - 6.0(1)		ns
SMC_{21}	Data Out Valid after NWR High (No Wait States)(3))	3.3V domain	0.5 * t_{CPMCK} - 5.5		ns
		1.8V domain	0.5 * t_{CPMCK} - 12		ns
SMC_{22}	Data Out Valid after NWR High (Wait States without Hold Cycles) (3)	3.3V domain	0.5 * t_{CPMCK} - 5.2		ns
		1.8V domain	0.5 * t_{CPMCK} - 8		ns
SMC_{23}	Data Out Valid after NWR High (Wait States with Hold Cycles) (3)	3.3V domain	h x t_{CPMCK} - 6.0(2)		ns
		1.8V domain	h x t_{CPMCK} - 12(2)		ns
SMC_{26}	NWR Minimum Pulse Width (No Wait States) (3)	3.3V domain	0.5 * t_{CPMCK} - 2.0		ns
		1.8V domain	0.5 * t_{CPMCK} - 6.5		ns
SMC_{27}	NWR Minimum Pulse Width (Wait States) (3)	3.3V domain	n x t_{CPMCK} - 2.5(1)		ns
		1.8V domain	n x t_{CPMCK} - 7.0(1)		ns

Notes: 1. n = Number of standard Wait States inserted.
2. h = Number of Hold Cycles inserted.
3. Not applicable when Address to Chip Select Setup Cycles are inserted.

Table 40-27. SAM7SE512/256 SMC Read Signals

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{35}	NRD High to NUB Change	3.3V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 1$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 1$ (4)	ns
SMC_{36}	NRD High to NLB/A0 Change	3.3V domain	$(h \times t_{CPMCK}) - 1.5$ (4)	$(h \times t_{CPMCK}) + 1.5$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 1.5$ (4)	$(h \times t_{CPMCK}) + 1$ (4)	ns
SMC_{37}	NRD High to A1-A22 Change	3.3V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 2$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 3.5$ (4)	ns
SMC_{38}	NRD High to Chip Select Inactive	3.3V domain	$(h \times t_{CPMCK}) - 3$ (4)	$(h \times t_{CPMCK}) + 1$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 3.5$ (4)	$(h \times t_{CPMCK}) + 2$ (4)	ns
SMC_{40}	Data Setup before NRD High	3.3V domain	22.2		ns
		1.8V domain	35		ns
SMC_{41}	Data Hold after NRD High	3.3V domain	0		ns
		1.8V domain	0		ns
SMC_{42}	Data Setup before NCS High	3.3V domain	23.2		ns
		1.8V domain	37		ns
SMC_{43}	Data Hold after NCS High	3.3V domain	0		ns
		1.8V domain	0		ns
SMC_{44}	NRD Minimum Pulse Width (1) (5)	3.3V domain	$(n+1) \times t_{CPMCK} - 1$ (3)		ns
		1.8V domain	$(n+1) \times t_{CPMCK} - 1.5$ (3)		ns
SMC_{45}	NRD Minimum Pulse Width (2) (5)	3.3V domain	$(2 \times n+1) \times 0.5 \times t_{CPMCK} - 1$ (3)		ns
		1.8V domain	$(2 \times n+1) \times 0.5 \times t_{CPMCK} - 1$ (3)		ns

Notes:
1. Early Read Protocol.
2. Standard Read Protocol.
3. n = Number of standard Wait States inserted.
4. h = Number of Hold Cycles inserted.
5. Not applicable when Address to Chip Select Setup Cycles are inserted.

Table 40-28. SAM7SE32 SMC Read Signals

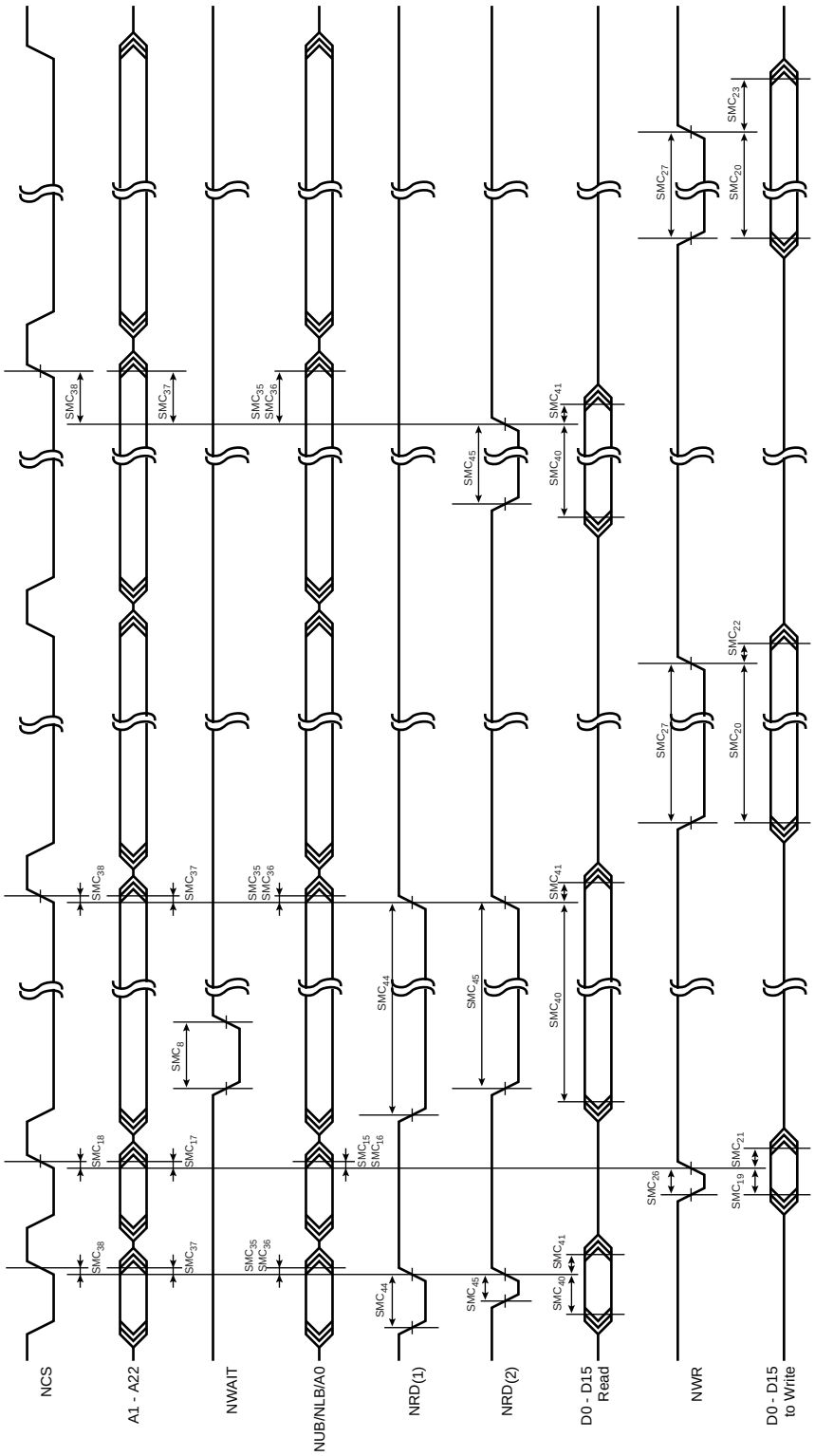
Symbol	Parameter	Conditions	Min	Max	Units
SMC_{35}	NRD High to NUB Change	3.3V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 1.5$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 7$ (4)	ns
SMC_{36}	NRD High to NLB/A0 Change	3.3V domain	$(h \times t_{CPMCK}) - 2$ (4)	$(h \times t_{CPMCK}) + 1.5$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 1.5$ (4)	$(h \times t_{CPMCK}) + 6.5$ (4)	ns
SMC_{37}	NRD High to A1-A22 Change	3.3V domain	$(h \times t_{CPMCK}) - 3$ (4)	$(h \times t_{CPMCK}) + 3$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 3$ (4)	$(h \times t_{CPMCK}) + 8$ (4)	ns
SMC_{38}	NRD High to Chip Select Inactive	3.3V domain	$(h \times t_{CPMCK}) - 2.5$ (4)	$(h \times t_{CPMCK}) + 2$ (4)	ns
		1.8V domain	$(h \times t_{CPMCK}) - 3$ (4)	$(h \times t_{CPMCK}) + 2$ (4)	ns

Table 40-28. SAM7SE32 SMC Read Signals (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{40}	Data Setup before NRD High	3.3V domain	23.2		ns
		1.8V domain	37		ns
SMC_{41}	Data Hold after NRD High	3.3V domain	-0		ns
		1.8V domain	-0		ns
SMC_{42}	Data Setup before NCS High	3.3V domain	25.2		ns
		1.8V domain	39		ns
SMC_{43}	Data Hold after NCS High	3.3V domain	0		ns
		1.8V domain	0		ns
SMC_{44}	NRD Minimum Pulse Width (1) (5)	3.3V domain	$(n+1) \times t_{CPMCK} - 2$ (3)		ns
		1.8V domain	$(n+1) \times t_{CPMCK} - 6$ (3)		ns
SMC_{45}	NRD Minimum Pulse Width (2) (5)	3.3V domain	$(2 \times n+1) \times 0.5 \times t_{CPMCK} - 2$ (3)		ns
		1.8V domain	$(2 \times n+1) \times 0.5 \times t_{CPMCK} - 6.5$ (3)		ns

Notes:
1. Early Read Protocol.
2. Standard Read Protocol.
3. n = Number of standard Wait States inserted.
4. h = Number of Hold Cycles inserted.
5. Not applicable when Address to Chip Select Setup Cycles are inserted.

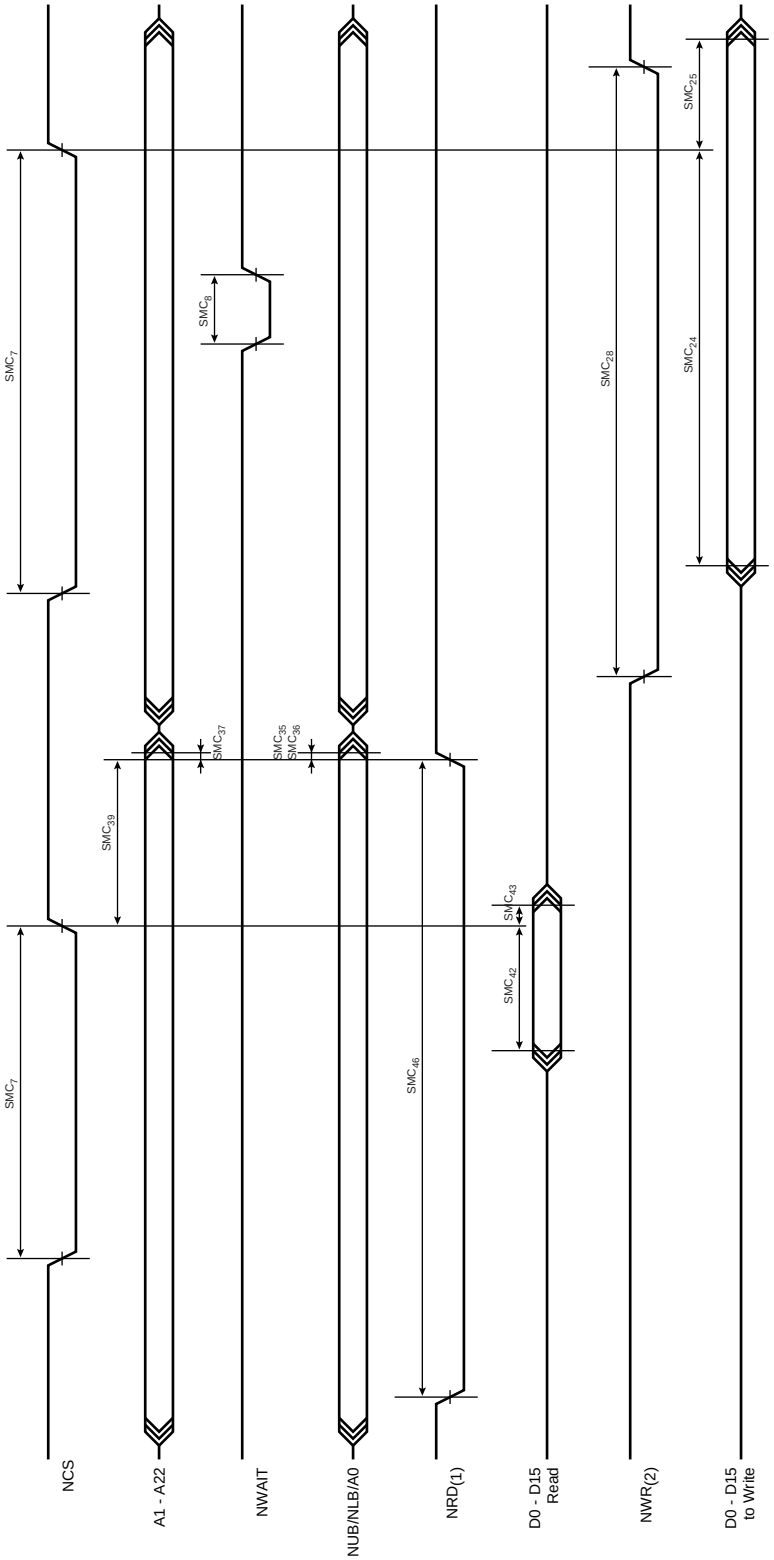
Figure 40-8. SMC Signals in Memory Interface Mode



Notes: 1. Early Read Protocol

2. Standard Read Protocol with or without Setup and Hold Cycles.

Figure 40-9. SM Signals in LCD Interface Mode



Notes: 1. Standard Read Protocol only.
2. With Standard Wait States inserted only.

40.8.5 SDRAMC Signals

These timings are given for a maximum 30 pF load on SDCK and a maximum 50 pF load on the databus.

Table 40-29. SDRAMC Clock Signal

Symbol	Parameter	Min		Max		Units
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
$1/(t_{CPSDCK})$	SDRAM Controller Clock Frequency			24	48.2	MHz
t_{CPSDCK}	SDRAM Controller Clock Period	41.7	20.7			ns

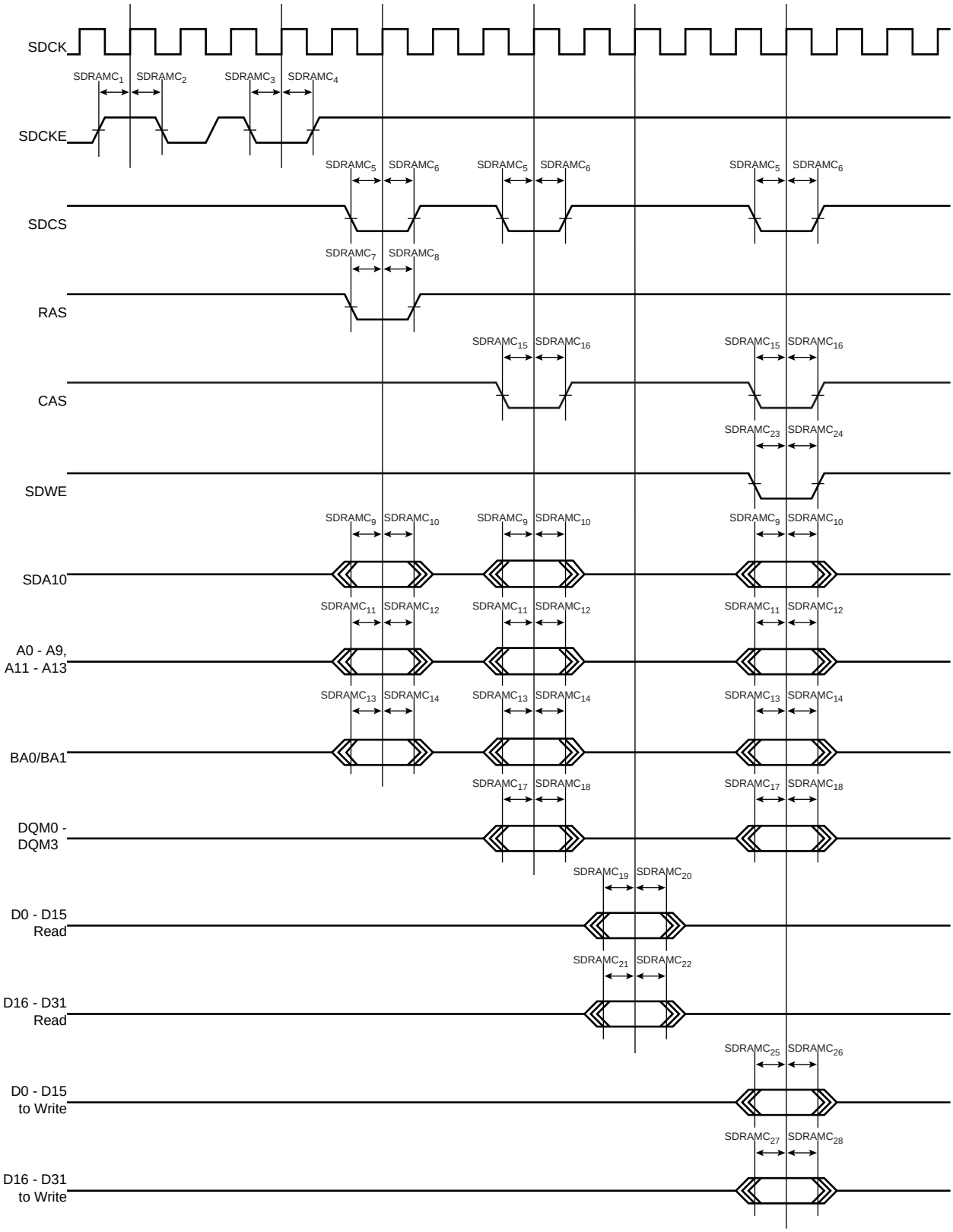
Table 40-30. SAM7SE512/256 SDRAMC Signals

Symbol	Parameter	Min		Max		Units
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
$SDRAMC_1$	SDCKE High before SDCK Rising Edge	17.5	12			ns
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge	22	9.5			ns
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge	11	10			ns
$SDRAMC_4$	SDCKE High after SDCK Rising Edge	20.5	8			ns
$SDRAMC_5$	SDCS Low before SDCK Rising Edge	11	10.5			ns
$SDRAMC_6$	SDCS High after SDCK Rising Edge	20.5	7.5			ns
$SDRAMC_7$	RAS Low before SDCK Rising Edge	10.5	10			ns
$SDRAMC_8$	RAS High after SDCK Rising Edge	20.5	8			ns
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge	10.5	10			ns
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge	20.5	8			ns
$SDRAMC_{11}$	Address Change before SDCK Rising Edge	8.5	7.5			ns
$SDRAMC_{12}$	Address Change after SDCK Rising Edge	20	9			ns
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge	9	8			ns
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge	20.5	9			ns
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge	10.5	10			ns
$SDRAMC_{16}$	CAS High after SDCK Rising Edge	20.5	8			ns
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge	10	9.5			ns
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge	20.5	9			ns
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge	16	12.5			ns
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge	3	2			ns
$SDRAMC_{21}$	D16-D31 in Setup before SDCK Rising Edge	16	12.5			ns
$SDRAMC_{22}$	D16-D31 in Hold after SDCK Rising Edge	3	2			ns
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge	10.5	10			ns
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge	20.5	8			ns
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge	6.5	5.5			ns
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge	17	4.5			ns
$SDRAMC_{27}$	D16-D31 Out Valid before SDCK Rising Edge	6.5	5.5			ns
$SDRAMC_{28}$	D16-D31 Out Valid after SDCK Rising Edge	17	4.5			ns

Table 40-31. SAM7SE32 SDRAMC Signals

Symbol	Parameter	Min		Max		Units
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
$SDRAMC_1$	SDCKE High before SDCK Rising Edge	11.5	6.5			ns
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge	23.5	11.5			ns
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge	10.5	5.5			ns
$SDRAMC_4$	SDCKE High after SDCK Rising Edge	22.5	11			ns
$SDRAMC_5$	SDCS Low before SDCK Rising Edge	11.5	7.5			ns
$SDRAMC_6$	SDCS High after SDCK Rising Edge	22	10.5			ns
$SDRAMC_7$	RAS Low before SDCK Rising Edge	12.5	8			ns
$SDRAMC_8$	RAS High after SDCK Rising Edge	22	10			ns
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge	12.5	8			ns
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge	22	10			ns
$SDRAMC_{11}$	Address Change before SDCK Rising Edge	10	5			ns
$SDRAMC_{12}$	Address Change after SDCK Rising Edge	22	10.5			ns
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge	9.5	4.5			ns
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge	22.5	10.5			ns
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge	12	7			ns
$SDRAMC_{16}$	CAS High after SDCK Rising Edge	22	10.5			ns
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge	8.5	4.5			ns
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge	22	10.5			ns
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge	8.5	8.5			ns
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge	2	1			ns
$SDRAMC_{21}$	D16-D31 in Setup before SDCK Rising Edge	8.5	8.5			ns
$SDRAMC_{22}$	D16-D31 in Hold after SDCK Rising Edge	2	1			ns
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge	12	7.5			ns
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge	22	10.5			ns
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge	6.5	2			ns
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge	20	9			ns
$SDRAMC_{27}$	D16-D31 Out Valid before SDCK Rising Edge	6.5	2			ns
$SDRAMC_{28}$	D16-D31 Out Valid after SDCK Rising Edge	20	9			ns

Figure 40-10. SDRAMC Signals



40.8.6 Embedded Flash Characteristics

The maximum operating frequency is given in Table 40-32 and Table 40-33 but is limited by the Embedded Flash access time when the processor is fetching code out of it. Table 40-32 and Table 40-33 give the device maximum operating frequency depending on the FWS field of the MC_FMR register. This field defines the number of wait states required to access the Embedded Flash Memory.

Table 40-32. Embedded Flash Wait States (VDDCORE = 1.65V)

FWS[1]	Read Operations	Maximum Operating Frequency (MHz)
0	1 cycle	25
1	2 cycles	44
2	3 cycles	48.2
3[2]	4 cycles	48.2

Notes: 1. FWS = Flash Wait States
2. It is not necessary to use 3 wait states because the Flash can operate at maximum frequency with only 2 wait states.

Table 40-33. Embedded Flash Wait States (VDDCORE = 1.8V)

FWS[1]	Read Operations	Maximum Operating Frequency (MHz)
0	1 cycle	30
1	2 cycles	55
2[2]	3 cycles	55
3[2]	4 cycles	55

Notes: 1. FWS = Flash Wait States
2. It is not necessary to use 2 or 3 wait states because the Flash can operate at maximum frequency with only 1 wait state.

Table 40-34. AC Flash Characteristics

Parameter	Conditions	Min	Max	Units
Program Cycle Time	per page including auto-erase		6	ms
	per page without auto-erase		3	ms
Full Chip Erase		15		ms

40.8.7 JTAG/ICE Timings

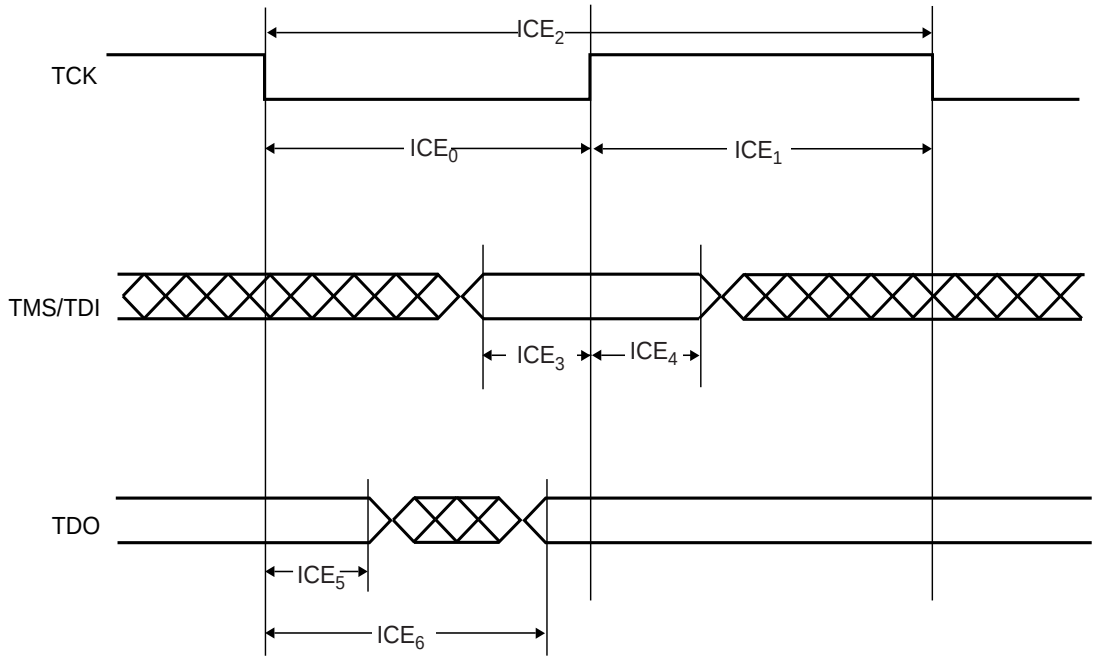
40.8.7.1 ICE Interface Signals

Table 40-35. ICE Interface Timing Specification

Symbol	Parameter	Conditions	Min	Max	Units
ICE_0	TCK Low Half-period	(1)	51		ns
ICE_1	TCK High Half-period	(1)	51		ns
ICE_2	TCK Period	(1)	102		ns
ICE_3	TDI, TMS, Setup before TCK High	(1)	0		ns
ICE_4	TDI, TMS, Hold after TCK High	(1)	3		ns
ICE_5	TDO Hold Time	(1)	13		ns
ICE_6	TCK Low to TDO Valid	(1)		20	ns

Note: 1. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF.

Figure 40-11. ICE Interface Signals



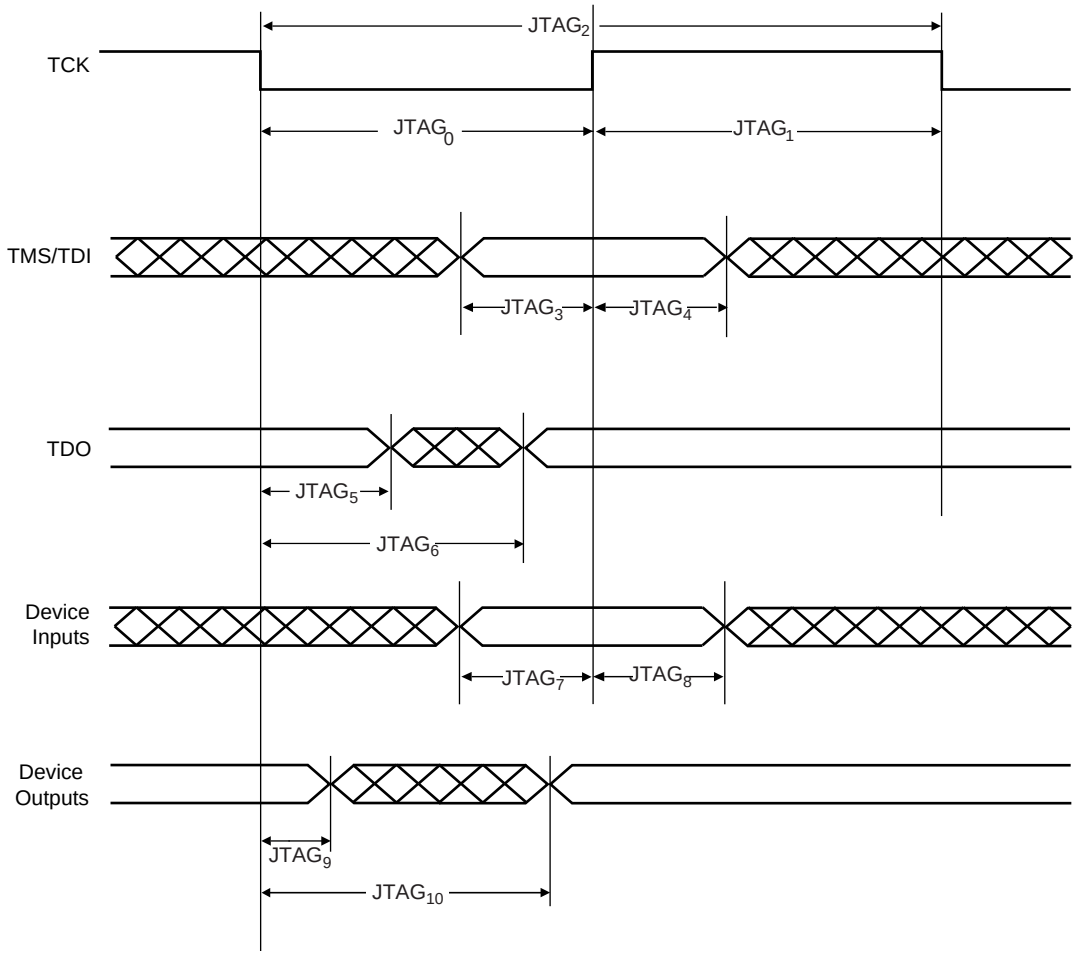
40.8.7.2 JTAG Interface Signals

Table 40-36. JTAG Interface Timing specification

Symbol	Parameter	Conditions	Min	Max	Units
$JTAG_0$	TCK Low Half-period	(1)	6.5		ns
$JTAG_1$	TCK High Half-period	(1)	5.5		ns
$JTAG_2$	TCK Period	(1)	12		ns
$JTAG_3$	TDI, TMS Setup before TCK High	(1)	2		ns
$JTAG_4$	TDI, TMS Hold after TCK High	(1)	3		ns
$JTAG_5$	TDO Hold Time	(1)	4		ns
$JTAG_6$	TCK Low to TDO Valid	(1)		16	ns
$JTAG_7$	Device Inputs Setup Time	(1)	0		ns
$JTAG_8$	Device Inputs Hold Time	(1)	3		ns
$JTAG_9$	Device Outputs Hold Time	(1)	6		ns
$JTAG_{10}$	TCK to Device Outputs Valid	(1)		18	ns

Note: 1. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF.

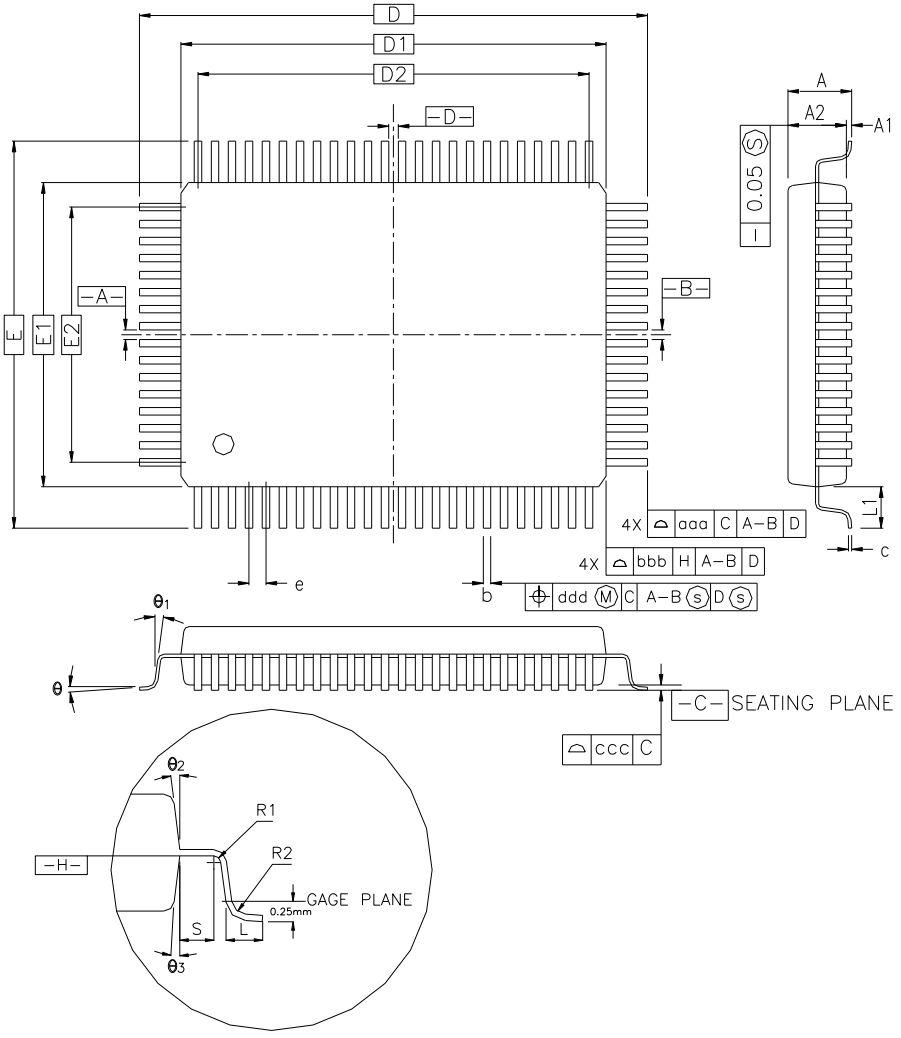
Figure 40-12. JTAG Interface Signals



41. SAM7SE512/256/32 Mechanical Characteristics

41.1 Package Drawings

Figure 41-1. LQFP128 Package Drawing



COTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A_1	0.05	—	0.15	0.002	—	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
D	22.00 BSC.			0.866 BSC.		
D_1	20.00 BSC.			0.787 BSC.		
E	16.00 BSC.			0.630 BSC.		
E_1	14.00 BSC.			0.551 BSC.		
R_2	0.08	—	0.20	0.003	—	0.008
R_1	0.08	—	—	0.003	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
θ_1	0°	—	—	0°	—	—
θ_2	11°	12°	13°	11°	12°	13°
θ_3	11°	12°	13°	11°	12°	13°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1	1.00 REF			0.039 REF		
S	0.20	—	—	0.008	—	—
b	0.17	0.20	0.27	0.007	0.008	0.011
e	0.50 BSC.			0.020 BSC.		
D2	18.50			0.728		
E2	12.50			0.492		
TOLERANCES OF FORM AND POSITION						
aaa	0.20			0.008		
bbb	0.20			0.008		
ccc	0.08			0.003		
ddd	0.08			0.003		

Table 41-1. Device and LQFP Package Maximum Weight

SAM7SE512/256/32	800	mg

Table 41-2. Package Reference

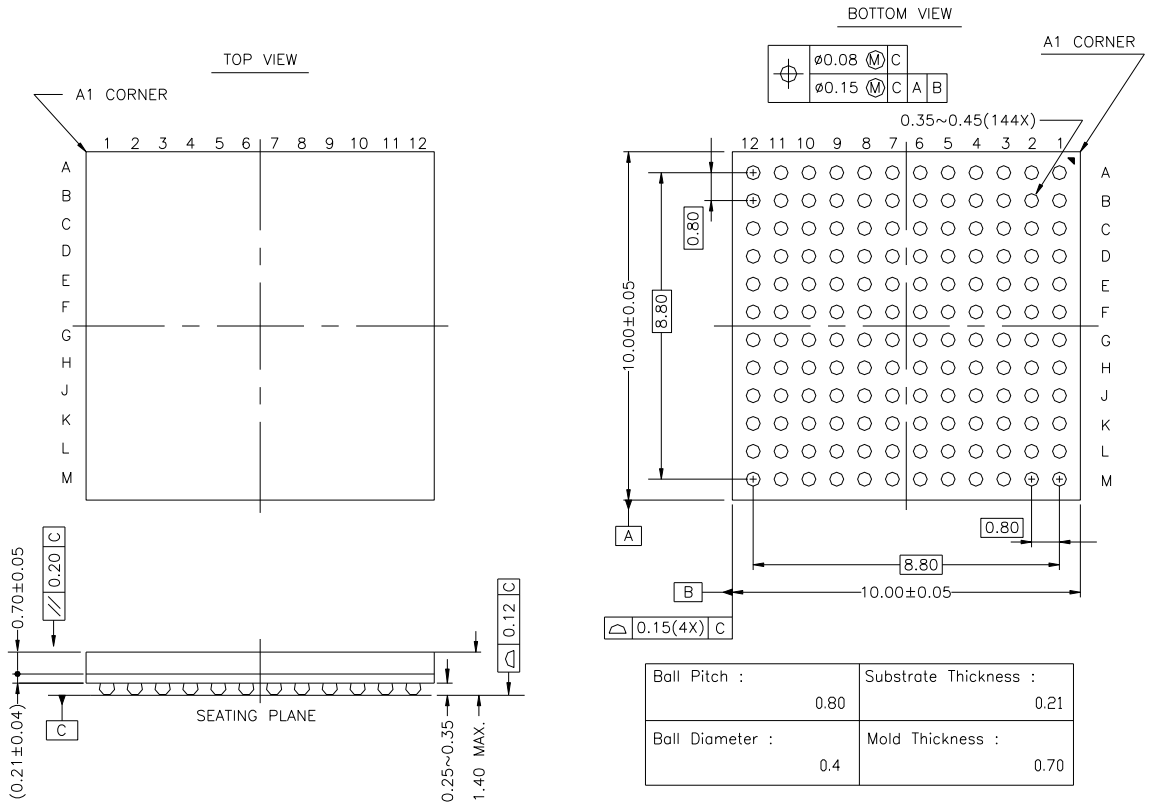
JEDEC Drawing Reference	MS-026
JESD97 Classification	e3

Table 41-3. LQFP Package Characteristics

Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Figure 41-2. 144-ball LFBGA Package Drawing



All dimensions are in mm

Table 41-4. Device and LFBGA Package Maximum Weight

SAM7SE512/256/32		mg

Table 41-5. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	e1

Table 41-6. LFBGA Package Characteristics

Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

41.2 Soldering Profile

Table 41-7 gives the recommended soldering profile from J-STD-020C.

Table 41-7. Soldering Profile

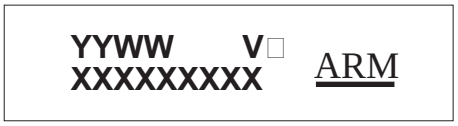
Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3· C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5· C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260· C
Ramp-down Rate	6· C/sec. max.
Time 25· C to Peak Temperature	8 min. max.

Note: The package is certified to be backward compatible with Pb/Sn soldering profile.
A maximum of three reflow passes is allowed per component.

41.3 Marking

All devices are marked with the Atmel logo and the ordering code.

Additional marking has the following format:



where

- “YY”: manufactory year
- “WW”: manufactory week
- “V”: revision
- “XXXXXXXXX”: lot number

42. SAM7SE512/256/32 Ordering Information

Table 42-1. Ordering Information

Ordering Code	MRL	Package	Package Type	Temperature Operating Range
AT91SAM7SE512B-AU	B	LQFP128	Green	Industrial (-40· C to 85· C)
AT91SAM7SE256B-AU	B	LQFP128	Green	Industrial (-40· C to 85· C)
AT91SAM7SE32B-AU	B	LQFP128	Green	Industrial (-40· C to 85· C)
AT91SAM7SE512B-CU	B	LFBGA144	Green	Industrial (-40· C to 85· C)
AT91SAM7SE256B-CU	B	LFBGA144	Green	Industrial (-40· C to 85· C)
AT91SAM7SE32B-CU	B	LFBGA144	Green	Industrial (-40· C to 85· C)
AT91SAM7SE512-AU	A	LQFP128	Green	Industrial (-40· C to 85· C)
AT91SAM7SE256-AU	A	LQFP128	Green	Industrial (-40· C to 85· C)
AT91SAM7SE32-AU	A	LQFP128	Green	Industrial (-40· C to 85· C)
AT91SAM7SE512-CU	A	LFBGA144	Green	Industrial (-40· C to 85· C)
AT91SAM7SE256-CU	A	LFBGA144	Green	Industrial (-40· C to 85· C)
AT91SAM7SE32-CU	A	LFBGA144	Green	Industrial (-40· C to 85· C)

43. SAM7SE512/256/32 Errata

43.1 Errata Summary by Product and Revision or Manufacturing Number

Table 43-1. Errata Summary Table

Part	AT91SAM7SE Product Revision or Manufacturing Number Errata	SAM7SE512/256/32 rev A	SAM7SE512/256 rev B	SAM7SE32 rev B
ADC	DRDY Bit Cleared	X	X	X
ADC	DRDY not Cleared on Disable	X		
ADC	DRDY Possibly Skipped due to CDR Read	X		
ADC	Possible Skip on DRDY when Disabling a Channel	X		
ADC	GOVRE Bit is not Updated	X	X	X
ADC	GOVRE Bit is Not Set when Reading CDR	X		
ADC	GOVRE Bit is Not Set when Disabling a Channel	X		
ADC	OVRE Flag Behavior	X	X	X
ADC	EOC Set although Channel Disabled	X		
ADC	Spurious Clear of EOC Flag	X		
ADC	Sleep Mode	X	X	X
EFC	Embedded Flash Access Time		X	
FLASH	Power consumption with data read access with multiple load of two words	X	X	X
PWM	Update when PWM_CCNTx = 0 or 1	X	X	X
PWM	Update when PWM_CPRDx = 0	X	X	X
PWM	Counter Start Value	X	X	X
PWM	Behavior of CHIDx Status Bits in the PWM_SR Register	X		
RTT	Possible Event Loss when Reading RTT_SR	X		
SDRAMC	PDC buffer in 16-bit SDRAM while the Core Accesses SDRAM	X		
SPI	Software Reset Must be Written Twice	X		
SPI	Baudrate Set to 1	X		
SPI	Bad Serial Clock Generation on 2nd Chip Select	X	X	X
SSC	Periodic Transmission Limitations in Master Mode	X		
SSC	Transmitter Limitations in Slave Mode	X	X	X
SSC	Transmitter Limitations in Slave Mode	X		
SSC	Last RK Clock Cycle when RK Outputs a Clock during Data Transfer	X	X	X

Table 43-1. Errata Summary Table (Continued)

Part	AT91SAM7SE Product Revision or Manufacturing Number Errata	SAM7SE512/256/32 rev A	SAM7SE512/256 rev B	SAM7SE32 rev B
SSC	First RK Clock Cycle when RK Outputs a Clock during Data Transfer	X	X	X
TWI	Switching from Slave to Master Mode	X		
USART	CTS in Hardware Handshaking	X		
USART	Two Characters Sent with Hardware Handshaking	X		
USART	RXBRK Flag Error in Asynchronous Mode	X		
USART	DCD is Active High instead of Low	X		

43.2 SAM7SE512/256/32 Errata - Rev. A Parts

Refer to Section 41.3 ”Marking” .

Notes:
1. AT91SAM7SE512 Revision A chip ID is 0x272A 0A40.
2. AT91SAM7SE256 Revision A chip ID is 0x272A 0940.
3. AT91SAM7SE32 Revision A chip ID is 0x2728 0340.

43.2.1 Analog-to-Digital Converter (ADC)

43.2.1.1 ADC: DRDY Bit Cleared

The DRDY Flag should be cleared only after a read of ADC_LCDR (Last Converted Data Register). A read of any ADC_CDRx register (Channel Data Register) automatically clears the DRDY flag.

Problem Fix/Workaround

None.

43.2.1.2 ADC: DRDY not Cleared on Disable

When reading LCDR at the same instant as an end of conversion, with DRDY already active, DRDY is kept active regardless of the enable status of the current channel. This sets DRDY, whereas new data is not stored.

Problem Fix/Workaround

None.

43.2.1.3 ADC: DRDY Possibly Skipped due to CDR Read

Reading CDR for channel "y" at the same instant as an end of conversion on channel "x" with EOC[x] already active, leads to skipping to set the DRDY flag if channel "x" is enabled.

Problem Fix/Workaround

Use of DRDY functionality with access to CDR registers should be avoided.

43.2.1.4 ADC: Possible Skip on DRDY when Disabling a Channel

DRDY does not rise when disabling channel "y" at the same time as an end of "x" channel conversion, although data is stored into CDRx and LCDR.

Problem Fix/Workaround

None.

43.2.1.5 ADC: GOVRE Bit is Not Updated

Read of the Status Register at the same instant as an end of conversion leads to skipping the update of the GOVRE (general overrun) flag. GOVRE is neither reset nor set.

For example, if reading the status while an end of conversion is occurring and:

1. GOVRE is active but DRDY is inactive, does not correspond to a new general overrun condition but the GOVRE flag is not reset.
2. GOVRE is inactive but DRDY is active, does correspond to a new general overrun condition but the GOVRE flag is not set.

Problem Fix/Workaround

None.

43.2.1.6 ADC: GOVRE Bit is not Set when Reading CDR

When reading CDRy (Channel Data Register y) at the same instant as an end of conversion on channel "x" with the following conditions:

- EOC[x] already active,
- DRDY already active,
- GOVRE inactive,
- previous data stored in LCDR being neither data from channel "y", nor data from channel "x".

GOVRE should be set but is not.

Problem Fix/Workaround

None.

43.2.1.7 ADC: GOVRE Bit is not Set when Disabling a Channel

When disabling channel "y" at the same instant as an end of conversion on channel "x", EOC[x] and DRDY being already active, GOVRE does not rise.

Note: OVRE[x] rises as expected.

Problem Fix/Workaround

None.

43.2.1.8 ADC: OVRE Flag Behavior

When the OVRE flag (on channel i) has been set but the related EOC status (of channel i) has been cleared (by a read of CDRi or LCDR), reading the Status register at the same instant as an end of conversion (causing the set of EOC status on channel i), does not lead to a reset of the OVRE flag (on channel i) as expected.

Problem Fix/Workaround

None.

43.2.1.9 ADC: EOC Set although Channel Disabled

If a channel is disabled while a conversion is running and if a read of CDR is performed at the same time as an end of conversion of any channel, the EOC of the channel with the conversion running may rise (whereas it has been disabled).

Problem Fix/Workaround

Do not take into account the EOC of a disabled channel

43.2.1.10 ADC: Spurious Clear of EOC Flag

If "x" and "y" are two successively converted channels and "z" is yet another enabled channel ("z" being neither "x" nor "y"), reading CDR on channel "z" at the same instant as an end of conversion on channel "y" automatically clears EOC[x] instead of EOC[z].

Problem Fix/Workaround

None.

43.2.1.11 ADC: Sleep Mode

If Sleep mode is activated while there is no activity (no conversion is being performed), it will take effect only after a conversion occurs.

Problem Fix/Workaround

To activate sleep mode as soon as possible, it is recommended to write successively, ADC Mode Register (SLEEP) then ADC Control Register (START bit field); to start an analog-to-digital conversion, put ADC into sleep mode at the end of this conversion.

43.2.2 Flash Memory

43.2.2.1 Flash: Power Consumption with data read access with multiple load of two words

When no Wait State (FWS = 0) is programmed and when data read access is performed with a multiple load of two words, the internal Flash may stay in read mode.

It implies a potential increase of power consumption on VDDCORE (around 2 mA). Note that it does not concern the program execution; thus, no issue is present when the program is fetching out of Flash.

Problem Fix/Workaround

2 workarounds are possible:

- Add one Wait State when performing these data read accesses (FWS =1)
- After the multiple load, perform a single read data access to an address different from the previous address accesses.

43.2.3 Pulse Width Modulation Controller (PWM)

43.2.3.1 PWM: Update when PWM_CCNTx = 0 or 1

If the Channel Counter Register value is 0 or 1, the Channel Period Register or Channel Duty Cycle Register is directly modified when writing the Channel Update Register.

Problem Fix/Workaround

Check the Channel Counter Register before writing the Channel Update Register.

43.2.3.2 PWM: Update when PWM_CPRDx = 0

When the Channel Period Register equals 0, the period update is not operational.

Problem Fix/Workaround

Do not write 0 in the Channel Period Register.

43.2.3.3 PWM: Counter Start Value

In left aligned mode, the first start value of the counter is 0. For the other periods, the counter starts at 1.

Problem Fix/Workaround

None.

43.2.3.4 PWM: Behavior of CHIDx Status Bits in the PWM_SR Register

There is an erratic behavior of the CHIDx status bit in the PWM_SR Register. When a channel is disabled by writing in the PWM_DIS Register just after enabling it (before completion of a Clock Period of the clock selected for the channel), the PWM line is internally disabled but the CHIDx status bit in the PWM_SR stays at 1.

Problem Fix/Workaround

Do not disable a channel before completion of one period of the selected clock.

43.2.4 Real-Time Timer (RTT)

43.2.4.1 RTT: Possible Event Loss when Reading RTT_SR

If an event (RTTINC or ALMS) occurs within the same slow clock cycle when RTT_SR is read, the corresponding bit might be cleared. This might lead to the loss of this event.

Problem Fix/Workaround

The software must handle RTT event as interrupt and should not poll RTT_SR.

43.2.5 SDRAM Controller (SDRAMC)

43.2.5.1 SDRAMC: PDC Buffer in 16-bit SDRAM while the Core Accesses SDRAM

When the SAM7SE interfaces with 16-bit SDRAM memory and the processor accesses the SDRAM, either for instruction fetch or data read/write, the data transferred by the PDC from SDRAM buffers to the peripherals might be corrupted. Transfers from peripherals to SDRAM buffers are not affected.

Problem Fix/Workaround

Map the transmit PDC buffers in internal SRAM or Flash.

43.2.6 Serial Peripheral Interface (SPI)

43.2.6.1 SPI: Baudrate Set to 1

When the Baudrate is set to 1 (so, the serial clock frequency equals the master clock), and when the BITS field (number of bits to be transmitted) in SPI_CSRx equals an odd value (in this case 9, 11, 13 or 15), an additional pulse will be generated on SPCK.

It does not occur when the BITS field is equal to 8, 10, 12, 14 or 16 and the Baudrate is equal to 1.

Problem Fix/Workaround

None.

43.2.6.2 SPI: Bad Serial Clock Generation on 2nd Chip Select

There is a bad Serial clock generation on the 2nd chip select when SCBR = 1, CPOL = 1 and NCPHA = 0.

This occurs using SPI with the following conditions:

- Master Mode
- CPOL = 1 and NCPHA = 0
- Multiple chip selects are used with one transfer with Baud rate (SCBR) equal to 1 (i.e., when serial clock frequency equals the system clock frequency); the other transfers set with SCBR are not equal to 1.
- Transmitting with the slowest chip select and then with the fastest one, then an additional pulse is generated on output SPCK during the second transfer.

Problem Fix/Workaround

Do not use a multiple Chip Select configuration where at least one SCRx register is configured with SCBR = 1 and the others differ from 1 if NCPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

43.2.6.3 SPI: Software Reset Must Be Written Twice

If a software reset (SWRST in the SPI control register) is performed, the SPI may not work properly (the clock is enabled before the chip select).

Problem Fix/Workaround

The SPI Control Register field SWRST (Software Reset) needs to be written twice to be correctly set.

43.2.7 Synchronous Serial Controller (SSC)

43.2.7.1 SSC: Periodic Transmission Limitations in Master Mode

If the Least Significant Bit is sent first (MSBF = 0), the first TAG during the frame synchro is not sent.

Problem Fix/Workaround

None.

43.2.7.2 SSC: Transmitter Limitations in Slave Mode

If TK is programmed as an output and TF is programmed as an input, it is impossible to emit data when the starting edge (rising or falling) of synchro has a Start Delay equal to zero.

Problem Fix/Workaround

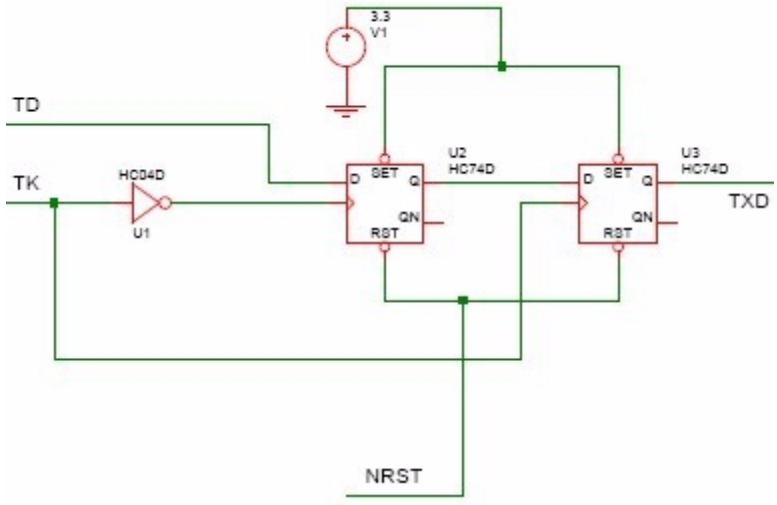
None.

43.2.7.3 SSC: Transmitter Limitations in Slave Mode

If TK is programmed as an input and TF is programmed as an output and requested to be set to low/high during data emission, the Frame Synchro signal is generated one bit clock period after the data start and one data bit is lost. This problem does not exist when generating a periodic synchro.

Problem Fix/Workaround

The data need to be delayed for one bit clock period with an external assembly. In the following schematic, TD, TK and NRST are SAM7SE signals, TXD is the delayed data to connect to the device.



43.2.7.4 SSC: Last RK Clock Cycle when RK Outputs a Clock During Data Transfer

When the SSC receiver is used with the following conditions:

- the internal clock divider is used (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0),

At the end of the data, the RK pin is set in high impedance which might be seen as an unexpected clock cycle.

Problem Fix/Workaround

Enable the pull-up on RK pin.

43.2.7.5 SSC: First RK Clock Cycle when RK Outputs a Clock During Data Transfer

When the SSC receiver is used with the following conditions:

- RX clock is divided clock (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0),

The first clock cycle time generated by the RK pin is equal to MCK/(2 x (value +1)).

Problem Fix/Workaround

None.

43.2.8 Two Wire Interface (TWI)

43.2.8.1 TWI: Switching from Slave to Master Mode

When the TWI is set in slave mode and if a master write access is performed, the start event is correctly generated but the SCL line is stuck at 1, so no transfer is possible.

Problem Fix/Workaround

Two software workarounds are possible:

1. Perform a software reset before going to master mode (TWI must be reconfigured).
2. Perform a slave read access before switching to master mode.

43.2.9 Universal Synchronous Asynchronous Receiver Transmitter (USART)

43.2.9.1 USART: CTS in Hardware Handshaking

When Hardware Handshaking is used and if CTS goes high near the end of the starting bit, a character can be lost.

CTS must not go high during a time slot occurring between 2 Master Clock periods before the starting bit and 16 Master Clock periods after the rising edge of the starting bit.

Problem Fix/Workaround

None.

43.2.9.2 USART: Two Characters Sent with Hardware Handshaking

When Hardware Handshaking is used and if CTS goes high during the TX of a character and if the holding register (US_THR) is not empty, the content of the US_THR will also be transmitted.

Problem Fix/Workaround

Do not use the PDC in transmit mode and do not fill US_THR before TXRDY is set to 1.

43.2.9.3 USART: DCD is Active High Instead of Low

DCD signal is active at “High” level in USART block (Modem Mode).

DCD should be active at “Low” level.

Problem Fix/Workaround

Add an inverter.

43.2.9.4 USART: RXBRK Flag Error in Asynchronous Mode

In Receiver mode, when 2 characters are consecutive (without a timeguard in between), the RXBRK is not taken into account. As a result, the RXBRK flag is not enabled correctly, and the frame error flag is set.

Problem Fix/Workaround

Constraints on the Transmitter device connected to the AT91 USART Receiver:

The Transmitter may use the timeguard feature, or send 2 STOP conditions. Only 1 STOP condition is taken into account by the Receiver state machine; after this STOP condition, as there is no valid data, the Receiver state machine will go in idle mode and will enable the RXBRK condition.

43.3 SAM7SE512/256 Errata - Rev. B Parts

Refer to Section 41.3 ”Marking” .

Notes: 1. AT91SAM7SE512 Revision B chip ID is 0x272A 0A41.
2. AT91SAM7SE256 Revision B chip ID is 0x272A 0941.

43.3.1 Analog-to-Digital Converter (ADC)

43.3.1.1 ADC: DRDY Bit Cleared

The DRDY Flag should be cleared only after a read of ADC_LCDR (Last Converted Data Register). A read of any ADC_CDRx register (Channel Data Register) automatically clears the DRDY flag.

Problem Fix/Workaround

None.

43.3.1.2 ADC: GOVRE Bit is Not Updated

Read of the Status Register at the same instant as an end of conversion leads to skipping the update of the GOVRE (general overrun) flag. GOVRE is neither reset nor set.

For example, if reading the status while an end of conversion is occurring and:

1. GOVRE is active but DRDY is inactive, does not correspond to a new general overrun condition but the GOVRE flag is not reset.
2. GOVRE is inactive but DRDY is active, does correspond to a new general overrun condition but the GOVRE flag is not set.

Problem Fix/Workaround

None.

43.3.1.3 ADC: OVRE Flag Behavior

When the OVRE flag (on channel i) has been set but the related EOC status (of channel i) has been cleared (by a read of CDRi or LCDR), reading the Status register at the same instant as an end of conversion (causing the set of EOC status on channel i), does not lead to a reset of the OVRE flag (on channel i) as expected.

Problem Fix/Workaround

None.

43.3.1.4 ADC: Sleep Mode

If Sleep mode is activated while there is no activity (no conversion is being performed), it will take effect only after a conversion occurs.

Problem Fix/Workaround

To activate sleep mode as soon as possible, it is recommended to write successively, ADC Mode Register (SLEEP) then ADC Control Register (START bit field); to start an analog-to-digital conversion, put ADC into sleep mode at the end of this conversion.

43.3.2 Flash Controller

43.3.2.1 EFC : Embedded Flash Access Time

The embedded Flash maximum access time is lower than expected. The tables below show the frequencies:

Table 43-2. Embedded Flash Wait State VDDCORE set at 1.65V

FWS[1]	Read Operations	Maximum Operating Frequency (MHz)
0	1 cycle	20
1	2 cycles	40
2	3 cycles	48.2
3[2]	4 cycles	48.2

Notes: 1. FWS = Flash Wait States

2. It is not necessary to use 3 wait states because the Flash can operate at maximum with only 2 wait states.

Table 43-3. Embedded Flash Wait State VDDCORE set at 1.80V

FWS[1]	Read Operations	Maximum Operating Frequency (MHz)
0	1 cycle	21.5
1	2 cycles	43
2	3 cycles	55
3[2]	4 cycles	55

Notes: 1. FWS = Flash Wait States

2. It is not necessary to use 3 wait states because the Flash can operate at maximum with only 2 wait states.

Problem Fix/Workaround

Set the number of Wait states (FWS) according to the frequency requirements described in the errata.

43.3.3 Flash Memory

43.3.3.1 Flash: Power Consumption with data read access with multiple load of two words

When no Wait State (FWS = 0) is programmed and when data read access is performed with a multiple load of two words, the internal Flash may stay in read mode.

It implies a potential increase of power consumption on VDDCORE (around 2 mA). Note that it does not concern the program execution; thus, no issue is present when the program is fetching out of Flash.

Problem Fix/Workaround

2 workarounds are possible:

- Add one Wait State when performing these data read accesses (FWS =1)
- After the multiple load, perform a single read data access to an address different from the previous address accesses.

43.3.4 Pulse Width Modulation Controller (PWM)

43.3.4.1 PWM: Update when PWM_CCNTx = 0 or 1

If the Channel Counter Register value is 0 or 1, the Channel Period Register or Channel Duty Cycle Register is directly modified when writing the Channel Update Register.

Problem Fix/Workaround

Check the Channel Counter Register before writing the Channel Update Register.

43.3.4.2 PWM: Update when PWM_CPRDx = 0

When the Channel Period Register equals 0, the period update is not operational.

Problem Fix/Workaround

Do not write 0 in the Channel Period Register.

43.3.4.3 PWM: Counter Start Value

In left aligned mode, the first start value of the counter is 0. For the other periods, the counter starts at 1.

Problem Fix/Workaround

None.

43.3.5 Serial Peripheral Interface (SPI)

43.3.5.1 SPI: Bad Serial Clock Generation on 2nd Chip Select

There is a bad Serial clock generation on the 2nd chip select when SCBR = 1, CPOL = 1 and NCPHA = 0.

This occurs using SPI with the following conditions:

- Master Mode
- CPOL = 1 and NCPHA = 0
- Multiple chip selects are used with one transfer with Baud rate (SCBR) equal to 1 (i.e., when serial clock frequency equals the system clock frequency); the other transfers set with SCBR are not equal to 1.
- Transmitting with the slowest chip select and then with the fastest one, then an additional pulse is generated on output SPCK during the second transfer.

Problem Fix/Workaround

Do not use a multiple Chip Select configuration where at least one SCRx register is configured with SCBR = 1 and the others differ from 1 if NCPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

43.3.6 Synchronous Serial Controller (SSC)

43.3.6.1 SSC: Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when the starting edge (rising or falling) of synchro has a Start Delay equal to zero.

Problem Fix/Workaround

None.

43.3.6.2 SSC: Last RK Clock Cycle when RK Outputs a Clock During Data Transfer

When the SSC receiver is used with the following conditions:

- the internal clock divider is used (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0),

At the end of the data, the RK pin is set in high impedance which might be seen as an unexpected clock cycle.

Problem Fix/Workaround

Enable the pull-up on RK pin.

43.3.6.3 SSC: First RK Clock Cycle when Rk Outputs a Clock During Data Transfer

When the SSC receiver is used with the following conditions:

- RX clock is divided clock (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0)

The first clock cycle time generated by the RK pin is equal to MCK/(2 x (value +1)).

Problem Fix/Workaround

None.

43.4 SAM7SE32 Errata - Rev. B Parts

Refer to Section 41.3 ”Marking” .

Notes: 1. AT91SAM7SE32 Revision B chip ID is 0x2728 0341.

43.4.1 Analog-to-Digital Converter (ADC)

43.4.1.1 ADC: DRDY Bit Cleared

The DRDY Flag should be cleared only after a read of ADC_LCDR (Last Converted Data Register). A read of any ADC_CDRx register (Channel Data Register) automatically clears the DRDY flag.

Problem Fix/Workaround

None.

43.4.1.2 ADC: GOVRE Bit is Not Updated

Read of the Status Register at the same instant as an end of conversion leads to skipping the update of the GOVRE (general overrun) flag. GOVRE is neither reset nor set.

For example, if reading the status while an end of conversion is occurring and:

1. GOVRE is active but DRDY is inactive, does not correspond to a new general overrun condition but the GOVRE flag is not reset.
2. GOVRE is inactive but DRDY is active, does correspond to a new general overrun condition but the GOVRE flag is not set.

Problem Fix/Workaround

None.

43.4.1.3 ADC: OVRE Flag Behavior

When the OVRE flag (on channel i) has been set but the related EOC status (of channel i) has been cleared (by a read of CDRi or LCDR), reading the Status register at the same instant as an end of conversion (causing the set of EOC status on channel i), does not lead to a reset of the OVRE flag (on channel i) as expected.

Problem Fix/Workaround

None.

43.4.1.4 ADC: Sleep Mode

If Sleep mode is activated while there is no activity (no conversion is being performed), it will take effect only after a conversion occurs.

Problem Fix/Workaround

To activate sleep mode as soon as possible, it is recommended to write successively, ADC Mode Register (SLEEP) then ADC Control Register (START bit field); to start an analog-to-digital conversion, put ADC into sleep mode at the end of this conversion.

43.4.2 Flash Memory

43.4.2.1 Flash: Power Consumption with data read access with multiple load of two words

When no Wait State (FWS = 0) is programmed and when data read access is performed with a multiple load of two words, the internal Flash may stay in read mode.

It implies a potential increase of power consumption on VDDCORE (around 2 mA). Note that it does not concern the program execution; thus, no issue is present when the program is fetching out of Flash.

Problem Fix/Workaround

2 workarounds are possible:

- Add one Wait State when performing these data read accesses (FWS =1)
- After the multiple load, perform a single read data access to an address different from the previous address accesses.

43.4.3 Pulse Width Modulation Controller (PWM)

43.4.3.1 PWM: Update when PWM_CCNTx = 0 or 1

If the Channel Counter Register value is 0 or 1, the Channel Period Register or Channel Duty Cycle Register is directly modified when writing the Channel Update Register.

Problem Fix/Workaround

Check the Channel Counter Register before writing the Channel Update Register.

43.4.3.2 PWM: Update when PWM_CPRDx = 0

When the Channel Period Register equals 0, the period update is not operational.

Problem Fix/Workaround

Do not write 0 in the Channel Period Register.

43.4.3.3 PWM: Counter Start Value

In left aligned mode, the first start value of the counter is 0. For the other periods, the counter starts at 1.

Problem Fix/Workaround

None.

43.4.4 Serial Peripheral Interface (SPI)

43.4.4.1 SPI: Bad Serial Clock Generation on 2nd Chip Select

There is a bad Serial clock generation on the 2nd chip select when SCBR = 1, CPOL = 1 and NCPHA = 0.

This occurs using SPI with the following conditions:

- Master Mode
- CPOL = 1 and NCPHA = 0
- Multiple chip selects are used with one transfer with Baud rate (SCBR) equal to 1 (i.e., when serial clock frequency equals the system clock frequency); the other transfers set with SCBR are not equal to 1.
- Transmitting with the slowest chip select and then with the fastest one, then an additional pulse is generated on output SPCK during the second transfer.

Problem Fix/Workaround

Do not use a multiple Chip Select configuration where at least one SCRx register is configured with SCBR = 1 and the others differ from 1 if NCPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

43.4.5 Synchronous Serial Controller (SSC)

43.4.5.1 SSC: Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when the starting edge (rising or falling) of synchro has a Start Delay equal to zero.

Problem Fix/Workaround

None.

43.4.5.2 SSC: Last RK Clock Cycle when RK Outputs a Clock During Data Transfer

When the SSC receiver is used with the following conditions:

- the internal clock divider is used (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0),

At the end of the data, the RK pin is set in high impedance which might be seen as an unexpected clock cycle.

Problem Fix/Workaround

Enable the pull-up on RK pin.

43.4.5.3 SSC: First RK Clock Cycle when Rk Outputs a Clock During Data Transfer

When the SSC receiver is used with the following conditions:

- RX clock is divided clock (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0)

The first clock cycle time generated by the RK pin is equal to MCK/(2 x (value +1)).

Problem Fix/Workaround

None.

44. Revision History

In the tables that follow, the most recent version of the document appears first.

Note: “rfo” indicates changes requested during document review and approval loop.

Version 6222H	Comments	Change Request Ref.
	Electrical Characteristics: Section 40.2 ”DC Characteristics”: Table 40.2, “DC Characteristics”, changed values for ‘R pull-up Resistor’	8124
	Errata: Section 43.3 ”SAM7SE512/256 Errata - Rev. B Parts”: “Flash Controller”: “EFC : Embedded Flash Access Time”, added Section 43.1 ”Errata Summary by Product and Revision or Manufacturing Number”, added column named “SAM7SE32 Rev B” Section 43.4 ”SAM7SE32 Errata - Rev. B Parts”, added	8124 8156

Version 6222G	Comments	Change Request Ref.
	‘Preliminary’ removed from 1st page, and from all headers and footers.	rfo
	ADC: Section 39.6.2 ”ADC Mode Register”, formula updated in SHTIM bitfield description.	7890
	Errata: Table 43-1, “Errata Summary Table” added. Notes added on top of Section 43.2 ”SAM7SE512/256/32 Errata - Rev. A Parts”. Section 43.5 ”SAM7SE512/256/32 Errata - Rev. B Parts” added Typos fixed within Section 43.2 ”SAM7SE512/256/32 Errata - Rev. A Parts”.	7749 rfo
	SAM7SE512/256/32 Ordering Information: MRL B Ordering Codes added to Table 42-1, “Ordering Information”	7749

Version 6222F	Comments	Change Request Ref.
	Boot ROM: SAM7SE32 user area addresses updated in Section 25.5 ”Hardware and Software Constraints”. Variables - only used in this section - changed into text (Yy, Yy_prod, Yz, Yz_prod, DRXD_PIO, DTXD_PIO).	7312 rfo
	SAM7SE512/256/32 Errata - Rev. A Parts: Section 43.2.2 ”Flash Memory” added.	7541
	‘AT91SAM’ product prefix changed to ‘SAM’ (except for Chip ID and ordering codes).	rfo

Version 6222E	Comments	Change Request Ref.
	Features: "Mode for General Purpose Two-wire UART Serial Communication" added to "Debug Unit (DBGU)". **Signal Description:** Table 3-1, "Signal Description List", AD0-AD3 and AD4-AD7 comments reversed. **System Controller:** Figure 9-1 "System Controller Block Diagram", 'periph_nreset' changed into 'power_on_reset' for RTT.	5846 5271 5222
	AT91SAM7SE512/256/32 Electrical Characteristics: Section 40.7 "ADC Characteristics", Table 40-17 and Table 40-18 edited.	6774
	AT91SAM7SE512/256/32 Errata - Rev. A Parts: Section 43.2.9.4 "USART: RXBRK Flag Error in Asynchronous Mode" description edited. Section 43.2.6.3 "SPI: Software Reset Must Be Written Twice" added. *USART: XOFF Character Bad Behavior* removed from Section 43.2.9	6626 5785 5337
	Embedded Flash Controller (EFC): Section 19.2.4.4 "General-purpose NVM Bits", bit values edited in last paragraph. Text added below Figure 19-6 "Example of Partial Page Programming:"	6236 6774
	External Bus Interface (EBI): Note (8) added to row NWR0/NWE/CFWE in Table 21-3. Note (1) added to Figure 21-6.	6774
	Memory Controller (MC): Section 18.5.2 "MC Abort Status Register", MST0, MST1, SVMST0, SVMST1 edited.	5687
	Reset Controller (RSTC): Section 13.2.4.4 "Software Reset", text added at the end of PERRST description.	5436
	USB Transceiver Characteristics: Latest Programmer Datasheet used (UDP_6083S instead of UDP_6083M).	6774

Version 6222D	Comments	Change Request Ref.
	"Two Wire Interface (TWI)" Erroneous text references to PDC functionality removed from the TWI section of the datasheet: page 353, page 355. (32.7.7 "Using The Peripheral DMA Controller (PDC)" removed from page 357), subsequent chapter numbering effected. (32.9.45 "PDC" removed from page 368), subsequent chapter numbering effected. Table 32-4, "Register Mapping", reserved offset for PDC removed Section 32.10.6 "TWI Status Register", TXBUFE, RXBUFF, ENDTX, ENDRX bit fields and descriptions removed. Section 32.10.7 "TWI Interrupt Enable Register", TXBUFE, RXBUFF, ENDTX, ENDRX bit fields and descriptions removed. Section 32.10.8 "TWI Interrupt Disable Register",TXBUFE, RXBUFF, ENDTX, ENDRX bit fields and descriptions removed. Section 32.10.9 "TWI Interrupt Mask Register",TXBUFE, RXBUFF, ENDTX, ENDRX bit fields and descriptions removed.	5187

Version 6222C	Comments	Change Request Ref.
	Overview:	
	Figure 8-1 "SAM7SE Memory Mapping", Compact Flash not shown w/EBI Chip Select 5. Compact Flash is shown with EBI Chip Select 2	4804
	Section 8.1.2.1 "Flash Overview", updated AT91SAM7SE32 ..."reads as 8192 32-bit words."	4512
	Section 6. "I/O Lines Considerations", "JTAG Port Pins","Test Pin","Reset Pin","ERASE Pin"; descriptions updated.	5062
	PMC	
	Section 29.9.10 "PMC Master Clock Register", MDIV removed from bit fields 9 and 8.	4766
	TWI	
	Important changes to this datasheet include a clarification of Atmel TWI compatibility with I2C Standard. (See Section 32.1 "Overview" and Table 32-1)	4373
	Section 32.7 "Master Mode", rewritten. New Master Read-write flowcharts, new Read-write transfer waveforms, bit field description modification etc.	
	Figure 32-2 "Application Block Diagram", updated Figure 32-5 "Master Mode Typical Application Block Diagram", updated	
	New sections; Section 32.7.4 "Master Transmitter Mode" and Section 32.7.5 "Master Receiver Mode" replace "Transmitting Data". See also: Figure 32-6, Figure 32-7, Figure 32-8, Figure 32-9 and Figure 32-10	
	Section 32.7.6 "Internal Address" added and includes, Section 32.7.6.1 "7-bit Slave Addressing" and Section 32.7.6.2 "10-bit Slave Addressing" See also: Figure 32-11, Figure 32-12 and Figure 32-13	
	Section 32.9.6 "Read Write Flowcharts", updated and new flowcharts added.	
	Fixed typo in ARBLST bit fields; "TWI Interrupt Enable Register", "TWI Interrupt Disable Register" and "TWI Interrupt Mask Register"	4584
	Inserted EOSACC bit field description in "TWI Interrupt Enable Register"	4586

Version 6222C	Comments	Change Request Ref.
	Section 40. ”SAM7SE512/256/32 Electrical Characteristics”	
	Table 40-12, “XIN Clock Electrical Characteristics” VXIN_IL, VXIN_IH updated	5007
	Table 40-2, “DC Characteristics”, junction temperature removed and	
	VDDIO DC supplies 3.3V and 1.8V defined	rfo
	Table 40-7, “Power Consumption for Different Modes”: Footnote assigned to Flash In standby mode. Footnote assigned to Ultra Low Power mode.	4657 4598
	Table 40-5, “DC Flash Characteristics SAM7SE32”, Max standby current updated.	rfo
	Section 41. ”SAM7SE512/256/32 Mechanical Characteristics”	
	LQFP-package, JESD97 Classification is e3.	4971/5007
	Thermal Considerations removed.	4657
	Section 43. ”SAM7SE512/256/32 Errata”	
	Section 43.2.1 ”Analog-to-Digital Converter (ADC)”, added to errata.	
	Section 43.2.5 ”SDRAM Controller (SDRAMC)”, added to errata.	
	Section 43.2.6.1 ”SPI: Baudrate Set to 1”, Problem Fix/Workaround = None.	
	Section 43.2.6.2 ”SPI: Bad Serial Clock Generation on 2nd Chip Select”, added to errata.	5007/4751 /4642
	Section 43.2.7.4 ”SSC: Last RK Clock Cycle when RK Outputs a Clock During Data Transfer”, added to errata.	
	Section 43.2.7.5 ”SSC: First RK Clock Cycle when RK Outputs a Clock During Data Transfer”, added to errata.	
	Section 43.2.9.3 ”USART: DCD is Active High Instead of Low”, added to errata.	
	Section 43.2.9.4 ”USART: RXBRK Flag Error in Asynchronous Mode”, added to errata.	

Version 6222B	Comments	Change Request Ref.
	Overview, Section 6.1 ”JTAG Port Pins”, Section 6.3 ”Reset Pin”, Section 6.5 ”SDCK Pin”, removed statement: “not 5V tolerant”. Section 7.6 ”SDRAM Controller” Mobile SDRAM controller added to SDRAMC description	3826
	INL and DNL updated in Section 10.14 “Analog-to-Digital Converter” on page 42	4005
	“Features” on page 2, Fully Static Operation: added up to 55 MHz at 1.8V and 85°C worst case conditions	3924
	Section 7.1 ”ARM7TDMI Processor”, Runs at up to 55 MHz, providing 0.9 MIPS/MHz (core supplied with 1.8V)	
	Section 7.8 ”Peripheral DMA Controller” PDC priority list added.	3833
	Section 7.5 ”Static Memory Controller” Multiple device adaptability: compliant w/PSRAM in synchronous operations	review
	Clock Generator, Removed information on capacitor load value in Section 28.3.1 ”Main Oscillator Connections” Figure 28-2 ”Typical Crystal Connection” on page 272, updated, CL1 and CL2 labels removed.	3282 3861
	DBGU, Debug Unit Chip ID Register, “SRAMSIZ: Internal SRAM Size” on page 320 updated w/AT91SAM7L internal RAM size and “ARCH: Architecture Identifier” on page 321 updated bin values for 0x60 and 0xF0, and added descriptions for CAP7, AT91SAM7AQxx series and CAP11	3828 3369 3807
	EBI, Table 21-3, “EBI Pins and External Static Device Connections,” on page 138, I/O[8:15] bits added in NAND Flash column, added notes to table for SDRAM, NAND FLash and references to app notes.	3742/3743 / 3852
	Figure 21-1 ”Organization of the External Bus Interface” SDCK is not multiplexed with PIO	3924
	Section 21.7.6.1 ”Hardware Configuration” A25 removed from CFRNW in CompactFlash	4044/3836
	Section 21.7.7.1 ”Hardware Configuration” A25 removed from CFRNW in CompactFlash True IDE	

Version 6222B	Comments	Change Request Ref.
	Electrical Characteristics,	
	Section 40.4.3 ”Crystal Characteristics” TCHXIN and TCHLXIN updated, TCLCH and TCHCL added to Table 40-12, “XIN Clock Electrical Characteristics” and Figure 40-2 ”XIN Clock Timing” has been added.	3966
	Section 40.7 ”ADC Characteristics” INL and DNL updated and Absolute accuracy added to Table 40-19, “Transfer Characteristics”. Reference to Data Converter Terminology added below table. INL and DNL updated in Section 10.14 “Analog-to-Digital Converter” on page 42	4005
	Section 40.8.4 ”SMC Signals”,A25 Address line changed to A22. Table 40-25 on page 632 thru Table 40-28 on page 634 and in the following two figures. Figure 40-8 ”SMC Signals in Memory Interface Mode” and Figure 40-9 ”SM Signals in LCD Interface Mode” SMC timings updated to be concordant with signals listed in Table 40-25 thru Table 40-28.	4044/3836
	Section 40.8.6 ”Embedded Flash Characteristics” updated. Note added t oTable 40-32, “Embedded Flash Wait States (VDDCORE = 1.65V)” and added Table 40-33, “Embedded Flash Wait States (VDDCORE = 1.8V)”	3924
	Table 40-20, “Master Clock Waveform Parameters”, updated w/V_{DDCORE} = 1.8V, Max = 55 MHz	
	Table 40-10, “Main Oscillator Characteristics” added schematic in footnote to C_L and C_{LEXT} symbols	3868
	Table 40-7, “Power Consumption for Different Modes” DDM and DDP pins must be left floating.	3829
	Table 40-32, “Embedded Flash Wait States (VDDCORE = 1.65V)” footnote (2) added.	review
	ECCC, Section 24.3 ”Functional Description” and Section 24.3.1 ”Write Access” and Section 24.3.2 “Read Access” on page 220 updated. Section 24.4.4 ”ECC Parity Register” and Section 24.4.5 “ECC NParity Register” on page 228 instruction updated.	3970
	ERRATA, Section 43.2.9.1 ”USART: CTS in Hardware Handshaking”, updated.....”if CTS goes high near the end of the starting bit, a character can be lost”.....	3955
	MC, Section 18.4.5 ”Memory Protection Unit”, initialization guidelines updated at end of section.	4045
	PIO, Section 34.4.5 ”Synchronous Data Output”, PIO_OWSR typo corrected. User Interface, Table 34-2, “PIO Register Mapping,” on page 446, footnotes updated on PIO_PSR, PIO_ODSR, PIO_PDSR table cells.	3289 3974
	SDRAMC, Section 23.1 “Overview” on page 199, Mobile SDRAM controller added to SDRAMC description	3826
	Figure 23-1 on page 199, SDCK signal in the Block Diagram updated.	review
	SMC, Figure 22-9, Figure 22-10, Figure 22-11, Figure 22-12, Figure 22-13 and Figure 22-25 replaced	3846
	32-bit bus removed from bit field description “BAT: Byte Access Type” on page 196	3847
	“SMC Chip Select Registers” on page 196, section restructured with table moved from the end of the section to appear in the bit field description: “NWS: Number of Wait States” on page 196. “Don’t Care” and “Number of Wait States” column added to this table and NRD Pulse Length is defined in Standard Read and Early Read Protocols.	3848/4182
	Note 1 assigned to table describing bit fields “RWSETUP: Read and Write Signal Setup Time”and “RWHOLD: Read and Write Signal Hold Time” on page 197.	
	GLOBAL All references to A25 address line changed to be A22 (23-bit address bus)	3863/3864
	Note specific to ECC Controller added to “RWHOLD: Read and Write Signal Hold Time”bit field description.	3886
	“Overview” on page 161, Address space is 64 Mbytes and the address bus is 23 bits.	review
	“External Memory Mapping” on page 163, external address bus is 23 bits.	
	Figure 22-3 on page 164, maximum address space per device is 8 Mbytes.	
	Figure 22-32 on page 183,change in values on [D15:0] line.	
	Figure 22-45, Figure 22-46 and Figure 22-47 on page 198 replaced.	

Version 6222B	Comments	Change Request Ref.
	SSC, Section 35.6.6.1 “Compare Functions” on page 474**,** updated	review
	UDP, Table 38-2, “USB Communication Flow”, Supported end point size updated for transfer interrupt Control endpoints are not effected by the “EPEDS: Endpoint Enable Disable”bit field in the USB_CSR register. write 1 updated in “RX_DATA_BK0: Receive Data Bank 0”bit field in USB_CSR register. write 0 updated in “TXPKTRDY: Transmit Packet Ready”bit field in USB_CSR register.	3476 4063 4099
	USART, In the US_MR register, typo fixed in bit field description “CLKO: Clock Output Select” on page 422 and DIV value given in bit field description “USCLKS: Clock Selection” on page 421 Section 33.5.1 “I/O Lines” on page 392, 3rd paragraph updated. In the US_CSR register the bit field description “TXEMPTY: TXEMPTY Interrupt Enable” on page 424 has been updated	3306 3763 3851 3895

Version 6222A	Comments	Change Request Ref.
	First issue: Preliminary	

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 487-2600

International

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
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
JAPAN
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com
www.atmel.com/AT91SAM

Technical Support
AT91SAM Support
Atmel technical support

Sales Contacts
www.atmel.com/contacts/

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2011 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, SAMBA®, DataFlash®and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM®, ARMPowered®logo, Cortex®, Thumb®-2 and others are registered trademarks or trademarks of ARM Ltd. Windows® and others are registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other terms and product names may be trademarks of others.