Features

- High Performance, Low Power AVR®32 UC 32-Bit Microcontroller
  - Compact Single-Cycle RISC Instruction Set Including DSP Instruction Set
  - Read-Modify-Write Instructions and Atomic Bit Manipulation
  - Performing 1.49 DMIPS/MHz
    - Up to 91 DMIPS Running at 66 MHz from Flash (1 Wait-State)
    - Up to 54 DMIPS Running at 36 MHz from Flash (0 Wait-State)
  - Memory Protection Unit
- Multi-Layer Bus System
  - High-Performance Data Transfers on Separate Buses for Increased Performance
  - 8 Peripheral DMA Channels (PDCA) Improves Speed for Peripheral Communication
  - 4 generic DMA Channels for High Bandwidth Data Paths
- Internal High-Speed Flash
  - 256 KBytes, 128 KBytes, 64 KBytes versions
  - Single-Cycle Flash Access up to 36 MHz
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 4 ms Page Programming Time and 8 ms Full-Chip Erase Time
  - 100,000 Write Cycles, 15-year Data Retention Capability
  - Flash Security Locks and User Defined Configuration Area
- Internal High-Speed SRAM
  - 64 KBytes Single-Cycle Access at Full Speed, Connected to CPU Local Bus
  - 64 KBytes on the Multi-Layer Bus System
- Interrupt Controller
  - Autovectored Low Latency Interrupt Service with Programmable Priority
- System Functions
  - Power and Clock Manager Including Internal RC Clock and One 32 KHz Oscillator
  - Two Multipurpose Oscillators and Two Phase-Lock-Loop (PLL),
  - Watchdog Timer, Real-Time Clock Timer
- External Memories
  - Support SDRAM, SRAM, NandFlash (1-bit and 4-bit ECC), Compact Flash
  - Up to 66 MHz
- External Storage device support
  - MultiMediaCard (MMC), Secure-Digital (SD), SDIO V1.1
  - CE-ATA, FastSD, SmartMedia, Compact Flash
  - Memory Stick: Standard Format V1.40, PRO Format V1.00, Micro
  - IDE Interface
- One Advanced Encryption System (AES) for AT32UC3A3256S, AT32UC3A3128S and AT32UC3A364S
  - 256-, 192-, 128-bit Key Algorithm, Compliant with FIPS PUB 197 Specifications
  - Buffer Encryption/Decryption Capabilities
- Universal Serial Bus (USB)
  - High-Speed USB (480 Mbit/s) Device/MiniHost with On-The-Go (OTG)
  - Flexible End-Point Configuration and Management with Dedicated DMA Channels
  - On-Chip Transceivers Including Pull-Ups
- One 8-channel 10-bit Analog-To-Digital Converter, multiplexed with Digital IOs.
- Two Three-Channel 16-bit Timer/Counter (TC)
- Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Independent Baudrate Generator, Support for SPI, IrDA and ISO7816 interfaces

AVR®32 32-Bit Microcontroller

AT32UC3A3256S
AT32UC3A3256
AT32UC3A3128S
AT32UC3A3128
AT32UC3A364S
AT32UC3A364

Preliminary

32072A–AVR32–03/09

  - Support for Hardware Handshaking, RS485 Interfaces and Modem Line
- Two Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
- One Synchronous Serial Protocol Controller
  - Supports I2S and Generic Frame-Based Protocols
- Two Master/Slave Two-Wire Interface (TWI), 400 kbit/s I2C-compatible
- On-Chip Debug System (JTAG interface)
  - Nexus Class 2+, Runtime Control, Non-Intrusive Data and Program Trace
- 110 General Purpose Input/Output (GPIOs)
  - Standard or High Speed mode
  - Toggle capability: up to 66 MHz
- 144-pin TBGA and LQFP
- Single 3.3 V Power Supply

1. Description

The AT32UC3A3 is a complete System-On-Chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 66MHz. AVR32 UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density and high performance.

The processor implements a Memory Protection Unit (MPU) and a fast and flexible interrupt controller for supporting modern operating systems and real-time operating systems. Higher computation capabilities are achievable using a rich set of DSP instructions.

The AT32UC3A3 incorporates on-chip Flash and SRAM memories for secure and fast access.

The Peripheral Direct Memory Access Controller (PDCA) enables data transfers between peripherals and memories without processor involvement. The PDCA drastically reduces processing overhead when transferring continuous and large data streams.

The Direct Memory Access controller (DMACA) allows high bandwidth data flows between high speed peripherals (USB, External Memories, MMC, SDIO, ...) and through high speed internal features (AES, internal memories).

The Power Manager improves design flexibility and security: the on-chip Brown-Out Detector monitors the power supply, the CPU runs from the on-chip RC oscillator or from one of external oscillator sources, a Real-Time Clock and its associated timer keeps track of the time.

The Device includes two sets of three identical 16-bit Timer/Counter (TC) channels. Each channel can be independently programmed to perform frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation. 16-bit channels are combined to operate as 32-bit channels.

The AT32UC3A3 also features many communication interfaces for communication intensive applications like UART, SPI or TWI. Additionally, a flexible Synchronous Serial Controller (SSC) and an USB are available.

The SSC provides easy access to serial communication protocols and audio standards like I2S.

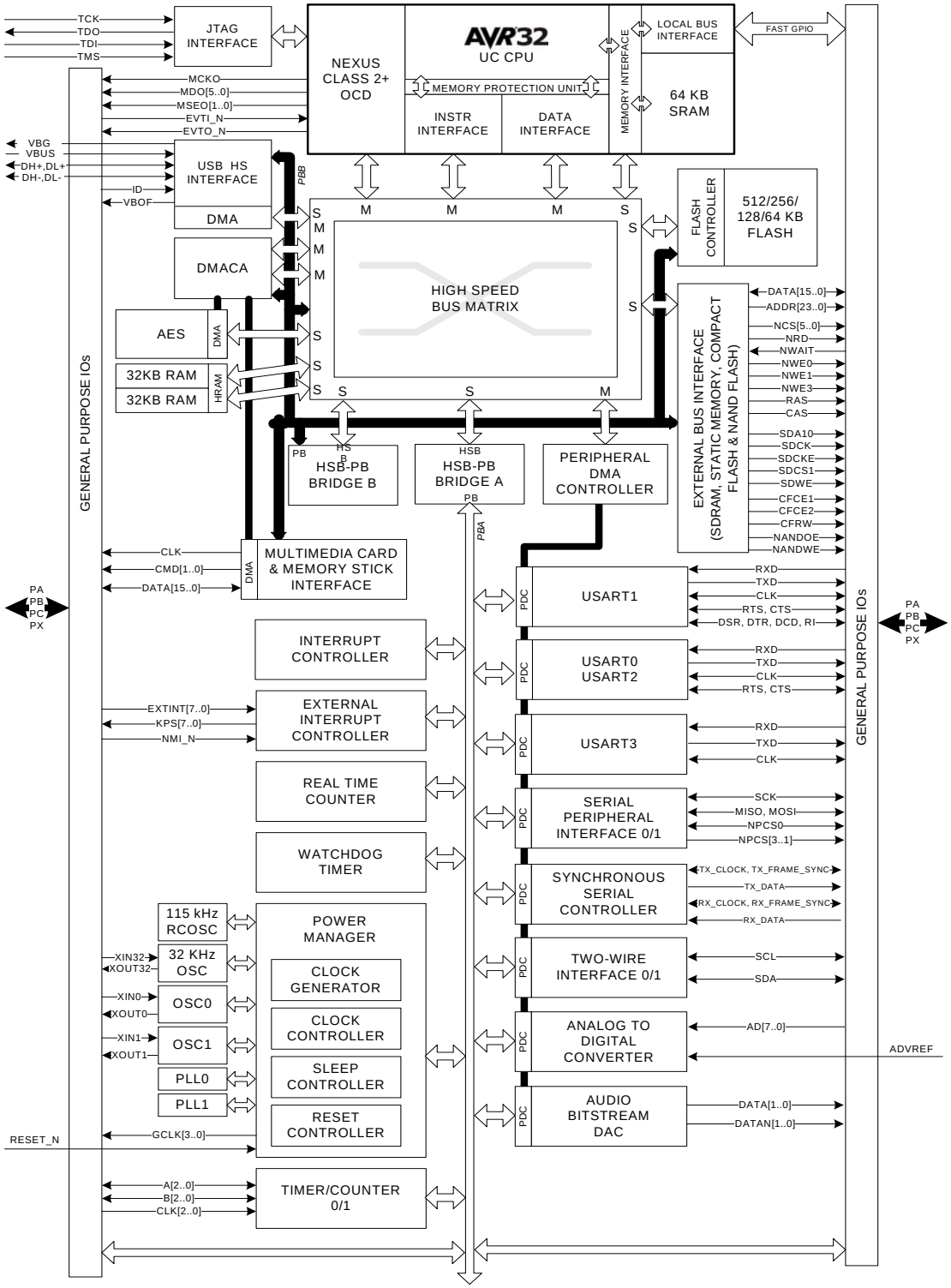
The High-Speed (480MBit/s) USB 2.0 Device interface supports several USB Classes at the same time thanks to the rich Endpoint configuration. The On-The-Go (OTG) Host interface allows device like a USB Flash disk or a USB printer to be directly connected to the processor.

AT32UC3A3 integrates a class 2+ Nexus 2.0 On-Chip Debug (OCD) System, with non-intrusive real-time trace, full-speed read/write memory access in addition to basic runtime control.

2. Overview

2.1 Block Diagram

Figure 2-1. Block Diagram



3. Configuration Summary

The table below lists all AT32UC3A3 memory and package configurations:

Table 3-1. Memory and Package Configurations

Device	Flash	SRAM	AES	Package
AT32UC3A3256S	256KB	128KB	Yes	144 balls TBGA/ 144 lead LQFP
AT32UC3A3256	256KB	128KB	No	144 balls TBGA/ 144 lead LQFP
AT32UC3A3128S	128KB	128KB	Yes	144 balls TBGA/ 144 lead LQFP
AT32UC3A3128	128KB	128KB	No	144 balls TBGA/ 144 lead LQFP
AT32UC3A364S	64KB	128KB	Yes	144 balls TBGA/ 144 lead LQFP
AT32UC3A364	64KB	128KB	No	144 balls TBGA/ 144 lead LQFP

4. Package and Pinout

4.1 Package

The device pins are multiplexed with peripheral functions as described in the Peripheral Multiplexing on I/O Line section.

Figure 4-1. TBGA144 Pinout (top view)

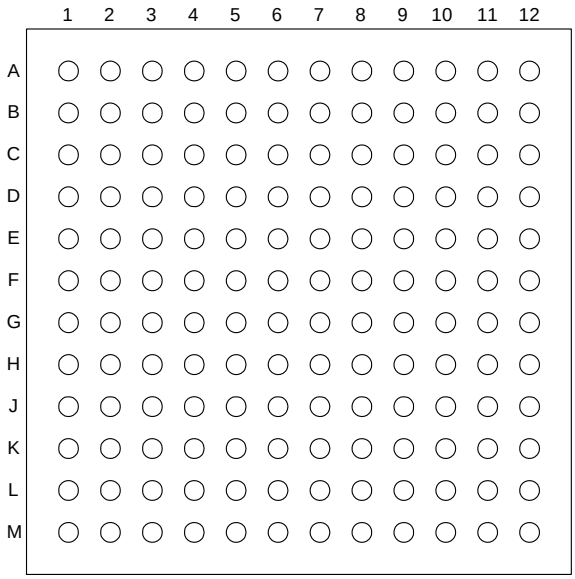


Table 4-1. BGA144 Package Pinout A1..M8

	1	2	3	4	5	6	7	8	9	10	11	12
A	PX40	PB00	PA28	PA27	PB03	PA29	PC02	PC04	PC05	DPHS	DMHS	USB_VBUS
B	PX10	PB11	PA31	PB02	VDDIO	PB04	PC03	VDDIO	USB_ VBIAS	DMFS	GNDPLL	PA09
C	PX09	PX35	GNDIO	PB01	PX16	PX13	PA30	PB08	DPFS	GNDCORE	PA08	PA10
D	PX08	PX37	PX36	PX47	PX19	PX12	PB10	PA02	PA26	PA11	PB07	PB06
E	PX38	VDDIO	PX54	PX53	VDDIO	PX15	PB09	VDDIN	PA25	PA07	VDDCORE	PA12
F	PX39	PX07	PX06	PX49	PX48	GNDIO	GNDIO	PA06	PA04	PA05	PA13	PA16
G	PX00	PX05	PX59	PX50	PX51	GNDIO	GNDIO	PA23	PA24	PA03	PA00	PA01
H	PX01	VDDIO	PX58	PX57	VDDIO	PC01	PA17	VDDIO	PA21	PA22	VDDANA	PB05
J	PX04	PX02	PX34	PX56	PX55	PA14	PA15	PA19	PA20	TMS	TDO	RESET_N
K	PX03	PX44	GNDIO	PX46	PC00	PX17	PX52	PA18	PX27	GNDIO	PX29	TCK
L	PX11	GNDIO	PX45	PX20	VDDIO	PX18	PX43	ONREG	PX26	PX28	GNDANA	TDI
M	PX22	PX41	PX42	PX14	PX21	PX23	PX24	PX25	PX32	PX31	PX30	PX33

Figure 4-2. LQFP144 Pinout

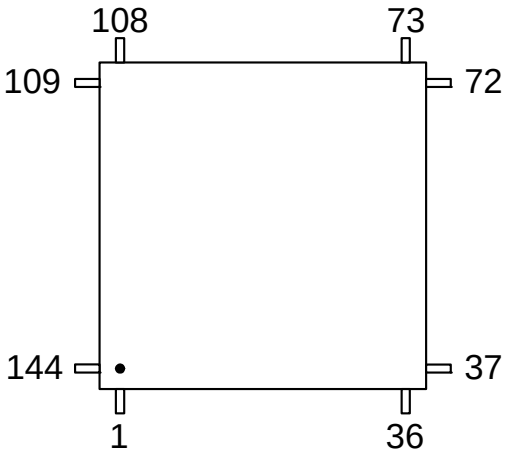


Table 4-2. Package Pinout

1	USB_VBUS	**37**	PX10	**73**	PX20	**109**	PA21
2	VDDIO	**38**	PX35	**74**	PX46	**110**	PA22
3	USB_VBIAS	**39**	PX47	**75**	PX50	**111**	PA23
4	GNDIO	**40**	PX15	**76**	PX57	**112**	PA24
5	DMHS	**41**	PX48	**77**	PX51	**113**	PA20
6	DPHS	**42**	PX53	**78**	PX56	**114**	PA19
7	GNDIO	**43**	PX49	**79**	PX55	**115**	PA18
8	DMFS	**44**	PX36	**80**	PX21	**116**	PA17
9	DPFS	**45**	PX37	**81**	VDDIO	**117**	GNDANA
10	VDDIO	**46**	PX54	**82**	GNDIO	**118**	VDDANA
11	PB08	**47**	GNDIO	**83**	PX17	**119**	PA25
12	PC05	**48**	VDDIO	**84**	PX18	**120**	PA26
13	PC04	**49**	PX09	**85**	PX23	**121**	PB05
14	PA30	**50**	PX08	**86**	PX24	**122**	PA00
15	PA02	**51**	PX38	**87**	PX52	**123**	PA01
16	PB10	**52**	PX39	**88**	PX43	**124**	PA05
17	PB09	**53**	PX06	**89**	PX27	**125**	PA03
18	PC02	**54**	PX07	**90**	PX26	**126**	PA04
19	PC03	**55**	PX00	**91**	PX28	**127**	PA06
20	GNDIO	**56**	PX59	**92**	PX25	**128**	PA16
21	VDDIO	**57**	PX58	**93**	PX32	**129**	PA13
22	PB04	**58**	PX05	**94**	PX29	**130**	VDDIO
23	PA29	**59**	PX01	**95**	PX33	**131**	GNDIO
24	PB03	**60**	PX04	**96**	PX30	**132**	PA12
25	PB02	**61**	PX34	**97**	PX31	**133**	PA07
26	PA27	**62**	PX02	**98**	PC00	**134**	PB06

Table 4-2. Package Pinout

27	PB01	**63**	PX03	**99**	PC01	**135**	PB07
28	PA28	**64**	VDDIO	**100**	PA14	**136**	PA11
29	PA31	**65**	GNDIO	**101**	PA15	**137**	PA08
30	PB00	**66**	PX44	**102**	GNDIO	**138**	PA10
31	PB11	**67**	PX11	**103**	VDDIO	**139**	PA09
32	PX16	**68**	PX14	**104**	TMS	**140**	GNDCORE
33	PX13	**69**	PX42	**105**	TDO	**141**	VDDCORE
34	PX12	**70**	PX45	**106**	RESET_N	**142**	VDDIN
35	PX19	**71**	PX41	**107**	TCK	**143**	ONREG
36	PX40	**72**	PX22	**108**	TDI	**144**	GNDPLL

4.2 Peripheral Multiplexing on I/O lines

Each GPIO line can be assigned to one of 4 peripheral functions; A, B, C, or D. The following table define how the I/O lines on the peripherals A, B, C, or D are multiplexed by the GPIO.

Table 4-3. GPIO Controller Function Multiplexing

BGA144	QFP144	PIN	GPIO Pin	Function A	Function B	Function C	Function D
G11	122	PA00	GPIO 0	USART0 - RTS	TC0 - CLK1	SPI1 - NPCS[3]	
G12	123	PA01	GPIO 1	USART0 - CTS	TC0 - A1	USART2 - RTS	
D8	15	PA02	GPIO 2	USART0 - CLK	TC0 - B1	SPI0 - NPCS[0]	
G10	125	PA03	GPIO 3	USART0 - RXD	EIC - EXTINT[4]	DAC - DATA[0]	
F9	126	PA04	GPIO 4	USART0 - TXD	EIC - EXTINT[5]	DAC - DATAN[0]	
F10	124	PA05	GPIO 5	USART1 - RXD	TC1 - CLK0	USB - USB_ID	
F8	127	PA06	GPIO 6	USART1 - TXD	TC1 - CLK1	USB - USB_VBOF	
E10	133	PA07	GPIO 7	SPI0 - NPCS[3]	DAC - DATAN[0]	USART1 - CLK	
C11	137	PA08	GPIO 8	SPI0 - SCK	DAC - DATA[0]	TC1 - B1	
B12	139	PA09	GPIO 9	SPI0 - NPCS[0]	EIC - EXTINT[6]	TC1 - A1	
C12	138	PA10	GPIO 10	SPI0 - MOSI	USB - USB_VBOF	TC1 - B0	
D10	136	PA11	GPIO 11	SPI0 - MISO	USB - USB_ID	TC1 - A2	
E12	132	PA12	GPIO 12	USART1 - CTS	SPI0 - NPCS[2]	TC1 - A0	
F11	129	PA13	GPIO 13	USART1 - RTS	SPI0 - NPCS[1]	EIC - EXTINT[7]	]
J6	100	PA14	GPIO 14	SPI0 - NPCS[1]	TWIMS0 - TWALM	TWIMS1 - TWCK	
J7	101	PA15	GPIO 15	MCI - CMD[1]	SPI1 - SCK	TWIMS1 - TWD	
F12	128	PA16	GPIO 16	MCI - DATA[11]	SPI1 - MOSI	TC1 - CLK2	
H7	116	PA17	GPIO 17	MCI - DATA[10]	SPI1 - NPCS[1]	ADC - AD[7]	
K8	115	PA18	GPIO 18	MCI - DATA[9]	SPI1 - NPCS[2]	ADC - AD[6]	
J8	114	PA19	GPIO 19	MCI - DATA[8]	SPI1 - MISO	ADC - AD[5]	

Table 4-3. GPIO Controller Function Multiplexing

J9	113	PA20	GPIO 20	EIC - EXTINT[8]	SSC - RX_FRAME_SYNC	ADC - AD[4]	
H9	109	PA21	GPIO 21	ADC - AD[0]	EIC - EXTINT[0]	USB - USB_ID	
H10	110	PA22	GPIO 22	ADC - AD[1]	EIC - EXTINT[1]	USB - USB_VBOF	
G8	111	PA23	GPIO 23	ADC - AD[2]	EIC - EXTINT[2]	DAC - DATA[1]	
G9	112	PA24	GPIO 24	ADC - AD[3]	EIC - EXTINT[3]	DAC - DATAN[1]	
E9	119	PA25	GPIO 25	TWIMS0 - TWD	TWIMS1 - TWALM	USART1 - DCD	
D9	120	PA26	GPIO 26	TWIMS0 - TWCK	USART2 - CTS	USART1 - DSR	
A4	26	PA27	GPIO 27	MCI - CLK	SSC - RX_DATA	USART3 - RTS	MSI - SCLK
A3	28	PA28	GPIO 28	MCI - CMD[0]	SSC - RX_CLOCK	USART3 - CTS	MSI - BS
A6	23	PA29	GPIO 29	MCI - DATA[0]	USART3 - TXD	TC0 - CLK0	MSI - DATA[0]
C7	14	PA30	GPIO 30	MCI - DATA[1]	USART3 - CLK	DMACA - DMAACK[0]	MSI - DATA[1]
B3	29	PA31	GPIO 31	MCI - DATA[2]	USART2 - RXD	DMACA - DMARQ[0]	MSI - DATA[2]
A2	30	PB00	GPIO 32	MCI - DATA[3]	USART2 - TXD	ADC - TRIGGER	MSI - DATA[3]
C4	27	PB01	GPIO 33	MCI - DATA[4]	DAC - DATA[1]	EIC - SCAN[0]	MSI - INS
B4	25	PB02	GPIO 34	MCI - DATA[5]	DAC - DATAN[1]	EIC - SCAN[1]	
A5	24	PB03	GPIO 35	MCI - DATA[6]	USART2 - CLK	EIC - SCAN[2]	
B6	22	PB04	GPIO 36	MCI - DATA[7]	USART3 - RXD	EIC - SCAN[3]	
H12	121	PB05	GPIO 37	USB - USB_ID	TC0 - A0	EIC - SCAN[4]	
D12	134	PB06	GPIO 38	USB - USB_VBOF	TC0 - B0	EIC - SCAN[5]	
D11	135	PB07	GPIO 39	SPI1 - SCK	SSC - TX_CLOCK	EIC - SCAN[6]	
C8	11	PB08	GPIO 40	SPI1 - MISO	SSC - TX_DATA	EIC - SCAN[7]	
E7	17	PB09	GPIO 41	SPI1 - NPCS[0]	SSC - RX_DATA	EBI - NCS[4]	
D7	16	PB10	GPIO 42	SPI1 - MOSI	SSC - RX_FRAME_SYNC	EBI - NCS[5]	
B2	31	PB11	GPIO 43	USART1 - RXD	SSC - TX_FRAME_SYNC	PM - GCLK[1]	
K5	98	PC00	GPIO 45				
H6	99	PC01	GPIO 46				
A7	18	PC02	GPIO 47				
B7	19	PC03	GPIO 48				
A8	13	PC04	GPIO 49				
A9	12	PC05	GPIO 50				
G1	55	PX00	GPIO 51	EBI - DATA[10]	USART0 - RXD	USART1 - RI	
H1	59	PX01	GPIO 52	EBI - DATA[9]	USART0 - TXD	USART1 - DTR	

Table 4-3. GPIO Controller Function Multiplexing

J2	62	PX02	GPIO 53	EBI - DATA[8]	USART0 - CTS	PM - GCLK[0]	
K1	63	PX03	GPIO 54	EBI - DATA[7]	USART0 - RTS		
J1	60	PX04	GPIO 55	EBI - DATA[6]	USART1 - RXD		
G2	58	PX05	GPIO 56	EBI - DATA[5]	USART1 - TXD		
F3	53	PX06	GPIO 57	EBI - DATA[4]	USART1 - CTS		
F2	54	PX07	GPIO 58	EBI - DATA[3]	USART1 - RTS		
D1	50	PX08	GPIO 59	EBI - DATA[2]	USART3 - RXD		
C1	49	PX09	GPIO 60	EBI - DATA[1]	USART3 - TXD		
B1	37	PX10	GPIO 61	EBI - DATA[0]	USART2 - RXD		
L1	67	PX11	GPIO 62	EBI - NWE1	USART2 - TXD		
D6	34	PX12	GPIO 63	EBI - NWE0	USART2 - CTS		
C6	33	PX13	GPIO 64	EBI - NRD	USART2 - RTS		
M4	68	PX14	GPIO 65	EBI - NCS[1]		TC0 - A0	
E6	40	PX15	GPIO 66	EBI - ADDR[19]	USART3 - RTS	TC0 - B0	
C5	32	PX16	GPIO 67	EBI - ADDR[18]	USART3 - CTS	TC0 - A1	
K6	83	PX17	GPIO 68	EBI - ADDR[17]	DMACA - DMARQ[1]	TC0 - B1	
L6	84	PX18	GPIO 69	EBI - ADDR[16]	DMACA - DMAACK[1]	TC0 - A2	
D5	35	PX19	GPIO 70	EBI - ADDR[15]	EIC - SCAN[0]	TC0 - B2	
L4	73	PX20	GPIO 71	EBI - ADDR[14]	EIC - SCAN[1]	TC0 - CLK0	
M5	80	PX21	GPIO 72	EBI - ADDR[13]	EIC - SCAN[2]	TC0 - CLK1	
M1	72	PX22	GPIO 73	EBI - ADDR[12]	EIC - SCAN[3]	TC0 - CLK2	
M6	85	PX23	GPIO 74	EBI - ADDR[11]	EIC - SCAN[4]	SSC - TX_CLOCK	
M7	86	PX24	GPIO 75	EBI - ADDR[10]	EIC - SCAN[5]	SSC - TX_DATA	
M8	92	PX25	GPIO 76	EBI - ADDR[9]	EIC - SCAN[6]	SSC - RX_DATA	
L9	90	PX26	GPIO 77	EBI - ADDR[8]	EIC - SCAN[7]	SSC - RX_FRAME_SYNC	
K9	89	PX27	GPIO 78	EBI - ADDR[7]	SPI0 - MISO	SSC - TX_FRAME_SYNC	
L10	91	PX28	GPIO 79	EBI - ADDR[6]	SPI0 - MOSI	SSC - RX_CLOCK	
K11	94	PX29	GPIO 80	EBI - ADDR[5]	SPI0 - SCK		
M11	96	PX30	GPIO 81	EBI - ADDR[4]	SPI0 - NPCS[0]		
M10	97	PX31	GPIO 82	EBI - ADDR[3]	SPI0 - NPCS[1]		
M9	93	PX32	GPIO 83	EBI - ADDR[2]	SPI0 - NPCS[2]		
M12	95	PX33	GPIO 84	EBI - ADDR[1]	SPI0 - NPCS[3]		
J3	61	PX34	GPIO 85	EBI - ADDR[0]	SPI1 - MISO	PM - GCLK[0]	
C2	38	PX35	GPIO 86	EBI - DATA[15]	SPI1 - MOSI	PM - GCLK[1]	
D3	44	PX36	GPIO 87	EBI - DATA[14]	SPI1 - SCK	PM - GCLK[2]	
D2	45	PX37	GPIO 88	EBI - DATA[13]	SPI1 - NPCS[0]	PM - GCLK[3]	

Table 4-3. GPIO Controller Function Multiplexing

E1	51	PX38	GPIO 89	EBI - DATA[12]	SPI1 - NPCS[1]	USART1 - DCD	
F1	52	PX39	GPIO 90	EBI - DATA[11]	SPI1 - NPCS[2]	USART1 - DSR	
A1	36	PX40	GPIO 91	EBI - SDCS			
M2	71	PX41	GPIO 92	EBI - CAS			
M3	69	PX42	GPIO 93	EBI - RAS			
L7	88	PX43	GPIO 94	EBI - SDA10	USART1 - RI		
K2	66	PX44	GPIO 95	EBI - SDWE	USART1 - DTR		
L3	70	PX45	GPIO 96	EBI - SDCK			
K4	74	PX46	GPIO 97	EBI - SDCKE			
D4	39	PX47	GPIO 98	EBI - NANDOE	ADC - TRIGGER	MCI - DATA[11]	
F5	41	PX48	GPIO 99	EBI - ADDR[23]	USB - USB_VBOF	MCI - DATA[10]	
F4	43	PX49	GPIO 100	EBI - CFRNW	USB - USB_ID	MCI - DATA[9]	
G4	75	PX50	GPIO 101	EBI - CFCE2	TC1 - B2	MCI - DATA[8]	
G5	77	PX51	GPIO 102	EBI - CFCE1	DMACA - DMAACK[0]	MCI - DATA[15]	
K7	87	PX52	GPIO 103	EBI - NCS[3]	DMACA - DMARQ[0]	MCI - DATA[14]	
E4	42	PX53	GPIO 104	EBI - NCS[2]		MCI - DATA[13]	
E3	46	PX54	GPIO 105	EBI - NWAIT	USART3 - TXD	MCI - DATA[12]	
J5	79	PX55	GPIO 106	EBI - ADDR[22]	EIC - SCAN[3]	USART2 - RXD	
J4	78	PX56	GPIO 107	EBI - ADDR[21]	EIC - SCAN[2]	USART2 - TXD	
H4	76	PX57	GPIO 108	EBI - ADDR[20]	EIC - SCAN[1]	USART3 - RXD	
H3	57	PX58	GPIO 109	EBI - NCS[0]	EIC - SCAN[0]	USART3 - TXD	
G3	56	PX59	GPIO 110	EBI - NANDWE		MCI - CMD[1]	

4.2.1 Oscillator Pinout

Table 4-4. Oscillator Pinout

pin	pin	Pad	Oscillator pin
A7	18	PC02	xin0
A8	13	PC04	xin1
K5	98	PC00	xin32
B7	19	PC03	xout0
A9	12	PC05	xout1
H6	99	PC01	xout32

4.3 Signal Descriptions

The following table gives details on signal name classified by peripheral.

Table 4-5. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDIO	I/O Power Supply	Power		3.0 to 3.6V
VDDANA	Analog Power Supply	Power		3.0 to 3.6V
VDDIN	Voltage Regulator Input Supply	Power		2.7 to 3.6V
ONREG	Voltage Regulator ON/OFF	Power Control	1	2.7 to 3.6 V
VDDCORE	Voltage Regulator Output for Digital Supply	Power Output		1.65 to 1.95 V
GNDANA	Analog Ground	Ground		
GNDIO	I/O Ground	Ground		
GNDCORE	DIgital Ground	Ground		
GNDPLL	PLL Ground	Ground		
Clocks, Oscillators, and PLL's				
XIN0, XIN1, XIN32	Crystal 0, 1, 32 Input	Analog		
XOUT0, XOUT1, XOUT32	Crystal 0, 1, 32 Output	Analog		
JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		
Auxiliary Port - AUX				
MCKO	Trace Data Output Clock	Output		
MDO[5:0]	Trace Data Output	Output		
MSEO[1:0]	Trace Frame Control	Output		
EVTI_N	Event In	Output	Low	
EVTO_N	Event Out	Output	Low	
Power Manager - PM				

Table 4-5. Signal Description List

Signal Name	Function	Type	Active Level	Comments
GCLK[2:0]	Generic Clock Pins	Output		
RESET_N	Reset Pin	Input	Low	
DMA Controller - DMACA (optional)				
DMAACK[1:0]	DMA Acknowledge	Output		
DMARQ[1:0]	DMA Requests	Input		
External Interrupt Module - EIM				
EXTINT[7:0]	External Interrupt Pins	Input		
KPS0 - KPS7	Keypad Scan Pins	Output		
NMI_N	Non-Maskable Interrupt Pin	Input	Low	
General Purpose Input/Output pin - GPIOA, GPIOB, GPIOC, GPIOX				
PA[31:0]	Parallel I/O Controller GPIO port A	I/O		
PB[11:0]	Parallel I/O Controller GPIO port B	I/O		
PC[5:0]	Parallel I/O Controller GPIO port C	I/O		
PX[59:0]	Parallel I/O Controller GPIO port X	I/O		
External Bus Interface - EBI				
ADDR[23:0]	Address Bus	Output		
CAS	Column Signal	Output	Low	
CFCE1	Compact Flash 1 Chip Enable	Output	Low	
CFCE2	Compact Flash 2 Chip Enable	Output	Low	
CFRNW	Compact Flash Read Not Write	Output		
DATA[15:0]	Data Bus	I/O		
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
NCS[5:0]	Chip Select	Output	Low	
NRD	Read Signal	Output	Low	
NWAIT	External Wait Signal	Input	Low	
NWE0	Write Enable 0	Output	Low	
NWE1	Write Enable 1	Output	Low	

Table 4-5. Signal Description List

Signal Name	Function	Type	Active Level	Comments
RAS	Row Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output		
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output		
SDCS	SDRAM Chip Select	Output	Low	
SDWE	SDRAM Write Enable	Output	Low	
MultiMedia Card Interface - MCI				
CLK	Multimedia Card Clock	Output		
CMD[1:0]	Multimedia Card Command	I/O		
DATA[15:0]	Multimedia Card Data	I/O		
Serial Peripheral Interface - SPI0				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
NPCS[3:0]	SPI Peripheral Chip Select	I/O	Low	
SCK	Clock	Output		
Synchronous Serial Controller - SSC				
RX_CLOCK	SSC Receive Clock	I/O		
RX_DATA	SSC Receive Data	Input		
RX_FRAME_SYNC	SSC Receive Frame Sync	I/O		
TX_CLOCK	SSC Transmit Clock	I/O		
TX_DATA	SSC Transmit Data	Output		
TX_FRAME_SYNC	SSC Transmit Frame Sync	I/O		
Timer/Counter - TC0, TC1				
A0	Channel 0 Line A	I/O		
A1	Channel 1 Line A	I/O		
A2	Channel 2 Line A	I/O		
B0	Channel 0 Line B	I/O		
B1	Channel 1 Line B	I/O		

Table 4-5. Signal Description List

Signal Name	Function	Type	Active Level	Comments
B2	Channel 2 Line B	I/O		
CLK0	Channel 0 External Clock Input	Input		
CLK1	Channel 1 External Clock Input	Input		
CLK2	Channel 2 External Clock Input	Input		
Two-wire Interface - TWI0, TWI1				
SCL	Serial Clock	I/O		
SDA	Serial Data	I/O		
Universal Synchronous Asynchronous Receiver Transmitter - USART0, USART1, USART2, USART3				
CLK	Clock	I/O		
CTS	Clear To Send	Input		
DCD	Data Carrier Detect			Only USART1
DSR	Data Set Ready			Only USART1
DTR	Data Terminal Ready			Only USART1
RI	Ring Indicator			Only USART1
RTS	Request To Send	Output		
RXD	Receive Data	Input		
RXDN	Inverted Receive Data	Input	Low	
TXD	Transmit Data	Output		
TXDN	Inverted Transmit Data	Output	Low	
Analog to Digital Converter - ADC				
AD0 - AD7	Analog input pins	Analog input		
Audio Bitstream DAC (ABDAC)				
DATA0-DATA1	D/A Data out	Output		
DATAN0-DATAN1	D/A Data inverted out	Output		
Universal Serial Bus Device - USB				
FSDM	USB Full Speed Data -	Analog		
FSDP	USB Full Speed Data +	Analog		
HSDM	USB High Speed Data -	Analog		

Table 4-5. Signal Description List

Signal Name	Function	Type	Active Level	Comments
HSDP	USB High Speed Data +	Analog		
USB_VBIAS	USB VBIAS reference	Analog		Connect to the ground through a 6810 ohms (+/- 0.5%) resistor
USB_VBUS	USB VBUS for OTG feature	Output		

4.3.1 JTAG Pins

TMS and TDI pins have pull-up resistors. TDO pin is an output, driven at up to VDDIO, and has no pull-up resistor.

4.3.2 RESET_N Pin

The RESET_N pin is a schmitt input and integrates a permanent pull-up resistor to VDDIO. As the product integrates a power-on reset cell, the RESET_N pin can be left unconnected in case no reset from the system needs to be applied to the product.

4.3.3 TWI Pins

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with inputs with spike filtering. When used as GPIO pins or used for other peripherals, the pins have the same characteristics as other GPIO pins.

4.3.4 GPIO Pins

All the I/O lines integrate a programmable pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the I/O Controller. After reset, I/O lines default as inputs with pull-up resistors disabled, except when indicated otherwise in the column “Reset State” of the I/O Controller multiplexing tables.

4.4 Power Considerations

4.4.1 Power Supplies

The AT32UC3A3 has several types of power supply pins:

- **VDDIO: Powers I/O lines. Voltage is 3.3V nominal**
- **VDDANA: Powers the ADC Voltage and provides the ADVREF voltage is 3.3V nominal**
- **VDDIN: Input voltage for the voltage regulator. Voltage is 3.3V nominal**
- **VDDCORE: Output voltage from regulator for filtering purpose and provides the supply to the core, memories, and peripherals. Voltage is 1.8V nominal**

The ground pins GNDCORE are common to VDDCORE and VDDIN. The ground pin for VDDANA is GNDANA. The ground pin for VDDIO is GNDIO.

Refer to Electrical Characteristics chapter for power consumption on the various supply pins.

4.4.2 Voltage Regulator

The AT32UC3A3 embeds a voltage regulator that converts from 3.3V to 1.8V with a load of up to 100 mA. The regulator takes its input voltage from VDDIN, and supplies the output voltage on VDDCORE and powers the core, memories and peripherals.

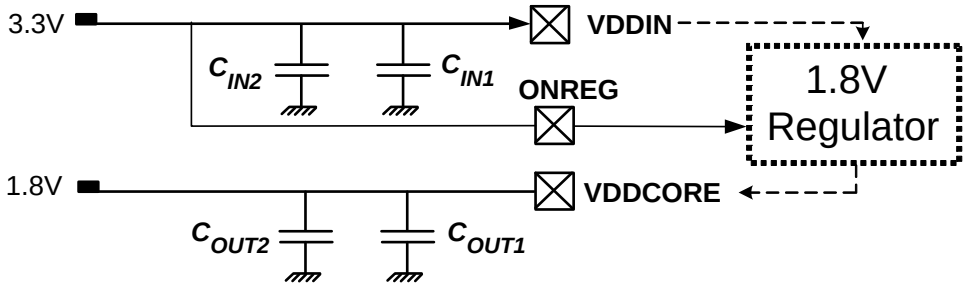
Adequate output supply decoupling is mandatory for VDDCORE to reduce ripple and avoid oscillations.

The best way to achieve this is to use two capacitors in parallel between VDDCORE and GNDCORE:

- One external 470 pF (or 1 nF) NPO capacitor (C_{OUT1}) should be connected as close to the chip as possible.
- One external 2.2 µF (or 3.3 µF) X7R capacitor (C_{OUT2}).

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, e.g., two capacitors can be used in parallel (100 nF NPO and 4.7 µF X7R).



ONREG input must be tied to VDDIN.

5. Processor and Architecture

Rev: 1.4.2.0

This chapter gives an overview of the AVR32UC CPU. AVR32UC is an implementation of the AVR32 architecture. A summary of the programming model, instruction set, and MPU is presented. For further details, see the *AVR32 Architecture Manual* and the *AVR32UC Technical Reference Manual*.

5.1 Features

- **32-bit load/store AVR32A RISC architecture**
  - **15 general-purpose 32-bit registers**
  - **32-bit Stack Pointer, Program Counter and Link Register reside in register file**
  - **Fully orthogonal instruction set**
  - **Privileged and unprivileged modes enabling efficient and secure Operating Systems**
  - **Innovative instruction set together with variable instruction length ensuring industry leading code density**
  - **DSP extention with saturating arithmetic, and a wide variety of multiply instructions**
- **3-stage pipeline allows one instruction per clock cycle for most instructions**
  - **Byte, halfword, word and double word memory access**
  - **Multiple interrupt priority levels**
- **MPU allows for operating systems with memory protection**

5.2 AVR32 Architecture

AVR32 is a new, high-performance 32-bit RISC microprocessor architecture, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption and high code density. In addition, the instruction set architecture has been tuned to allow a variety of microarchitectures, enabling the AVR32 to be implemented as low-, mid-, or high-performance processors. AVR32 extends the AVR family into the world of 32- and 64-bit applications.

Through a quantitative approach, a large set of industry recognized benchmarks has been compiled and analyzed to achieve the best code density in its class. In addition to lowering the memory requirements, a compact code size also contributes to the core's low power characteristics. The processor supports byte and halfword data types without penalty in code size and performance.

Memory load and store operations are provided for byte, halfword, word, and double word data with automatic sign- or zero extension of halfword and byte data. The C-compiler is closely linked to the architecture and is able to exploit code optimization features, both for size and speed.

In order to reduce code size to a minimum, some instructions have multiple addressing modes. As an example, instructions with immediates often have a compact format with a smaller immediate, and an extended format with a larger immediate. In this way, the compiler is able to use the format giving the smallest code size.

Another feature of the instruction set is that frequently used instructions, like add, have a compact format with two operands as well as an extended format with three operands. The larger format increases performance, allowing an addition and a data move in the same instruction in a single cycle. Load and store instructions have several different formats in order to reduce code size and speed up execution.

The register file is organized as sixteen 32-bit registers and includes the Program Counter, the Link Register, and the Stack Pointer. In addition, register R12 is designed to hold return values from function calls and is used implicitly by some instructions.

5.3 The AVR32UC CPU

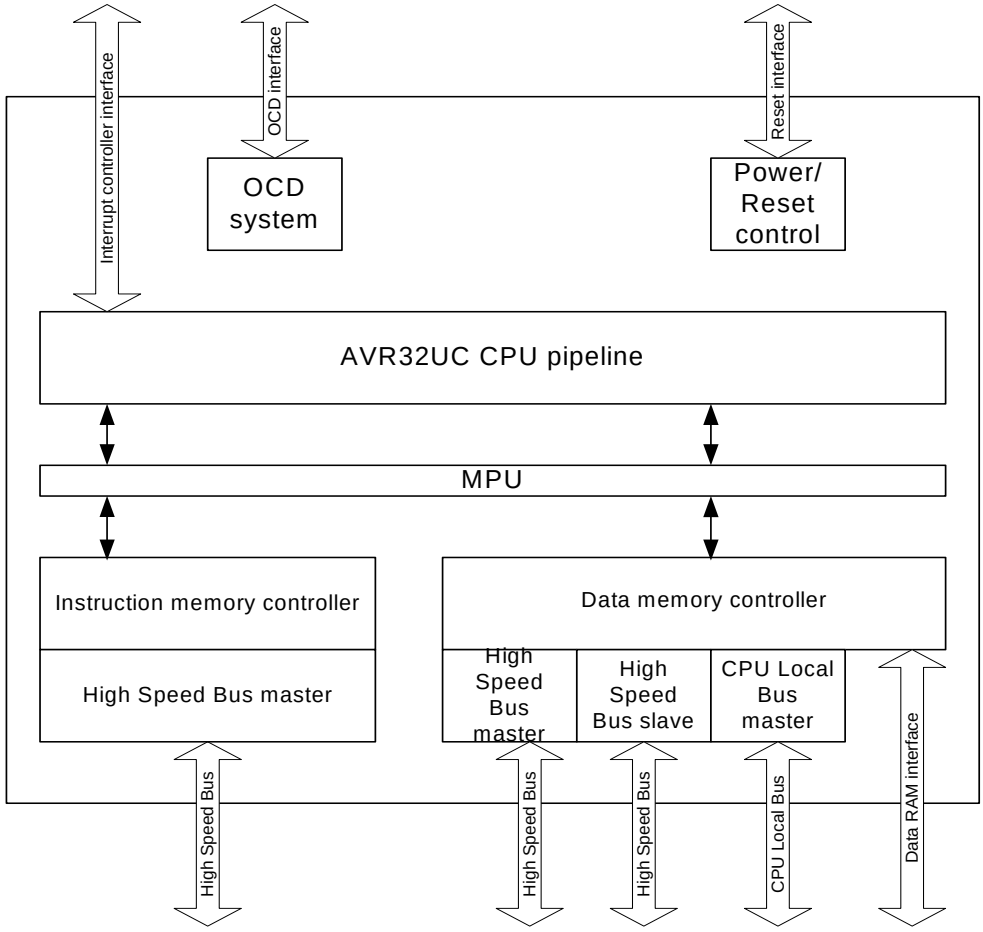
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced OCD system, no caches, and a Memory Protection Unit (MPU). Java acceleration hardware is not implemented.

AVR32UC provides three memory interfaces, one High Speed Bus master for instruction fetch, one High Speed Bus master for data access, and one High Speed Bus slave interface allowing other bus masters to access data RAMs internal to the CPU. Keeping data RAMs internal to the CPU allows fast access to the RAMs, reduces latency, and guarantees deterministic timing. Also, power consumption is reduced by not needing a full High Speed Bus access for memory accesses. A dedicated data RAM interface is provided for communicating with the internal data RAMs.

A local bus interface is provided for connecting the CPU to device-specific high-speed systems, such as floating-point units and fast GPIO ports. This local bus has to be enabled by writing the LOCEN bit in the CPUCR system register. The local bus is able to transfer data between the CPU and the local bus slave in a single clock cycle. The local bus has a dedicated memory range allocated to it, and data transfers are performed using regular load and store instructions. Details on which devices that are mapped into the local bus space is given in the Memories chapter of this data sheet.

Figure 5-1 on page 20 displays the contents of AVR32UC.

Figure 5-1. Overview of the AVR32UC CPU



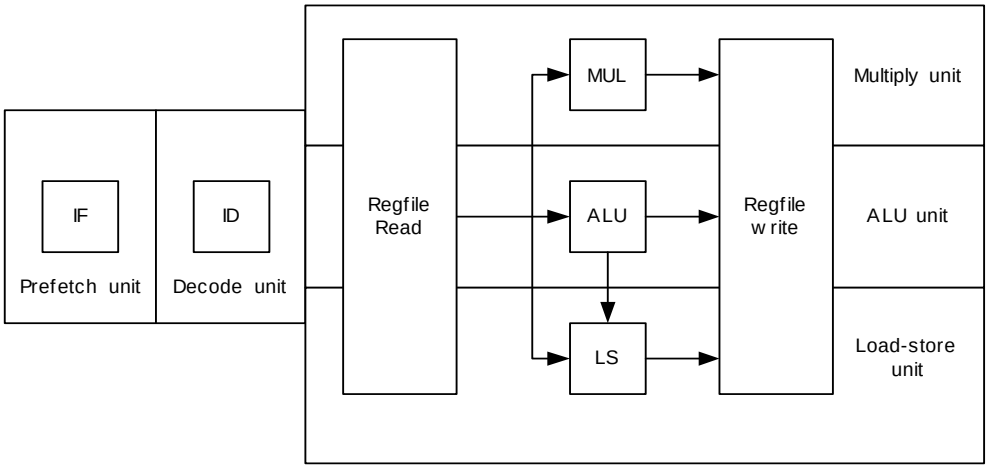
5.3.1 Pipeline Overview

AVR32UC has three pipeline stages, Instruction Fetch (IF), Instruction Decode (ID), and Instruction Execute (EX). The EX stage is split into three parallel subsections, one arithmetic/logic (ALU) section, one multiply (MUL) section, and one load/store (LS) section.

Instructions are issued and complete in order. Certain operations require several clock cycles to complete, and in this case, the instruction resides in the ID and EX stages for the required number of clock cycles. Since there is only three pipeline stages, no internal data forwarding is required, and no data dependencies can arise in the pipeline.

Figure 5-2 on page 21 shows an overview of the AVR32UC pipeline stages.

Figure 5-2. The AVR32UC Pipeline



5.3.2 AVR32A Microarchitecture Compliance

AVR32UC implements an AVR32A microarchitecture. The AVR32A microarchitecture is targeted at cost-sensitive, lower-end applications like smaller microcontrollers. This microarchitecture does not provide dedicated hardware registers for shadowing of register file registers in interrupt contexts. Additionally, it does not provide hardware registers for the return address registers and return status registers. Instead, all this information is stored on the system stack. This saves chip area at the expense of slower interrupt handling.

Upon interrupt initiation, registers R8-R12 are automatically pushed to the system stack. These registers are pushed regardless of the priority level of the pending interrupt. The return address and status register are also automatically pushed to stack. The interrupt handler can therefore use R8-R12 freely. Upon interrupt completion, the old R8-R12 registers and status register are restored, and execution continues at the return address stored popped from stack.

The stack is also used to store the status register and return address for exceptions and *scall*. Executing the *rete* or *rets* instruction at the completion of an exception or system call will pop this status register and continue execution at the popped return address.

5.3.3 Java Support

AVR32UC does not provide Java hardware acceleration.

5.3.4 Memory Protection

The MPU allows the user to check all memory accesses for privilege violations. If an access is attempted to an illegal memory address, the access is aborted and an exception is taken. The MPU in AVR32UC is specified in the AVR32UC Technical Reference manual.

5.3.5 Unaligned Reference Handling

AVR32UC does not support unaligned accesses, except for doubleword accesses. AVR32UC is able to perform word-aligned *st.d* and *ld.d*. Any other unaligned memory access will cause an address exception. Doubleword-sized accesses with word-aligned pointers will automatically be performed as two word-sized accesses.

The following table shows the instructions with support for unaligned addresses. All other instructions require aligned addresses.

Table 5-1. Instructions with Unaligned Reference Support

Instruction	Supported alignment
ld.d	Word
st.d	Word

5.3.6 Unimplemented Instructions

The following instructions are unimplemented in AVR32UC, and will cause an Unimplemented Instruction Exception if executed:

- All SIMD instructions
- All coprocessor instructions if no coprocessors are present
- retj, incjosp, popjc, pushjc
- tlbr, tlbs, tlbw
- cache

5.3.7 CPU and Architecture Revision

Three major revisions of the AVR32UC CPU currently exist. The device described in this datasheet uses CPU revision 3.

The Architecture Revision field in the CONFIG0 system register identifies which architecture revision is implemented in a specific device.

AVR32UC CPU revision 3 is fully backward-compatible with revisions 1 and 2, ie. code compiled for revision 1 or 2 is binary-compatible with revision 3 CPUs.

5.4 Programming Model

5.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 5-3. The AVR32UC Register File

Application	Supervisor	INT0	INT1	INT2	INT3	Exception	NMI	Secure
Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0
PC	PC	PC	PC	PC	PC	PC	PC	PC
LR	LR	LR	LR	LR	LR	LR	LR	LR
SP_APP	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SEC
R12	R12	R12	R12	R12	R12	R12	R12	R12
R11	R11	R11	R11	R11	R11	R11	R11	R11
R10	R10	R10	R10	R10	R10	R10	R10	R10
R9	R9	R9	R9	R9	R9	R9	R9	R9
R8	R8	R8	R8	R8	R8	R8	R8	R8
R7	R7	R7	R7	R7	R7	R7	R7	R7
R6	R6	R6	R6	R6	R6	R6	R6	R6
R5	R5	R5	R5	R5	R5	R5	R5	R5
R4	R4	R4	R4	R4	R4	R4	R4	R4
R3	R3	R3	R3	R3	R3	R3	R3	R3
R2	R2	R2	R2	R2	R2	R2	R2	R2
R1	R1	R1	R1	R1	R1	R1	R1	R1
R0	R0	R0	R0	R0	R0	R0	R0	R0
SR	SR	SR	SR	SR	SR	SR	SR	SR

SS_STATUS
SS_ADRF
SS_ADRR
SS_ADR0
SS_ADR1
SS_SP_SYS
SS_SP_APP
SS_RAR
SS_RSR

5.4.2 Status Register Configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 5-4 on page 23 and Figure 5-5 on page 24. The lower word contains the C, Z, N, V, and Q condition code flags and the R, T, and L bits, while the upper halfword contains information about the mode and state the processor executes in. Refer to the *AVR32 Architecture Manual* for details.

Figure 5-4. The Status Register High Halfword

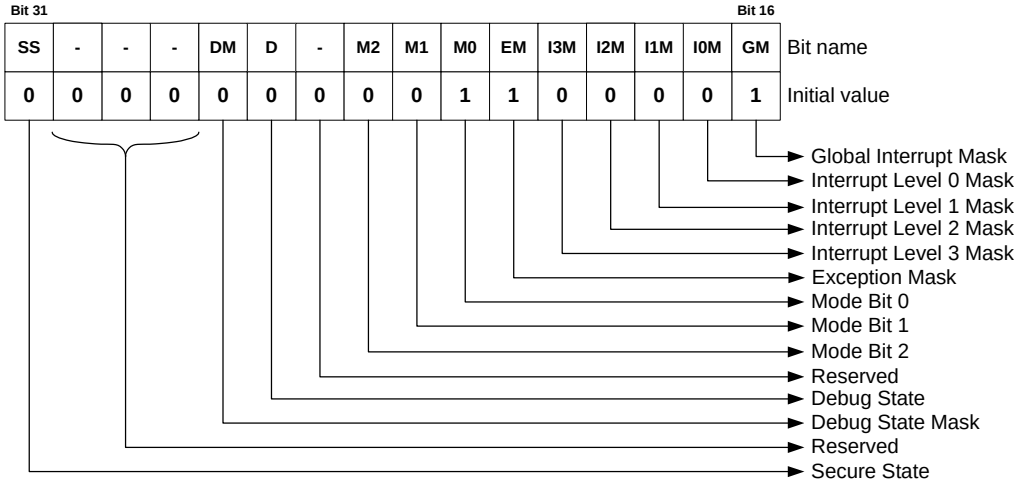
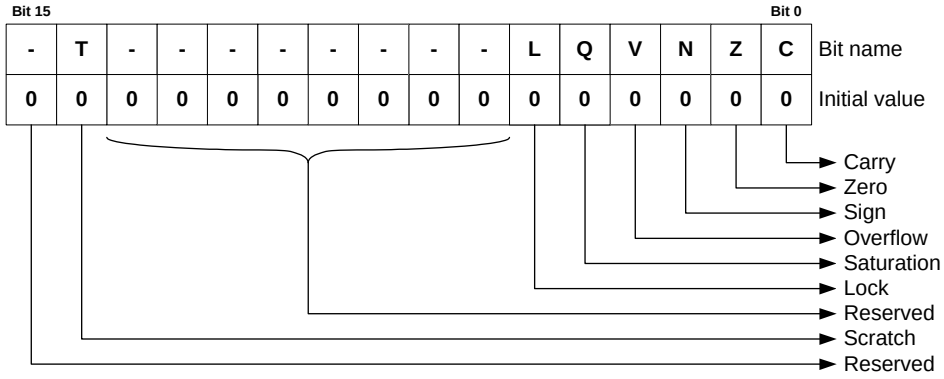


Figure 5-5. The Status Register Low Halfword



5.4.3 Processor States

5.4.3.1 Normal RISC State

The AVR32 processor supports several different execution contexts as shown in Table 5-2 on page 24.

Table 5-2. Overview of Execution Modes, their Priorities and Privilege Levels.

Priority	Mode	Security	Description
1	Non Maskable Interrupt	Privileged	Non Maskable high priority interrupt mode
2	Exception	Privileged	Execute exceptions
3	Interrupt 3	Privileged	General purpose interrupt mode
4	Interrupt 2	Privileged	General purpose interrupt mode
5	Interrupt 1	Privileged	General purpose interrupt mode
6	Interrupt 0	Privileged	General purpose interrupt mode
N/A	Supervisor	Privileged	Runs supervisor calls
N/A	Application	Unprivileged	Normal program execution mode

Mode changes can be made under software control, or can be caused by external interrupts or exception processing. A mode can be interrupted by a higher priority mode, but never by one with lower priority. Nested exceptions can be supported with a minimal software overhead.

When running an operating system on the AVR32, user processes will typically execute in the application mode. The programs executed in this mode are restricted from executing certain instructions. Furthermore, most system registers together with the upper halfword of the status register cannot be accessed. Protected memory areas are also not available. All other operating modes are privileged and are collectively called System Modes. They have full access to all privileged and unprivileged resources. After a reset, the processor will be in supervisor mode.

5.4.3.2 Debug State

The AVR32 can be set in a debug state, which allows implementation of software monitor routines that can read out and alter system information for use during application development. This implies that all system and application registers, including the status registers and program counters, are accessible in debug state. The privileged instructions are also available.

All interrupt levels are by default disabled when debug state is entered, but they can individually be switched on by the monitor routine by clearing the respective mask bit in the status register.

Debug state can be entered as described in the *AVR32UC Technical Reference Manual*.

Debug state is exited by the *retd* instruction.

5.4.4 System Registers

The system registers are placed outside of the virtual memory space, and are only accessible using the privileged *mfsr* and *mtsr* instructions. The table below lists the system registers specified in the AVR32 architecture, some of which are unused in AVR32UC. The programmer is responsible for maintaining correct sequencing of any instructions following a *mtsr* instruction. For detail on the system registers, refer to the *AVR32UC Technical Reference Manual*.

Table 5-3. System Registers

Reg #	Address	Name	Function
0	0	SR	Status Register
1	4	EVBA	Exception Vector Base Address
2	8	ACBA	Application Call Base Address
3	12	CPUCR	CPU Control Register
4	16	ECR	Exception Cause Register
5	20	RSR_SUP	Unused in AVR32UC
6	24	RSR_INT0	Unused in AVR32UC
7	28	RSR_INT1	Unused in AVR32UC
8	32	RSR_INT2	Unused in AVR32UC
9	36	RSR_INT3	Unused in AVR32UC
10	40	RSR_EX	Unused in AVR32UC
11	44	RSR_NMI	Unused in AVR32UC
12	48	RSR_DBG	Return Status Register for Debug mode
13	52	RAR_SUP	Unused in AVR32UC
14	56	RAR_INT0	Unused in AVR32UC
15	60	RAR_INT1	Unused in AVR32UC
16	64	RAR_INT2	Unused in AVR32UC
17	68	RAR_INT3	Unused in AVR32UC
18	72	RAR_EX	Unused in AVR32UC
19	76	RAR_NMI	Unused in AVR32UC
20	80	RAR_DBG	Return Address Register for Debug mode
21	84	JECR	Unused in AVR32UC
22	88	JOSP	Unused in AVR32UC
23	92	JAVA_LV0	Unused in AVR32UC
24	96	JAVA_LV1	Unused in AVR32UC
25	100	JAVA_LV2	Unused in AVR32UC

Table 5-3. System Registers (Continued)

Reg #	Address	Name	Function
26	104	JAVA_LV3	Unused in AVR32UC
27	108	JAVA_LV4	Unused in AVR32UC
28	112	JAVA_LV5	Unused in AVR32UC
29	116	JAVA_LV6	Unused in AVR32UC
30	120	JAVA_LV7	Unused in AVR32UC
31	124	JTBA	Unused in AVR32UC
32	128	JBCR	Unused in AVR32UC
33-63	132-252	Reserved	Reserved for future use
64	256	CONFIG0	Configuration register 0
65	260	CONFIG1	Configuration register 1
66	264	COUNT	Cycle Counter register
67	268	COMPARE	Compare register
68	272	TLBEHI	Unused in AVR32UC
69	276	TLBELO	Unused in AVR32UC
70	280	PTBR	Unused in AVR32UC
71	284	TLBEAR	Unused in AVR32UC
72	288	MMUCR	Unused in AVR32UC
73	292	TLBARLO	Unused in AVR32UC
74	296	TLBARHI	Unused in AVR32UC
75	300	PCCNT	Unused in AVR32UC
76	304	PCNT0	Unused in AVR32UC
77	308	PCNT1	Unused in AVR32UC
78	312	PCCR	Unused in AVR32UC
79	316	BEAR	Bus Error Address Register
80	320	MPUAR0	MPU Address Register region 0
81	324	MPUAR1	MPU Address Register region 1
82	328	MPUAR2	MPU Address Register region 2
83	332	MPUAR3	MPU Address Register region 3
84	336	MPUAR4	MPU Address Register region 4
85	340	MPUAR5	MPU Address Register region 5
86	344	MPUAR6	MPU Address Register region 6
87	348	MPUAR7	MPU Address Register region 7
88	352	MPUPSR0	MPU Privilege Select Register region 0
89	356	MPUPSR1	MPU Privilege Select Register region 1
90	360	MPUPSR2	MPU Privilege Select Register region 2
91	364	MPUPSR3	MPU Privilege Select Register region 3

Table 5-3. System Registers (Continued)

Reg #	Address	Name	Function
92	368	MPUPSR4	MPU Privilege Select Register region 4
93	372	MPUPSR5	MPU Privilege Select Register region 5
94	376	MPUPSR6	MPU Privilege Select Register region 6
95	380	MPUPSR7	MPU Privilege Select Register region 7
96	384	MPUCRA	Unused in this version of AVR32UC
97	388	MPUCRB	Unused in this version of AVR32UC
98	392	MPUBRA	Unused in this version of AVR32UC
99	396	MPUBRB	Unused in this version of AVR32UC
100	400	MPUAPRA	MPU Access Permission Register A
101	404	MPUAPRB	MPU Access Permission Register B
102	408	MPUCR	MPU Control Register
103	412	SS_STATUS	Secure State Status Register
104	416	SS_ADRF	Secure State Address Flash Register
105	420	SS_ADRR	Secure State Address RAM Register
106	424	SS_ADR0	Secure State Address 0 Register
107	428	SS_ADR1	Secure State Address 1 Register
108	432	SS_SP_SYS	Secure State Stack Pointer System Register
109	436	SS_SP_APP	Secure State Stack Pointer Application Register
110	440	SS_RAR	Secure State Return Address Register
111	444	SS_RSR	Secure State Return Status Register
112-191	448-764	Reserved	Reserved for future use
192-255	768-1020	IMPL	IMPLEMENTATION DEFINED

5.5 Exceptions and Interrupts

AVR32UC incorporates a powerful exception handling scheme. The different exception sources, like Illegal Op-code and external interrupt requests, have different priority levels, ensuring a well-defined behavior when multiple exceptions are received simultaneously. Additionally, pending exceptions of a higher priority class may preempt handling of ongoing exceptions of a lower priority class.

When an event occurs, the execution of the instruction stream is halted, and execution control is passed to an event handler at an address specified in Table 5-4 on page 30. Most of the handlers are placed sequentially in the code space starting at the address specified by EVBA, with four bytes between each handler. This gives ample space for a jump instruction to be placed there, jumping to the event routine itself. A few critical handlers have larger spacing between them, allowing the entire event routine to be placed directly at the address specified by the EVBA-relative offset generated by hardware. All external interrupt sources have autovectored interrupt service routine (ISR) addresses. This allows the interrupt controller to directly specify the ISR address as an address relative to EVBA. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes. The target address of the event handler is calculated as (EVBA | event_handler_offset), not (EVBA + event_handler_offset), so EVBA and exception

code segments must be set up appropriately. The same mechanisms are used to service all different types of events, including external interrupt requests, yielding a uniform event handling scheme.

An interrupt controller does the priority handling of the external interrupts and provides the autovector offset to the CPU.

5.5.1 System Stack Issues

Event handling in AVR32UC uses the system stack pointed to by the system stack pointer, SP_SYS, for pushing and popping R8-R12, LR, status register, and return address. Since event code may be timing-critical, SP_SYS should point to memory addresses in the IRAM section, since the timing of accesses to this memory section is both fast and deterministic.

The user must also make sure that the system stack is large enough so that any event is able to push the required registers to stack. If the system stack is full, and an event occurs, the system will enter an UNDEFINED state.

5.5.2 Exceptions and Interrupt Requests

When an event other than *scall* or debug request is received by the core, the following actions are performed atomically:

1. The pending event will not be accepted if it is masked. The I3M, I2M, I1M, I0M, EM, and GM bits in the Status Register are used to mask different events. Not all events can be masked. A few critical events (NMI, Unrecoverable Exception, TLB Multiple Hit, and Bus Error) can not be masked. When an event is accepted, hardware automatically sets the mask bits corresponding to all sources with equal or lower priority. This inhibits acceptance of other events of the same or lower priority, except for the critical events listed above. Software may choose to clear some or all of these bits after saving the necessary state if other priority schemes are desired. It is the event source's responsability to ensure that their events are left pending until accepted by the CPU.
2. When a request is accepted, the Status Register and Program Counter of the current context is stored to the system stack. If the event is an INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also automatically stored to stack. Storing the Status Register ensures that the core is returned to the previous execution mode when the current event handling is completed. When exceptions occur, both the EM and GM bits are set, and the application may manually enable nested exceptions if desired by clearing the appropriate bit. Each exception handler has a dedicated handler address, and this address uniquely identifies the exception source.
3. The Mode bits are set to reflect the priority of the accepted event, and the correct register file bank is selected. The address of the event handler, as shown in Table 5-4, is loaded into the Program Counter.

The execution of the event handler routine then continues from the effective address calculated.

The *rete* instruction signals the end of the event. When encountered, the Return Status Register and Return Address Register are popped from the system stack and restored to the Status Register and Program Counter. If the *rete* instruction returns from INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also popped from the system stack. The restored Status Register contains information allowing the core to resume operation in the previous execution mode. This concludes the event handling.

5.5.3 Supervisor Calls

The AVR32 instruction set provides a supervisor mode call instruction. The *scall* instruction is designed so that privileged routines can be called from any context. This facilitates sharing of

code between different execution modes. The *scall* mechanism is designed so that a minimal execution cycle overhead is experienced when performing supervisor routine calls from time-critical event handlers.

The *scall* instruction behaves differently depending on which mode it is called from. The behaviour is detailed in the instruction set reference. In order to allow the *scall* routine to return to the correct context, a return from supervisor call instruction, *rets*, is implemented. In the AVR32UC CPU, *scall* and *rets* uses the system stack to store the return address and the status register.

5.5.4 Debug Requests

The AVR32 architecture defines a dedicated Debug mode. When a debug request is received by the core, Debug mode is entered. Entry into Debug mode can be masked by the DM bit in the status register. Upon entry into Debug mode, hardware sets the SR[D] bit and jumps to the Debug Exception handler. By default, Debug mode executes in the exception context, but with dedicated Return Address Register and Return Status Register. These dedicated registers remove the need for storing this data to the system stack, thereby improving debuggability. The mode bits in the status register can freely be manipulated in Debug mode, to observe registers in all contexts, while retaining full privileges.

Debug mode is exited by executing the *retd* instruction. This returns to the previous context.

5.5.5 Entry Points for Events

Several different event handler entry points exists. In AVR32UC, the reset address is 0x8000_0000. This places the reset address in the boot flash memory area.

TLB miss exceptions and *scall* have a dedicated space relative to EVBA where their event handler can be placed. This speeds up execution by removing the need for a jump instruction placed at the program address jumped to by the event hardware. All other exceptions have a dedicated event routine entry point located relative to EVBA. The handler routine address identifies the exception source directly.

AVR32UC uses the ITLB and DTLB protection exceptions to signal a MPU protection violation. ITLB and DTLB miss exceptions are used to signal that an access address did not map to any of the entries in the MPU. TLB multiple hit exception indicates that an access address did map to multiple TLB entries, signalling an error.

All external interrupt requests have entry points located at an offset relative to EVBA. This autovector offset is specified by an external Interrupt Controller. The programmer must make sure that none of the autovector offsets interfere with the placement of other code. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes.

Special considerations should be made when loading EVBA with a pointer. Due to security considerations, the event handlers should be located in non-writeable flash memory, or optionally in a privileged memory protection region if an MPU is present.

If several events occur on the same instruction, they are handled in a prioritized way. The priority ordering is presented in Table 5-4. If events occur on several instructions at different locations in the pipeline, the events on the oldest instruction are always handled before any events on any younger instruction, even if the younger instruction has events of higher priority than the oldest instruction. An instruction B is younger than an instruction A if it was sent down the pipeline later than A.

The addresses and priority of simultaneous events are shown in Table 5-4. Some of the exceptions are unused in AVR32UC since it has no MMU, coprocessor interface, or floating-point unit.

Table 5-4. Priority and Handler Addresses for Events

Priority	Handler Address	Name	Event source	Stored Return Address
1	0x8000_0000	Reset	External input	Undefined
2	Provided by OCD system	OCD Stop CPU	OCD system	First non-completed instruction
3	EVBA+0x00	Unrecoverable exception	Internal	PC of offending instruction
4	EVBA+0x04	TLB multiple hit	MPU	
5	EVBA+0x08	Bus error data fetch	Data bus	First non-completed instruction
6	EVBA+0x0C	Bus error instruction fetch	Data bus	First non-completed instruction
7	EVBA+0x10	NMI	External input	First non-completed instruction
8	Autovectored	Interrupt 3 request	External input	First non-completed instruction
9	Autovectored	Interrupt 2 request	External input	First non-completed instruction
10	Autovectored	Interrupt 1 request	External input	First non-completed instruction
11	Autovectored	Interrupt 0 request	External input	First non-completed instruction
12	EVBA+0x14	Instruction Address	CPU	PC of offending instruction
13	EVBA+0x50	ITLB Miss	MPU	
14	EVBA+0x18	ITLB Protection	MPU	PC of offending instruction
15	EVBA+0x1C	Breakpoint	OCD system	First non-completed instruction
16	EVBA+0x20	Illegal Opcode	Instruction	PC of offending instruction
17	EVBA+0x24	Unimplemented instruction	Instruction	PC of offending instruction
18	EVBA+0x28	Privilege violation	Instruction	PC of offending instruction
19	EVBA+0x2C	Floating-point	UNUSED	
20	EVBA+0x30	Coprocessor absent	Instruction	PC of offending instruction
21	EVBA+0x100	Supervisor call	Instruction	PC(Supervisor Call) +2
22	EVBA+0x34	Data Address (Read)	CPU	PC of offending instruction
23	EVBA+0x38	Data Address (Write)	CPU	PC of offending instruction
24	EVBA+0x60	DTLB Miss (Read)	MPU	
25	EVBA+0x70	DTLB Miss (Write)	MPU	
26	EVBA+0x3C	DTLB Protection (Read)	MPU	PC of offending instruction
27	EVBA+0x40	DTLB Protection (Write)	MPU	PC of offending instruction
28	EVBA+0x44	DTLB Modified	UNUSED	

6. Memories

6.1 Embedded Memories

- **Internal High-Speed Flash**
  - **256KBytes (AT32UC3A3256/S)**
  - **128Kbytes (AT32UC3A3128/S)**
  - **64Kbytes (AT32UC3A364/S)**
    - **0 wait state access at up to 36MHz in worst case conditions**
    - **1 wait state access at up to 66MHz in <orst case conditions**
    - **Pipelined Flash architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access**
    - **Pipelined Flash architecture typically reduces the cycle penalty of 1 wait state operation to only 15% compared to 0 wait state operation**
    - **100 000 write cycles, 15-year data retention capability**
    - **Sector lock capabilities, Bootloader protection, Security Bit**
    - **32 fuses, preserved during Chip Erase**
    - **User page for data to be preserved during Chip Erase**
- **Internal High-Speed SRAM**
  - **64KBytes, Single-cycle access at full speed on CPU Local Bus and accessible through the High Speed Bud (HSB) matrix**
  - **2x32KBytes, accessible independently through the High Speed Bud (HSB) matrix**

6.2 Physical Memory Map

The System Bus is implemented as a bus matrix. All system bus addresses are fixed, and they are never remapped in any way, not even in boot.

Note that AVR32UC CPU uses unsegmented translation, as described in the AVR32 Architecture Manual.

The 32-bit physical address space is mapped as follows:

Table 6-1. AT32UC3A3 Physical Memory Map

Device	Start Address	Size	Size	Size
		AT32UC3A3256	AT32UC3A3128	AT32UC3A364
Embedded CPU SRAM	0x00000000	64KByte	64KByte	64KByte
Embedded Flash	0x80000000	256KByte	128KByte	64KByte
EBI SRAM CS0	0xC0000000	16MByte	16MByte	16MByte
EBI SRAM CS2	0xC8000000	16MByte	16MByte	16MByte
EBI SRAM CS3	0xCC000000	16MByte	16MByte	16MByte
EBI SRAM CS4	0xD8000000	16MByte	16MByte	16MByte
EBI SRAM CS5	0xDC000000	16MByte	16MByte	16MByte
EBI SRAM CS1 /SDRAM CS0	0xD0000000	128MByte	128MByte	128MByte
USB Data	0xE0000000	64KByte	64KByte	64KByte
Embedded System SRAM 0	0xFF000000	32KByte	32KByte	32KByte

Table 6-1. AT32UC3A3 Physical Memory Map

Device	Start Address	Size AT32UC3A3256	Size AT32UC3A3128	Size AT32UC3A364
Embedded System SRAM 1	0xFF008000	32KByte	32KByte	32KByte
HSB-PB Bridge A	0xFFFF0000	64KByte	64KByte	64KByte
HSB-PB Bridge B	0xFFFE0000	64KByte	64KByte	64KByte

6.3 Peripheral Address Map

Table 6-2. Peripheral Address Mapping

Address		Peripheral Name	Bus
0xFF100000	DMACA	DMA Controller - DMACA	
0xFF200000	RESERVED		
0xFFFD0000	AES	Advanced Encryption Standard - AES	
0xFFFE0000	USB	USB 2.0 OTG Interface - USB	
0xFFFE1000	HMATRIX	HSB Matrix - HMATRIX	
0xFFFE1400	FLASHC	Flash Controller - FLASHC	
0xFFFE1C00	SMC	Static Memory Controller - SMC	
0xFFFE2000	SDRAMC	SDRAM Controller - SDRAMC	
0xFFFE2400	ECCHRS	Error code corrector Hamming and Reed Solomon - ECCHRS	
0xFFFE2800	BUSMON	Bus Monitor module - BUSMON	
0xFFFE4000	MCI	Mulitmedia Card Interface - MCI	
0xFFFE8000	MSI	Memory Stick Interface - MSI	
0xFFFF0000	PDMA	Peripheral DMA Controller - PDMA	
0xFFFF0800	INTC	Interrupt controller - INTC	

Table 6-2. Peripheral Address Mapping

0xFFFF0C00	PM	Power Manager - PM
0xFFFF0D00	RTC	Real Time Counter - RTC
0xFFFF0D30	WDT	Watchdog Timer - WDT
0xFFFF0D80	EIC	External Interrupt Controller - EIC
0xFFFF1000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF1400	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0
0xFFFF1800	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1
0xFFFF1C00	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2
0xFFFF2000	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter - USART3
0xFFFF2400	SPI0	Serial Peripheral Interface - SPI0
0xFFFF2800	SPI1	Serial Peripheral Interface - SPI1
0xFFFF2C00	TWIM0	Two-wire Master Interface - TWIM0
0xFFFF3000	TWIM1	Two-wire Master Interface - TWIM1
0xFFFF3400	SSC	Synchronous Serial Controller - SSC
0xFFFF3800	TC0	Timer/Counter - TC0
0xFFFF3C00	ADC	Analog to Digital Converter - ADC
0xFFFF4000	DAC	Audio Bitstream DAC - DAC
0xFFFF4400	TC1	Timer/Counter - TC1
0xFFFF4800	RESERVED	

Table 6-2. Peripheral Address Mapping

0xFFFF4c00	RESERVED	
0xFFFF5000	TWIS0	Two-wire Slave Interface - TWIS0
0xFFFF5400	TWIS1	Two-wire Slave Interface - TWIS1

6.4 CPU Local Bus Mapping

Some of the registers in the GPIO module are mapped onto the CPU local bus, in addition to being mapped on the Peripheral Bus. These registers can therefore be reached both by accesses on the Peripheral Bus, and by accesses on the local bus.

Mapping these registers on the local bus allows cycle-deterministic toggling of GPIO pins since the CPU and GPIO are the only modules connected to this bus. Also, since the local bus runs at CPU speed, one write or read operation can be performed per clock cycle to the local bus-mapped GPIO registers.

The following GPIO registers are mapped on the local bus:

Table 6-3. Local Bus Mapped GPIO Registers

Port	Register	Mode	Local Bus Address	Access
A	Output Driver Enable Register (ODER)	WRITE	0x40000040	Write-only
		SET	0x40000044	Write-only
		CLEAR	0x40000048	Write-only
		TOGGLE	0x4000004C	Write-only
	Output Value Register (OVR)	WRITE	0x40000050	Write-only
		SET	0x40000054	Write-only
		CLEAR	0x40000058	Write-only
		TOGGLE	0x4000005C	Write-only
	Pin Value Register (PVR)	-	0x40000060	Read-only
B	Output Driver Enable Register (ODER)	WRITE	0x40000240	Write-only
		SET	0x40000244	Write-only
		CLEAR	0x40000248	Write-only
		TOGGLE	0x4000024C	Write-only
	Output Value Register (OVR)	WRITE	0x40000250	Write-only
		SET	0x40000254	Write-only
		CLEAR	0x40000258	Write-only
		TOGGLE	0x4000025C	Write-only
	Pin Value Register (PVR)	-	0x40000260	Read-only

7. Boot Sequence

This chapter summarizes the boot sequence of the AT32UC3A3. The behavior after power-up is controlled by the Power Manager. For specific details, refer to Section 8. ”Power Manager (PM)” on page 36.

7.1 Starting of Clocks

After power-up, the device will be held in a reset state by the Power-On Reset circuitry, until the power has stabilized throughout the device. Once the power has stabilized, the device will use the internal RC Oscillator as clock source.

On system start-up, the PLLs are disabled. All clocks to all modules are running. No clocks have a divided frequency, all parts of the system receives a clock with the same frequency as the internal RC Oscillator.

7.2 Fetching of Initial Instructions

After reset has been released, the AVR32 UC CPU starts fetching instructions from the reset address, which is 0x8000_0000. This address points to the first address in the internal Flash.

The code read from the internal Flash is free to configure the system to use for example the PLLs, to divide the frequency of the clock routed to some of the peripherals, and to gate the clocks to unused peripherals.

8. Power Manager (PM)

Rev: 2.3.1.0

8.1 Features

- **Controls integrated oscillators and PLLs**
- **Generates clocks and resets for digital logic**
- **Supports 2 crystal oscillators 4MHZ-16MHz**
- **Supports 2 PLLs 48-150MHz**
- **Supports 32KHz ultra-low power oscillator**
- **Integrated low-power RC oscillator**
- **On-the fly frequency change of CPU, HSB, PBA, and PBB clocks**
- **Sleep modes allow simple disabling of logic clocks, PLLs, and oscillators**
- **Module-level clock gating through maskable peripheral clocks**
- **Wake-up from internal or external interrupts**
- **Generic clocks with wide frequency range provided**
- **Automatic identification of reset sources**
- **Controls brownout detector (BOD), RC oscillator, and bandgap voltage reference through control and calibration registers**

8.2 Overview

The Power Manager (PM) controls the oscillators and PLLs, and generates the clocks and resets in the device. The PM controls two fast crystal oscillators, as well as two PLLs, which can multiply the clock from either oscillator to provide higher frequencies. Additionally, a low-power 32KHz oscillator is used to generate the real-time counter clock for high accuracy real-time measurements. The PM also contains a low-power RC oscillator with fast start-up time, which can be used to clock the digital logic.

The provided clocks are divided into synchronous and generic clocks. The synchronous clocks are used to clock the main digital logic in the device, namely the CPU, and the modules and peripherals connected to the HSB, PBA, and PBB buses. The generic clocks are asynchronous clocks, which can be tuned precisely within a wide frequency range, which makes them suitable for peripherals that require specific frequencies, such as timers and communication modules.

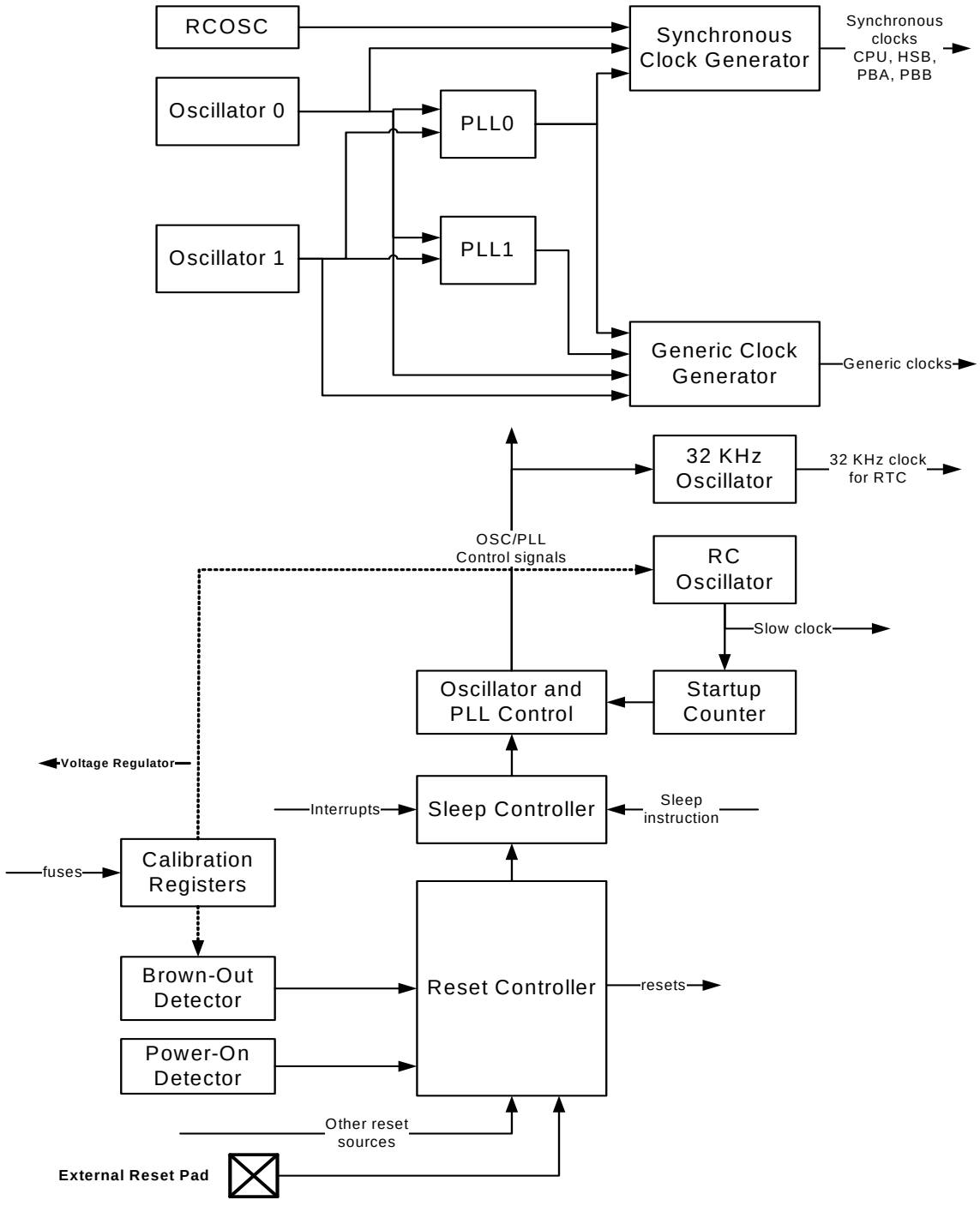
The PM also contains advanced power-saving features, allowing the user to optimize the power consumption for an application. The synchronous clocks are divided into three clock domains, one for the CPU and HSB, one for modules on the PBA bus, and one for modules on the PBB bus.The three clocks can run at different speeds, so the user can save power by running peripherals at a relatively low clock, while maintaining a high CPU performance. Additionally, the clocks can be independently changed on-the-fly, without halting any peripherals. This enables the user to adjust the speed of the CPU and memories to the dynamic load of the application, without disturbing or re-configuring active peripherals.

Each module also has a separate clock, enabling the user to switch off the clock for inactive modules, to save further power. Additionally, clocks and oscillators can be automatically switched off during idle periods by using the sleep instruction on the CPU. The system will return to normal on occurrence of interrupts.

The Power Manager also contains a Reset Controller, which collects all possible reset sources, generates hard and soft resets, and allows the reset source to be identified by software.

8.3 Block Diagram

Figure 8-1. Power Manager Block Diagram



8.4 Product Dependencies

8.4.1 I/O Lines

The PM provides a number of generic clock outputs, which can be connected to output pins, multiplexed with GPIO lines. The programmer must first program the GPIO controller to assign these pins to their peripheral function. If the I/O pins of the PM are not used by the application, they can be used for other purposes by the GPIO controller.

8.4.2 Interrupt

The PM interrupt line is connected to one of the internal sources of the interrupt controller. Using the PM interrupt requires the interrupt controller to be programmed first.

8.5 Functional Description

8.5.1 Slow Clock

The slow clock is generated from an internal RC oscillator which is always running, except in Static mode. The slow clock can be used for the main clock in the device, as described in Section 8.5.5. The slow clock is also used for the Watchdog Timer and measuring various delays in the Power Manager.

The RC oscillator has a 3 cycles startup time, and is always available when the CPU is running. The RC oscillator operates at approximately 115 kHz, and can be calibrated to a narrow range by the RCOSCCAL fuses. Software can also change RC oscillator calibration through the use of the RCCR register. Please see the Electrical Characteristics section for details.

RC oscillator can also be used as the RTC clock when crystal accuracy is not required.

8.5.2 Oscillator 0 and 1 Operation

The two main oscillators are designed to be used with an external 450 kHz to 16 MHz crystal and two biasing capacitors, as shown in Figure 8-2 on page 39. Oscillator 0 can be used for the main clock in the device, as described in Section 8.5.5. Both oscillators can be used as source for the generic clocks, as described in Section 8.5.8.

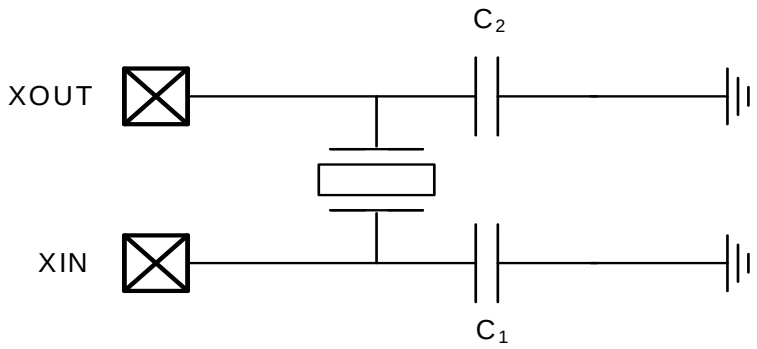
The oscillators are disabled by default after reset. When the oscillators are disabled, the XIN and XOUT pins can be used as general purpose I/Os. When the oscillators are configured to use an external clock, the clock must be applied to the XIN pin while the XOUT pin can be used as a general purpose I/O.

The oscillators can be enabled by writing to the OSCnEN bits in MCCTRL. Operation mode (external clock or crystal) is chosen by writing to the MODE field in OSCCTRLn. Oscillators are automatically switched off in certain sleep modes to reduce power consumption, as described in Section 8.5.7.

After a hard reset, or when waking up from a sleep mode that disabled the oscillators, the oscillators may need a certain amount of time to stabilize on the correct frequency. This start-up time can be set in the OSCCTRLn register.

The PM masks the oscillator outputs during the start-up time, to ensure that no unstable clocks propagate to the digital logic. The OSCnRDY bits in POSCSR are automatically set and cleared according to the status of the oscillators. A zero to one transition on these bits can also be configured to generate an interrupt, as described in Section 8.6.7.

Figure 8-2. Oscillator Connections



8.5.3 32 KHz Oscillator Operation

The 32 KHz oscillator operates as described for Oscillator 0 and 1 above. The 32 KHz oscillator is used as source clock for the Real-Time Counter.

The oscillator is disabled by default, but can be enabled by writing OSC32EN in OSCCTRL32. The oscillator is an ultra-low power design and remains enabled in all sleep modes except Static mode.

While the 32 KHz oscillator is disabled, the XIN32 and XOUT32 pins are available as general purpose I/Os. When the oscillator is configured to work with an external clock (MODE field in OSCCTRL32 register), the external clock must be connected to XIN32 while the XOUT32 pin can be used as a general purpose I/O.

The startup time of the 32 KHz oscillator can be set in the OSCCTRL32, after which OSC32RDY in POSCSR is set. An interrupt can be generated on a zero to one transition of OSC32RDY.

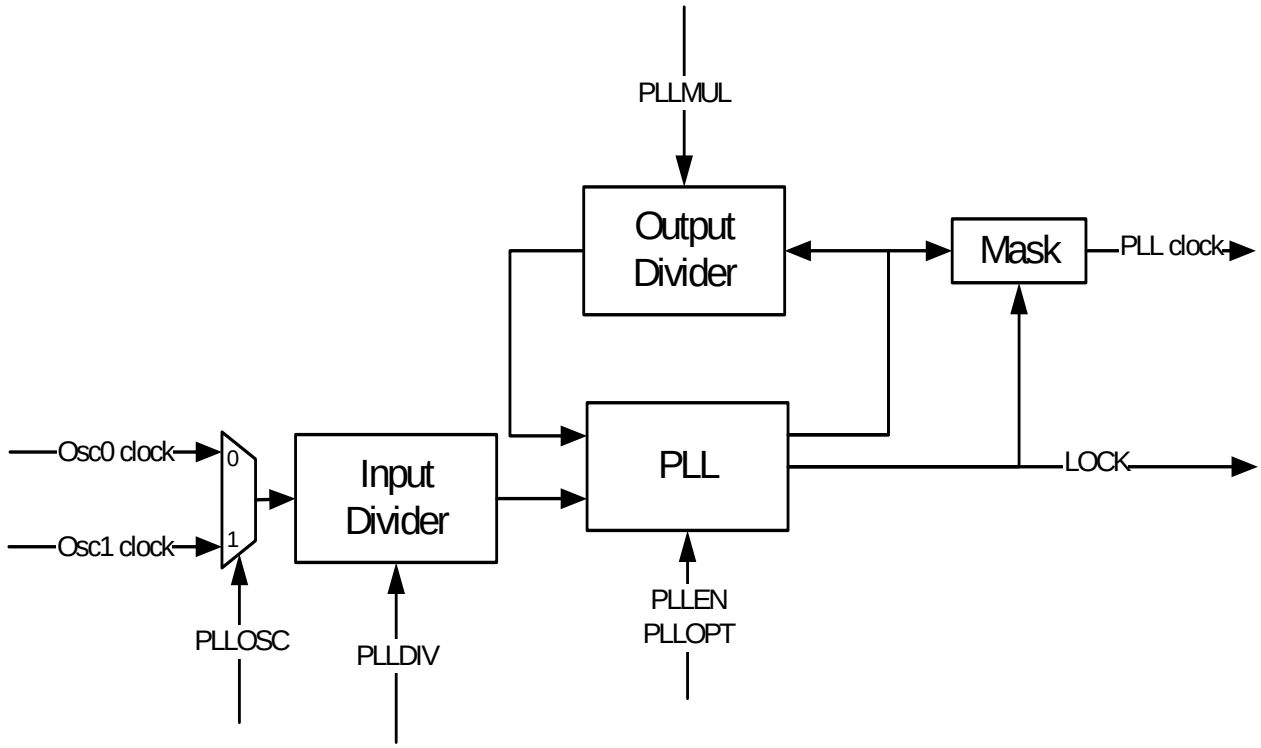
As a crystal oscillator usually requires a very long startup time (up to 1 second), the 32 KHz oscillator will keep running across resets, except Power-On-Reset.

8.5.4 PLL Operation

The device contains two PLLs, PLL0 and PLL1. These are disabled by default, but can be enabled to provide high frequency source clocks for synchronous or generic clocks. The PLLs can take either Oscillator 0 or 1 as reference clock. The PLL output is divided by a multiplication factor, and the PLL compares the resulting clock to the reference clock. The PLL will adjust its output frequency until the two compared clocks are equal, thus locking the output frequency to a multiple of the reference clock frequency.

When the PLL is switched on, or when changing the clock source or multiplication factor for the PLL, the PLL is unlocked and the output frequency is undefined. The PLL clock for the digital logic is automatically masked when the PLL is unlocked, to prevent connected digital logic from receiving a too high frequency and thus become unstable.

Figure 8-3. PLL with Control Logic and Filters



8.5.4.1 Enabling the PLL

PLLn is enabled by writing the PLLEN bit in the PLLn register. PLLOSC selects Oscillator 0 or 1 as clock source. The PLLMUL and PLLDIV fields must be written with the multiplication and division factors, respectively, creating the PLL frequency:

$$f_{PLL} = 2*(PLLMUL+1)/(PLLDIV+1) \bullet f_{OSC}$$

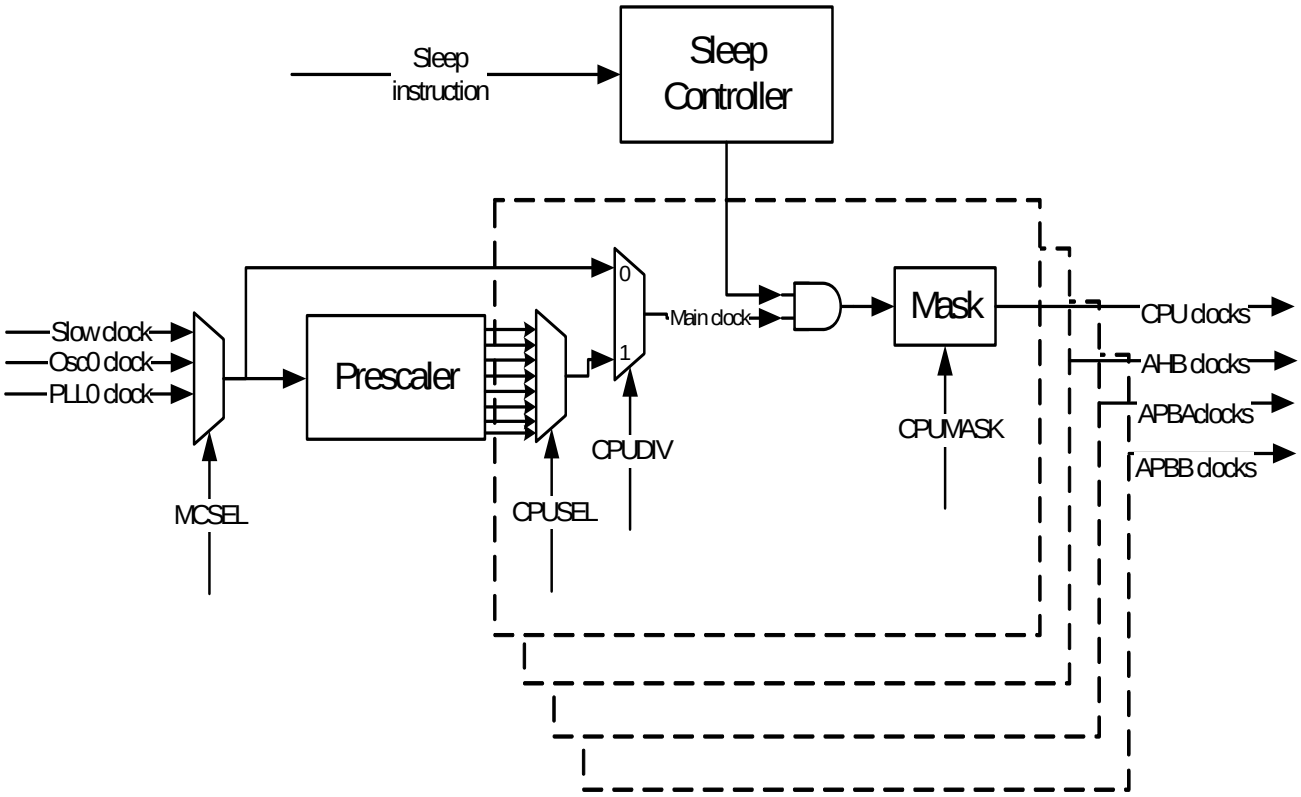
The PLLn.PLLOPT field should be set to proper values according to the PLL operating frequency. The PLLOPT field can also be set to divide the output frequency of the PLLs by 2.

The lock signal for each PLL is available as a LOCKn flag in POSCSR. An interrupt can be generated on a 0 to 1 transition of these bits.

8.5.5 Synchronous Clocks

The slow clock (default), Oscillator 0, or PLL0 provide the source for the main clock, which is the common root for the synchronous clocks for the CPU/HSB, PBA, and PBB modules. The main clock is divided by an 8-bit prescaler, and each of these four synchronous clocks can run from any tapping of this prescaler, or the undivided main clock, as long as $f_{CPU} \square f_{PBA,B,}$. The synchronous clock source can be changed on-the fly, responding to varying load in the application. The clock domains can be shut down in sleep mode, as described in Section 8.5.7. Additionally, the clocks for each module in the four domains can be individually masked, to avoid power consumption in inactive modules.

Figure 8-4. Synchronous Clock Generation



8.5.5.1 Selecting PLL or oscillator for the main clock

The common main clock can be connected to the slow clock, Oscillator 0, or PLL0. By default, the main clock will be connected to the slow clock. The user can connect the main clock to Oscillator 0 or PLL0 by writing the MCSEL field in the Main Clock Control Register (MCCTRL). This must only be done after that unit has been enabled, otherwise a deadlock will occur. Care should also be taken that the new frequency of the synchronous clocks does not exceed the maximum frequency for each clock domain.

8.5.5.2 Selecting synchronous clock division ratio

The main clock feeds an 8-bit prescaler, which can be used to generate the synchronous clocks. By default, the synchronous clocks run on the undivided main clock. The user can select a prescaler division for the CPU clock by writing CKSEL.CPUDIV to 1 and CPUSEL to the prescaling value, resulting in a CPU clock frequency:

$$f_{CPU} = f_{main} / 2^{(CPUSEL+1)}$$

Similarly, the clock for the PBA, and PBB can be divided by writing their respective fields. To ensure correct operation, frequencies must be selected so that $f_{CPU} \geq f_{PBA,B}$. Also, frequencies must never exceed the specified maximum frequency for each clock domain.

CKSEL can be written without halting or disabling peripheral modules. Writing CKSEL allows a new clock setting to be written to all synchronous clocks at the same time. It is possible to keep

one or more clocks unchanged by writing the same value a before to the xxxDIV and xxxSEL fields. This way, it is possible to e.g. scale CPU and HSB speed according to the required performance, while keeping the PBA and PBB frequency constant.

8.5.5.3 *Clock ready flag*

There is a slight delay from CKSEL is written and the new clock setting becomes effective. During this interval, the Clock Ready (CKRDY) flag in ISR will read as 0. If IER.CKRDY is written to 1, the Power Manager interrupt can be triggered when the new clock setting is effective. CKSEL must not be re-written while CKRDY is 0, or the system may become unstable or hang.

8.5.6 Peripheral Clock Masking

By default, the clock for all modules are enabled, regardless of which modules are actually being used. It is possible to disable the clock for a module in the CPU, HSB, PBA, or PBB clock domain by writing the corresponding bit in the Clock Mask register (CPU/HSB/PBA/PBB) to 0. When a module is not clocked, it will cease operation, and its registers cannot be read or written. The module can be re-enabled later by writing the corresponding mask bit to 1.

A module may be connected to several clock domains, in which case it will have several mask bits.

Table 8-6 on page 52 contains a list of implemented maskable clocks.

8.5.6.1 *Cautionary note*

The OCD clock must never be switched off if the user wishes to debug the device with a JTAG debugger.

Note that clocks should only be switched off if it is certain that the module will not be used. Switching off the clock for the internal RAM will cause a problem if the stack is mapped there. Switching off the clock to the Power Manager (PM), which contains the mask registers, or the corresponding PBx bridge, will make it impossible to write the mask registers again. In this case, they can only be re-enabled by a system reset.

8.5.6.2 *Mask ready flag*

Due to synchronization in the clock generator, there is a slight delay from a mask register is written until the new mask setting goes into effect. When clearing mask bits, this delay can usually be ignored. However, when setting mask bits, the registers in the corresponding module must not be written until the clock has actually be re-enabled. The status flag MSKRDY in ISR provides the required mask status information. When writing either mask register with any value, this bit is cleared. The bit is set when the clocks have been enabled and disabled according to the new mask setting. Optionally, the Power Manager interrupt can be enabled by writing the MSKRDY bit in IER.

8.5.7 Sleep Modes

In normal operation, all clock domains are active, allowing software execution and peripheral operation. When the CPU is idle, it is possible to switch off the CPU clock and optionally other clock domains to save power. This is activated by the sleep instruction, which takes the sleep mode index number as argument.

8.5.7.1 *Entering and exiting sleep modes*

The sleep instruction will halt the CPU and all modules belonging to the stopped clock domains. The modules will be halted regardless of the bit settings of the mask registers.

Oscillators and PLLs can also be switched off to save power. Some of these modules have a relatively long start-up time, and are only switched off when very low power consumption is required.

The CPU and affected modules are restarted when the sleep mode is exited. This occurs when an interrupt triggers. Note that even if an interrupt is enabled in sleep mode, it may not trigger if the source module is not clocked.

8.5.7.2 *Supported sleep modes*

The following sleep modes are supported. These are detailed in Table 8-1 on page 43.

•Idle: The CPU is stopped, the rest of the chip is operating. Wake-up sources are any interrupt.

•Frozen: The CPU and HSB modules are stopped, peripherals are operating. Wake-up sources are any interrupt from PB modules.

•Standby: All synchronous clocks are stopped, but oscillators and PLLs are running, allowing quick wake-up to normal mode. Wake-up sources are RTC or external interrupt.

•Stop: As Standby, but Oscillator 0 and 1, and the PLLs are stopped. 32 KHz (if enabled) and RC oscillators and RTC/WDT still operate. Wake-up sources are RTC, external interrupt, or external reset pin.

•DeepStop: All synchronous clocks, Oscillator 0 and 1 and PLL 0 and 1 are stopped. 32 KHz oscillator can run if enabled. RC oscillator still operates. Bandgap voltage reference and BOD is turned off.

•Static: All oscillators, including 32 KHz and RC oscillator are stopped. Bandgap voltage reference BOD detector is turned off.

Table 8-1. Sleep Modes

Index	Sleep Mode	CPU	HSB	PBA,B GCLK	Osc0,1 PLL0,1	Osc32	RCOsc	BOD & Bandgap	Voltage Regulator
0	Idle	Stop	Run	Run	Run	Run	Run	On	Full power
1	Frozen	Stop	Stop	Run	Run	Run	Run	On	Full power
2	Standby	Stop	Stop	Stop	Run	Run	Run	On	Full power
3	Stop	Stop	Stop	Stop	Stop	Run	Run	On	Low power
4	DeepStop	Stop	Stop	Stop	Stop	Run	Run	Off	Low power
5	Static	Stop	Stop	Stop	Stop	Stop	Stop	Off	Low power

The power level of the internal voltage regulator is also adjusted according to the sleep mode to reduce the internal regulator power consumption.

8.5.7.3 *Precautions when entering sleep mode*

Modules communicating with external circuits should normally be disabled before entering a sleep mode that will stop the module operation. This prevents erratic behavior when entering or exiting sleep mode. Please refer to the relevant module documentation for recommended actions.

Communication between the synchronous clock domains is disturbed when entering and exiting sleep modes. This means that bus transactions are not allowed between clock domains affected by the sleep mode. The system may hang if the bus clocks are stopped in the middle of a bus transaction.

The CPU is automatically stopped in a safe state to ensure that all CPU bus operations are complete when the sleep mode goes into effect. Thus, when entering Idle mode, no further action is necessary.

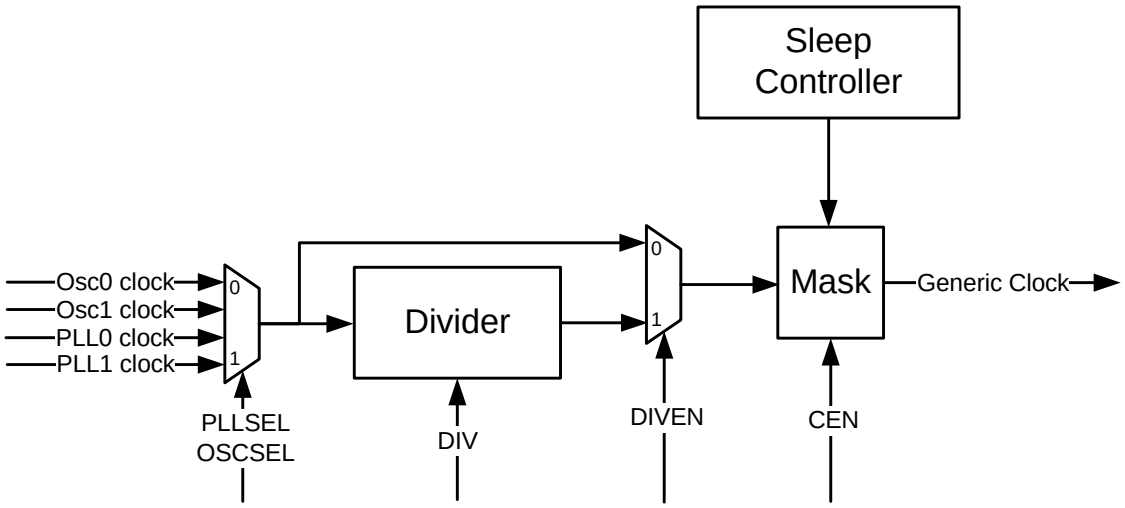
When entering a sleep mode (except Idle mode), all HSB masters must be stopped before entering the sleep mode. Also, if there is a chance that any PB write operations are incomplete, the CPU should perform a read operation from any register on the PB bus before executing the sleep instruction. This will stall the CPU while waiting for any pending PB operations to complete.

8.5.8 Generic Clocks

Timers, communication modules, and other modules connected to external circuitry may require specific clock frequencies to operate correctly. The Power Manager contains an implementation defined number of generic clocks that can provide a wide range of accurate clock frequencies.

Each generic clock module runs from either Oscillator 0 or 1, or PLL0 or 1. The selected source can optionally be divided by any even integer up to 256. Each clock can be independently enabled and disabled, and is also automatically disabled along with peripheral clocks by the Sleep Controller.

Figure 8-5. Generic Clock Generation



8.5.8.1 Enabling a generic clock

A generic clock is enabled by writing the CEN bit in GCCTRL to 1. Each generic clock can use either Oscillator 0 or 1 or PLL0 or 1 as source, as selected by the PLLSEL and OSCSEL bits. The source clock can optionally be divided by writing DIVEN to 1 and the division factor to DIV, resulting in the output frequency:

$$f_{GCLK} = f_{SRC}/(2 \times (DIV + 1))$$

8.5.8.2 *Disabling a generic clock*

The generic clock can be disabled by writing CEN to 0 or entering a sleep mode that disables the PB clocks. In either case, the generic clock will be switched off on the first falling edge after the disabling event, to ensure that no glitches occur. If CEN is written to 0, the bit will still read as 1 until the next falling edge occurs, and the clock is actually switched off. When writing CEN to 0, the other bits in GCCTRL should not be changed until CEN reads as 0, to avoid glitches on the generic clock.

When the clock is disabled, both the prescaler and output are reset.

8.5.8.3 *Changing clock frequency*

When changing generic clock frequency by writing GCCTRL, the clock should be switched off by the procedure above, before being re-enabled with the new clock source or division setting. This prevents glitches during the transition.

8.5.8.4 *Generic clock implementation*

The generic clocks are allocated to different functions as shown in Table 8-2 on page 45.

Table 8-2. Generic Clock Allocation

Clock number	Function
0	**GCLK0 pin**
1	**GCLK1 pin**
2	**GCLK2 pin**
3	**GCLK3 pin**
4	**GLCK_USBB**

8.5.9 Divided PB Clocks

The clock generator in the Power Manager provides divided PBA and PBB clocks for use by peripherals that require a prescaled PBx clock. This is described in the documentation for the relevant modules.

The divided clocks are not directly maskable, but are stopped in sleep modes where the PBx clocks are stopped.

8.5.10 Debug Operation

The OCD clock must never be switched off if the user wishes to debug the device with a JTAG debugger.

During a debug session, the user may need to halt the system to inspect memory and CPU registers. The clocks normally keep running during this debug operation, but some peripherals may require the clocks to be stopped, e.g. to prevent timer overflow, which would cause the program to fail. For this reason, peripherals on the PBA and PBB buses may use “debug qualified” PBx clocks. This is described in the documentation for the relevant modules. The divided PBx clocks are always debug qualified clocks.

Debug qualified PBx clocks are stopped during debug operation. The debug system can optionally keep these clocks running during the debug operation. This is described in the documentation for the On-Chip Debug system.

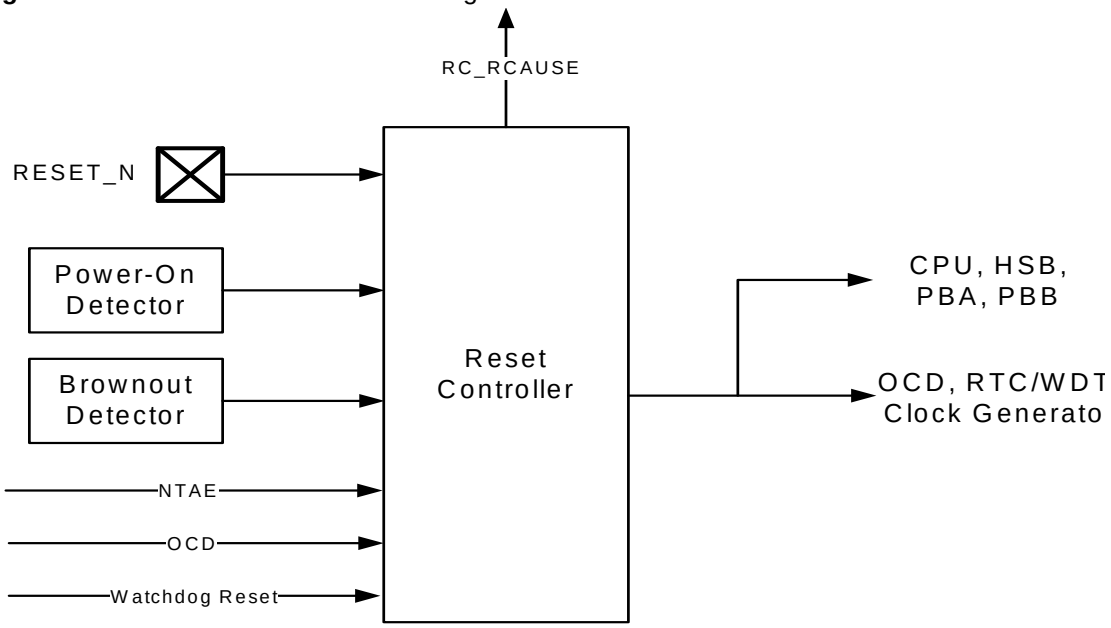
8.5.11 Reset Controller

The Reset Controller collects the various reset sources in the system and generates hard and soft resets for the digital logic.

The device contains a Power-On Detector, which keeps the system reset until power is stable. This eliminates the need for external reset circuitry to guarantee stable operation when powering up the device.

It is also possible to reset the device by asserting the RESET_N pin. This pin has an internal pull-up, and does not need to be driven externally when negated. Table 8-4 on page 47 lists these and other reset sources supported by the Reset Controller.

Figure 8-6. Reset Controller Block Diagram



In addition to the listed reset types, the JTAG can keep parts of the device statically reset through the JTAG Reset Register. See JTAG documentation for details.

Table 8-3. Reset Description

Reset source	Description
Power-on Reset	Supply voltage below the power-on reset detector threshold voltage
External Reset	RESET_N pin asserted
Brownout Reset	Supply voltage below the brownout reset detector threshold voltage
CPU Error	Caused by an illegal CPU access to external memory while in Supervisor mode
Watchdog Timer	See watchdog timer documentation.
OCD	See On-Chip Debug documentation

When a reset occurs, some parts of the chip are not necessarily reset, depending on the reset source. Only the Power On Reset (POR) will force a reset of the whole chip.

Table 8-4 on page 47 lists parts of the device that are reset, depending on the reset source.

Table 8-4. Effect of the Different Reset Events

	Power-On Reset	External Reset	Watchdog Reset	BOD Reset	CPU Error Reset	OCD Reset
CPU/HSB/PBA/PBB (excluding Power Manager)	Y	Y	Y	Y	Y	Y
32 KHz oscillator	Y	N	N	N	N	N
RTC control register	Y	N	N	N	N	N
GPLP registers	Y	N	N	N	N	N
Watchdog control register	Y	Y	N	Y	Y	Y
Voltage calibration register	Y	N	N	N	N	N
RC Oscillator Calibration register	Y	N	N	N	N	N
BOD control register	Y	Y	N	N	N	N
Bandgap control register	Y	Y	N	N	N	N
Clock control registers	Y	Y	Y	Y	Y	Y
Osc0/Osc1 and control registers	Y	Y	Y	Y	Y	Y
PLL0/PLL1 and control registers	Y	Y	Y	Y	Y	Y
OCD system and OCD registers	Y	Y	N	Y	Y	N

The cause of the last reset can be read from the RCAUSE register. This register contains one bit for each reset source, and can be read during the boot sequence of an application to determine the proper action to be taken.

8.5.11.1 Power-On detector

The Power-On Detector monitors the VDDCORE supply pin and generates a reset when the device is powered on. The reset is active until the supply voltage from the linear regulator is above the power-on threshold level. The reset will be re-activated if the voltage drops below the power-on threshold level. See Electrical Characteristics for parametric details.

8.5.11.2 Brown-Out detector

The Brown-Out Detector (BOD) monitors the VDDCORE supply pin and compares the supply voltage to the brown-out detection level, as set in BOD.LEVEL. The BOD is disabled by default, but can be enabled either by software or by flash fuses. The Brown-Out Detector can either generate an interrupt or a reset when the supply voltage is below the brown-out detection level. In any case, the BOD output is available in bit POSCR.BODET bit.

Note that any change to the BOD.LEVEL field of the BOD register should be done with the BOD deactivated to avoid spurious reset or interrupt.

See Electrical Characteristics chapter for parametric details.

8.5.11.3 *External reset*

The external reset detector monitors the state of the RESET_N pin. By default, a low level on this pin will generate a reset.

8.5.12 Calibration Registers

The Power Manager controls the calibration of the RC oscillator, voltage regulator, bandgap voltage reference through several calibrations registers.

Those calibration registers are loaded after a Power On Reset with default values stored in factory-programmed flash fuses.

Although it is not recommended to override default factory settings, it is still possible to override these default values by writing to those registers. To prevent unexpected writes due to software bugs, write access to these registers is protected by a “key”. First, a write to the register must be made with the field “KEY” equal to 0x55 then a second write must be issued with the “KEY” field equal to 0xAA.

8.6 User Interface

Table 8-5. PM Register Memory Map

Offset	Register	Register Name	Access	Reset State
0x000	Main Clock Control	MCCTRL	Read/Write	0x00000000
0x0004	Clock Select	CKSEL	Read/Write	0x00000000
0x008	CPU Mask	CPUMASK	Read/Write	0x00000003
0x00C	HSB Mask	HSBMASK	Read/Write	0x00000FFF
0x010	PBA Mask	PBAMASK	Read/Write	0x001FFFFF
0x014	PBB Mask	PBBMASK	Read/Write	0x000003FF
0x020	PLL0 Control	PLL0	Read/Write	0x00000000
0x024	PLL1 Control	PLL1	Read/Write	0x00000000
0x028	Oscillator 0 Control Register	OSCCTRL0	Read/Write	0x00000000
0x02C	Oscillator 1 Control Register	OSCCTRL1	Read/Write	0x00000000
0x030	Oscillator 32 Control Register	OSCCTRL32	Read/Write	0x00000000
0x040	PM Interrupt Enable Register	IER	Write-only	0x00000000
0x044	PM Interrupt Disable Register	IDR	Write-only	0x00000000
0x048	PM Interrupt Mask Register	IMR	Read-only	0x00000000
0x04C	PM Interrupt Status Register	ISR	Read-only	0x00000000
00050	PM Interrupt Clear Register	ICR	Write-only	0x00000000
0x054	Power and Oscillators Status Register	POSCSR	Read/Write	0x00000000
0x060	Generic Clock Control	GCCTRL	Read/Write	0x00000000
0x0C0	RC Oscillator Calibration Register	RCCR	Read/Write	Factory settings
0x0C4	Bandgap Calibration Register	BGCR	Read/Write	Factory settings
0x0C8	Linear Regulator Calibration Register	VREGCR	Read/Write	Factory settings
0x0D0	BOD Level Register	BOD	Read/Write	BOD fuses in Flash
0x200	General Purpose Low-Power register	GPLP	Read/Write	0x00000000

8.6.1 Main Clock Control Register

Name: MCCTRL

Access Type: Read/Write

Offset: 0x00

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
-	-	-	-	OSC1EN	OSC0EN	MCSEL	

- **OSC1EN: Oscillator 1 Enable**
  1: Oscillator 1 is enabled
  0: Oscillator 1 is disabled
- **OSC0EN: Oscillator 0 Enable**
  1: Oscillator 0 is enabled
  0: Oscillator 0 is disabled
- **MCSEL: Main Clock Select**
  This field contains the clock selected as the main clock.

MCSEL	Selected Clock
0b00	Slow Clock
0b01	Oscillator 0
0b10	PLL 0
0b11	Reserved

8.6.2 Clock Select Register

Name: CKSEL

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
PBBDIV	-	-	-	-	PBBSEL		

23	22	21	20	19	18	17	16
PBADIV	-	-	-	-	PBASEL		

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
CPUDIV	-	-	-	-	CPUSEL		

- **PBBDIV: PBB Division Enable**
  PBBDIV = 0: PBB clock equals main clock.
  PBBDIV = 1: PBB clock equals main clock divided by $2^{(PBBSEL+1)}$.
- **PBADIV, PBASEL: PBA Division and Clock Select**
  PBADIV = 0: PBA clock equals main clock.
  PBADIV = 1: PBA clock equals main clock divided by $2^{(PBASEL+1)}$.
- **CPUDIV, CPUSEL: CPU/HSB Division and Clock Select**
  CPUDIV = 0: CPU/HSB clock equals main clock.
  CPUDIV = 1: CPU/HSB clock equals main clock divided by $2^{(CPUSEL+1)}$.

Note that if xxxDIV is written to 0, xxxSEL should also be written to 0 to ensure correct operation.

Also note that writing this register clears POSCSR.CKRDY. The register must not be re-written until CKRDY goes high.

8.6.3 Clock Mask Registers

Name: CPU/HSB/PBA/PBBMASK

Access Type: Read/Write

Offset: 0x08-0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MASK[31:24]							

23	22	21	20	19	18	17	16
MASK[23:16]							

15	14	13	12	11	10	9	8
MASK[15:8]							

7	6	5	4	3	2	1	0
MASK[7:0]							

- **MASK: Clock Mask**

  If bit n is cleared, the clock for module n is stopped. If bit n is set, the clock for module n is enabled according to the current power mode. The number of implemented bits in each mask register, as well as which module clock is controlled by each bit, is shown in Table 8-6 on page 52.

Table 8-6. Maskable module clocks in AT32UC3A3.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
0	-	FLASHC	INTC	HMATRIX
1	OCD[1]	PBA bridge	GPIO	USBB
2	-	PBB bridge	PDCA	FLASHC
3	-	USBB	PM/RTC/EIM	SMC
4	-	PDCA	ADC	SDRAMC
5	-	EBI	SPI0	HECC
6	-	PBC bridge	SPI1	MCI
7	-	DMACA	TWIM0	BUSMON
8	-	BUSMON	TWIM1	MSI
9	-	HRAMC0	TWIS0	AES
10	-	HRAMC1	TWIS1	-
11	-	-	USART0	-
12	-	-	USART1	-
13	-	-	USART2	-
14	-	-	USART3	-
15	-	-	SSC	-

Table 8-6. Maskable module clocks in AT32UC3A3.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
16	-	-	TC0	-
17	-	-	TC1	-
18	-	-	DAC	-
19	-	-	JTAGM	-
20	-	-	AWM	-
31:21	-	-	-	-

Note: 1. This bit must be one if the user wishes to debug the device with a JTAG debugger.

8.6.4 PLL Control Registers

Name: PLL0,1

Access Type: Read/Write

Offset: 0x20-0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
PLLTEST	-	PLLCOUNT					

23	22	21	20	19	18	17	16
PLLMUL							

15	14	13	12	11	10	9	8
PLLDIV							

7	6	5	4	3	2	1	0
-	-	-	PLLOPT			PLLOSC	PLLEN

- **PLLTEST: PLL Test**
  Reserved for internal use. Always write to 0.
- **PLLCOUNT: PLL Count**
  Specifies the number of slow clock cycles before ISR.LOCKn will be set after PLLn has been written, or after PLLn has been automatically re-enabled after exiting a sleep mode.
- **PLLMUL: PLL Multiply Factor**
- **PLLDIV: PLL Division Factor**
  These fields determine the ratio of the PLL output frequency to the source oscillator frequency:

$$f_{PLL} = 2 \times f_{osc} \times (PLLMUL + 1)/(PLLDIV + 1)$$

  fNote that the PLLMUL field cannot be equal to 0 or 1, or the behavior of the PLL will be undefined.
- **PLLOPT: PLL Option**
  Select the operating range for the PLL.
  PLLOPT[0]: Select the VCO frequency range
  PLLOPT[1]: Enable the extra output divider
  PLLOPT[2]: Disable the Wide-Bandwidth mode (Wide-Bandwidth mode allows a faster startup time and out-of-lock time)

	Description
PLLOPT[0]: VCO frequency	
0	**160MHz<f_{vco}<240MHz**
1	**80MHz<f_{vco}<180MHz**
PLLOPT[1]: Output divider	
0	**$f_{out} = f_{vco}$**
1	**$f_{out} = f_{vco}/2$**

	Description
PLLOPT[2]	
0	**Wide Bandwidth Mode enabled**
1	**Wide Bandwidth Mode disabled**

- **PLLOSC: PLL Oscillator Select**
  0: Oscillator 0 is the source for the PLL.
  1: Oscillator 1 is the source for the PLL.
- **PLLEN: PLL Enable**
  0: PLL is disabled.
  1: PLL is enabled.

8.6.5 Oscillator 0/1 Control Registers

Name: OSCCTRL0,1

Access Type: Read/Write

Offset: 0x28-0x2C

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
-	-	-	-	-	STARTUP		

7	6	5	4	3	2	1	0
-	-	-	-	-	MODE		

- **STARTUP: Oscillator Startup Time**
  Select startup time for the oscillator.

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCOsc = 115 kHz)
0	**0**	**0**
1	**64**	**560 us**
2	**128**	**1.1 ms**
3	**2048**	**18 ms**
4	**4096**	**36 ms**
5	**8192**	**71 ms**
6	**16384**	**142 ms**
7	**Reserved**	**Reserved**

- **MODE: Oscillator Mode**
  Choose between crystal, or external clock
  0: External clock connected on XIN, XOUT can be used as an I/O (no crystal)
  1: Crystal is connected to XIN/XOUT - Oscillator is used with automatic gain control
  2 to 7: Reserved

8.6.6 32 KHz Oscillator Control Register

Name: OSCCTRL32

Access Type: Read/Write

Offset: 0x30

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	STARTUP		

15	14	13	12	11	10	9	8
-	-	-	-	-	MODE		

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OSC32EN

- **STARTUP: Oscillator Startup Time**
  Select startup time for 32 KHz oscillator

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCOsc = 115 kHz)
0	**0**	**0**
1	**128**	**1.1 ms**
2	**8192**	**72.3 ms**
3	**16384**	**143 ms**
4	**65536**	**570 ms**
5	**131072**	**1.1 s**
6	**262144**	**2.3 s**
7	**524288**	**4.6 s**

Note: This register is only reset by Power-On Reset

- **MODE: Oscillator Mode**
  Choose between crystal, or external clock
  0: External clock connected on XIN32, XOUT32 can be used as a I/O (no crystal)
  1: Crystal is connected to XIN32/XOUT32 - Oscillator is used with automatic gain control
  2 to 7: Reserved
- **OSC32EN: Enable the 32 KHz oscillator**
  0: 32 KHz Oscillator is disabled
  1: 32 KHz Oscillator is enabled

8.6.7 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x40

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

Writing a one to a bit in this register will set the corresponding bit in IMR.
Writing a zero to a bit in this register has no effect.

8.6.8 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x44

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

Writing a one to a bit in this register will clear the corresponding bit in IMR.
Writing a zero to a bit in this register has no effect.

8.6.9 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x48

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

8.6.10 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x4C

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

- **BODDET: Brown out detection**
  This bit is set when a 0 to 1 transition on POSCSR.BODDET bit is detected: BOD has detected that power supply is going below BOD reference value.
  This bit is cleared when the corresponding bit in ICR is written to one.
- **OSC32RDY: 32 KHz oscillator Ready**
  This bit is set when a 0 to 1 transition on the POSCSR.OSC32RDY bit is detected: The 32 KHz oscillator is stable and ready to be used as clock source.
  This bit is cleared when the corresponding bit in ICR is written to one.
- **OSC1RDY: Oscillator 1 Ready**
  This bit is set when a 0 to 1 transition on the POSCSR.OSC1RDY bit is detected: Oscillator 1 is stable and ready to be used as clock source.
  This bit is cleared when the corresponding bit in ICR is written to one.
- **OSC0RDY: Oscillator 0 Ready**
  This bit is set when a 0 to 1 transition on the POSCSR.OSC1RDY bit is detected: Oscillator 1 is stable and ready to be used as clock source.
  This bit is cleared when the corresponding bit in ICR is written to one.
- **MSKRDY: Mask Ready**
  This bit is set when a 0 to 1 transition on the POSCSR.MSKRDY bit is detected: Clocks are now masked according to the (CPU/HSB/PBA/PBB)_MASK registers.
  This bit is cleared when the corresponding bit in ICR is written to one.
- **CKRDY: Clock Ready**
  0: The CKSEL register has been written, and the new clock setting is not yet effective.
  1: The synchronous clocks have frequencies as indicated in the CKSEL register.
  Note: Writing a one to ICR.CKRDY has no effect.
- **LOCK1: PLL1 locked**
  This bit is set when a 0 to 1 transition on the POSCSR.LOCK1 bit is detected: PLL 1 is locked and ready to be selected as clock source.
  This bit is cleared when the corresponding bit in ICR is written to one.

- **LOCK0: PLL0 locked**

  This bit is set when a 0 to 1 transition on the POSCSR.LOCK0 bit is detected: PLL 0 is locked and ready to be selected as clock source.

  This bit is cleared when the corresponding bit in ICR is written to one.

8.6.11 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x50

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in ISR and the corresponding interrupt request.

8.6.12 Power and Oscillators Status Register

Name: POSCCR

Access Type: Read-only

Offset: 0x54

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

- **BODDET: Brown out detection**
  0: No BOD event
  1: BOD has detected that power supply is going below BOD reference value.
- **OSC32RDY: 32 KHz oscillator Ready**
  0: The 32 KHz oscillator is not enabled or not ready.
  1: The 32 KHz oscillator is stable and ready to be used as clock source.
- **OSC1RDY: OSC1 ready**
  0: Oscillator 1 not enabled or not ready.
  1: Oscillator 1 is stable and ready to be used as clock source.
- **OSC0RDY: OSC0 ready**
  0: Oscillator 0 not enabled or not ready.
  1: Oscillator 0 is stable and ready to be used as clock source.
- **MSKRDY: Mask ready**
  0: Mask register has been changed, masking in progress.
  1: Clock are masked according to xxx_MASK
- **CKRDY:**
  0: The CKSEL register has been written, and the new clock setting is not yet effective.
  1: The synchronous clocks have frequencies as indicated in the CKSEL register.
- **LOCK1: PLL 1 locked**
  0:PLL 1 is unlocked
  1:PLL 1 is locked, and ready to be selected as clock source.
- **LOCK0: PLL 0 locked**
  0: PLL 0 is unlocked
  1: PLL 0 is locked, and ready to be selected as clock source.

8.6.13 Generic Clock Control Register

Name: GCCTRL

Access Type: Read/Write

Offset: 0x60

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
DIV[7:0]							

7	6	5	4	3	2	1	0
-	-	-	DIVEN	-	CEN	PLLSEL	OSCSEL

There is one GCCTRL register per generic clock in the design.

- **DIV: Division Factor**
- **DIVEN: Divide Enable**
  0: The generic clock equals the undivided source clock.
  1: The generic clock equals the source clock divided by 2*(DIV+1).
- **CEN: Clock Enable**
  0: Clock is stopped.
  1: Clock is running.
- **PLLSEL: PLL Select**
  0: Oscillator is source for the generic clock.
  1: PLL is source for the generic clock.
- **OSCSEL: Oscillator Select**
  0: Oscillator (or PLL) 0 is source for the generic clock.
  1: Oscillator (or PLL) 1 is source for the generic clock.

8.6.14 Reset Cause Register

Name: RCAUSE

Access Type: Read-only

Offset: 0x140

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
-	-	-	-	-	-	JTAGHARD	OCDRST

7	6	5	4	3	2	1	0
CPUERR	-	-	JTAG	WDT	EXT	BOD	POR

- **JTAGHARD: JTAG Hard Reset**
  The chip was reset by setting the bit RC_OCD in the JTAG reset register or by using the JTAG HALT instruction.
- **OCDRST: OCD Reset**
  The CPU was reset because the RES strobe in the OCD Development Control register has been written to one.
- **CPUERR: CPU Error**
  The CPU was reset because it had detected an illegal access.
- **JTAG: JTAG reset**
  The CPU was reset by setting the bit RC_CPU in the JTAG reset register.
- **WDT: Watchdog Reset**
  The CPU was reset because of a watchdog timeout.
- **EXT: External Reset Pin**
  The CPU was reset due to the RESET pin being asserted.
- **BOD: Brown-out Reset**
  The CPU was reset due to the supply voltage being lower than the brown-out threshold level.
- **POR Power-on Reset**
  The CPU was reset due to the supply voltage being lower than the power-on threshold level.

8.6.15 BOD Control Register

Name: BOD

Access Type: Read/Write

Offset: 0xD0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CTRL	

7	6	5	4	3	2	1	0
-	HYST	LEVEL					

- **KEY: Register Write protection**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.
- **FCD: BOD Fuse calibration done**
  Set to 1 when CTRL, HYST and LEVEL fields has been updated by the Flash fuses after power-on reset or Flash fuses update
  If one, the CTRL, HYST and LEVEL values will not be updated again by Flash fuses
  Can be cleared to allow subsequent overwriting of the value by Flash fuses
- **CTRL: BOD Control**
  0: BOD is off
  1: BOD is enabled and can reset the chip
  2: BOD is enabled and but cannot reset the chip. Only interrupt will be sent to interrupt controller, if enabled in the IMR register.
  3: BOD is off
- **HYST: BOD Hysteresis**
  0: No hysteresis
  1: Hysteresis On
- **LEVEL: BOD Level**
  This field sets the triggering threshold of the BOD. See Electrical Characteristics for actual voltage levels.
  Note that any change to the LEVEL field of the BOD register should be done with the BOD deactivated to avoid spurious reset or interrupt.

8.6.16 RC Oscillator Calibration Register

Name: RCCR

Access Type: Read/Write

Offset: 0xC0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CALIB	

7	6	5	4	3	2	1	0
CALIB							

- **KEY: Register Write protection**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.
- **FCD: Flash Calibration Done**
  Set to 1 when CTRL, HYST, and LEVEL fields have been updated by the Flash fuses after power-on reset, or after Flash fuses are reprogrammed. The CTRL, HYST and LEVEL values will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.
- **CALIB: Calibration Value**
  Calibration Value for the RC oscillator.

8.6.17 Bandgap Calibration Register

Name: BGCR

Access Type: Read/Write

Offset: 0xC4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	CALIB		

- **KEY: Register Write protection**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.
- **FCD: Flash Calibration Done**
  Set to 1 when the CALIB field has been updated by the Flash fuses after power-on reset or when the Flash fuses are reprogrammed. The CALIB field will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.
- **CALIB: Calibration value**
  Calibration value for Bandgap. See Electrical Characteristics for voltage values.

8.6.18 PM Voltage Regulator Calibration Register

Name: VREGCR

Access Type: Read/Write

Offset: 0xC8

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	CALIB		

- **KEY: Register Write protection**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.
  Calibration value for Voltage Regulator. See Electrical Characteristics for voltage values.
- **FCD: Flash Calibration Done**
  Set to 1 when the CALIB field has been updated by the Flash fuses after power-on reset or when the Flash fuses are reprogrammed. The CALIB field will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.
- **CALIB: Calibration value**

8.6.19 General Purpose Low-power Register

Name: GPLP

Access Type: Read/Write

Offset: 0x200

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
GPLP							

23	22	21	20	19	18	17	16
GPLP							

15	14	13	12	11	10	9	8
GPLP							

7	6	5	4	3	2	1	0
GPLP							

These registers are general purpose 32-bit registers that are reset only by power-on-reset. Any other reset will keep the content of these registers untouched.

9. Real Time Counter (RTC)

Rev: 2.3.1.1

9.1 Features

- **32-bit real-time counter with 16-bit prescaler**
- **Clocked from RC oscillator or 32KHz oscillator**
- **Long delays**
  - **Max timeout 272years**
- **High resolution: Max count frequency 16KHz**
- **Extremely low power consumption**
- **Available in all sleep modes except Static**
- **Interrupt on wrap**

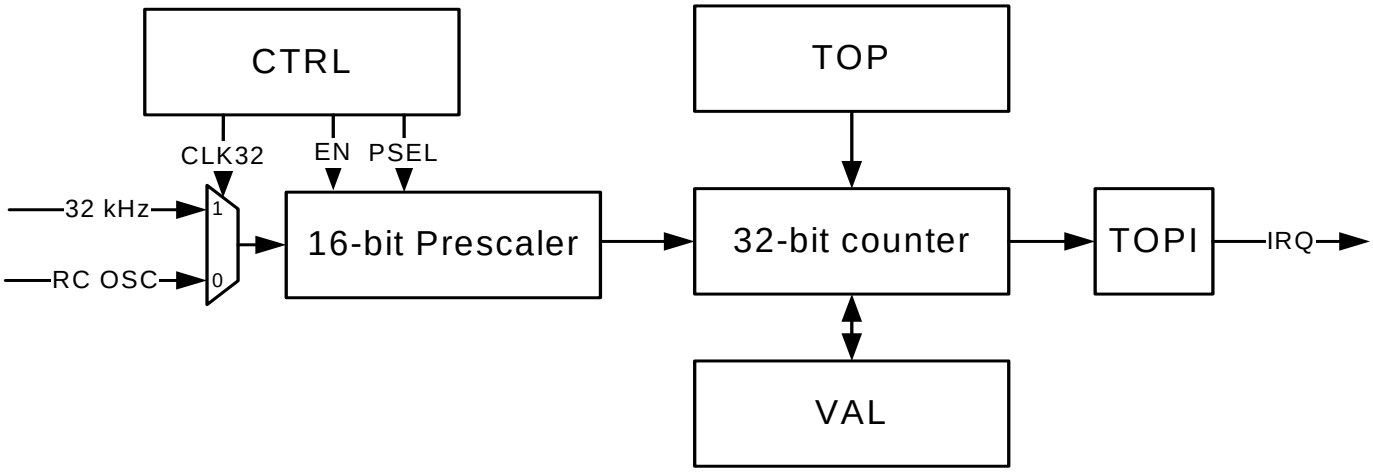
9.2 Overview

The Real Time Counter (RTC) enables periodic interrupts at long intervals, or accurate measurement of real-time sequences. The RTC is fed from a 16-bit prescaler, which is clocked from the system RC oscillator or the 32KHz crystal oscillator. Any tapping of the prescaler can be selected as clock source for the RTC, enabling both high resolution and long timeouts. The prescaler cannot be written directly, but can be cleared by the user.

The RTC can generate an interrupt when the counter wraps around the value stored in the top register (TOP), producing accurate periodic interrupts.

9.3 Block Diagram

Figure 9-1. Real Time Counter Block Diagram



9.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

9.4.1 Power Management

The RTC remains operating in all sleep modes except Static mode. Interrupts are not available in DeepStop mode.

9.4.2 Clocks

The RTC can use the system RC oscillator as clock source. This oscillator is always enabled whenever this module is active. Please refer to the Electrical Characteristics chapter for the characteristic frequency of this oscillator (f_{RC}).

The RTC can also use the 32 KHz crystal oscillator as clock source. This oscillator must be enabled before use. Please refer to the Power Manager chapter for details.

The clock for the RTC bus interface (CLK_RTC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the RTC before disabling the clock, to avoid freezing the RTC in an undefined state.

9.4.3 Interrupts

The RTC interrupt request line is connected to the interrupt controller. Using the RTC interrupt requires the interrupt controller to be programmed first.

9.4.4 Debug Operation

The RTC prescaler is frozen during debug operation, unless the OCD system keeps peripherals running in debug operation.

9.5 Functional Description

9.5.1 RTC Operation

9.5.1.1 Source clock

The RTC is enabled by writing a one to the Enable bit in the Control Register (CTRL.EN). The 16-bit prescaler will then increment on the selected clock. The prescaler cannot be read or written, but it can be reset by writing a one to the Prescaler Clear bit in CTRL register (CTRL.PCLR).

The 32KHz Oscillator Select bit in CTRL register (CTRL.CLK32) selects either the RC oscillator or the 32KHz oscillator as clock source for the prescaler.

The Prescale Select field in CTRL register (CTRL.PSEL) selects the prescaler tapping, selecting the source clock for the RTC:

$$f_{RTC} = 2^{-(PSEL+1)} \times (f_{RC} \text{ or } 32\ \text{KHz})$$

9.5.1.2 Counter operation

When enabled, the RTC will increment until it reaches TOP, and then wraps to 0x0. The status bit TOPI in Interrupt Status Register (ISR) is set to one when this occurs. From 0x0 the counter will count TOP+1 cycles of the source clock before it wraps back to 0x0.

The RTC count value can be read from or written to the Value register (VAL). Due to synchronization, continuous reading of the VAL register with the lowest prescaler setting will skip every other value.

9.5.1.3 RTC interrupt

The RTC interrupt is enabled by writing a one to the Top Interrupt bit in the Interrupt Enable Register (IER.TOPI), and is disabled by writing a one to the Top Interrupt bit in the Interrupt Disable Register (IDR.TOPI). The Interrupt Mask Register (IMR) can be read to see whether or not the interrupt is enabled. If enabled, an interrupt will be generated if the TOPI bit in the Interrupt Status Register (ISR) is set. The TOPI bit in ISR can be cleared by writing a one to the TOPI bit in the Interrupt Clear Register (ICR.TOPI).

The RTC interrupt can wake the CPU from all sleep modes except DeepStop and Static modes.

9.5.1.4 RTC wakeup

The RTC can also wake up the CPU directly without triggering an interrupt when the ISR.TOPI bit is set. In this case, the CPU will continue executing from the instruction following the sleep instruction.

This direct RTC wake-up is enabled by writing a one to the Wake Enable bit in the CTRL register (CTRL.WAKEN). When the CPU wakes from sleep, the CTRL.WAKEN bit must be written to zero to clear the internal wake signal to the sleep controller, otherwise a new sleep instruction will have no effect.

The RTC wakeup is available in all sleep modes except Static mode. The RTC wakeup can be configured independently of the RTC interrupt.

9.5.1.5 Busy bit

Due to the crossing of clock domains, the RTC uses a few clock cycles to propagate the values stored in CTRL, TOP, and VAL to the RTC. The RTC Busy bit in CTRL (CTRL.BUSY) indicates that a register write is still going on and all writes to TOP, CTRL, and VAL will be discarded until the CTRL.BUSY bit goes low again.

9.6 User Interface

Table 9-1. RTC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Value Register	VAL	Read/Write	0x00000000
0x08	Top Register	TOP	Read/Write	0x00000000
0x10	Interrupt Enable Register	IER	Write-only	0x00000000
0x14	Interrupt Disable Register	IDR	Write-only	0x00000000
0x18	Interrupt Mask Register	IMR	Read-only	0x00000000
0x1C	Interrupt Status Register	ISR	Read-only	0x00000000
0x20	Interrupt Clear Register	ICR	Write-only	0x00000000

9.6.1 Control Register

Name: CTRL

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	CLKEN
15	14	13	12	11	10	9	8
-	-	-	-	PSEL			
7	6	5	4	3	2	1	0
-	-	-	BUSY	CLK32	WAKEN	PCLR	EN

- **CLKEN: Clock Enable**
  1: The clock is enabled.
  0: The clock is disabled.
- **PSEL: Prescale Select**
  Selects prescaler bit PSEL as source clock for the RTC.
- **BUSY: RTC Busy**
  This bit is set when the RTC is busy and will discard writes to TOP, VAL, and CTRL.
  This bit is cleared when the RTC accepts writes to TOP, VAL, and CTRL.
- **CLK32: 32 KHz Oscillator Select**
  1: The RTC uses the 32 KHz oscillator as clock source.
  0: The RTC uses the RC oscillator as clock source.
- **WAKEN: Wakeup Enable**
  1: The RTC wakes up the CPU from sleep modes.
  0: The RTC does not wake up the CPU from sleep modes.
- **PCLR: Prescaler Clear**
  Writing a one to this bit clears the prescaler.
  Writing a zero to this bit has no effect.
  This bit always reads as zero.
- **EN: Enable**
  1: The RTC is enabled.
  0: The RTC is disabled.

9.6.2 Value Register

Name: VAL

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
VAL[31:24]							

23	22	21	20	19	18	17	16
VAL[23:16]							

15	14	13	12	11	10	9	8
VAL[15:8]							

7	6	5	4	3	2	1	0
VAL[7:0]							

- **VAL[31:0]: RTC Value**
  This value is incremented on every rising edge of the source clock.

9.6.3 Top Register

Name: TOP

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
VAL[31:24]							

23	22	21	20	19	18	17	16
VAL[23:16]							

15	14	13	12	11	10	9	8
VAL[15:8]							

7	6	5	4	3	2	1	0
VAL[7:0]							

- **VAL[31:0]: RTC Top Value**
  VAL wraps at this value.

9.6.4 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

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
-	-	-	-	-	-	-	TOPI

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

9.6.5 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

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
-	-	-	-	-	-	-	TOPI

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

9.6.6 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

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
-	-	-	-	-	-	-	TOPI

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

9.6.7 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x1C

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
-	-	-	-	-	-	-	TOPI

- **TOPI: Top Interrupt**
  This bit is set when VAL has wrapped at its top value.
  This bit is cleared when the corresponding bit in ICR is written to one.

9.6.8 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x20

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
-	-	-	-	-	-	-	TOPI

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

10. Watchdog Timer (WDT)

Rev: 2.3.1.1

10.1 Features

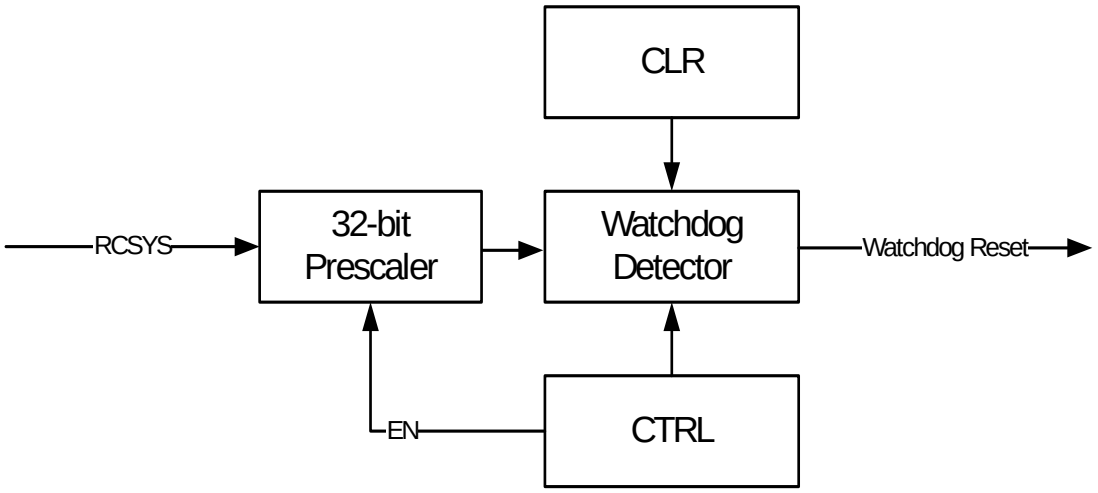
- **Watchdog timer counter with 32-bit prescaler**
- **Clocked from the system RC oscillator (RCSYS)**

10.2 Overview

The Watchdog Timer (WDT) has a prescaler generating a time-out period. This prescaler is clocked from the RC oscillator. The watchdog timer must be periodically reset by software within the time-out period, otherwise, the device is reset and starts executing from the boot vector. This allows the device to recover from a condition that has caused the system to be unstable.

10.3 Block Diagram

Figure 10-1. WDT Block Diagram



10.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

10.4.1 Power Management

When the WDT is enabled, the WDT remains clocked in all sleep modes, and it is not possible to enter Static mode.

10.4.2 Clocks

The WDT can use the system RC oscillator (RCSYS) as clock source. This oscillator is always enabled whenever these modules are active. Please refer to the Electrical Characteristics chapter for the characteristic frequency of this oscillator (f_{RC}).

10.4.3 Debug Operation

The WDT prescaler is frozen during debug operation, unless the On-Chip Debug (OCD) system keeps peripherals running in debug operation.

10.5 Functional Description

The WDT is enabled by writing a one to the Enable bit in the Control register (CTRL.EN). This also enables the system RC clock (CLK_RCSYS) for the prescaler. The Prescale Select field (PSEL) in the CTRL register selects the watchdog time-out period:

$$T_{WDT} = 2^{(PSEL+1)} / f_{RC}$$

The next time-out period will begin as soon as the watchdog reset has occurred and count down during the reset sequence. Care must be taken when selecting the PSEL field value so that the time-out period is greater than the startup time of the chip, otherwise a watchdog reset can reset the chip before any code has been run.

To avoid accidental disabling of the watchdog, the CTRL register must be written twice, first with the KEY field set to 0x55, then 0xAA without changing the other bits. Failure to do so will cause the write operation to be ignored, and the CTRL register value will not change.

The Clear register (CLR) must be written with any value with regular intervals shorter than the watchdog time-out period. Otherwise, the device will receive a soft reset, and the code will start executing from the boot vector.

When the WDT is enabled, it is not possible to enter Static mode. Attempting to do so will result in entering Shutdown mode, leaving the WDT operational.

10.6 User Interface

Table 10-1. WDT Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Clear Register	CLR	Write-only	0x00000000

10.6.1 Control Register

Name: CTRL

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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
-	-	-	PSEL				

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **KEY: Write protection key**

  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to be effective.

  This field always reads as zero.
- **PSEL: Prescale Select**

  PSEL is used as watchdog timeout period.
- **EN: WDT Enable**

  1: WDT is enabled.

  0: WDT is disabled.

10.6.2 Clear Register

Name: CLR

Access Type: Write-only

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CLR[31:24]							

23	22	21	20	19	18	17	16
CLR[23:16]							

15	14	13	12	11	10	9	8
CLR[15:8]							

7	6	5	4	3	2	1	0
CLR[7:0]							

- **CLR:**
  Writing periodically any value to this field when the WDT is enabled, within the watchdog time-out period, will prevent a watchdog reset.
  This field always reads as zero.

11. Interrupt Controller (INTC)

Rev: 1.0.1.4

11.1 Features

- **Autovectored low latency interrupt service with programmable priority**
  - **4 priority levels for regular, maskable interrupts**
  - **One Non-Maskable Interrupt**
- **Up to 64 groups of interrupts with up to 32 interrupt requests in each**

11.2 Overview

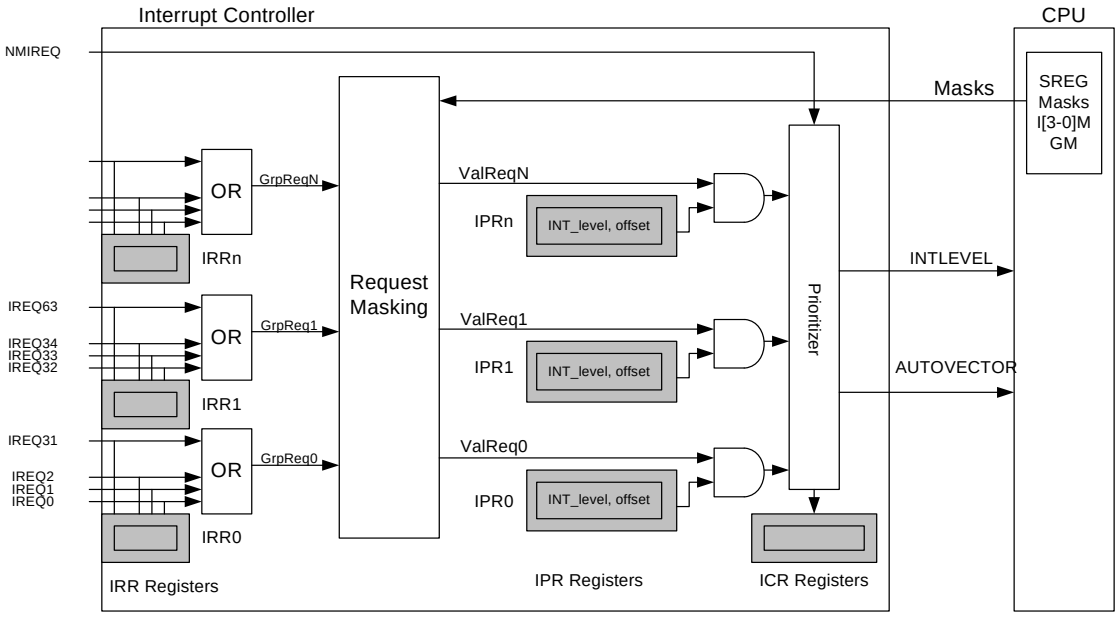
The INTC collects interrupt requests from the peripherals, prioritizes them, and delivers an interrupt request and an autovector to the CPU. The AVR32 architecture supports 4 priority levels for regular, maskable interrupts, and a Non-Maskable Interrupt (NMI).

The INTC supports up to 64 groups of interrupts. Each group can have up to 32 interrupt request lines, these lines are connected to the peripherals. Each group has an Interrupt Priority Register (IPR) and an Interrupt Request Register (IRR). The IPRs are used to assign a priority level and an autovector to each group, and the IRRs are used to identify the active interrupt request within each group. If a group has only one interrupt request line, an active interrupt group uniquely identifies the active interrupt request line, and the corresponding IRR is not needed. The INTC also provides one Interrupt Cause Register (ICR) per priority level. These registers identify the group that has a pending interrupt of the corresponding priority level. If several groups have an pending interrupt of the same level, the group with the lowest number takes priority.

11.3 Block Diagram

Figure 11-1 on page 89 gives an overview of the INTC. The grey boxes represent registers that can be accessed via the user interface. The interrupt requests from the peripherals (IREQn) and the NMI are input on the left side of the figure. Signals to and from the CPU are on the right side of the figure.

Figure 11-1. Block Diagram of the Interrupt Controller



11.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

11.4.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the INTC, the INTC will stop functioning and resume operation after the system wakes up from sleep mode.

11.4.2 Clocks

The clock for the INTC bus interface (CLK_INTC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager.

11.4.3 Debug Operation

When an external debugger forces the CPU into debug mode, the INTC continues normal operation.

11.5 Functional Description

All of the incoming interrupt requests (IREQs) are sampled into the corresponding Interrupt Request Register (IRR). The IRRs must be accessed to identify which IREQ within a group that is active. If several IREQs within the same group is active, the interrupt service routine must prioritize between them. All of the input lines in each group are logically-ORed together to form the GrpReqN lines, indicating if there is a pending interrupt in the corresponding group.

The Request Masking hardware maps each of the GrpReq lines to a priority level from INT0 to INT3 by associating each group with the Interrupt Level (INTLEVEL) field in the corresponding Interrupt Priority Register (IPR). The GrpReq inputs are then masked by the mask bits from the CPU status register. Any interrupt group that has a pending interrupt of a priority level that is not masked by the CPU status register, gets its corresponding ValReq line asserted.

Masking of the interrupt requests is done based on five interrupt mask bits of the CPU status register, namely Interrupt Level 3 Mask (I3M) to Interrupt Level 0 Mask (I0M), and Global Interrupt Mask (GM). An interrupt request is masked if either the GM or the corresponding interrupt level mask bit is set.

The Prioritizer hardware uses the ValReq lines and the INTLEVEL field in the IPRs to select the pending interrupt of the highest priority. If an NMI interrupt request is pending, it automatically gets the highest priority of any pending interrupt. If several interrupt groups of the highest pending interrupt level have pending interrupts, the interrupt group with the highest number is selected.

The INTLEVEL and handler autovector offset (AUTOVECTOR) of the selected interrupt are transmitted to the CPU for interrupt handling and context switching. The CPU doesn't need to know which interrupt is requesting handling, but only the level and the offset of the handler address. The IRR registers contain the interrupt request lines of the groups and can be read via user interface registers for checking which interrupts of the group are actually active.

11.5.1 Non-Maskable Interrupts

A NMI request has priority over all other interrupt requests. NMI has a dedicated exception vector address defined by the AVR32 architecture, so AUTOVECTOR is undefined when INTLEVEL indicates that an NMI is pending.

11.5.2 CPU Response

When the CPU receives an interrupt request it checks if any other exceptions are pending. If no exceptions of higher priority are pending, interrupt handling is initiated. When initiating interrupt handling, the corresponding interrupt mask bit is set automatically for this and lower levels in status register. E.g, if an interrupt of level 3 is approved for handling, the interrupt mask bits I3M, I2M, I1M, and I0M are set in status register. If an interrupt of level 1 is approved, the masking bits I1M and I0M are set in status register. The handler address is calculated by adding AUTOVECTOR to the CPU system register Exception Vector Base Address (EVBA). The CPU will then jump to the calculated address and start executing the interrupt handler.

Setting the interrupt mask bits prevents the interrupts from the same and lower levels to be passed through the interrupt controller. Setting of the same level mask bit prevents also multiple requests of the same interrupt to happen.

It is the responsibility of the handler software to clear the interrupt request that caused the interrupt before returning from the interrupt handler. If the conditions that caused the interrupt are not cleared, the interrupt request remains active.

11.5.3 Clearing an Interrupt Request

Clearing of the interrupt request is done by writing to registers in the corresponding peripheral module, which then clears the corresponding NMIREQ/IREQ signal.

The recommended way of clearing an interrupt request is a store operation to the controlling peripheral register, followed by a dummy load operation from the same register. This causes a pipeline stall, which prevents the interrupt from accidentally re-triggering in case the handler is exited and the interrupt mask is cleared before the interrupt request is cleared.

11.6 User Interface

.

Table 11-1. INTC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Priority Register 0	IPR0	Read/Write	0x00000000
0x004	Interrupt Priority Register 1	IPR1	Read/Write	0x00000000
...	...	...	...	...
0x0FC	Interrupt Priority Register 63	IPR63	Read/Write	0x00000000
0x100	Interrupt Request Register 0	IRR0	Read-only	N/A
0x104	Interrupt Request Register 1	IRR1	Read-only	N/A
...	...	...	...	...
0x1FC	Interrupt Request Register 63	IRR63	Read-only	N/A
0x200	Interrupt Cause Register 3	ICR3	Read-only	N/A
0x204	Interrupt Cause Register 2	ICR2	Read-only	N/A
0x208	Interrupt Cause Register 1	ICR1	Read-only	N/A
0x20C	Interrupt Cause Register 0	ICR0	Read-only	N/A

11.6.1 Interrupt Priority Registers

Register Name: IPR0...IPR63

Access Type: Read/Write

Offset: 0x000 - 0x0FC

Reset Value: 0x00000000

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

- **INTLEVEL: Interrupt Level**

  Indicates the EVBA-relative offset of the interrupt handler of the corresponding group:

  00: INT0.

  01: INT1.

  10: INT2.

  11: INT3.

- **AUTOVECTOR: Autovector Address**

  Handler offset is used to give the address of the interrupt handler. The least significant bit should be written to zero to give halfword alignment.

11.6.2 Interrupt Request Registers

Name: IRR0...IRR63

Access Type: Read-only

Offset: 0x0FF - 0x1FC

Reset Value: N/A

31	30	29	28	27	26	25	24
IRR[32*x+31]	IRR[32*x+30]	IRR[32*x+29]	IRR[32*x+28]	IRR[32*x+27]	IRR[32*x+26]	IRR[32*x+25]	IRR[32*x+24]
23	22	21	20	19	18	17	16
IRR[32*x+23]	IRR[32*x+22]	IRR[32*x+21]	IRR[32*x+20]	IRR[32*x+19]	IRR[32*x+18]	IRR[32*x+17]	IRR[32*x+16]
15	14	13	12	11	10	9	8
IRR[32*x+15]	IRR[32*x+14]	IRR[32*x+13]	IRR[32*x+12]	IRR[32*x+11]	IRR[32*x+10]	IRR[32*x+9]	IRR[32*x+8]
7	6	5	4	3	2	1	0
IRR[32*x+7]	IRR[32*x+6]	IRR[32*x+5]	IRR[32*x+4]	IRR[32*x+3]	IRR[32*x+2]	IRR[32*x+1]	IRR[32*x+0]

- **IRR: Interrupt Request line**

  This bit is cleared when no interrupt request is pending on this input request line.

  This bit is set when an interrupt request is pending on this input request line.

  The are 64 IRRs, one for each group. Each IRR has 32 bits, one for each possible interrupt request, for a total of 2048 possible input lines. The IRRs are read by the software interrupt handler in order to determine which interrupt request is pending. The IRRs are sampled continuously, and are read-only.

11.6.3 Interrupt Cause Registers

Register Name: ICR0...ICR3

Access Type: Read-only

Offset: 0x200 - 0x20C

Reset Value: N/A

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

- **CAUSE: Interrupt Group Causing Interrupt of Priority n**

  ICRn identifies the group with the highest priority that has a pending interrupt of level n. This value is only defined when at least one interrupt of level n is pending.

0.1 Interrupt Request Signal Map

The various modules may output Interrupt request signals. These signals are routed to the Interrupt Controller (INTC), described in a later chapter. The Interrupt Controller supports up to 64 groups of interrupt requests. Each group can have up to 32 interrupt request signals. All interrupt signals in the same group share the same autovector address and priority level. Refer to the documentation for the individual submodules for a description of the semantics of the different interrupt requests.

The interrupt request signals are connected to the INTC as follows.

Table 0-1. Interrupt Request Signal Map

Group	Line	Module	Signal
0	0	Stiletto CPU with optional MPU and optional OCD	SYSBLOCK COMPARE
1	0	External Interrupt Controller	EIC 0
	1	External Interrupt Controller	EIC 1
	2	External Interrupt Controller	EIC 2
	3	External Interrupt Controller	EIC 3
	4	External Interrupt Controller	EIC 4
	5	External Interrupt Controller	EIC 5
	6	External Interrupt Controller	EIC 6
	7	External Interrupt Controller	EIC 7
	8	Real Time Counter	RTC
	9	Power Manager	PM
2	0	General Purpose Input/Output Controller	GPIO 0
	1	General Purpose Input/Output Controller	GPIO 1
	2	General Purpose Input/Output Controller	GPIO 2
	3	General Purpose Input/Output Controller	GPIO 3
	4	General Purpose Input/Output Controller	GPIO 4
	5	General Purpose Input/Output Controller	GPIO 5
	6	General Purpose Input/Output Controller	GPIO 6
	7	General Purpose Input/Output Controller	GPIO 7
	8	General Purpose Input/Output Controller	GPIO 8
	9	General Purpose Input/Output Controller	GPIO 9
	10	General Purpose Input/Output Controller	GPIO 10
	11	General Purpose Input/Output Controller	GPIO 11
	12	General Purpose Input/Output Controller	GPIO 12
	13	General Purpose Input/Output Controller	GPIO 13

Table 0-1. Interrupt Request Signal Map

3	0	Peripheral DMA Controller	PDCA 0
	1	Peripheral DMA Controller	PDCA 1
	2	Peripheral DMA Controller	PDCA 2
	3	Peripheral DMA Controller	PDCA 3
	4	Peripheral DMA Controller	PDCA 4
	5	Peripheral DMA Controller	PDCA 5
	6	Peripheral DMA Controller	PDCA 6
	7	Peripheral DMA Controller	PDCA 7
4	0	Flash Controller	FLASHC
5	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART0
6	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART1
7	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART2
8	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART3
9	0	Serial Peripheral Interface	SPI0
10	0	Serial Peripheral Interface	SPI1
11	0	Two-wire Master Interface	TWIM0
12	0	Two-wire Master Interface	TWIM1
13	0	Synchronous Serial Controller	SSC
14	0	Timer/Counter	TC00
	1	Timer/Counter	TC01
	2	Timer/Counter	TC02
15	0	Analog to Digital Converter	ADC
16	0	Timer/Counter	TC10
	1	Timer/Counter	TC11
	2	Timer/Counter	TC12
17	0	USB 2.0 OTG Interface	USBB
18	0	SDRAM Controller	SDRAMC
19	0	Audio Bitstream DAC	DAC
20	0	Mulitmedia Card Interface	MCI
21	0	Advanced Encryption Standard	AES

Table 0-1. Interrupt Request Signal Map

22	0	DMA Controller	DMACA BLOCK
	1	DMA Controller	DMACA DSTT
	2	DMA Controller	DMACA ERR
	3	DMA Controller	DMACA SRCT
	4	DMA Controller	DMACA TFR
26	0	Memory Stick Interface	MSI
27	0	Two-wire Slave Interface	TWIS0
28	0	Two-wire Slave Interface	TWIS1
29	0	Error code corrector Hamming and Reed Solomon	RS4

12. External Interrupt Controller (EIC)

Rev: 2.3.1.0

12.1 Features

- **Dedicated interrupt request for each interrupt**
- **Individually maskable interrupts**
- **Interrupt on rising or falling edge**
- **Interrupt on high or low level**
- **Asynchronous interrupts for sleep modes without clock**
- **Filtering of interrupt lines**
- **Maskable NMI interrupt**
- **Keypad scan support**

12.2 Overview

The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked. Each external interrupt can generate an interrupt on rising or falling edge, or high or low level. Every interrupt input has a configurable filter to remove spikes from the interrupt source. Every interrupt pin can also be configured to be asynchronous in order to wake up the part from sleep modes where the CLK_SYNC clock has been disabled.

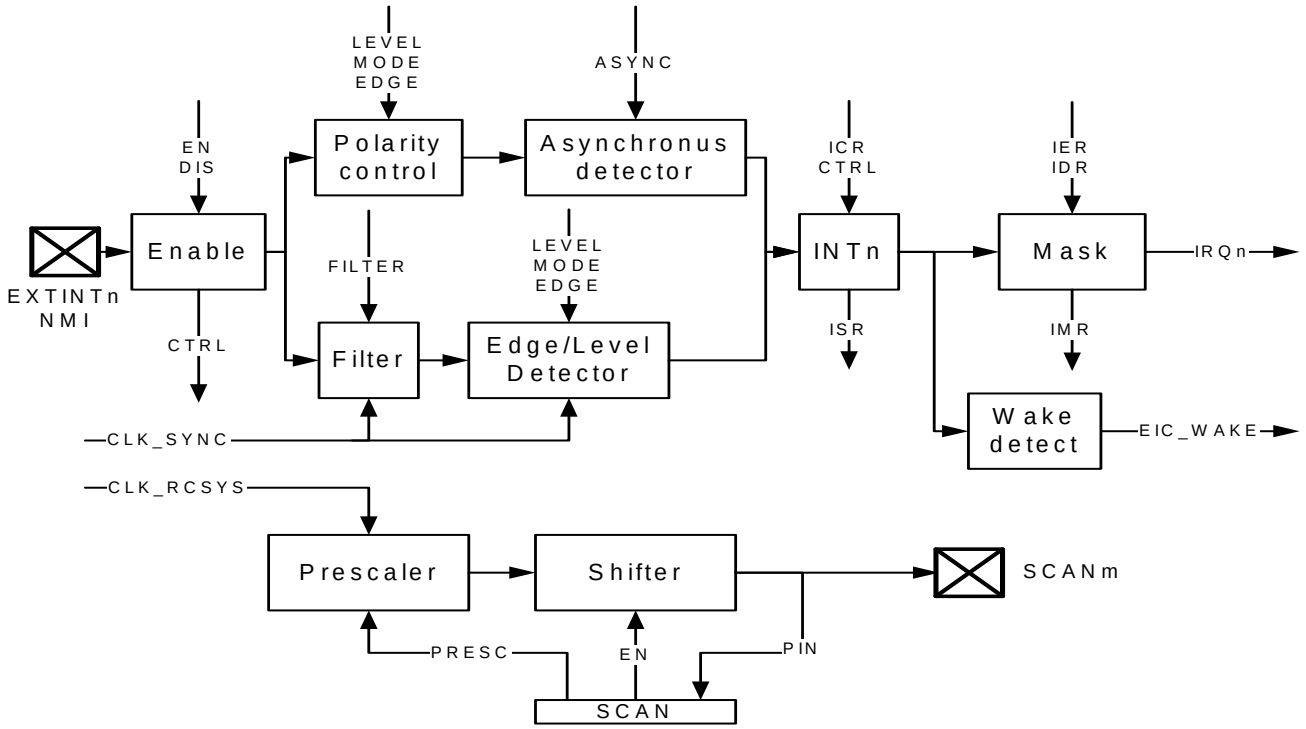
A Non-Maskable Interrupt (NMI) is also supported. This has the same properties as the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

The EIC can wake up the part from sleep modes without triggering an interrupt. In this mode, code execution starts from the instruction following the sleep instruction.

The External Interrupt Controller has support for keypad scanning for keypads laid out in rows and columns. Columns are driven by a separate set of scanning outputs, while rows are sensed by the external interrupt lines. The pressed key will trigger an interrupt, which can be identified through the user registers of the module.

12.3 Block Diagram

Figure 12-1. EIC Block Diagram



12.4 I/O Lines Description

Table 12-1. I/O Lines Description

Pin Name	Pin Description	Type
NMI	Non-Maskable Interrupt	Input
EXTINTn	External Interrupt	Input
SCANm	Keypad scan pin m	Output

12.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

12.5.1 I/O Lines

The external interrupt pins (EXTINTn and NMI) are multiplexed with I/O lines. To generate an external interrupt from an external source the source pin must be configured as an input pins by the I/O Controller. It is also possible to trigger the interrupt by driving these pins from registers in the I/O Controller, or another peripheral output connected to the same pin.

12.5.2 Power Management

All interrupts are available in all sleep modes as long as the EIC module is powered. However, in sleep modes where CLK_SYNC is stopped, the interrupt must be configured to asynchronous mode.

12.5.3 Clocks

The clock for the EIC bus interface (CLK_EIC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager.

The filter and synchronous edge/level detector runs on a clock which is stopped in any of the sleep modes where the system RC oscillator is not running. This clock is referred to as CLK_SYNC. Refer to the Module Configuration section at the end of this chapter for details.

The Keypad scan function operates on the system RC oscillator clock CLK_RCSYS.

12.5.4 Interrupts

The external interrupt request lines are connected to the interrupt controller. Using the external interrupts requires the interrupt controller to be programmed first.

Using the Non-Maskable Interrupt does not require the interrupt controller to be programmed.

12.5.5 Debug Operation

The EIC is frozen during debug operation, unless the OCD system keeps peripherals running during debug operation.

12.6 Functional Description

12.6.1 External Interrupts

The external interrupts are not enabled by default, allowing the proper interrupt vectors to be set up by the CPU before the interrupts are enabled.

Each external interrupt INTn can be configured to produce an interrupt on rising or falling edge, or high or low level. External interrupts are configured by the MODE, EDGE, and LEVEL registers. Each interrupt n has a bit INTn in each of these registers. Writing a zero to the INTn bit in the MODE register enables edge triggered interrupts, while writing a one to the bit enables level triggered interrupts.

If INTn is configured as an edge triggered interrupt, writing a zero to the INTn bit in the EDGE register will cause the interrupt to be triggered on a falling edge on EXTINTn, while writing a one to the bit will cause the interrupt to be triggered on a rising edge on EXTINTn.

If INTn is configured as a level triggered interrupt, writing a zero to the INTn bit in the LEVEL register will cause the interrupt to be triggered on a low level on EXTINTn, while writing a one to the bit will cause the interrupt to be triggered on a high level on EXTINTn.

Each interrupt has a corresponding bit in each of the interrupt control and status registers. Writing a one to the INTn bit in the Interrupt Enable Register (IER) enables the external interrupt from pin EXTINTn to propagate from the EIC to the interrupt controller, while writing a one to INTn bit in the Interrupt Disable Register (IDR) disables this propagation. The Interrupt Mask Register (IMR) can be read to check which interrupts are enabled. When an interrupt triggers, the corresponding bit in the Interrupt Status Register (ISR) will be set. This bit remains set until a one is written to the corresponding bit in the Interrupt Clear Register (ICR) or the interrupt is disabled.

Writing a one to the INTn bit in the Enable Register (EN) enables the external interrupt on pin EXTINTn, while writing a one to INTn bit in the Disable Register (DIS) disables the external interrupt. The Control Register (CTRL) can be read to check which interrupts are enabled. If a bit in the CTRL register is set, but the corresponding bit in IMR is not set, an interrupt will not propa-

gate to the interrupt controller. However, the corresponding bit in ISR will be set, and EIC_WAKE will be set.

If the CTRL.INTn bit is zero, then the corresponding bit in ISR will always be zero. Disabling an external interrupt by writing to the DIS.INTn bit will clear the corresponding bit in ISR.

12.6.2 Synchronization and Filtering of External Interrupts

In synchronous mode the pin value of the EXTINTn pin is synchronized to CLK_SYNC, so spikes shorter than one CLK_SYNC cycle are not guaranteed to produce an interrupt. The synchronization of the EXTINTn to CLK_SYNC will delay the propagation of the interrupt to the interrupt controller by two cycles of CLK_SYNC, see Figure 12-2 on page 102 and Figure 12-3 on page 102 for examples (FILTER off).

It is also possible to apply a filter on EXTINTn by writing a one to INTn bit in the FILTER register. This filter is a majority voter, if the condition for an interrupt is true for more than one of the latest three cycles of CLK_SYNC the interrupt will be set. This will additionally delay the propagation of the interrupt to the interrupt controller by one or two cycles of CLK_SYNC, see Figure 12-2 on page 102 and Figure 12-3 on page 102 for examples (FILTER on).

Figure 12-2. Timing Diagram, Synchronous Interrupts, High Level or Rising Edge

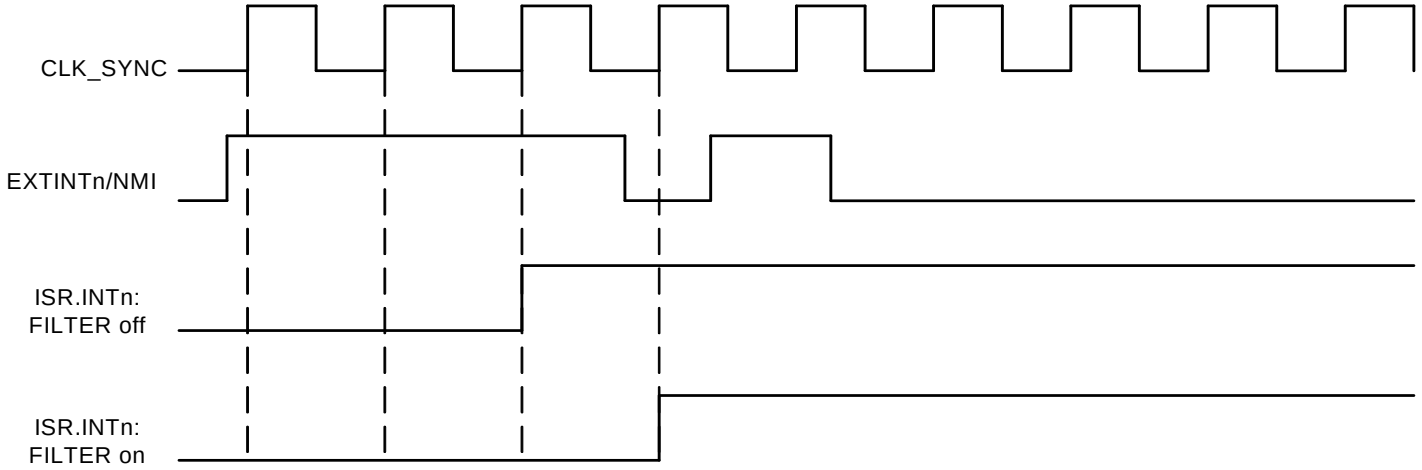
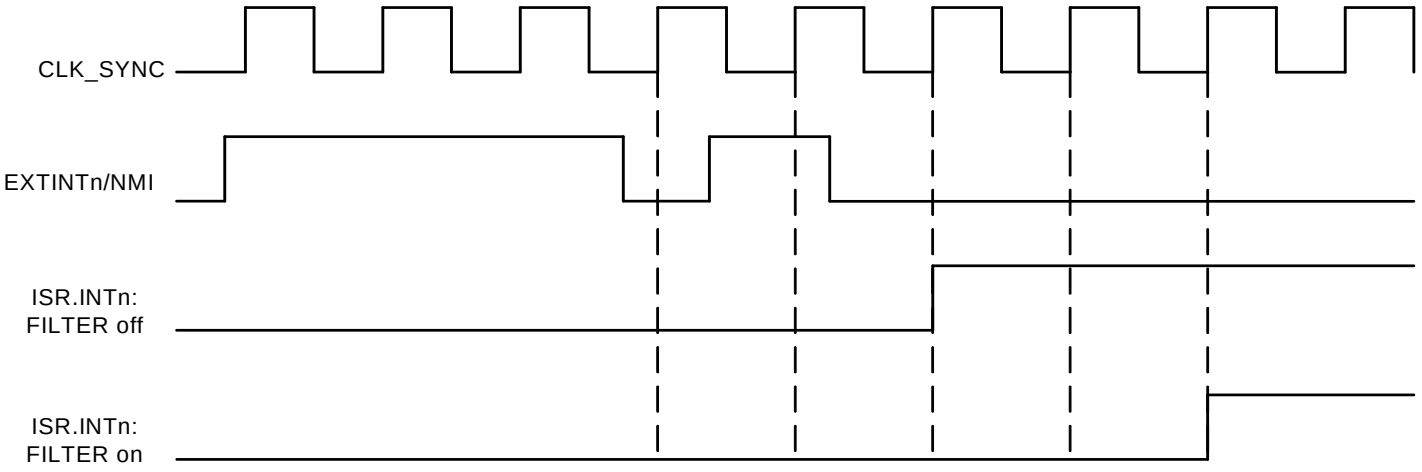


Figure 12-3. Timing Diagram, Synchronous Interrupts, Low Level or Falling Edge



12.6.3 Non-Maskable Interrupt

The NMI supports the same features as the external interrupts, and is accessed through the same registers. The description in Section 12.6.1 should be followed, accessing the NMI bit instead of the INTn bits.

The NMI is non-maskable within the CPU in the sense that it can interrupt any other execution mode. Still, as for the other external interrupts, the actual NMI input can be enabled and disabled by accessing the registers in the EIC.

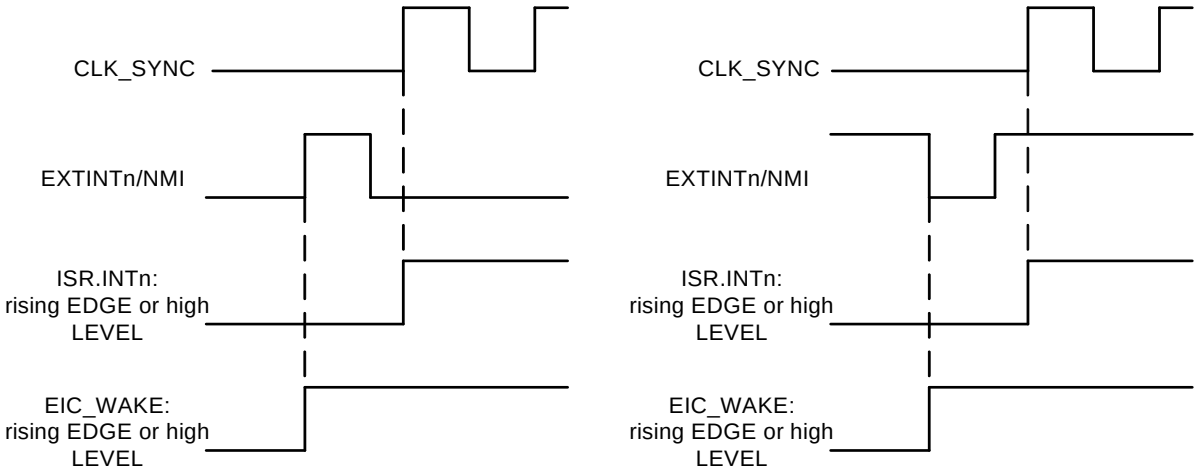
12.6.4 Asynchronous Interrupts

Each external interrupt can be made asynchronous by writing a one to INTn in the ASYNC register. This will route the interrupt signal through the asynchronous path of the module. All edge interrupts will be interpreted as level interrupts and the filter is disabled. If an interrupt is configured as edge triggered interrupt in asynchronous mode, a zero in EDGE.INTn will be interpreted as low level, and a one in EDGE.INTn will be interpreted as high level.

EIC_WAKE will be set immediately after the source triggers the interrupt, while the corresponding bit in ISR and the interrupt to the interrupt controller will be set on the next rising edge of CLK_SYNC. Please refere to Figure 12-4 on page 103 for details.

When CLK_SYNC is stopped only asynchronous interrupts remain active, and any short spike on this interrupt will wake up the device. EIC_WAKE will restart CLK_SYNC and ISR will be updated on the first rising edge of CLK_SYNC.

Figure 12-4. Timing Diagram, Asynchronous Interrupts



12.6.5 Wakeup

The external interrupts can be used to wake up the part from sleep modes. The wakeup can be interpreted in two ways. If the corresponding bit in IMR is one, then the execution starts at the interrupt handler for this interrupt. If the bit in IMR is zero, then the execution starts from the next instruction after the sleep instruction.

12.6.6 Keypad scan support

The External Interrupt Controller also includes support for keypad scanning. The keypad scan feature is compatible with keypads organized as rows and columns, where a row is shorted against a column when a key is pressed.

The rows should be connected to the external interrupt pins with pull-ups enabled in the I/O Controller. These external interrupts should be enabled as low level or falling edge interrupts. The columns should be connected to the available scan pins. The I/O Controller must be configured to let the required scan pins be controlled by the EIC. Unused external interrupt or scan pins can be left controlled by the I/O Controller or other peripherals.

The Keypad Scan function is enabled by writing SCAN.EN to 1, which starts the keypad scan counter. The SCAN outputs are tri-stated, except SCAN[0], which is driven to zero. After $2^{(SCAN.PRESC+1)}$ RC clock cycles this pattern is left shifted, so that SCAN[1] is driven to zero while the other outputs are tri-stated. This sequence repeats infinitely, wrapping from the most significant SCAN pin to SCAN[0].

When a key is pressed, the pulled-up row is driven to zero by the column, and an external interrupt triggers. The scanning stops, and the software can then identify the key pressed by the interrupt status register and the SCAN.PINS value.

The scanning stops whenever there is an active interrupt request from the EIC to the CPU. When the CPU clears the interrupt flags, scanning resumes.

12.7 User Interface

Table 12-2. EIC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Enable Register	IER	Write-only	0x00000000
0x004	Interrupt Disable Register	IDR	Write-only	0x00000000
0x008	Interrupt Mask Register	IMR	Read-only	0x00000000
0x00C	Interrupt Status Register	ISR	Read-only	0x00000000
0x010	Interrupt Clear Register	ICR	Write-only	0x00000000
0x014	Mode Register	MODE	Read/Write	0x00000000
0x018	Edge Register	EDGE	Read/Write	0x00000000
0x01C	Level Register	LEVEL	Read/Write	0x00000000
0x020	Filter Register	FILTER	Read/Write	0x00000000
0x024	Test Register	TEST	Read/Write	0x00000000
0x028	Asynchronous Register	ASYNC	Read/Write	0x00000000
0x2C	Scan Register	SCAN	Read/Write	0x00000000
0x030	Enable Register	EN	Write-only	0x00000000
0x034	Disable Register	DIS	Write-only	0x00000000
0x038	Control Register	CTRL	Read-only	0x00000000

12.7.1 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x000

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in IMR.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in IMR.

12.7.2 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x004

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.

12.7.3 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x008

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The corresponding interrupt is disabled.
  1: The corresponding interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is disabled.
  1: The Non-Maskable Interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.

12.7.4 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x00C

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: An interrupt event has not occurred
  1: An interrupt event has occurred
  This bit is cleared by writing a one to the corresponding bit in ICR.
- **NMI: Non-Maskable Interrupt**
  0: An interrupt event has not occurred
  1: An interrupt event has occurred
  This bit is cleared by writing a one to the corresponding bit in ICR.

12.7.5 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x010

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in ISR.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in ISR.

12.7.6 Mode Register

Name: MODE

Access Type: Read/Write

Offset: 0x014

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The external interrupt is edge triggered.
  1: The external interrupt is level triggered.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is edge triggered.
  1: The Non-Maskable Interrupt is level triggered.

12.7.7 Edge Register

Name: EDGE

Access Type: Read/Write

Offset: 0x018

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The external interrupt triggers on falling edge.
  1: The external interrupt triggers on rising edge.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt triggers on falling edge.
  1: The Non-Maskable Interrupt triggers on rising edge.

12.7.8 Level Register

Name: LEVEL

Access Type: Read/Write

Offset: 0x01C

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The external interrupt triggers on low level.
  1: The external interrupt triggers on high level.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt triggers on low level.
  1: The Non-Maskable Interrupt triggers on high level.

12.7.9 Filter Register

Name: FILTER

Access Type: Read/Write

Offset: 0x020

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The external interrupt is not filtered.
  1: The external interrupt is filtered.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is not filtered.
  1: The Non-Maskable Interrupt is filtered.

12.7.10 Test Register

Name: TEST

Access Type: Read/Write

Offset: 0x024

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **TESTEN: Test Enable**
  0: This bit disables external interrupt test mode.
  1: This bit enables external interrupt test mode.
- **INTn: External Interrupt n**
  If TESTEN is 1, the value written to this bit will be the value to the interrupt detector and the value on the pad will be ignored.
- **NMI: Non-Maskable Interrupt**
  If TESTEN is 1, the value written to this bit will be the value to the interrupt detector and the value on the pad will be ignored.

12.7.11 Asynchronous Register

Name: ASYNC

Access Type: Read/Write

Offset: 0x028

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
-	-	-	-	-	-	-	NMI
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The external interrupt is synchronized to CLK_SYNC.
  1: The external interrupt is asynchronous.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is synchronized to CLK_SYNC
  1: The Non-Maskable Interrupt is asynchronous.

12.7.12 Scan Register

Name: SCAN

Access Type: Read/Write

Offset: 0x2C

Reset Value: 0x0000000

31	30	29	28	27	26	25	24
-	-	-	-	-	PIN[2:0]		

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

15	14	13	12	11	10	9	8
-	-	-	PRESC[4:0]				

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **EN**

  0: Keypad scanning is disabled

  1: Keypad scanning is enabled

- **PRESC**

  Prescale select for the keypad scan rate:

  Scan rate = $2^{(SCAN:PRESC+1)}$ T_{RC}

  The RC clock period can be found in the Electrical Characteristics section.

- **PIN**

  The index of the currently active scan pin. Writing to this bitfield has no effect.

12.7.13 Enable Register

Name: EN

Access Type: Write-only

Offset: 0x030

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable the corresponding external interrupt.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable the Non-Maskable Interrupt.

12.7.14 Disable Register

Name: DIS

Access Type: Write-only

Offset: 0x034

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable the corresponding external interrupt.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable the Non-Maskable Interrupt.

12.7.15 Control Register

Name: CTRL

Access Type: Read-only

Offset: 0x038

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
-	-	-	-	-	-	-	NMI

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **INTn: External Interrupt n**
  0: The corresponding external interrupt is disabled.
  1: The corresponding external interrupt is enabled.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is disabled.
  1: The Non-Maskable Interrupt is enabled.

13. Flash Controller (FLASHC)

Rev: 2.1.0.4

13.1 Features

- **Controls flash block with dual read ports allowing staggered reads.**
- **Supports 0 and 1 wait state bus access.**
- **Allows interleaved burst reads for systems with one wait state, outputting one 32-bit word per clock cycle.**
- **32-bit HSB interface for reads from flash array and writes to page buffer.**
- **32-bit PB interface for issuing commands to and configuration of the controller.**
- **16 lock bits, each protecting a region consisting of (total number of pages in the flash block / 16) pages.**
- **Regions can be individually protected or unprotected.**
- **Additional protection of the Boot Loader pages.**
- **Supports reads and writes of general-purpose NVM bits.**
- **Supports reads and writes of additional NVM pages.**
- **Supports device protection through a security bit.**
- **Dedicated command for chip-erase, first erasing all on-chip volatile memories before erasing flash and clearing security bit.**
- **Interface to Power Manager for power-down of flash-blocks in sleep mode.**
-

13.2 Overview

The flash controller (FLASHC) interfaces a flash block with the 32-bit internal High-Speed Bus (HSB). Performance for uncached systems with high clock-frequency and one wait state is increased by placing words with sequential addresses in alternating flash subblocks. Having one read interface per subblock allows them to be read in parallel. While data from one flash subblock is being output on the bus, the sequential address is being read from the other flash subblock and will be ready in the next clock cycle.

The controller also manages the programming, erasing, locking and unlocking sequences with dedicated commands.

13.3 Product dependencies

13.3.1 Power Manager

The FLASHC has two bus clocks connected: One High speed bus clock (CLK_FLASHC_HSB) and one Peripheral bus clock (CLK_FLASHC_PB). These clocks are generated by the Power manager. Both clocks are turned on by default, but the user has to ensure that CLK_FLASHC_HSB is not turned off before reading the flash or writing the pagebuffer and that CLK_FLASHC_PB is not turned off before accessing the FLASHC configuration and control registers.

13.3.2 Interrupt Controller

The FLASHC interrupt lines are connected to internal sources of the interrupt controller. Using FLASHC interrupts requires the interrupt controller to be programmed first.

13.4 Functional description

13.4.1 Bus interfaces

The FLASHC has two bus interfaces, one HSB interface for reads from the flash array and writes to the page buffer, and one Peripheral Bus (PB) interface for writing commands and control to and reading status from the controller.

13.4.2 Memory organization

To maximize performance for high clock-frequency systems, FLASHC interfaces to a flash block with two read ports. The flash block has several parameters, given by the design of the flash block. Refer to the “Memories” chapter for the device-specific values of the parameters.

- *p* pages (*FLASH_P*)
- *w* words in each page and in the page buffer (*FLASH_W*)
- *pw* words in total (*FLASH_PW*)
- *f* general-purpose fuse bits (*FLASH_F*)
- 1 security fuse bit
- 1 User Page

13.4.3 User page

The User page is an additional page, outside the regular flash array, that can be used to store various data, like calibration data and serial numbers. This page is not erased by regular chip erase. The User page can only be written and erased by proprietary commands. Read accesses to the User page is performed just as any other read access to the flash. The address map of the User page is given in Figure 13-1.

13.4.4 Read operations

The FLASHC provides two different read modes:

- 0 wait state (0ws) for clock frequencies < (access time of the flash plus the bus delay)
- 1 wait state (1ws) for clock frequencies < (access time of the flash plus the bus delay)/2

Higher clock frequencies that would require more wait states are not supported by the flash controller.

The programmer can select the wait states required by writing to the FWS field in the Flash Control Register (FCR). It is the responsibility of the programmer to select a number of wait states compatible with the clock frequency and timing characteristics of the flash block.

In 0ws mode, only one of the two flash read ports is accessed. The other flash read port is idle. In 1ws mode, both flash read ports are active. One read port reading the addressed word, and the other reading the next sequential word.

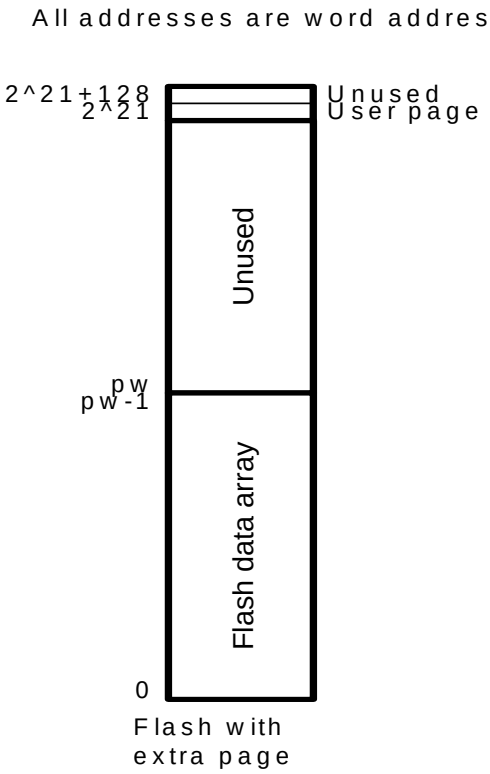
If the clock frequency allows, the user should use 0ws mode, because this gives the lowest power consumption for low-frequency systems as only one flash read port is read. Using 1ws mode has a power/performance ratio approaching 0ws mode as the clock frequency approaches twice the max frequency of 0ws mode. Using two flash read ports use twice the power, but also give twice the performance.

The flash controller supports flash blocks with up to 2^21 word addresses, as displayed in Figure 13-1. Reading the memory space between address *pw* and 2^21-1 returns an undefined result. The User page is permanently mapped to word address 2^21.

Table 13-1. User row addresses

Memory type	Start address, byte sized	Size
Main array	0	*pw* words = 4*pw* bytes
User	2^23 = 8388608	128 words = 512 bytes

Figure 13-1. Memory map for the Flash memories



13.4.5 Quick Page Read

A dedicated command, Quick Page Read (QPR), is provided to read all words in an addressed page. All bits in all words in this page are AND'ed together, returning a 1-bit result. This result is placed in the Quick Page Read Result (QPRR) bit in Flash Status Register (FSR). The QPR command is useful to check that a page is in an erased state. The QPR instruction is much faster than performing the erased-page check using a regular software subroutine.

13.4.6 Write page buffer operations

The internal memory area reserved for the embedded flash can also be written through a write-only page buffer. The page buffer is addressed only by the address bits required to address *w* words (since the page buffer is word addressable) and thus wrap around within the internal memory area address space and appear to be repeated within it.

When writing to the page buffer, the PAGEN field in the Flash Command register (FCMD) is updated with the page number corresponding to page address of the latest word written into the page buffer.

The page buffer is also used for writes to the User page.

Write operations can be prevented by programming the Memory Protection Unit of the CPU. Writing 8-bit and 16-bit data to the page buffer is not allowed and may lead to unpredictable data corruption.

Page buffer write operations are performed with 4 wait states.

Writing to the page buffer can only change page buffer bits from one to zero, i.e. writing 0xaaaaaaaa to a page buffer location that has the value 0x00000000, will not change the page buffer value. The only way to change a bit from zero to one, is to reset the entire page buffer with the Clear Page Buffer command.

The page buffer is not automatically reset after a page write. The programmer should do this manually by issuing the Clear Page Buffer flash command. This can be done after a page write, or before the page buffer is loaded with data to be stored to the flash page.

Example: Writing a word into word address 130 of a flash with 128 words in the page buffer. PAGEN will be updated with the value 1, and the word will be written into word 2 in the page buffer.

13.4.7 Writing words to a page that is not completely erased

This can be used for EEPROM emulation, i.e. writes with granularity of one word instead of an entire page. Only words that are in an completely erased state (0xFFFFFFFF) can be changed. The procedure is as follows:

1. Clear page buffer
2. Write to the page buffer the result of the logical bitwise AND operation between the contents of the flash page and the new data to write. Only words that were in an erased state can be changed from the original page.
3. Write Page.

13.5 Flash commands

The FLASHC offers a command set to manage programming of the flash memory, locking and unlocking of regions, and full flash erasing. See chapter 13.8.2 for a complete list of commands.

To run a command, the field FCMD.CMD has to be written with the command number. As soon as FCMD is written, the FRDY bit is automatically cleared. Once the current command is complete, the FRDY bit is automatically set. If an interrupt has been enabled by setting the bit FRDY in FCR, the interrupt line of the flash controller is activated. All flash commands except for Quick Page Read (QPR) will generate an interrupt request upon completion if FRDY is set.

After a command has been written to FCMD, the programming algorithm should wait until the command has been executed before attempting to read instructions or data from the flash or writing to the page buffer, as the flash will be busy. The waiting can be performed either by polling the Flash Status Register (FSR) or by waiting for the flash ready interrupt. The command written to FCMD is initiated on the first clock cycle where the HSB bus interface in FLASHC is IDLE. The user must make sure that the access pattern to the FLASHC HSB interface contains an IDLE cycle so that the command is allowed to start. Make sure that no bus masters such as DMA controllers are performing endless burst transfers from the flash. Also, make sure that the CPU does not perform endless burst transfers from flash. This is done by letting the CPU enter sleep mode after writing to FCMD, or by polling FSR for command completion. This polling will result in an access pattern with IDLE HSB cycles.

All the commands are protected by the same keyword, which has to be written in the eight highest bits of FCMD. Writing FCMD with data that does not contain the correct key and/or with an invalid command has no effect on the flash memory; however, the PROGE bit is set in FSR. This bit is automatically cleared by a read access to FSR.

Writing a command to FCMD while another command is being executed has no effect on the flash memory; however, the PROGE bit is set in FSR. This bit is automatically cleared by a read access to FSR.

If the current command writes or erases a page in a locked region, or a page protected by the BOOTPROT fuses, the command has no effect on the flash memory; however, the LOCKE bit is set in FSR . This bit is automatically cleared by a read access to FSR.

13.5.1 Write/erase page operation

Flash technology requires that an erase must be done before programming. The entire flash can be erased by an Erase All command. Alternatively, pages can be individually erased by the Erase Page command.

The User page can be written and erased using the mechanisms described in this chapter.

After programming, the page can be locked to prevent miscellaneous write or erase sequences. Locking is performed on a per-region basis, so locking a region locks all pages inside the region. Additional protection is provided for the lowermost address space of the flash. This address space is allocated for the Boot Loader, and is protected both by the lock bit(s) corresponding to this address space, and the BOOTPROT[2:0] fuses.

Data to be written are stored in an internal buffer called page buffer. The page buffer contains *w* words. The page buffer wraps around within the internal memory area address space and appears to be repeated by the number of pages in it. Writing of 8-bit and 16-bit data to the page buffer is not allowed and may lead to unpredictable data corruption.

Data must be written to the page buffer before the programming command is written to FCMD. The sequence is as follows:

- Reset the page buffer with the Clear Page Buffer command.
- Fill the page buffer with the desired contents, using only 32-bit access.
- Programming starts as soon as the programming key and the programming command are written to the Flash Command Register. The FCMD.PAGEN field must contain the address of the page to write. PAGEN is automatically updated when writing to the page buffer, but can also be written to directly. The FRDY bit in FSR is automatically cleared when the page write operation starts.
- When programming is completed, the bit FRDY in FSR is set. If an interrupt was enabled by setting the bit FRDY in FCR, the interrupt line of the flash controller is set.

Two errors can be detected in FSR after a programming sequence:

- Programming Error: A bad keyword and/or an invalid command have been written in FCMD.
- Lock Error: The page to be programmed belongs to a locked region. A command must be executed to unlock the corresponding region before programming can start.

13.5.2 Erase All operation

The entire memory is erased if the Erase All command (EA) is written to FCMD. Erase All erases all bits in the flash array. The User page is not erased. All flash memory locations, the general-purpose fuse bits, and the security bit are erased (reset to 0xFF) after an Erase All.

The EA command also ensures that all volatile memories, such as register file and RAMs, are erased before the security bit is erased.

Erase All operation is allowed only if no regions are locked, and the BOOTPROT fuses are programmed with a region size of 0. Thus, if at least one region is locked, the bit LOCKE in FSR is set and the command is cancelled. If the bit LOCKE has been written to 1 in FCR, the interrupt line rises.

When the command is complete, the bit FRDY bit in FSR is set. If an interrupt has been enabled by setting the bit FRDY in FCR, the interrupt line of the flash controller is set. Two errors can be detected in FSR after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in FCMD.
- Lock Error: At least one lock region to be erased is protected, or BOOTPROT is different from 0. 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.

13.5.3 Region lock bits

The flash block has *p* pages, and these pages are grouped into 16 lock regions, each region containing *p*/16 pages. Each region has a dedicated lock bit preventing writing and erasing pages in the region. After production, the device may have some regions locked. These locked regions are reserved for a boot or default application. Locked regions can be unlocked to be erased and then programmed with another application or other data.

To lock or unlock a region, the commands Lock Region Containing Page (LP) and Unlock Region Containing Page (UP) are provided. Writing one of these commands, together with the number of the page whose region should be locked/unlocked, performs the desired operation.

One error can be detected in FSR after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in FCMD.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that lock bits can also be set/cleared using the commands for writing/erasing general-purpose fuse bits, see chapter 13.6. The general-purpose bit being in an erased (1) state means that the region is unlocked.

The lowermost pages in the Flash can additionally be protected by the BOOTPROT fuses, see Section 13.6.

13.6 General-purpose fuse bits

Each flash block has a number of general-purpose fuse bits that the application programmer can use freely. The fuse bits can be written and erased using dedicated commands, and read

through a dedicated Peripheral Bus address. Some of the general-purpose fuse bits are reserved for special purposes, and should not be used for other functions.:

Table 13-2. General-purpose fuses with special functions

General-Purpose fuse number	Name	Usage
15:0	LOCK	Region lock bits.
16	EPFL	External Privileged Fetch Lock. Used to prevent the CPU from fetching instructions from external memories when in privileged mode. This bit can only be changed when the security bit is cleared. The address range corresponding to external memories is device-specific, and not known to the flash controller. This fuse bit is simply routed out of the CPU or bus system, the flash controller does not treat this fuse in any special way, except that it can not be altered when the security bit is set. If the security bit is set, only an external JTAG Chip Erase can clear EPFL. No internal commands can alter EPFL if the security bit is set. When the fuse is erased (i.e. "1"), the CPU can execute instructions fetched from external memories. When the fuse is programmed (i.e. "0"), instructions can not be executed from external memories.
19:17	BOOTPROT	Used to select one of eight different boot loader sizes. Pages included in the bootlegger area can not be erased or programmed except by a JTAG chip erase. BOOTPROT can only be changed when the security bit is cleared. If the security bit is set, only an external JTAG Chip Erase can clear BOOTPROT, and thereby allow the pages protected by BOOTPROT to be programmed. No internal commands can alter BOOTPROT or the pages protected by BOOTPROT if the security bit is set.

The BOOTPROT fuses protects the following address space for the Boot Loader:

Table 13-3. Boot Loader area specified by BOOTPROT

BOOTPROT	Pages protected by BOOTPROT	Size of protected memory
7	None	0
6	0-1	1kByte
5	0-3	2kByte
4	0-7	4kByte
3	0-15	8kByte
2	0-31	16kByte
1	0-63	32kByte
0	0-127	64kByte

To erase or write a general-purpose fuse bit, the commands Write General-Purpose Fuse Bit (WGPB) and Erase General-Purpose Fuse Bit (EGPB) are provided. Writing one of these commands, together with the number of the fuse to write/erase, performs the desired operation.

An entire General-Purpose Fuse byte can be written at a time by using the Program GP Fuse Byte (PGPFB) instruction. A PGPFB to GP fuse byte 2 is not allowed if the flash is locked by the security bit. The PFB command is issued with a parameter in the PAGEN field:

- PAGEN[2:0] - byte to write
- PAGEN[10:3] - Fuse value to write

All General-Purpose fuses can be erased by the Erase All General-Purpose fuses (EAGP) command. An EAGP command is not allowed if the flash is locked by the security bit.

Two errors can be detected in FSR after issuing these commands:

- Programming Error: A bad keyword and/or an invalid command have been written in FCMD.
- Lock Error: A write or erase of any of the special-function fuse bits in Table 13-3 was attempted while the flash is locked by the security bit.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that the 16 lowest general-purpose fuse bits can also be written/erased using the commands for locking/unlocking regions, see Section 13.5.3.

13.7 Security bit

The security bit allows the entire chip to be locked from external JTAG or other debug access for code security. The security bit can be written by a dedicated command, Set Security Bit (SSB). Once set, the only way to clear the security bit is through the JTAG Chip Erase command.

Once the Security bit is set, the following Flash controller commands will be unavailable and return a lock error if attempted:

- Write General-Purpose Fuse Bit (WGPB) to BOOTPROT or EPFL fuses
- Erase General-Purpose Fuse Bit (EGPB) to BOOTPROT or EPFL fuses
- Program General-Purpose Fuse Byte (PGPFB) of fuse byte 2
- Erase All General-Purpose Fuses (EAGPF)

One error can be detected in FSR after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in FCMD.

13.8 User Interface

Table 13-4. FLASHC Register Memory Map

Offset	Register	Name	Access	Reset
0x0	Flash Control Register	FCR	R/W	0
0x4	Flash Command Register	FCMD	R/W	0
0x8	Flash Status Register	FSR	R/W	0 (*)
0xc	Flash General Purpose Fuse Register Hi	FGPFRHI	R	NA (*)
0x10	Flash General Purpose Fuse Register Lo	FGPFRLO	R	NA (*)

(*) The value of the Lock bits is dependent of their programmed state. All other bits in FSR are 0. All bits in FGPFR and FCFR are dependent on the programmed state of the fuses they map to. Any bits in these registers not mapped to a fuse read 0.

13.8.1 Flash Control Register

Name: FCR
Access Type: Read/Write
Offset: 0x00
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
-	FWS	-	-	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.
- **FWS: Flash Wait State**
  0: The flash is read with 0 wait states.
  1: The flash is read with 1 wait state.

13.8.2 Flash Command Register

Name: FCMD
Access Type: Read/Write
Offset: 0x04
Reset value: 0x00000000

FCMD can not be written if the flash is in the process of performing a flash command. Doing so will cause the FCR write to be ignored, and the PROGE bit to be set.

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
PAGEN [15:8]							

15	14	13	12	11	10	9	8
PAGEN [7:0]							

7	6	5	4	3	2	1	0
-	-	CMD					

- **CMD: Command**
  This field defines the flash command. Issuing any unused command will cause the Programming Error bit to be set, and the corresponding interrupt to be requested if FCR.PROGE is set.

Table 13-5. Set of commands

Command	Value	Mnemonic
No operation	0	NOP
Write Page	1	WP
Erase Page	2	EP
Clear Page Buffer	3	CPB
Lock region containing given Page	4	LP
Unlock region containing given Page	5	UP
Erase All	6	EA
Write General-Purpose Fuse Bit	7	WGPB
Erase General-Purpose Fuse Bit	8	EGPB
Set Security Bit	9	SSB
Program GP Fuse Byte	10	PGPFB
Erase All GPFuses	11	EAGPF
Quick Page Read	12	QPR
Write User Page	13	WUP
Erase User Page	14	EUP

Table 13-5. Set of commands

Command	Value	Mnemonic
Quick Page Read User Page	15	QPRUP
High Speed Mode Enable	16	HSEN
High Speed Mode Disable	17	HSDIS

- **PAGEN: Page number**

  The PAGEN field is used to address a page or fuse bit for certain operations. In order to simplify programming, the PAGEN field is automatically updated every time the page buffer is written to. For every page buffer write, the PAGEN field is updated with the page number of the address being written to. Hardware automatically masks writes to the PAGEN field so that only bits representing valid page numbers can be written, all other bits in PAGEN are always 0. As an example, in a flash with 1024 pages (page 0 - page 1023), bits 15:10 will always be 0.

Table 13-6. Semantic of PAGEN field in different commands

Command	PAGEN description
No operation	Not used
Write Page	The number of the page to write
Clear Page Buffer	Not used
Lock region containing given Page	Page number whose region should be locked
Unlock region containing given Page	Page number whose region should be unlocked
Erase All	Not used
Write General-Purpose Fuse Bit	GPFUSE #
Erase General-Purpose Fuse Bit	GPFUSE #
Set Security Bit	Not used
Program GP Fuse Byte	WriteData[7:0], ByteAddress[2:0]
Erase All GP Fuses	Not used
Quick Page Read	Page number
Write User Page	Not used
Erase User Page	Not used
Quick Page Read User Page	Not used
High Speed Mode Enable	Not used
High Speed Mode Disable	Not used

- **KEY: Write protection key**

  This field should be written with the value 0xA5 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.

  This field always reads as 0.

13.8.3 Flash Status Register

Name: FSR
Access Type: Read/Write
Offset: 0x08
Reset value: 0x00000000

31	30	29	28	27	26	25	24
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8

23	22	21	20	19	18	17	16
LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0

15	14	13	12	11	10	9	8
FSZ			-	-	-	-	-

7	6	5	4	3	2	1	0
-	HSEN	QPRR	SECURITY	PROGE	LOCKE	-	FRDY

- **FRDY: Flash Ready Status**
  0: The flash controller is busy and the application must wait before running a new command.
  1: The flash controller is ready to run a new command.
- **LOCKE: Lock Error Status**
  Automatically cleared when FSR is read.
  0: No programming of at least one locked lock region has happened since the last read of FSR.
  1: Programming of at least one locked lock region has happened since the last read of FSR.
- **PROGE: Programming Error Status**
  Automatically cleared when FSR is read.
  0: No invalid commands and no bad keywords were written in FCMD.
  1: An invalid command and/or a bad keyword was/were written in FCMD.
- **SECURITY: Security Bit Status**
  0: The security bit is inactive.
  1: The security bit is active.
- **QPRR: Quick Page Read Result**
  0: The result is zero, i.e. the page is not erased.
  1: The result is one, i.e. the page is erased.
- **HSEN: High Speed Mode Enable**
  0: High Speed Mode disabled.
  1: High Speed Mode enabled.

- **FSZ: Flash Size**
  The size of the flash. Not all device families will provide all flash sizes indicated in the table.

Table 13-7. Flash size

FSZ	Flash Size
0	32 Kbytes
1	64 Kbytes
2	128 Kbytes
3	256 Kbytes
4	384 Kbytes
5	512 Kbytes
6	768 Kbytes
7	1024 Kbytes

- **LOCKx: Lock Region x Lock Status**
  0: The corresponding lock region is not locked.
  1: The corresponding lock region is locked.

13.8.4 Flash General Purpose Fuse Register High

Name: FGPFRHI
Access Type: Read
Offset: 0x0C
Reset value: N/A

31	30	29	28	27	26	25	24
GPF63	GPF62	GPF61	GPF60	GPF59	GPF58	GPF57	GPF56

23	22	21	20	19	18	17	16
GPF55	GPF54	GPF53	GPF52	GPF51	GPF50	GPF49	GPF48

15	14	13	12	11	10	9	8
GPF47	GPF46	GPF45	GPF44	GPF43	GPF42	GPF41	GPF40

7	6	5	4	3	2	1	0
GPF39	GPF38	GPF37	GPF36	GPF35	GPF34	GPF33	GPF32

This register is only used in systems with more than 32 GP fuses.

- **GPFxx: General Purpose Fuse xx**
  0: The fuse has a written/programmed state.
  1: The fuse has an erased state.

13.8.5 Flash General Purpose Fuse Register Low

Name: FGPFRLO
Access Type: Read
Offset: 0x10
Reset value: N/A

31	30	29	28	27	26	25	24
GPF31	GPF30	GPF29	GPF28	GPF27	GPF26	GPF25	GPF24

23	22	21	20	19	18	17	16
GPF23	GPF22	GPF21	GPF20	GPF19	GPF18	GPF17	GPF16

15	14	13	12	11	10	9	8
GPF15	GPF14	GPF13	GPF12	GPF11	GPF10	GPF09	GPF08

7	6	5	4	3	2	1	0
GPF07	GPF06	GPF05	GPF04	GPF03	GPF02	GPF01	GPF00

- **GPFxx: General Purpose Fuse xx**
  0: The fuse has a written/programmed state.
  1: The fuse has an erased state.

13.9 Fuses Settings

The flash block contains 64 general purpose fuses. These 64 fuses can be found in the Flash General Purpose Fuse Register Low (FGPFRLO) and in the Flash General Purpose Fuse Register High (FGPFRHI) of the Flash Controller (FLASHC).

Some of the FGPFRLO fuses have defined meanings outside the FLASHC and are described in this section.

The general purpose fuses are set by a JTAG chip erase.

13.9.1 Flash General Purpose Fuse Register Low (FGPFRLO)

Table 13-8. FGPFRLO Register Description

31	30	29	28	27	26	25	24
GPF31	GPF30	GPF29	BODEN		BODHYST	BODLEVEL[5:4]	
23	22	21	20	19	18	17	16
BODLEVEL[3:0]				BOOTPROT			EPFL
15	14	13	12	11	10	9	8
LOCK[15:8]							
7	6	5	4	3	2	1	0
LOCK[7:0]							

- **BODEN: Brown Out Detector Enable**

Table 13-9. BODEN Field Description

BODEN	Description
0x0	Brown Out Detector (BOD) disabled
0x1	BOD enabled, BOD reset enabled
0x2	BOD enabled, BOD reset disabled
0x3	BOD disabled

- **BODHYST: Brown Out Detector Hystersis**
  0: The BOD hysteresis is disabled
  1: The BOD hysteresis is enabled
- **BODLEVEL: Brown Out Detector Trigger Level**
  This controls the voltage trigger level for the Brown out detector. For value description refer to Electrical Characteristics chapter.
  If the BODLEVEL is set higher than VDDCORE and enabled by fuses, the part will be in constant reset. To recover from this situation, apply an external voltage on VDDCORE that is higher than the BOD Trigger level and disable the BOD.
- **LOCK, EPFL, BOOTPROT**
  These are Flash controller fuses and are described in the FLASHC chapter.

13.9.2 Default Fuse Value

The devices are shipped with the FGPFRLO register value: 0xFFF7FFFF:

- GPF31 fuse set to 0b1. This fuse is used by the pre-programmed USB bootloader.
- GPF30 fuse set to 0b1. This fuse is used by the pre-programmed USB bootloader.
- GPF29 fuse set to 0b1.
- BODEN fuses set to 0b11. BOD is disabled.
- BODHYST fuse set to 0b1. The BOD hystersis is enabled.
- BODLEVEL fuses set to 0b111111. This is the minimum voltage trigger level for BOD.
- BOOTPROT fuses set to 0b011. The bootloader protected size is 8KBytes.
- EPFL fuse set to 0b1. External privileged fetch is not locked.
- LOCK fuses set to 0b1111111111111111. No region locked.

See also the AT32UC3A3 Bootloader user guide document.

After the JTAG chip erase command, the FGPFRLO register value is 0xFFFFFFFF.

13.10 Module configuration

The specific configuration for the FLASHC instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 13-10. Module Configuration

Feature	FLASH
Flash size	256Kbytes
Number of pages	512
Page size	512 bytes

Table 13-11. Module Clock Name

Module name	Clock name	Clock name
FLASHC	CLK_FLASHC_HSB	CLK_FLASHC_PB

14. HSB Bus Matrix (HMATRIX)

Rev: 2.3.0.2

14.1 Features

- **User Interface on peripheral bus**
- **Configurable Number of Masters (Up to sixteen)**
- **Configurable Number of Slaves (Up to sixteen)**
- **One Decoder for Each Master**
- **Three Different Memory Mappings for Each Master (Internal and External boot, Remap)**
- **One Remap Function for Each Master**
- **Programmable Arbitration for Each Slave**
  - **Round-Robin**
  - **Fixed Priority**
- **Programmable Default Master for Each Slave**
  - **No Default Master**
  - **Last Accessed Default Master**
  - **Fixed Default Master**
- **One Cycle Latency for the First Access of a Burst**
- **Zero Cycle Latency for Default Master**
- **One Special Function Register for Each Slave (Not dedicated)**

14.2 Overview

The Bus Matrix implements a multi-layer bus structure, that enables parallel access paths between multiple High Speed Bus (HSB) masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 16 HSB Masters to up to 16 HSB Slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency). The Bus Matrix provides 16 Special Function Registers (SFR) that allow the Bus Matrix to support application specific features.

14.3 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

14.3.1 Clocks

The clock for the HMATRIX bus interface (CLK_HMATRIX) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the HMATRIX before disabling the clock, to avoid freezing the HMATRIX in an undefined state.

14.4 Functional Description

14.4.1 Memory Mapping

The Bus Matrix provides one decoder for every HSB Master Interface. The decoder offers each HSB Master several memory mappings. In fact, depending on the product, each memory area

may be assigned to several slaves. Booting at the same address while using different HSB slaves (i.e. external RAM, internal ROM or internal Flash, etc.) becomes possible.

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

14.4.2 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism reduces latency at first access of a burst or single transfer. This bus granting mechanism sets a different default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters: no default master, last access master and fixed default master.

14.4.2.1 No Default Master

At the end of the current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits low-power mode.

14.4.2.2 Last Access Master

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

14.4.2.3 Fixed Default Master

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

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that set a default master for each slave. The Slave Configuration Register contains two fields: DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Please refer to the Bus Matrix user interface description.

14.4.3 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when conflict cases occur, i.e. when two or more masters try to access the same slave at the same time. One arbiter per HSB slave is provided, thus arbitrating each slave differently.

The Bus Matrix provides the user with the possibility of choosing between 2 arbitration types for each slave:

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

This choice is made via the field ARBT of the Slave Configuration Registers (SCFG).

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

When a re-arbitration must be done, specific conditions apply. See Section 14.4.3.1 ”Arbitration Rules” on page 141.

14.4.3.1 *Arbitration Rules*

Each arbiter has the ability to arbitrate between two or more different master requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle Cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: When a slave is currently doing a single access.
3. End of Burst Cycles: When the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. See Section “•” on page 141.
4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See Section “•” on page 141.

• Undefined Length Burst Arbitration

In order to avoid long slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer. A predicted end of burst is used as a defined length burst transfer and can be selected from among the following five possibilities:

1. Infinite: No predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. One beat bursts: Predicted end of burst is generated at each single transfer inside the INCP transfer.
3. Four beat bursts: Predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
4. Eight beat bursts: Predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
5. Sixteen beat bursts: Predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the field ULBT of the Master Configuration Registers (MCFG).

• Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (SCFG) and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, half word or word transfer.

14.4.3.2 Round-Robin Arbitration

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

There are three round-robin algorithms implemented:

1. Round-Robin arbitration without default master
2. Round-Robin arbitration with last default master
3. Round-Robin arbitration with fixed default master

• Round-Robin Arbitration without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

• Round-Robin Arbitration with Last Default Master

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

• Round-Robin Arbitration with Fixed Default Master

This is another biased round-robin algorithm. It allows the Bus Matrix arbiters to remove the one latency cycle for the fixed default master per slave. At the end of the current access, the slave remains connected to its fixed default master. Every request attempted by this fixed default master will not cause any latency whereas other non privileged masters will still get one latency cycle. This technique can be used for masters that mainly perform single accesses.

14.4.3.3 Fixed Priority Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If two or more master requests are active at the same time, the master with the highest priority number is serviced first. If two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master may be defined through the Priority Registers for Slaves (PRAS and PRBS).

14.4.4 Slave and Master assignation

The index number assigned to Bus Matrix slaves and masters are described in Memories chapter.

14.5 User Interface

Table 14-1. HMATRICX Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Master Configuration Register 0	MCFG0	Read/Write	0x00000002
0x0004	Master Configuration Register 1	MCFG1	Read/Write	0x00000002
0x0008	Master Configuration Register 2	MCFG2	Read/Write	0x00000002
0x000C	Master Configuration Register 3	MCFG3	Read/Write	0x00000002
0x0010	Master Configuration Register 4	MCFG4	Read/Write	0x00000002
0x0014	Master Configuration Register 5	MCFG5	Read/Write	0x00000002
0x0018	Master Configuration Register 6	MCFG6	Read/Write	0x00000002
0x001C	Master Configuration Register 7	MCFG7	Read/Write	0x00000002
0x0020	Master Configuration Register 8	MCFG8	Read/Write	0x00000002
0x0024	Master Configuration Register 9	MCFG9	Read/Write	0x00000002
0x0028	Master Configuration Register 10	MCFG10	Read/Write	0x00000002
0x002C	Master Configuration Register 11	MCFG11	Read/Write	0x00000002
0x0030	Master Configuration Register 12	MCFG12	Read/Write	0x00000002
0x0034	Master Configuration Register 13	MCFG13	Read/Write	0x00000002
0x0038	Master Configuration Register 14	MCFG14	Read/Write	0x00000002
0x003C	Master Configuration Register 15	MCFG15	Read/Write	0x00000002
0x0040	Slave Configuration Register 0	SCFG0	Read/Write	0x00000010
0x0044	Slave Configuration Register 1	SCFG1	Read/Write	0x00000010
0x0048	Slave Configuration Register 2	SCFG2	Read/Write	0x00000010
0x004C	Slave Configuration Register 3	SCFG3	Read/Write	0x00000010
0x0050	Slave Configuration Register 4	SCFG4	Read/Write	0x00000010
0x0054	Slave Configuration Register 5	SCFG5	Read/Write	0x00000010
0x0058	Slave Configuration Register 6	SCFG6	Read/Write	0x00000010
0x005C	Slave Configuration Register 7	SCFG7	Read/Write	0x00000010
0x0060	Slave Configuration Register 8	SCFG8	Read/Write	0x00000010
0x0064	Slave Configuration Register 9	SCFG9	Read/Write	0x00000010
0x0068	Slave Configuration Register 10	SCFG10	Read/Write	0x00000010
0x006C	Slave Configuration Register 11	SCFG11	Read/Write	0x00000010
0x0070	Slave Configuration Register 12	SCFG12	Read/Write	0x00000010
0x0074	Slave Configuration Register 13	SCFG13	Read/Write	0x00000010
0x0078	Slave Configuration Register 14	SCFG14	Read/Write	0x00000010
0x007C	Slave Configuration Register 15	SCFG15	Read/Write	0x00000010
0x0080	Priority Register A for Slave 0	PRAS0	Read/Write	0x00000000
0x0084	Priority Register B for Slave 0	PRBS0	Read/Write	0x00000000
0x0088	Priority Register A for Slave 1	PRAS1	Read/Write	0x00000000

Table 14-1. HMATRICX Register Memory Map (Continued)

Offset	Register	Name	Access	Reset Value
0x008C	Priority Register B for Slave 1	PRBS1	Read/Write	0x00000000
0x0090	Priority Register A for Slave 2	PRAS2	Read/Write	0x00000000
0x0094	Priority Register B for Slave 2	PRBS2	Read/Write	0x00000000
0x0098	Priority Register A for Slave 3	PRAS3	Read/Write	0x00000000
0x009C	Priority Register B for Slave 3	PRBS3	Read/Write	0x00000000
0x00A0	Priority Register A for Slave 4	PRAS4	Read/Write	0x00000000
0x00A4	Priority Register B for Slave 4	PRBS4	Read/Write	0x00000000
0x00A8	Priority Register A for Slave 5	PRAS5	Read/Write	0x00000000
0x00AC	Priority Register B for Slave 5	PRBS5	Read/Write	0x00000000
0x00B0	Priority Register A for Slave 6	PRAS6	Read/Write	0x00000000
0x00B4	Priority Register B for Slave 6	PRBS6	Read/Write	0x00000000
0x00B8	Priority Register A for Slave 7	PRAS7	Read/Write	0x00000000
0x00BC	Priority Register B for Slave 7	PRBS7	Read/Write	0x00000000
0x00C0	Priority Register A for Slave 8	PRAS8	Read/Write	0x00000000
0x00C4	Priority Register B for Slave 8	PRBS8	Read/Write	0x00000000
0x00C8	Priority Register A for Slave 9	PRAS9	Read/Write	0x00000000
0x00CC	Priority Register B for Slave 9	PRBS9	Read/Write	0x00000000
0x00D0	Priority Register A for Slave 10	PRAS10	Read/Write	0x00000000
0x00D4	Priority Register B for Slave 10	PRBS10	Read/Write	0x00000000
0x00D8	Priority Register A for Slave 11	PRAS11	Read/Write	0x00000000
0x00DC	Priority Register B for Slave 11	PRBS11	Read/Write	0x00000000
0x00E0	Priority Register A for Slave 12	PRAS12	Read/Write	0x00000000
0x00E4	Priority Register B for Slave 12	PRBS12	Read/Write	0x00000000
0x00E8	Priority Register A for Slave 13	PRAS13	Read/Write	0x00000000
0x00EC	Priority Register B for Slave 13	PRBS13	Read/Write	0x00000000
0x00F0	Priority Register A for Slave 14	PRAS14	Read/Write	0x00000000
0x00F4	Priority Register B for Slave 14	PRBS14	Read/Write	0x00000000
0x00F8	Priority Register A for Slave 15	PRAS15	Read/Write	0x00000000
0x00FC	Priority Register B for Slave 15	PRBS15	Read/Write	0x00000000
0x0100	Master Remap Control Register	MRCR	Read/Write	0x00000000
0x0110	Special Function Register 0	SFR0	Read/Write	–
0x0114	Special Function Register 1	SFR1	Read/Write	–
0x0118	Special Function Register 2	SFR2	Read/Write	–
0x011C	Special Function Register 3	SFR3	Read/Write	–
0x0120	Special Function Register 4	SFR4	Read/Write	–
0x0124	Special Function Register 5	SFR5	Read/Write	–

Table 14-1. HMATRICX Register Memory Map (Continued)

Offset	Register	Name	Access	Reset Value
0x0128	Special Function Register 6	SFR6	Read/Write	–
0x012C	Special Function Register 7	SFR7	Read/Write	–
0x0130	Special Function Register 8	SFR8	Read/Write	–
0x0134	Special Function Register 9	SFR9	Read/Write	–
0x0138	Special Function Register 10	SFR10	Read/Write	–
0x013C	Special Function Register 11	SFR11	Read/Write	–
0x0140	Special Function Register 12	SFR12	Read/Write	–
0x0144	Special Function Register 13	SFR13	Read/Write	–
0x0148	Special Function Register 14	SFR14	Read/Write	–
0x014C	Special Function Register 15	SFR15	Read/Write	–

14.5.1 Master Configuration Registers

Name: MCFG0...MCFG15

Access Type: Read/Write

Offset: 0x00 - 0x3C

Reset Value: 0x00000002

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

- **ULBT: Undefined Length Burst Type**

  0: Infinite Length Burst

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

  1: Single Access

  The undefined length burst is treated as a succession of single accesses, allowing re-arbitration at each beat of the INCR burst.

  2: Four Beat Burst

  The undefined length burst is split into a four-beat burst, allowing re-arbitration at each four-beat burst end.

  3: Eight Beat Burst

  The undefined length burst is split into an eight-beat burst, allowing re-arbitration at each eight-beat burst end.

  4: Sixteen Beat Burst

  The undefined length burst is split into a sixteen-beat burst, allowing re-arbitration at each sixteen-beat burst end.

14.5.2 Slave Configuration Registers

Name: SCFG0...SCFG15

Access Type: Read/Write

Offset: 0x40 - 0x7C

Reset Value: 0x00000010

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

- **ARBT: Arbitration Type**
  - 0: Round-Robin Arbitration
  - 1: Fixed Priority Arbitration
- **FIXED_DEFMSTR: Fixed Default Master**

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

- **DEFMSTR_TYPE: Default Master Type**
  - 0: No Default Master
  - At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters.
  - This results in a one cycle latency for the first access of a burst transfer or for a single access.
  - 1: Last Default Master
  - At the end of the current slave access, if no other master request is pending, the slave stays connected to the last master having accessed it.
  - This results in not having one cycle latency when the last master tries to access the slave again.
  - 2: Fixed Default Master
  - At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master the number that has been written in the FIXED_DEFMSTR field.
  - This results in not having one cycle latency when the fixed master tries to access the slave again.
- **SLOT_CYCLE: Maximum Number of Allowed Cycles for a Burst**
  - When the SLOT_CYCLE limit is reached for a burst, it may be broken by another master trying to access this slave.
  - This limit has been placed to avoid locking a very slow slave when very long bursts are used.
  - This limit must not be very small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

14.5.3 Bus Matrix Priority Registers A For Slaves

Register Name: PRAS0...PRAS15

Access Type: Read/Write

Offset: -

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
M7PR				M6PR			
23	22	21	20	19	18	17	16
M5PR				M4PR			
15	14	13	12	11	10	9	8
M3PR				M2PR			
7	6	5	4	3	2	1	0
M1PR				M0PR			

- **MxPR: Master x Priority**

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

14.5.4 Priority Registers B For Slaves

Name: PRBS0...PRBS15

Access Type: Read/Write

Offset: -

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
M15PR				M14PR			

23	22	21	20	19	18	17	16
M13PR				M12PR			

15	14	13	12	11	10	9	8
M11PR				M10PR			

7	6	5	4	3	2	1	0
M9PR				M8PR			

- **MxPR: Master x Priority**

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

14.5.5 Master Remap Control Register

Name: MRCR

Access Type: Read/Write

Offset: 0x100

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
RCB15	RCB14	RCB13	RCB12	RCB11	RCB10	RCB9	RCB8

7	6	5	4	3	2	1	0
RCB7	RCB6	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

- **RCB: Remap Command Bit for Master x**
  0: Disable remapped address decoding for the selected Master
  1: Enable remapped address decoding for the selected Master

14.5.6 Special Function Registers

Name: SFR0...SFR15

Access Type: Read/Write

Offset: 0x110 - 0x115

Reset Value: -

31	30	29	28	27	26	25	24
SFR							

23	22	21	20	19	18	17	16
SFR							

15	14	13	12	11	10	9	8
SFR							

7	6	5	4	3	2	1	0
SFR							

- **SFR: Special Function Register Fields**

  Those registers are not a HMATRIX specific register. The field of those will be defined where they are used.

14.6 Bus Matrix Connections

Accesses to unused areas returns an error result to the master requesting such an access.

The bus matrix has the several masters and slaves. Each master has its own bus and its own decoder, thus allowing a different memory mapping per master. The master number in the table below can be used to index the HMATRIX control registers. For example, HMATRIX MCFG0 register is associated with the CPU Data master interface.

Table 14-2. High Speed Bus masters

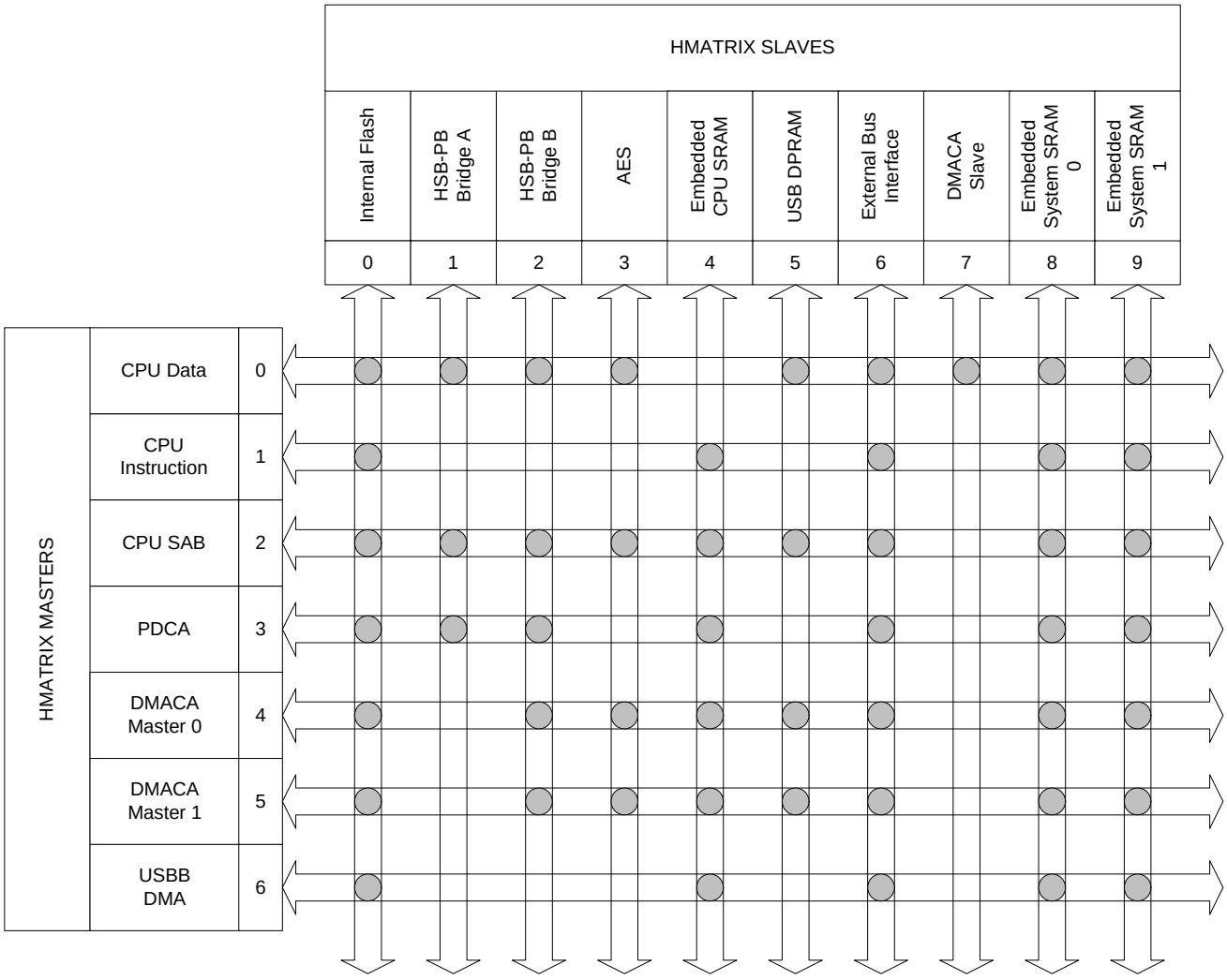
Master 0	CPU Data
Master 1	CPU Instruction
Master 2	CPU SAB
Master 3	PDCA
Master 4	DMACA Master 1
Master 5	DMACA Master 2
Master 6	USBB DMA

Each slave has its own arbiter, thus allowing a different arbitration per slave. The slave number in the table below can be used to index the HMATRIX control registers. For example, HMATRIX SCFG4 register is associated with the Embedded CPU SRAM Slave Interface.

Table 14-3. High Speed Bus slaves

Slave 0	Internal Flash
Slave 1	HSB-PB Bridge A
Slave 2	HSB-PB Bridge B
Slave 3	AES
Slave 4	Embedded CPU SRAM
Slave 5	USBB DPRAM
Slave 6	EBI
Slave 7	DMACA Slave
Slave 8	Embedded System SRAM 0
Slave 9	Embedded System SRAM 1

Figure 14-1. HMATRIX Master / Slave Connections



15. External Bus Interface (EBI)

Rev.: 1.7.0.0

15.1 Features

- **Optimized for application memory space support**
- **Integrates three external memory controllers:**
  - **Static Memory Controller (SMC)**
  - **SDRAM Controller (SDRAMC)**
  - **Error Corrected Code (ECCHRS) controller**
- **Additional logic for NAND Flash/SmartMedia™ 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-bit data bus**
  - **Up to 24-bit Address Bus, Up to 8-Mbytes Addressable**
  - **Optimized pin multiplexing to reduce latencies on external memories**
- **Up to 6 Chip Selects, Configurable Assignment:**
  - **Static Memory Controller on Chip Select 0**
  - **SDRAM Controller or Static Memory Controller on Chip Select 1**
  - **Static Memory Controller on Chip Select 2, Optional NAND Flash support**
  - **Static Memory Controller on Chip Select 3, Optional NAND Flash support**
  - **Static Memory Controller on Chip Select 4, Optional CompactFlash™ support**
  - **Static Memory Controller on Chip Select 5, Optional CompactFlash™ support**

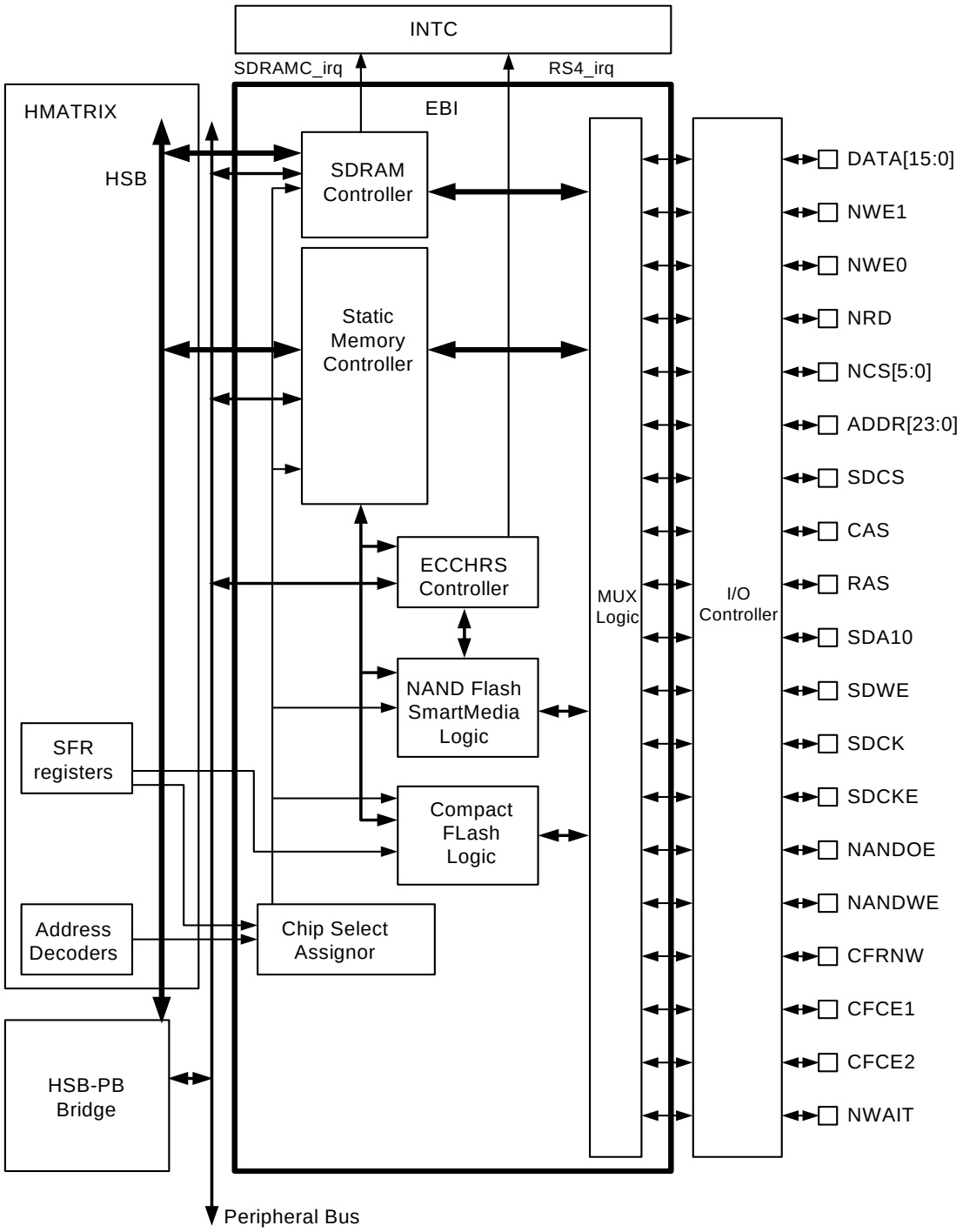
15.2 Overview

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded memory controller of an AVR32 device. The Static Memory, SDRAM and ECCHRS 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/SmartMedia protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded memory controller. Data transfers are performed through a 16-bit, an address bus of up to 23 bits, up to six chip select lines (NCS[5:0]), and several control pins that are generally multiplexed between the different external memory controllers.

15.3 Block Diagram

Figure 15-1. EBI Block Diagram



15.4 I/O Lines Description

Table 15-1. EBI I/O Lines Description

Pin Name	Alternate Name	Pin Description	Type	Active Level
EBI common lines				
DATA[15:0]		Data Bus	I/O	
SMC dedicated lines				
ADDR[1]		SMC Address Bus Line 1	Output	
ADDR[12]		SMC Address Bus Line 12	Output	
ADDR[15]		SMC Address Bus Line 15	Output	
ADDR[23:18]		SMC Address Bus Line [23:18]	Output	
NCS[0]		SMC Chip Select Line 0	Output	Low
NWAIT		SMC External Wait Signal	Input	Low
SDRAMC dedicated lines				
SDCK		SDRAM Clock	Output	
SDCKE		SDRAM Clock Enable	Output	High
SDCS	SDCS1	SDRAM Controller Chip Select Line 1	Output	Low
SDWE		SDRAM Write Enable	Output	Low
SDA10		SDRAM Address Bus Line 10	Output	Low
RAS - CAS		Row and Column Signal	Output	Low
CompactFlash dedicated lines				
CFCE1 - CFCE2		CompactFlash Chip Enable	Output	Low
CFRNW		CompactFlash Read Not Write Signal	Output	
NAND Flash/SmartMedia dedicated lines				
NANDOE		NAND Flash Output Enable	Output	Low
NANDWE		NAND Flash Write Enable	Output	Low
SMC/SDRAMC shared lines				
NCS[1]	NCS[1] SDCS0	SMC Chip Select Line 1 SDRAMC Chip Select Line 0	Output	Low
ADDR[0]	DQM0 ADDR[0]-NBS0	SDRAMC DQM1 SMC Address Bus Line 0 or Byte Select 1	Output	
ADDR[11:2]	ADDR[9:0] ADDR[11:2]	SDRAMC Address Bus Lines [9:0] SMC Address Bus Lines [11:2]	Output	
ADDR[14:13]	ADDR[9:0] ADDR[14:13]	SDRAMC Address Bus Lines [12:11] SMC Address Bus Lines [14:13]	Output	
ADDR[16]	BA0 ADDR[16]	SDRAMC Bank 0 SMC Address Bus Line 16	Output	

Pin Name	Alternate Name	Pin Description	Type	Active Level
ADDR[17]	BA1 ADDR[17]	SDRAMC Bank 1 SMCAddress Bus Line 17	Output	
SMC/CompactFlash shared lines				
NRD	NRD CFNOE	SMC Read Signal CompactFlash CFNOE	Output	Low
NWE0	NWE0-NWE CFNWE	SMC Write Enable10 or Write enable CompactFlash CFNWE	Output	Low
NCS[4]	NCS[4] CFCS[0]	SMC Chip Select Line 4 CompactFlash Chip Select Line 0	Output	Low
NCS[5]	NCS[5] CFCS[1]	SMC Chip Select Line 5 CompactFlash Chip Select Line 1	Output	Low
SMC/NAND Flash/SmartMedia shared lines				
NCS[2]	NCS[2] NANDCS[0]	SMC Chip Select Line 2 NANDFlash/SmartMedia Chip Select Line 0	Output	Low
NCS[3]	NCS[3] NANDCS[1]	SMC Chip Select Line 3 NANDFlash/SmartMedia Chip Select Line 1	Output	Low
SDRAMC/SMC/CompactFlash shared lines				
NWE1	DQM1/ NWE1-NBS1/ CFNIORD	SDRAMC DQM1 SMC Write Enable1 or Byte Select 1 CompactFlash CFNIORD	Output	

15.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

15.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O Controller lines. The user must first configure the I/O Controller to assign the EBI pins to their peripheral functions.

15.5.2 Power Management

To prevent bus errors EBI operation must be terminated before entering sleep mode.

15.5.3 Clocks

A number of clocks can be selected as source for the EBI. The selected clock must be enabled by the Power Manager.

The following clock sources are available:

- CLK_EBI
- CLK_SDRAMC
- CLK_SMC

- CLK_ECCHRS

Refer to Table 15-2 on page 158 to configure those clocks.

Table 15-2. EBI Clocks Configuration

Clocks name	Clocks type	Type of the Interfaced Device			
		SDRAM	SRAM, PROM, EPROM, EEPROM, Flash	NandFlash SmartMedia	CompactFlash
CLK_EBI	HSB	X	X	X	X
CLK_SDRAMC	PB	X			
CLK_SMC	PB		X	X	X
CLK_ECCHRS	PB			X	

15.5.4 Interrupts

The EBI interface has two interrupt lines connected to the Interrupt Controller:

- SDRAMC_IRQ: Interrupt signal coming from the SDRAMC
- RS4_IRQ: Interrupt signal coming from the ECCHRS

Handling the EBI interrupt requires configuring the interrupt controller before configuring the EBI.

15.5.5 HMATRIX

The EBI interface is connected to the HMATRIX Special Function Register 6 (SFR6). The user must first write to this HMATRIX.SFR6 to configure the EBI correctly.

Table 15-3. EBI Special Function Register Fields Description

SFR6 Bit Number	Bit name	Description
[31:6]		Reserved
5	CS5A	0 = Chip Select 5 (NCS[5]) is connected to a Static Memory device. For each access to the NCS[5] memory space, all related pins act as SMC pins 1 = Chip Select 5 (NCS[5]) is connected to a CompactFlash device. For each access to the NCS[5] memory space, all related pins act as CompactFlash pins
4	CS4A	0 = Chip Select 4 (NCS[4]) is connected to a Static Memory device. For each access to the NCS[4] memory space, all related pins act as SMC pins 1 = Chip Select 4 (NCS[4]) is connected to a CompactFlash device. For each access to the NCS[4] memory space, all related pins act as CompactFlash pins
3	CS3A	0 = Chip Select 3 (NCS[3]) is connected to a Static Memory device. For each access to the NCS[3] memory space, all related pins act as SMC pins 1 = Chip Select 3 (NCS[3]) is connected to a NandFlash or a SmartMedia device. For each access to the NCS[3] memory space, all related pins act as NandFlash or SmartMedia pins

Table 15-3. EBI Special Function Register Fields Description

SFR6 Bit Number	Bit name	Description
2	CS2A	0 = Chip Select 2 (NCS[2]) is connected to a Static Memory device. For each access to the NCS[2] memory space, all related pins act as SMC pins 1 = Chip Select 2 (NCS[2]) is connected to a NandFlash or a SmartMedia device. For each access to the NCS[2] memory space, all related pins act as NandFlash or SmartMedia pins
1	CS1A	0 = Chip Select 1 (NCS[1]) is connected to a Static Memory device. For each access to the NCS[1] memory space, all related pins act as SMC pins 1 = Chip Select 1 (NCS[1]) is connected to a SDRAM device. For each access to the NCS[1] memory space, all related pins act as SDRAM pins
0		Reserved

15.6 Functional Description

The EBI transfers data between the internal HSB bus (handled by the HMATRIX) 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 ECCHRS Controller (ECCHRS)
- A chip select assignment feature that assigns an HSB address space to the external devices
- A multiplex controller circuit that shares the pins between the different memory controllers
- Programmable CompactFlash support logic
- Programmable SmartMedia and NAND Flash support logic

15.6.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 16-bit data lines, the address lines of up to 24 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 SDRAMC without delaying the other external memory controller accesses.

15.6.2 Static Memory Controller

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

15.6.3 SDRAM Controller

Writing a one to the HMATRIX.SFR6.CS1A bit enables the SDRAM logic.

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

15.6.4 ECCHRS Controller

For information on the ECCHRS Controller, refer to the ECCHRS Section.

15.6.5 CompactFlash Support

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

The CompactFlash logic is driven by the SMC on the NCS[4] and/or NCS[5] address space. Writing to the HMATRIX.SFR6.CS4A and/or HMATRIX.SFR6.CS5A bits the appropriate value enables this logic. Access to an external CompactFlash device is then made by accessing the address space reserved to NCS[4] and/or NCS[5].

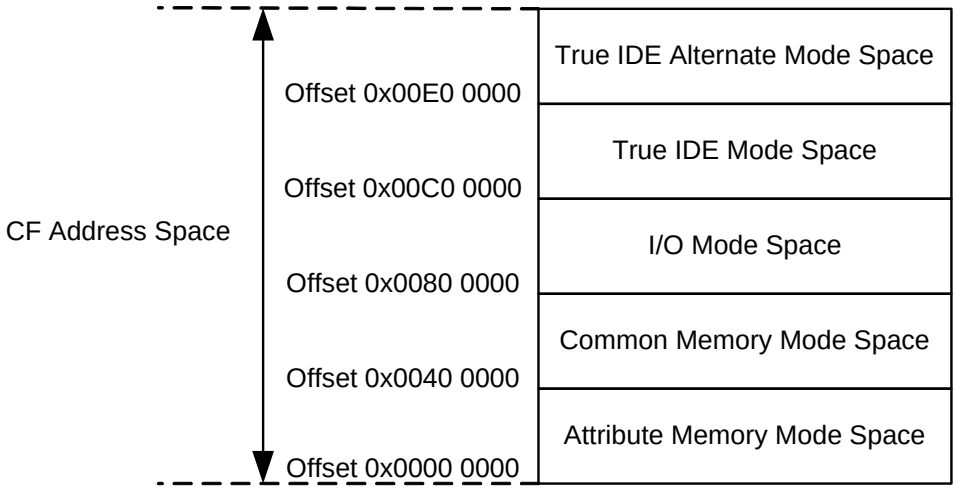
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOWR, _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

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

Within the NCS[4] and/or NCS[5] address space, the current transfer address is used to distinguish I/O mode, common memory mode, attribute memory mode and True IDE mode.

The different modes are accessed through a specific memory mapping as illustrated on Figure 15-2 on page 160. ADDR[23:21] bits of the transfer address are used to select the desired mode as described in Table 15-4 on page 160.

Figure 15-2. CompactFlash Memory Mapping



Note: The ADDR[22] I/O line is used to drive the REG signal of the CompactFlash Device (except in True IDE mode).

Table 15-4. CompactFlash Mode Selection

ADDR[23:21]	Mode Base Address
000	Attribute Memory
010	Common Memory
100	I/O Mode
110	True IDE Mode
111	Alternate True IDE Mode

15.6.5.2 CFCE1 and CFCE2 signals

To cover all types of access, the SMC must be alternatively set to drive 8-bit data bus or 16-bit data bus. The odd byte access on the DATA[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the corresponding NCS pin (NCS[4] or NCS[5]). The Data Bus Width (DBW) field in the SMC Mode (MODE) register of the NCS[4] and/or NCS[5] address space must be written as shown in Table 15-5 on page 161 to enable the required access type.

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

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

Table 15-5. CFCE1 and CFCE2 Truth Table

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

15.6.5.3 *Read/Write signals*

In I/O mode and True IDE mode, the CompactFlash logic drives the read command signals of the SMC on CFNIORD signal, while the CFNOE and CFNWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFNOE and CFNWE signals, while the CFNIORD is deactivated. Figure 15-3 on page 162 demonstrates a schematic representation of this logic.

Attribute memory mode, common memory mode and I/O mode are supported by writing the address setup and hold time on the NCS[4] (and/or NCS[5]) chip select to the appropriate values. For details on these signal waveforms, please refer to the section: Setup and Hold Cycles of the SMC Section.

Figure 15-3. CompactFlash Read/Write Control Signals

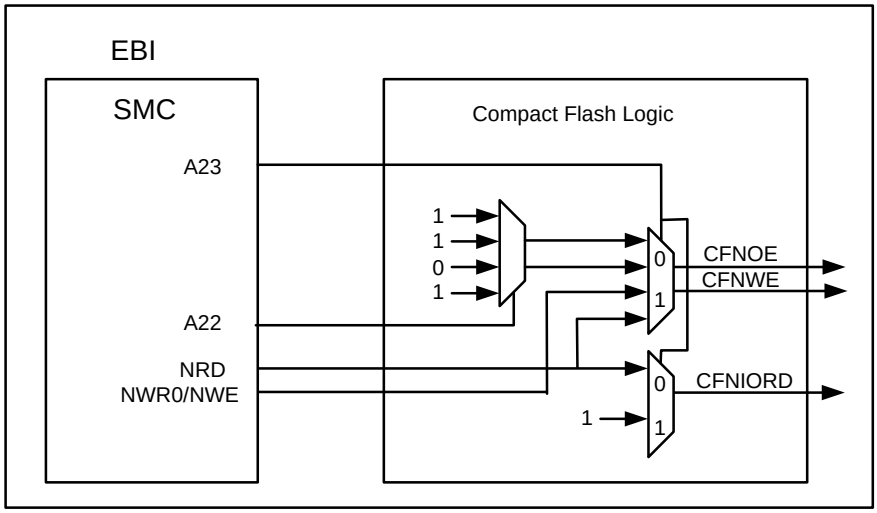


Table 15-6. CompactFlash Mode Selection

Mode Base Address	CFNOE	CFNWE	CFNIORD
Attribute Memory Common Memory	NRD_NOE	NWR0_NWE	1
I/O Mode	1	1	NRD_NOE
True IDE Mode	0	1	NRD_NOE

15.6.5.4 *Multiplexing of CompactFlash signals on EBI pins*

Table 15-7 on page 163 and Table on page 163 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 15-7 on page 163 are strictly dedicated to the CompactFlash interface as soon as the HMATRIX.SFR6.CS4A and/or HMATRIX.SFR6.CS5A bits is/are written. These pins must not be used to drive any other memory devices.

The EBI pins in Table 15-8 on page 163 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (CS4A = 1 and/or CS5A = 1).

Table 15-7. Dedicated CompactFlash Interface Multiplexing

Pins	CompactFlash Signals		EBI Signals	
	CS4A = 1	**CS5A = 1**	**CS4A = 0**	**CS5A = 0**
NCS[4]	CFCS0		NCS[4]	
NCS[5]		CFCS1		NCS[5]

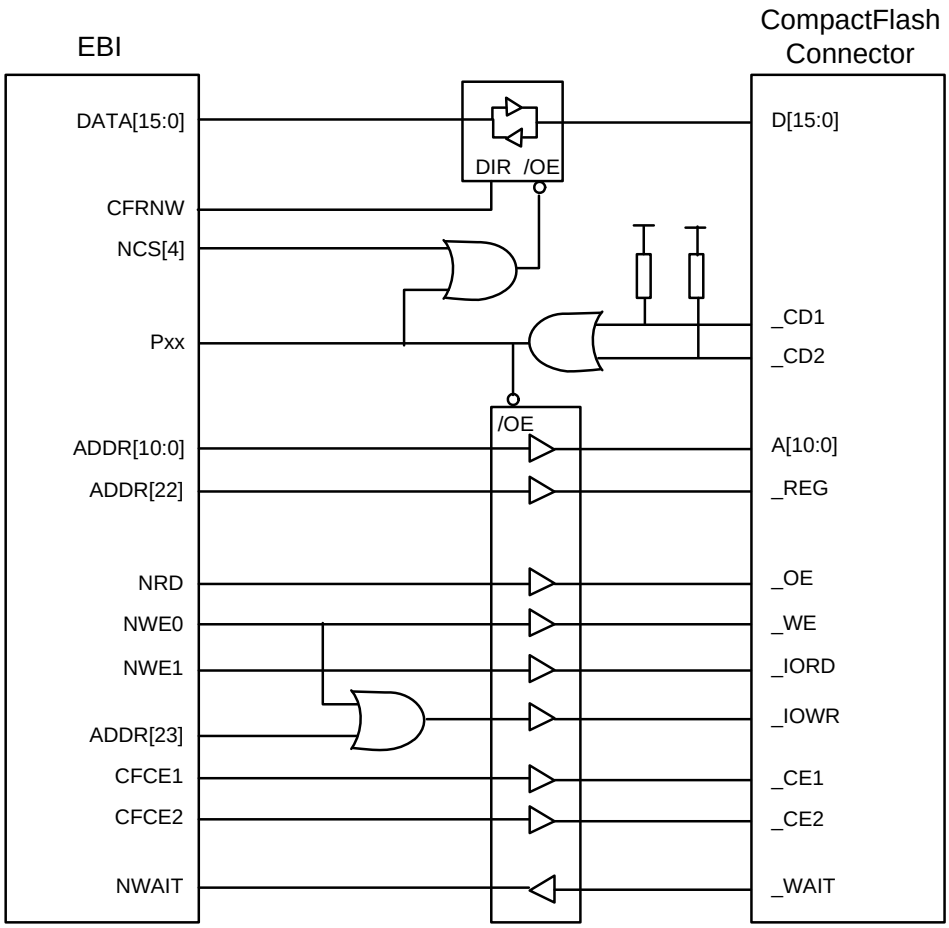
Table 15-8. Shared CompactFlash Interface Multiplexing

	Access to CompactFlash Device
Pins	**CompactFlash Signals**
NRD	CFNOE
NWE0	CFNWE
NWE1	CFNIORD
CFRNW	CFRNW

15.6.5.5 Application example

Figure 15-4 on page 164 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 NCS[4] signal. 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 SMC Section.

Figure 15-4. CompactFlash Application Example



15.6.6 SmartMedia and NAND Flash Support

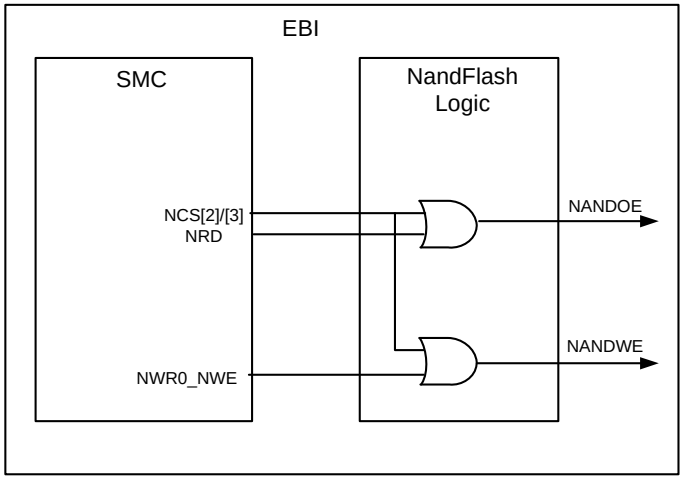
The EBI integrates circuitry that interfaces to SmartMedia and NAND Flash devices.

The NAND Flash logic is driven by the Static Memory Controller on the NCS[2] (and/or NCS[3]) address space. Writing to the HMATRIX.SFR6.CS2A (and/or HMATRIX.SFR6.CS3A) bit the appropriate value enables the NAND Flash logic. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS[2] (and/or NCS[3]).

The NAND Flash logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS[2] (and/or NCS[3]) signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS[2] (and/or NCS[3]) address space. See Figure 15-5 on page 165 for more informations. For details on these waveforms, refer to the SMC Section.

The SmartMedia device is connected the same way as the NAND Flash device.

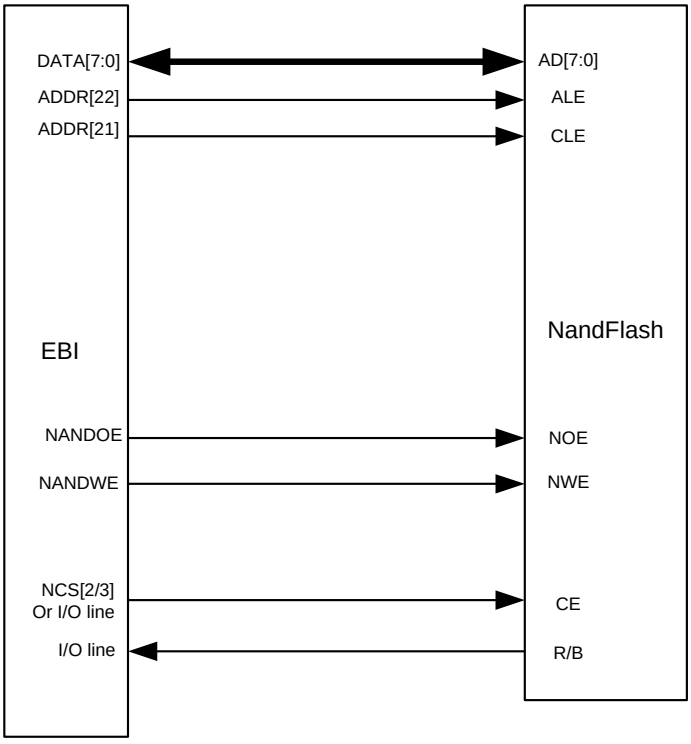
Figure 15-5. NAND Flash Signal Multiplexing on EBI Pins



15.6.6.1 *NAND Flash signals*

The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits ADDR[22] and ADDR[21] of the EBI address bus. The user should note that any bit on the EBI address bus can also be used for this purpose. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCSx address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to I/O Controller lines. The CE signal then remains asserted even when NCSx is not selected, preventing the device from returning to standby mode.

Figure 15-6. NAND Flash Application Example



Note: The External Bus Interfaces is also able to support 16-bits devices.

15.7 Application Example

15.7.1 Hardware Interface

Table 15-9. EBI Pins and External Static Devices Connections

Pins name	Pins of the Interfaced Device		
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device
Controller	**SMC**		
DATA[7:0]	D[7:0]	D[7:0]	D[7:0]
DATA[15:0	–	D[15:8]	D[15:8]
ADDR[0]	A[0]	–	NBS0(2)
ADDR[1]	A[1]	A[0]	A[0]
ADDR[23:2]	A[23:2]	A[22:1]	A[22:1]
NCS[0] - NCS[5]	CS	CS	CS
NRD	OE	OE	OE
NWE0	WE	WE(1)	WE
NWE1	–	WE(1)	NBS1(2)

Note: 1. NWE1 enables upper byte writes. NWE0 enables lower byte writes.
2. NBS1 enables upper byte writes. NBS0 enables lower byte writes.

Table 15-10. EBI Pins and External Devices Connections

Pins name	Pins of the Interfaced Device			
	SDRAM	Compact Flash	Compact Flash True IDE Mode	Smart Media or NAND Flash
Controller	**SDRAMC**	**SMC**		
DATA[7:0]	D[7:0]	D[7:0]	D[7:0]	AD[7:0]
DATA[15:8]	D[15:8]	D[15:8]	D[15:8]	AD[15:8]
ADDR[0]	DQM0	A[0]	A[0]	–
ADDR[1]	–	A[1]	A[1]	–
ADDR[10:2]	A[8:0]	A[10:2]	A[10:2]	–
ADDR[11]	A[9]	–	–	–
SDA10	A[10]	–	–	–
ADDR[12]	–	–	–	–
ADDR[14:13]	A[12:11]	–	–	–
ADDR[15]	–	–	–	–
ADDR[16]	BA0	–	–	–
ADDR[17]	BA1	–	–	–
ADDR[20:18]	–	–	–	–

Table 15-10. EBI Pins and External Devices Connections (Continued)

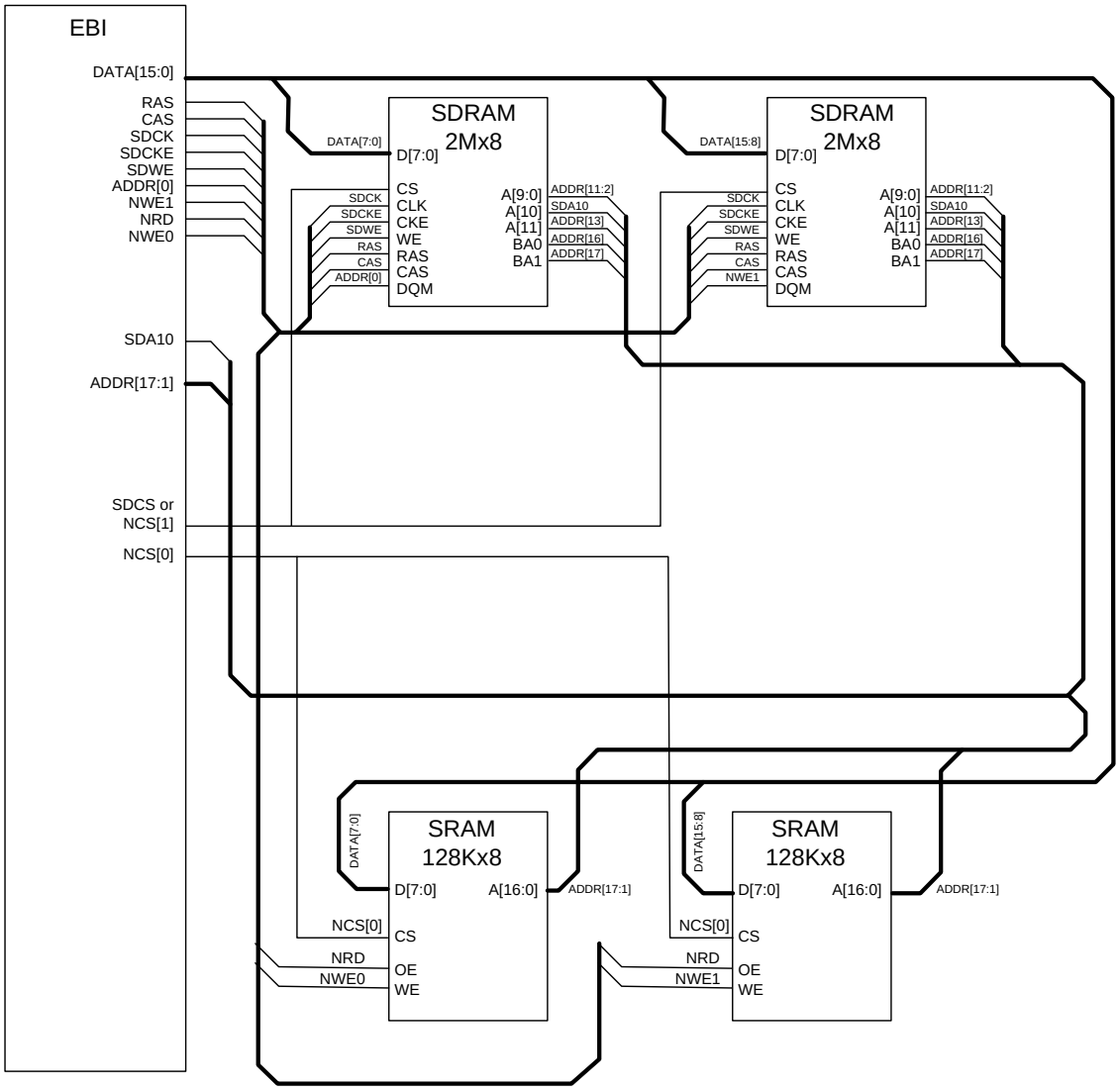
Pins name	Pins of the Interfaced Device			
	SDRAM	Compact Flash	Compact Flash True IDE Mode	Smart Media or NAND Flash
Controller	**SDRAMC**	**SMC**		
ADDR[21]	–	–	–	CLE(3)
ADDR[22]	–	REG	REG	ALE(3)
NCS[0]	–	–	–	–
NCS[1]	SDCS[0]	–	–	–
NCS[2]	–	–	–	CE0
NCS[3]	–	–	–	CE1
NCS[4]	–	CFCS0(1)	CFCS0(1)	–
NCS[5]	–	CFCS1(1)	CFCS1(1)	–
NANDOE	–	–	–	OE
NANDWE	–	–	–	WE
NRD	–	OE	–	–
NWE0	–	WE	WE	–
NWE1	DQM1	IOR	IOR	–
CFRNW	–	CFRNW(1)	CFRNW(1)	–
CFCE1	–	CE1	CS0	–
CFCE2	–	CE2	CS1	–
SDCS	SDCS[1]	–	–	–
SDCK	CLK	–	–	–
SDCKE	CKE	–	–	–
RAS	RAS	–	–	–
CAS	CAS	–	–	–
SDWE	WE	–	–	–
NWAIT	–	WAIT	WAIT	–
Pxx(2)	–	CD1 or CD2	CD1 or CD2	–
Pxx(2)	–	–	–	RDY

Note:
1. Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.
2. Any I/O Controller line.
3. The CLE and ALE signals of the NAND Flash device may be driven by any address bit. For details, see Section 15.6.6.

15.7.2 Connection Examples

Figure 15-7 on page 168shows an example of connections between the EBI and external devices.

Figure 15-7. EBI Connections to Memory Devices



16. Static Memory Controller (SMC)

Rev. 1.0.6.3

16.1 Features

- **6 chip selects available**
- **64-Mbytes address space per chip select**
- **8- or 16-bit data bus**
- **Word, halfword, byte transfers**
- **Byte write or byte select lines**
- **Programmable setup, pulse and hold time for read signals per chip select**
- **Programmable setup, pulse and hold time for write signals per chip select**
- **Programmable data float time per chip select**
- **Compliant with LCD module**
- **External wait request**
- **Automatic switch to slow clock mode**
- **Asynchronous read in page mode supported: page size ranges from 4 to 32 bytes**

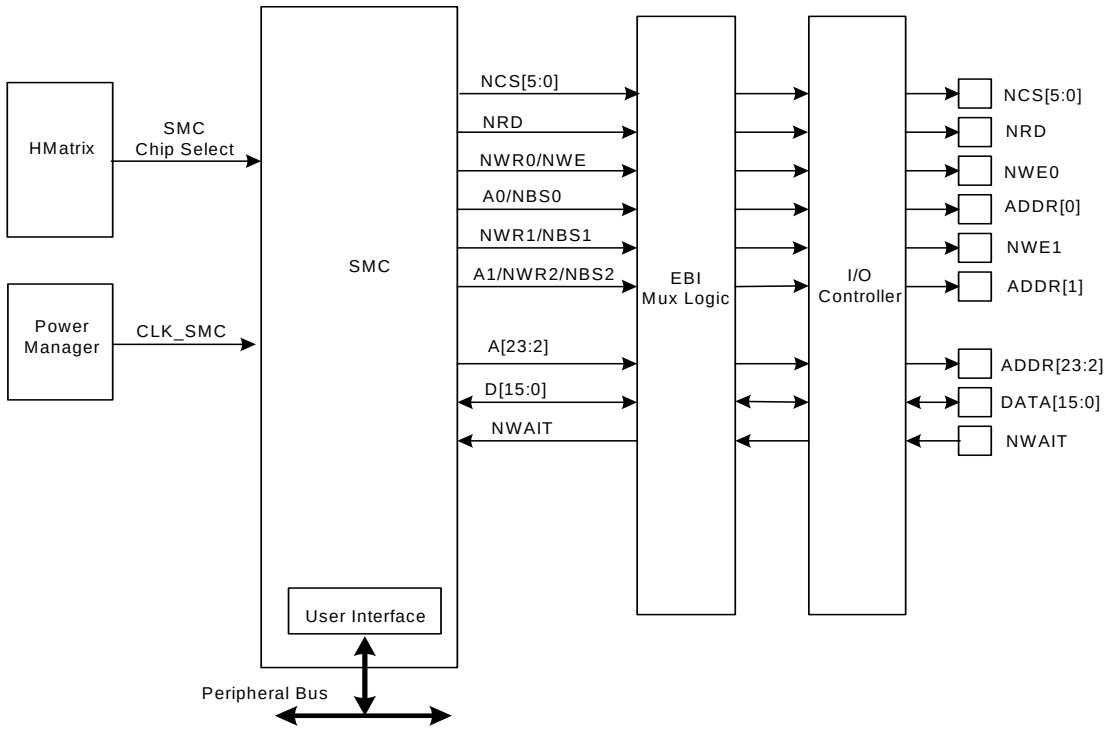
16.2 Overview

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 6 chip selects and a 26-bit address bus. The 16-bit data bus can be configured to interface with 8-16-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic slow clock mode. In slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in page mode access for page size up to 32 bytes.

16.3 Block Diagram

Figure 16-1. SMC Block Diagram



16.4 I/O Lines Description

Table 16-1. I/O Lines Description

Pin Name	Pin Description	Type	Active Level
NCS[5:0]	Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A[25:2]	Address Bus	Output	
D[15:0]	Data Bus	Input/Output	
NWAIT	External Wait Signal	Input	Low

16.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

16.5.1 I/O Lines

The SMC signals pass through the External Bus Interface (EBI) module where they are multiplexed. The user must first configure the I/O Controller to assign the EBI pins corresponding to SMC signals to their peripheral function. If the I/O lines of the EBI corresponding to SMC signals are not used by the application, they can be used for other purposes by the I/O Controller.

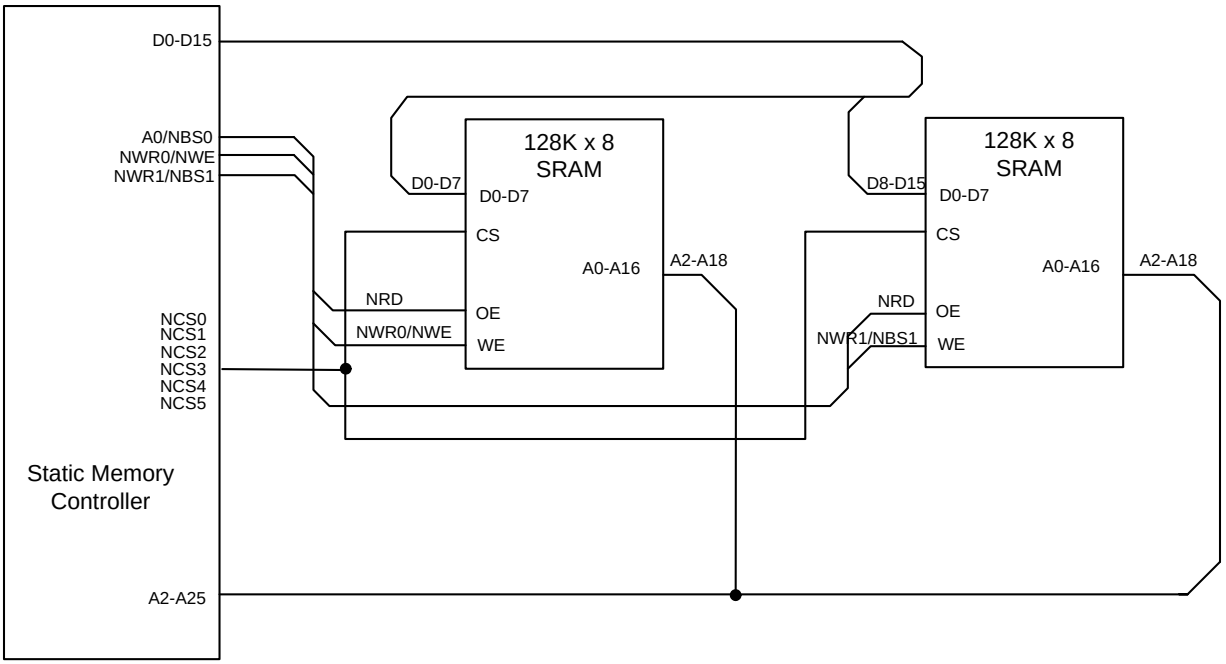
16.5.2 Clocks

The clock for the SMC bus interface (CLK_SMC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SMC before disabling the clock, to avoid freezing the SMC in an undefined state.

16.6 Functional Description

16.6.1 Application Example

Figure 16-2. SMC Connections to Static Memory Devices



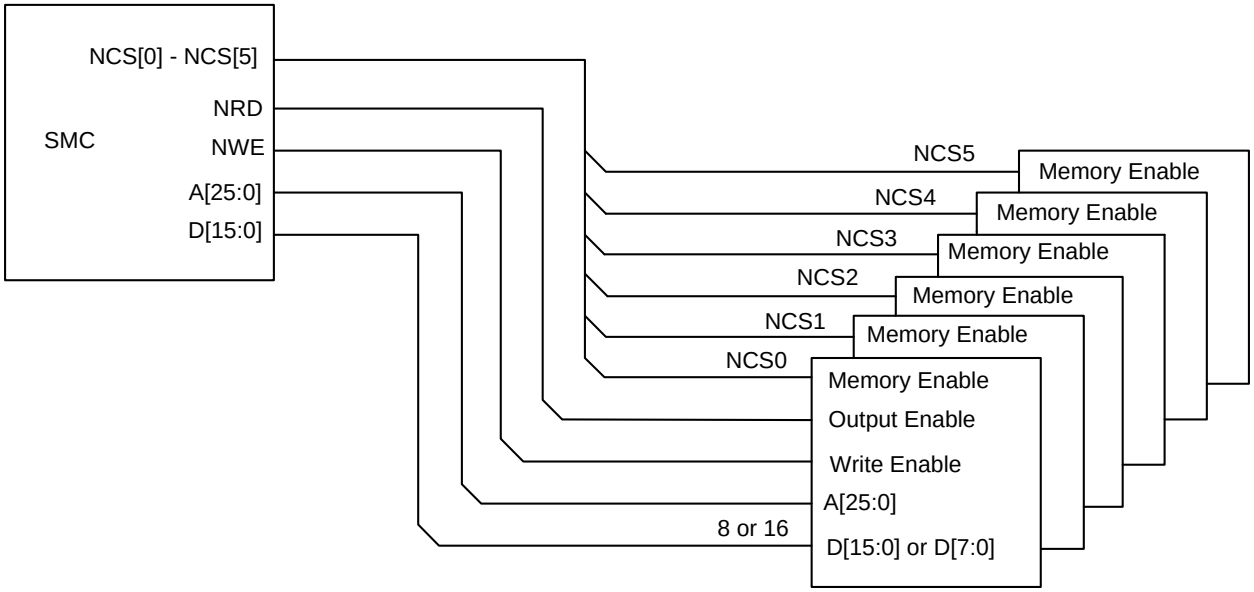
16.6.2 External Memory Mapping

The SMC provides up to 26 address lines, A[25:0]. This allows each chip select line to address up to 64 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 64 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 16-3 on page 172).

A[25:0] is only significant for 8-bit memory, A[25:1] is used for 16-bit memory.

Figure 16-3. Memory Connections for Six External Devices



16.6.3 Connection to External Devices

16.6.3.1 Data bus width

A data bus width of 8 or 16 bits can be selected for each chip select. This option is controlled by the Data Bus Width field in the Mode Register (MODE.DBW) for the corresponding chip select.

Figure 16-4 on page 172 shows how to connect a 512K x 8-bit memory on NCS2. Figure 16-5 on page 173 shows how to connect a 512K x 16-bit memory on NCS2.

16.6.3.2 Byte write or byte select access

Each chip select with a 16-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the Byte Access Type bit in the MODE register (MODE.BAT) for the corresponding chip select.

Figure 16-4. Memory Connection for an 8-bit Data Bus

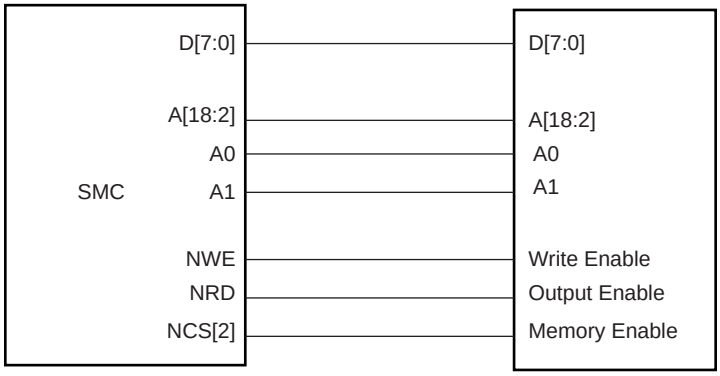
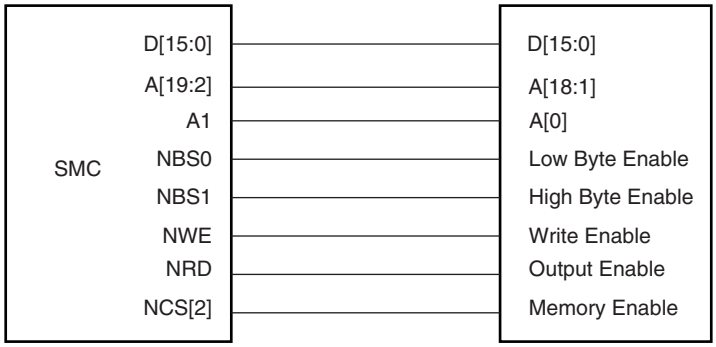


Figure 16-5. Memory Connection for a 16-bit Data Bus



•*Byte write access*

The byte write access mode supports one byte write signal per byte of the data bus and a single read signal.

Note that the SMC does not allow boot in byte write access mode.

- For 16-bit devices: the SMC provides NWR0 and NWR1 write signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

The byte write access mode is used to connect two 8-bit devices as a 16-bit memory.

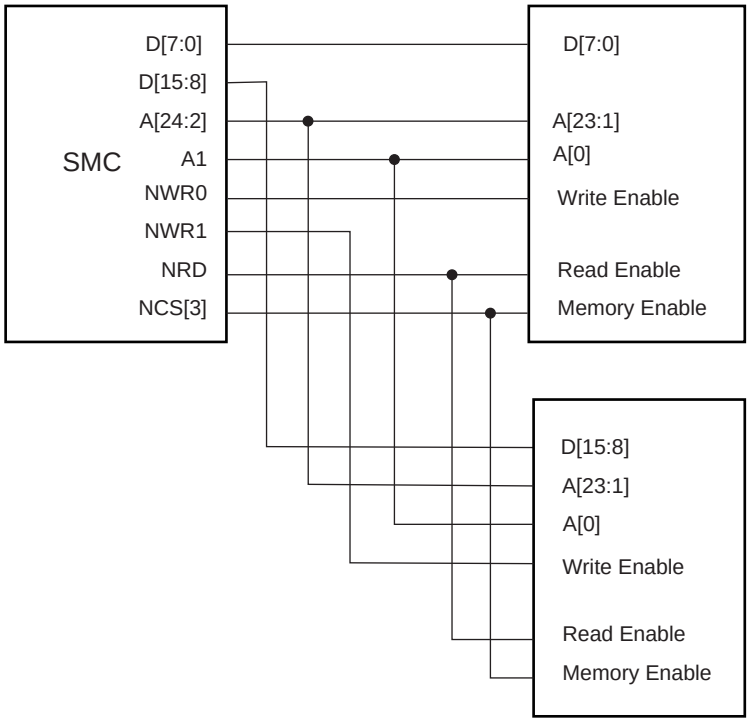
The byte write option is illustrated on Figure 16-6 on page 174.

•*Byte select access*

In this mode, read/write operations can be enabled/disabled at a byte level. One byte select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. The byte select access is used to connect one 16-bit device.

Figure 16-6. Connection of two 8-bit Devices on a 16-bit Bus: Byte Write Option



•*Signal multiplexing*

Depending on the MODE.BAT bit, only the write signals or the byte select signals are used. To save I/Os at the external bus interface, control signals at the SMC interface are multiplexed.

For 16-bit devices, bit A0 of address is unused. When byte select option is selected, NWR1 is unused. When byte write option is selected, NBS0 to NBS1 are unused.

Table 16-3. SMC Multiplexed Signal Translation

Signal Name	**16-bit Bus**		**8-bit Bus**
Device Type	**1 x 16-bit**	**2 x 8-bit**	**1 x 8-bit**
Byte Access Type (BAT)	Byte Select	Byte Write	
NBS0_A0	NBS0		A0
NWE_NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NWR1	
NBS2_NWR2_A1	A1	A1	A1

16.6.4 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS1) always have the same timing as the address bus (A). NWE represents either the NWE

signal in byte select access type or one of the byte write lines (NWR0 to NWR1) in byte write access type. NWR0 to NWR1 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..5] chip select lines.

16.6.4.1 *Read waveforms*

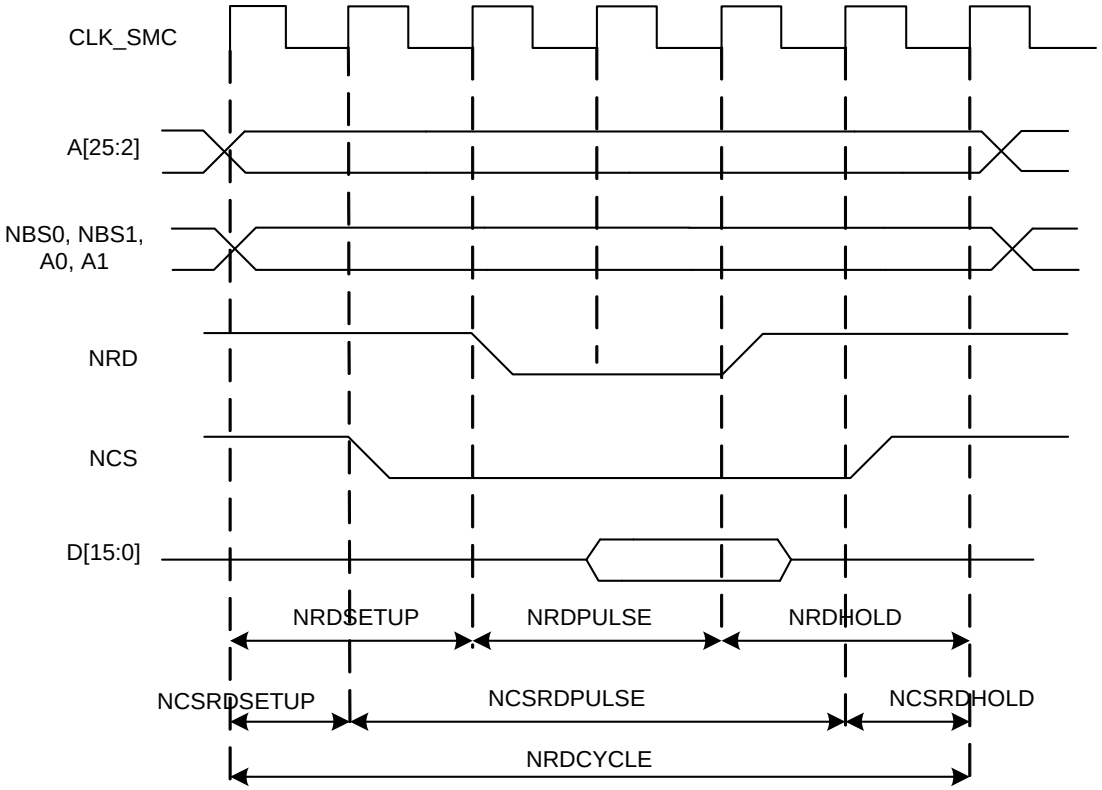
The read cycle is shown on Figure 16-7 on page 175.

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

{A[25:2], A1, A0} for 8-bit devices

{A[25:2], A1} for 16-bit devices

Figure 16-7. Standard Read Cycle



•NRD waveform

The NRD signal is characterized by a setup timing, a pulse width, and a hold timing.

1. NRDSETUP: the NRD setup time is defined as the setup of address before the NRD falling edge.
2. NRDPULSE: the NRD pulse length is the time between NRD falling edge and NRD rising edge.
3. NRDHOLD: the NRD hold time is defined as the hold time of address after the NRD rising edge.

•NCS waveform

Similarly, the NCS signal can be divided into a setup time, pulse length and hold time.

1. NCSRDSETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCSRDPULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge.
3. NCSRDHOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

•Read cycle

The NRDCYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is equal to:

$$NRDCYCLE = NRDSETUP + NRDPULSE + NRDHOLD$$

Similarly,

$$NRDCYCLE = NCSRDSETUP + NCSRDPULSE + NCSRDHOLD$$

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

$$NRDHOLD = NRDCYCLE - NRDSETUP - NRDPULSE$$

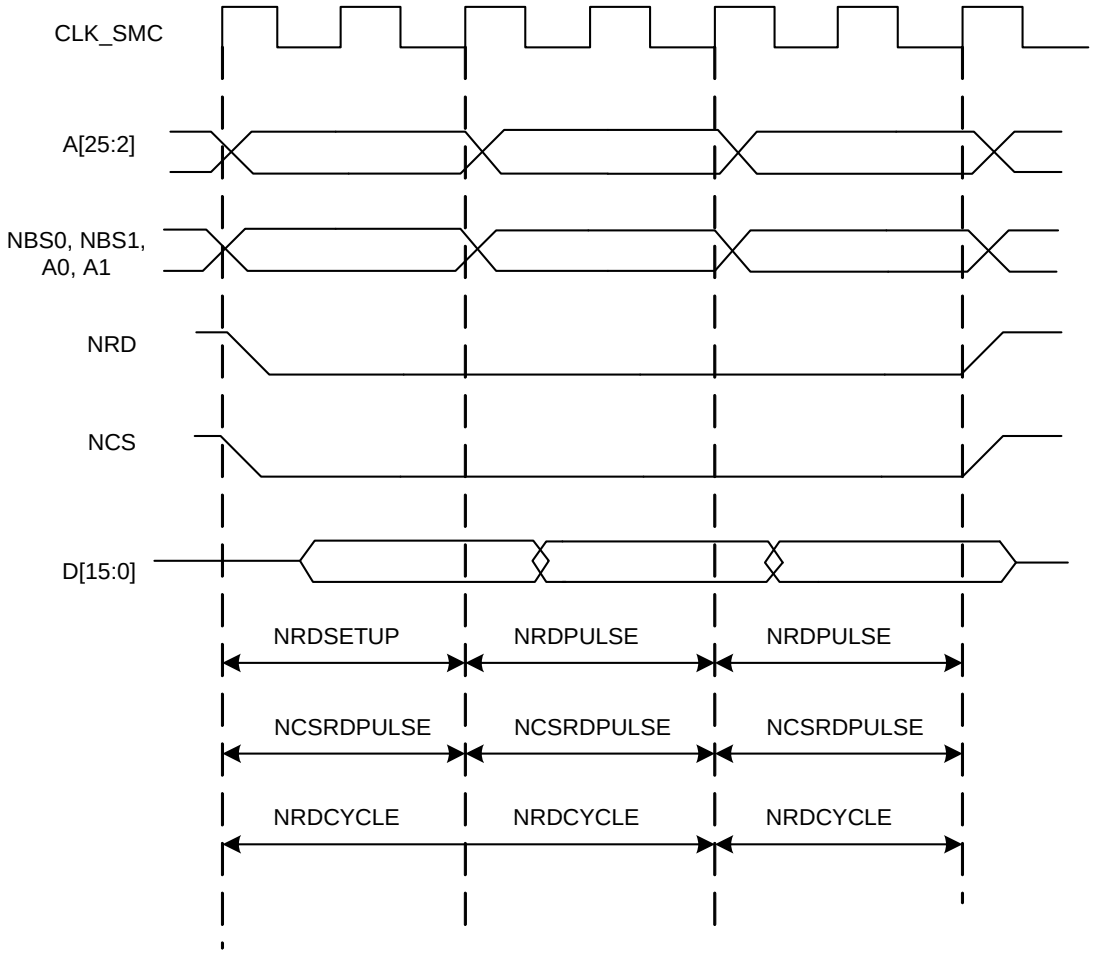
And,

$$NCSRDHOLD = NRDCYCLE - NCSRDSETUP - NCSRDPULSE$$

•Null delay setup and hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 16-8 on page 177).

Figure 16-8. No Setup, No Hold on NRD, and NCS Read Signals



•Null Pulse

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

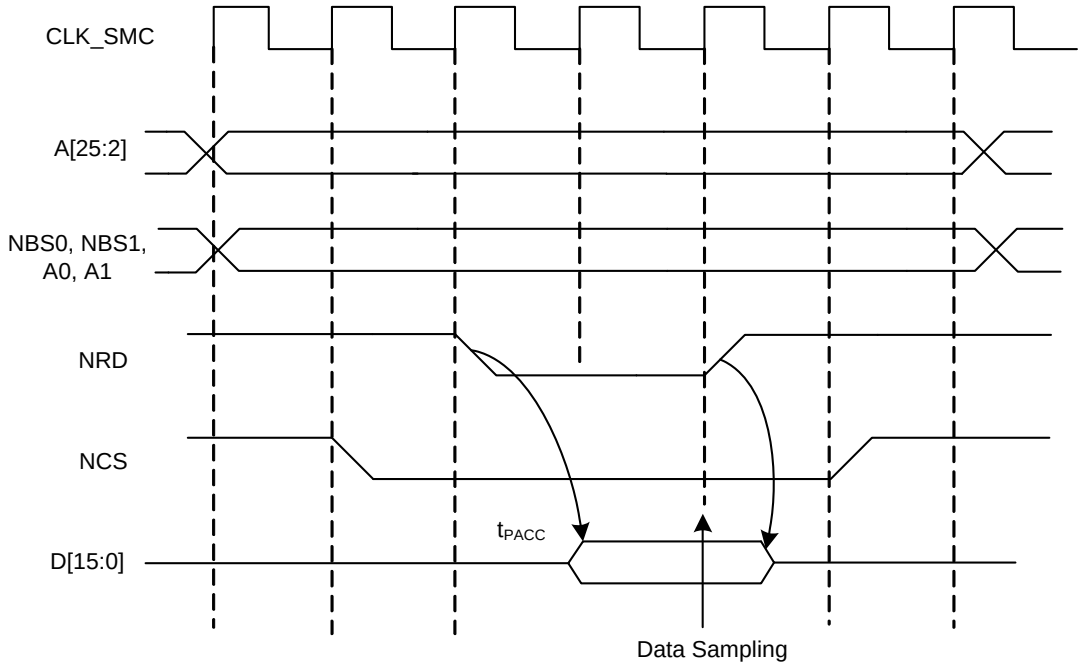
16.6.4.2 Read mode

As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The Read Mode bit in the MODE register (MODE.READMODE) of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

•Read is controlled by NRD (MODE.READMODE = 1)

Figure 16-9 on page 178 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to 'Z' after the rising edge of NRD. In this case, the MODE.READMODE bit must be written to one (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of CLK_SMC that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

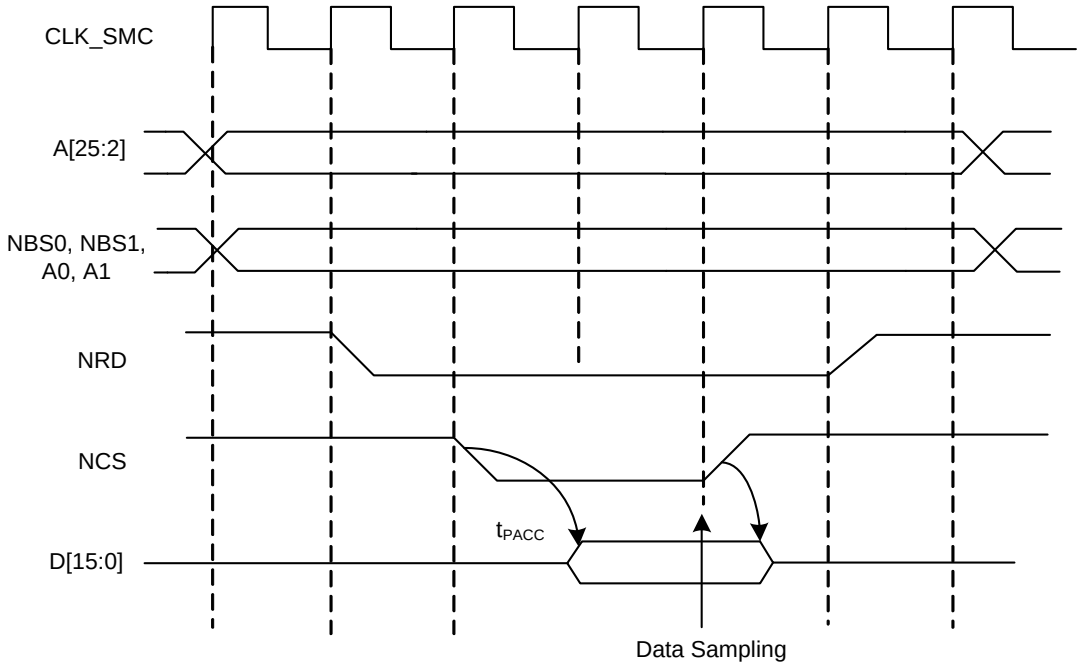
Figure 16-9. READMODE = 1: Data Is Sampled by SMC Before the Rising Edge of NRD



•*Read is controlled by NCS (MODE.READMODE = 0)*

Figure 16-10 on page 179 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In that case, the MODE.READMODE bit must be written to zero (read is controlled by NCS): the SMC internally samples the data on the rising edge of CML_SMC that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 16-10. READMODE = 0: Data Is Sampled by SMC Before the Rising Edge of NCS



16.6.4.3 *Write waveforms*

The write protocol is similar to the read protocol. It is depicted in Figure 16-11 on page 180. The write cycle starts with the address setting on the memory address bus.

- *NWE waveforms*

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

1. NWESETUP: the NWE setup time is defined as the setup of address and data before the NWE falling edge.
2. NWEPULSE: the NWE pulse length is the time between NWE falling edge and NWE rising edge.
3. NWEHOLD: the NWE hold time is defined as the hold time of address and data after the NWE rising edge.

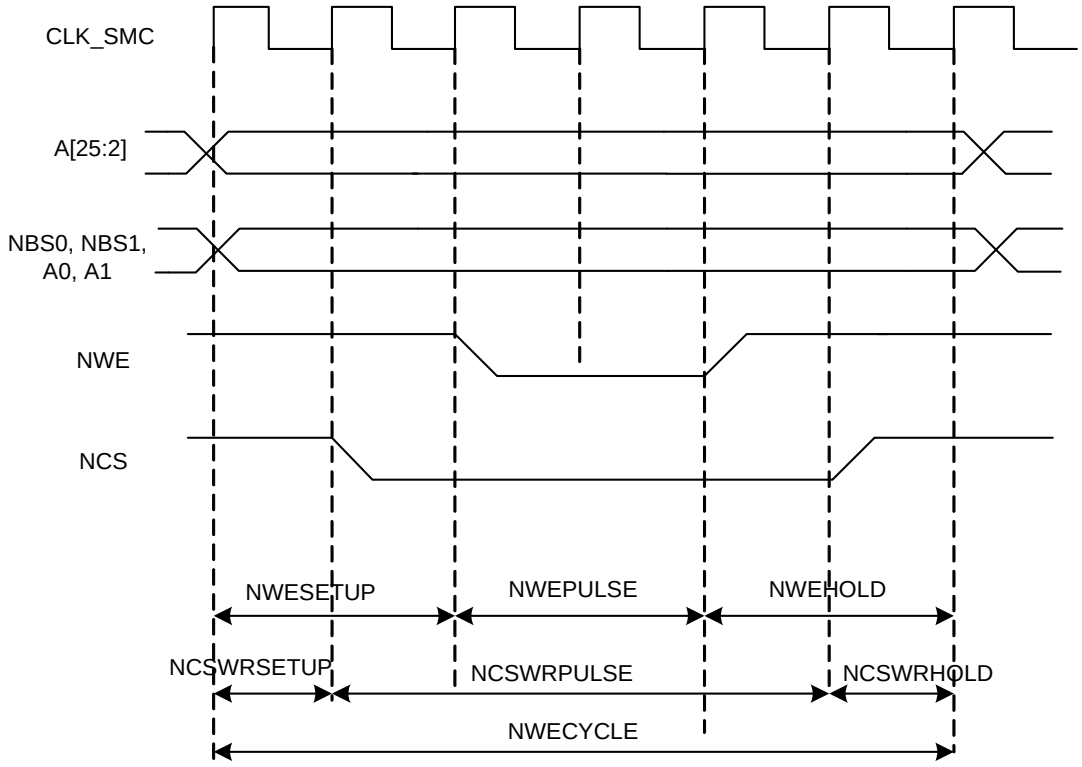
The NWE waveforms apply to all byte-write lines in byte write access mode: NWR0 to NWR3.

16.6.4.4 *NCS waveforms*

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined.

1. NCSWRSETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCSWRPULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
3. NCSWRHOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 16-11. Write Cycle



•*Write cycle*

The write cycle time is defined as the total duration of the write cycle, that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is equal to:

$$NWECYCLE = NWESETUP + NWEPULSE + NWEHOLD$$

Similarly,

$$NWECYCLE = NCSWRSETUP + NCSWRPULSE + NCSWRHOLD$$

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

$$NWEHOLD = NWECYCLE - NWESETUP - NWEPULSE$$

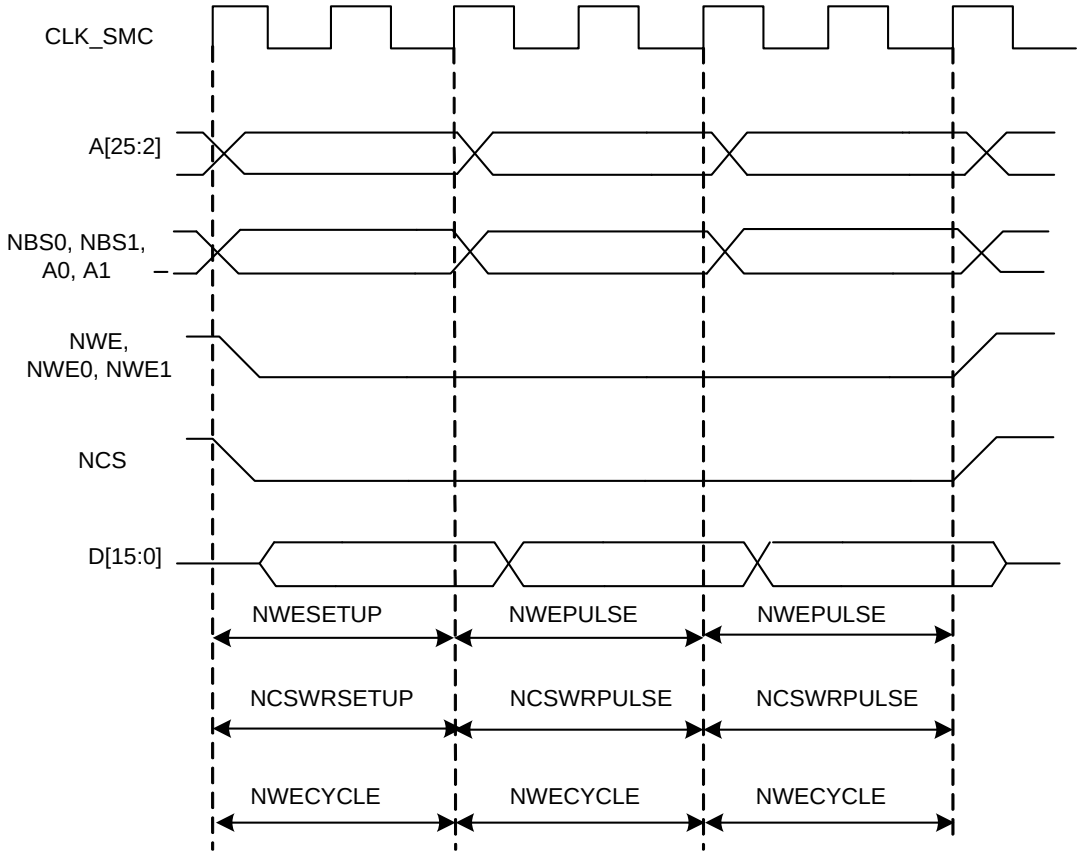
And,

$$NCSWRHOLD = NWECYCLE - NCSWRSETUP - NCSWRPULSE$$

•*Null delay setup and hold*

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 16-12 on page 181). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 16-12. Null Setup and Hold Values of NCS and NWE in Write Cycle



•*Null pulse*

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

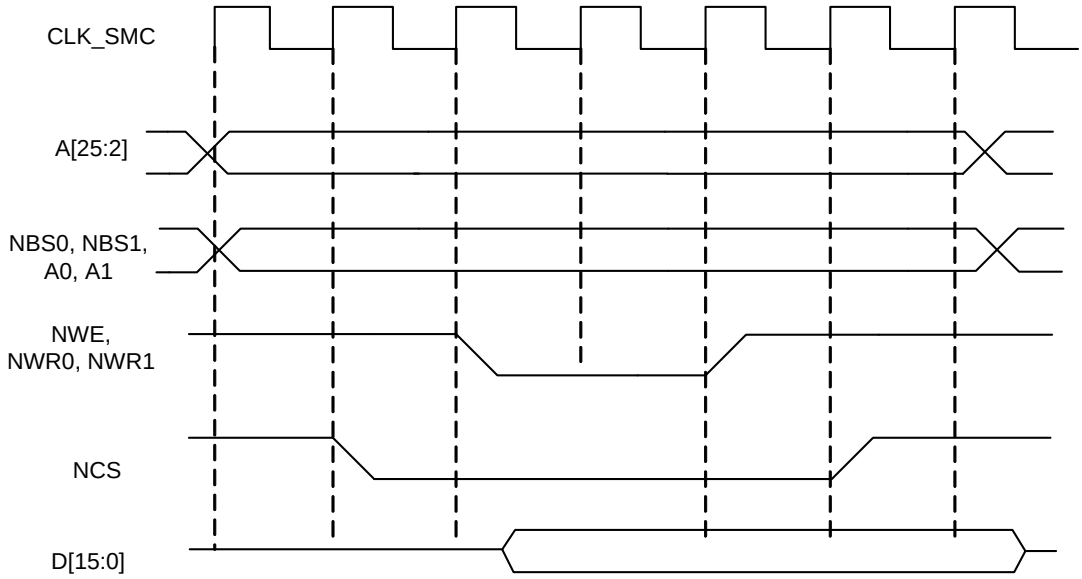
16.6.4.5 *Write mode*

The Write Mode bit in the MODE register (MODE.WRITEMODE) of the corresponding chip select indicates which signal controls the write operation.

•*Write is controlled by NWE (MODE.WRITEMODE = 1)*

Figure 16-13 on page 182 shows the waveforms of a write operation with MODE.WRITEMODE equal to one. The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are turned out after the NWESETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

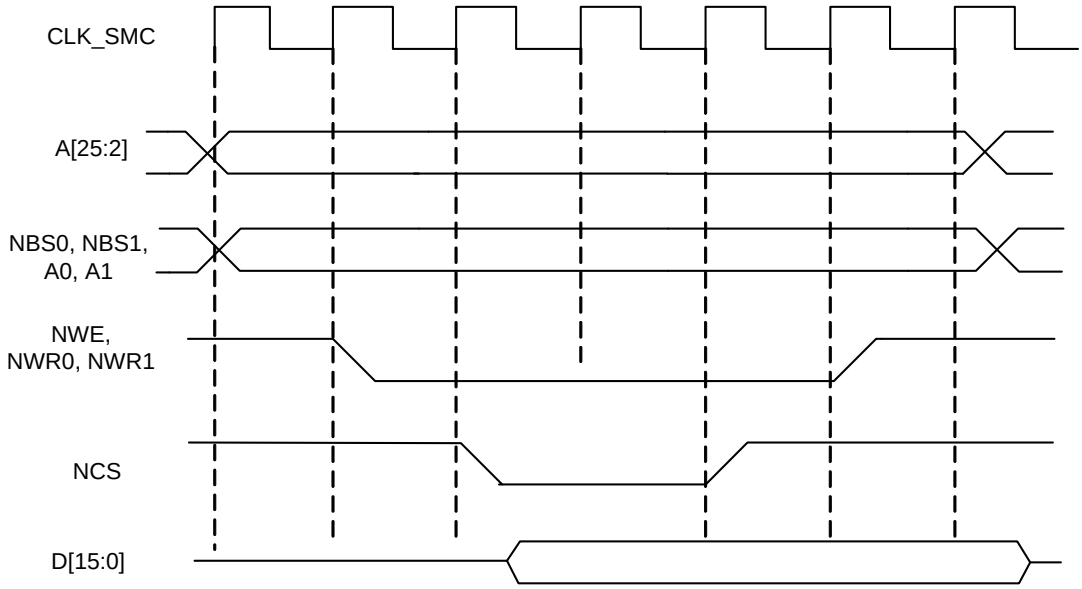
Figure 16-13. WRITEMODE = 1. The Write Operation Is Controlled by NWE



•*Write is controlled by NCS (MODE.WRITEMODE = 0)*

Figure 16-14 on page 182 shows the waveforms of a write operation with MODE.WRITEMODE written to zero. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are turned out after the NCSWRSETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 16-14. WRITEMODE = 0. The Write Operation Is Controlled by NCS



16.6.4.6 *Coding timing parameters*

All timing parameters are defined for one chip select and are grouped together in one register according to their type.

The Setup register (SETUP) groups the definition of all setup parameters:

- NRDSETUP, NCSRDSETUP, NWESETUP, and NCSWRSETUP.

The Pulse register (PULSE) groups the definition of all pulse parameters:

- NRDPULSE, NCSRDPULSE, NWEPULSE, and NCSWRPULSE.

The Cycle register (CYCLE) groups the definition of all cycle parameters:

- NRDCYCLE, NWECYCLE.

Table 16-4 on page 183 shows how the timing parameters are coded and their permitted range.

Table 16-4. Coding and Range of Timing Parameters

			Permitted Range	
Coded Value	**Number of Bits**	**Effective Value**	**Coded Value**	**Effective Value**
setup [5:0]	6	128 x setup[5] + setup[4:0]	0 ≤ value ≤ 31	128 ≤ value ≤ 128+31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤ value ≤ 63	256 ≤ value ≤ 256+63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤ value ≤ 127	256 ≤ value ≤ 256+127 512 ≤ value ≤ 512+127 768 ≤ value ≤ 768+127

16.6.4.7 *Usage restriction*

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

For read operations:

Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

For write operations:

If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true if the MODE.WRITEMODE bit is written to one. See Section 16.6.5.2.

For read and write operations: a null value for pulse parameters is forbidden and may lead to unpredictable behavior.

In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

16.6.5 Automatic Wait States

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

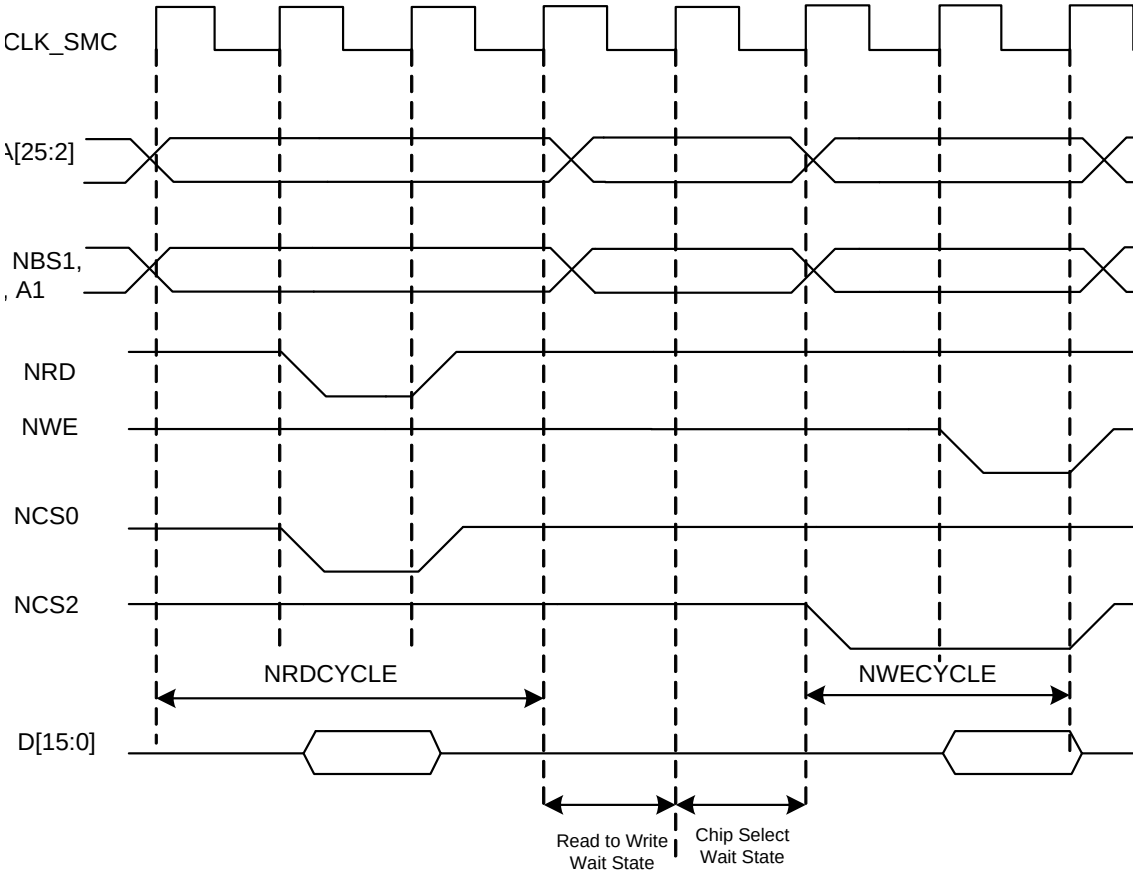
16.6.5.1 Chip select wait states

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

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..5], NRD lines are all set to high level.

Figure 16-15 on page 184 illustrates a chip select wait state between access on Chip Select 0 (NCS0) and Chip Select 2 (NCS2).

Figure 16-15. Chip Select Wait State Between a Read Access on NCS0 and a Write Access on NCS2



16.6.5.2 Early read wait state

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 16-16 on page 185).

- in NCS write controlled mode (MODE.WRITEMODE = 0), if there is no hold timing on the NCS signal and the NCSRDSETUP parameter is set to zero, regardless of the read mode (Figure 16-17 on page 186). The write operation must end with a NCS rising edge. Without an early read wait state, the write operation could not complete properly.
- in NWE controlled mode (MODE.WRITEMODE = 1) and if there is no hold timing (NWEHOLD = 0), the feedback of the write control signal is used to control address, data, chip select, and byte select lines. If the external write control signal is not inactivated as expected due to load capacitances, an early read wait state is inserted and address, data and control signals are maintained one more cycle. See Figure 16-18 on page 187.

Figure 16-16. Early Read Wait State: Write with No Hold Followed by Read with No Setup.

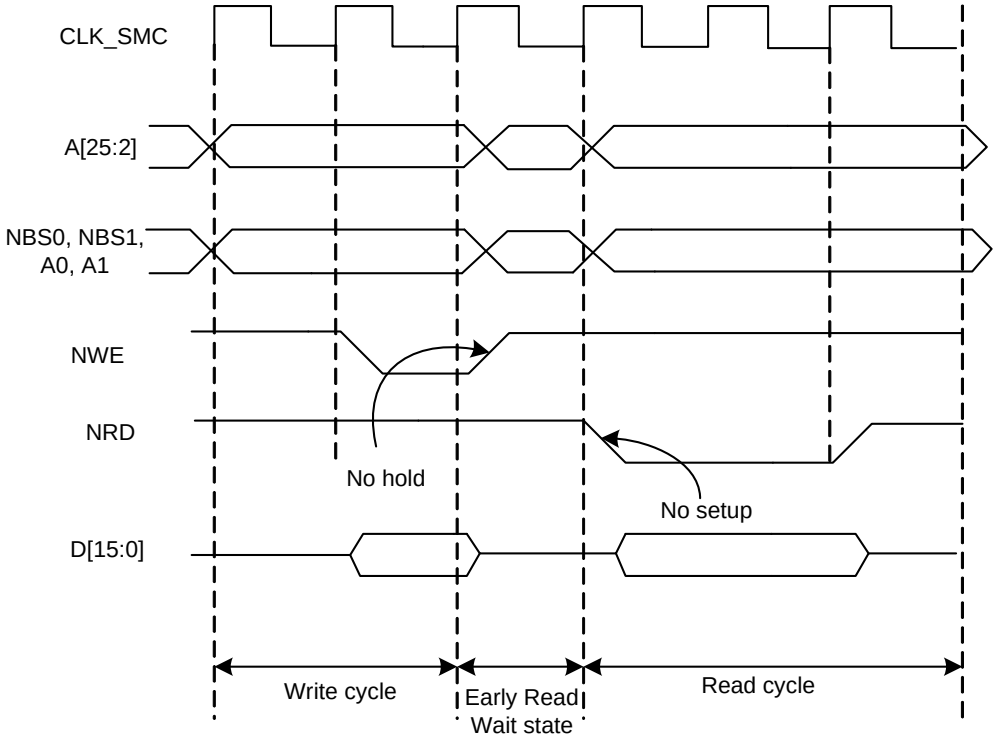


Figure 16-17. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No Setup.

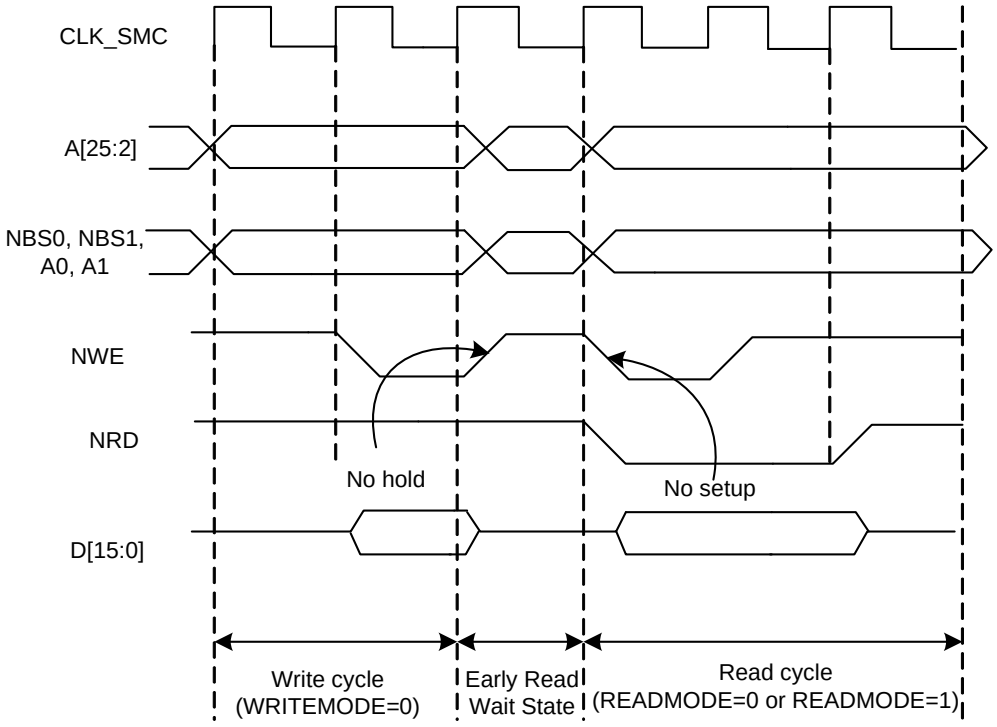
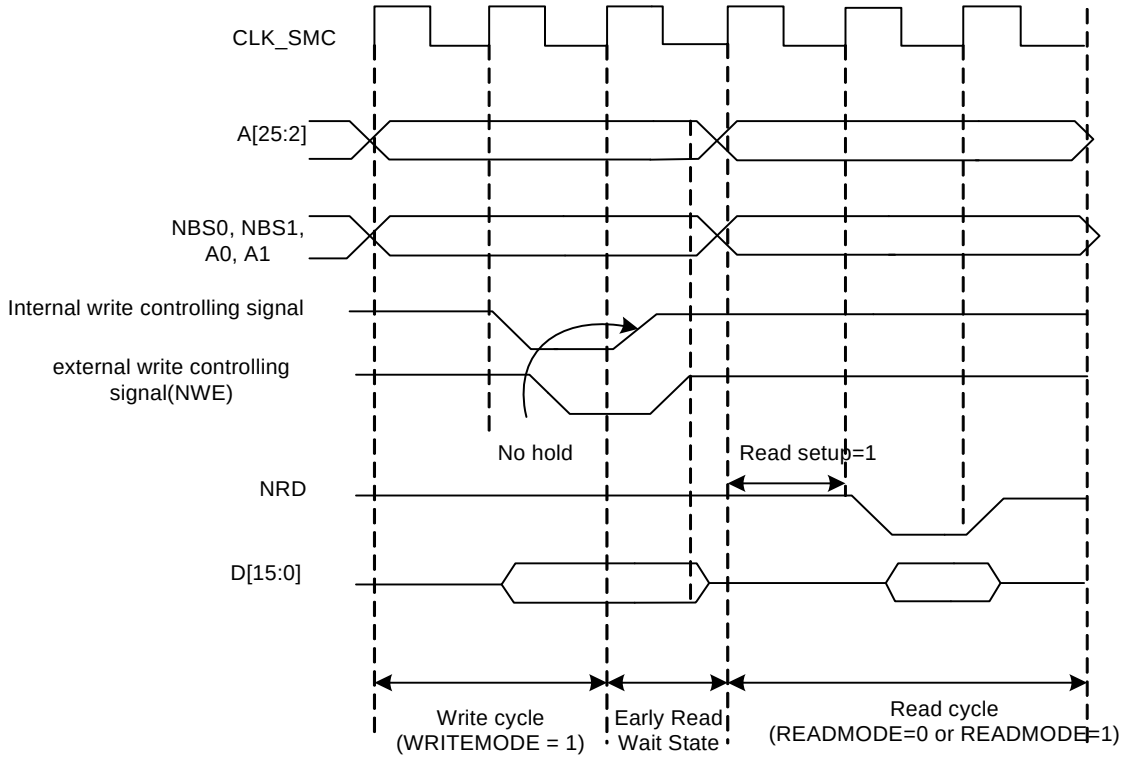


Figure 16-18. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle.



16.6.5.3 *Reload user configuration wait state*

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. The so called “reload user configuration wait state” is used by the SMC to load the new set of parameters to apply to next accesses.

The reload configuration wait state is not applied in addition to the chip select wait state. If accesses before and after reprogramming the user interface are made to different devices (different chip selects), then one single chip select wait state is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a reload configuration wait state is inserted, even if the change does not concern the current chip select.

•User procedure

To insert a reload configuration wait state, the SMC detects a write access to any MODE register of the user interface. If the user only modifies timing registers (SETUP, PULSE, CYCLE registers) in the user interface, he must validate the modification by writing the MODE register, even if no change was made on the mode parameters.

•*Slow clock mode transition*

A reload configuration wait state is also inserted when the slow clock mode is entered or exited, after the end of the current transfer (see Section 16.6.8).

16.6.5.4 Read to write wait state

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 16-15 on page 184.

16.6.6 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory.
- before starting a write access to the same device or to a different external one.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the Data Float Time field of the MODE register (MODE.TDFCYCLES) for the corresponding chip select. The value of MODE.TDFCYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The data float wait states management depends on the MODE.READMODE bit and the TDF Optimization bit of the MODE register (MODE.TDFMODE) for the corresponding chip select.

16.6.6.1 Read mode

Writing a one to the MODE.READMODE bit indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The data float period then begins after the rising edge of the NRD signal and lasts MODE.TDFCYCLES cycles of the CLK_SMC clock.

When the read operation is controlled by the NCS signal (MODE.READMODE = 0), the MODE.TDFCYCLES field gives the number of CLK_SMC cycles during which the data bus remains busy after the rising edge of NCS.

Figure 16-19 on page 189 illustrates the data float period in NRD-controlled mode (MODE.READMODE =1), assuming a data float period of two cycles (MODE.TDFCYCLES = 2). Figure 16-20 on page 189 shows the read operation when controlled by NCS (MODE.READMODE = 0) and the MODE.TDFCYCLES field equals to three.

Figure 16-19. TDF Period in NRD Controlled Read Access (TDFCYCLES = 2)

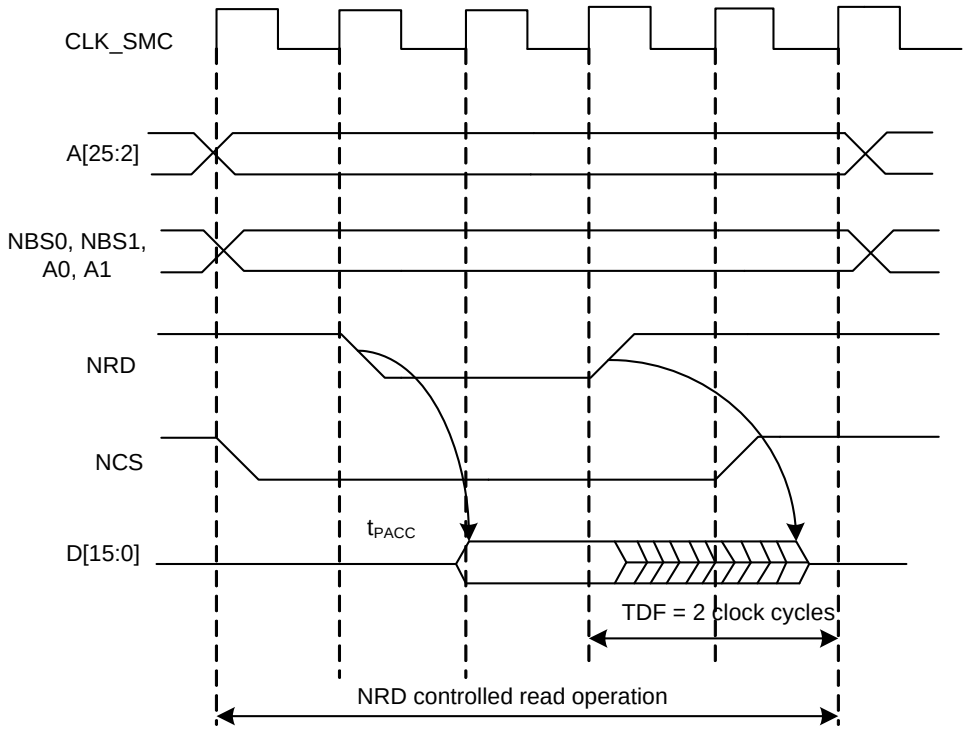
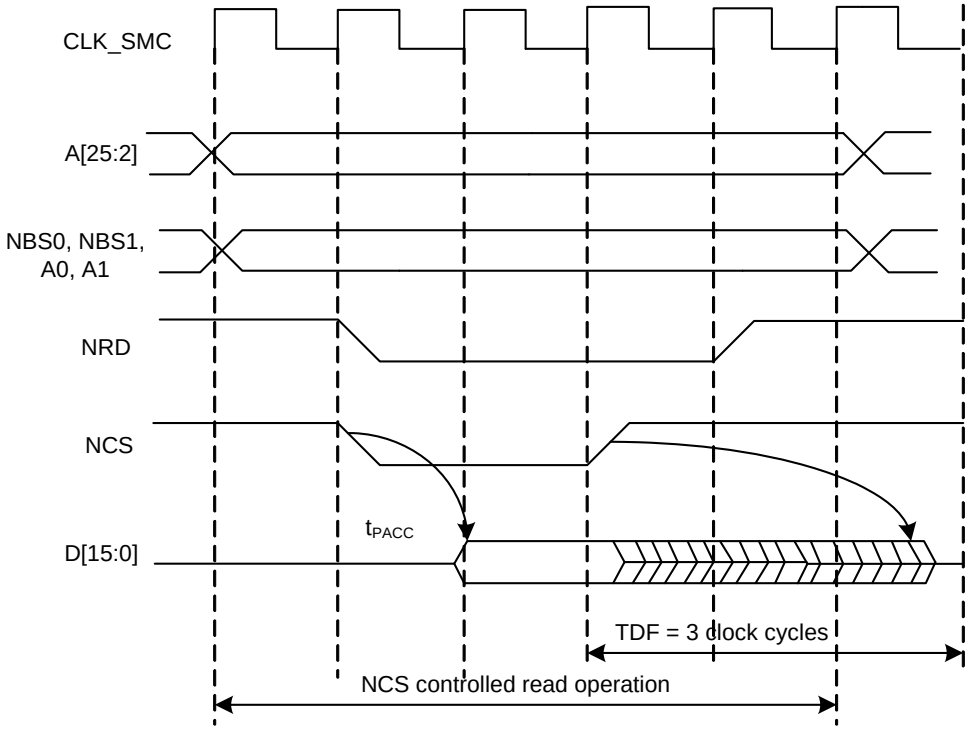


Figure 16-20. TDF Period in NCS Controlled Read Operation (TDFCYCLES = 3)



16.6.6.2 *TDF optimization enabled (MODE.TDFMODE = 1)*

When the MODE.TDFMODE bit is written to one (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

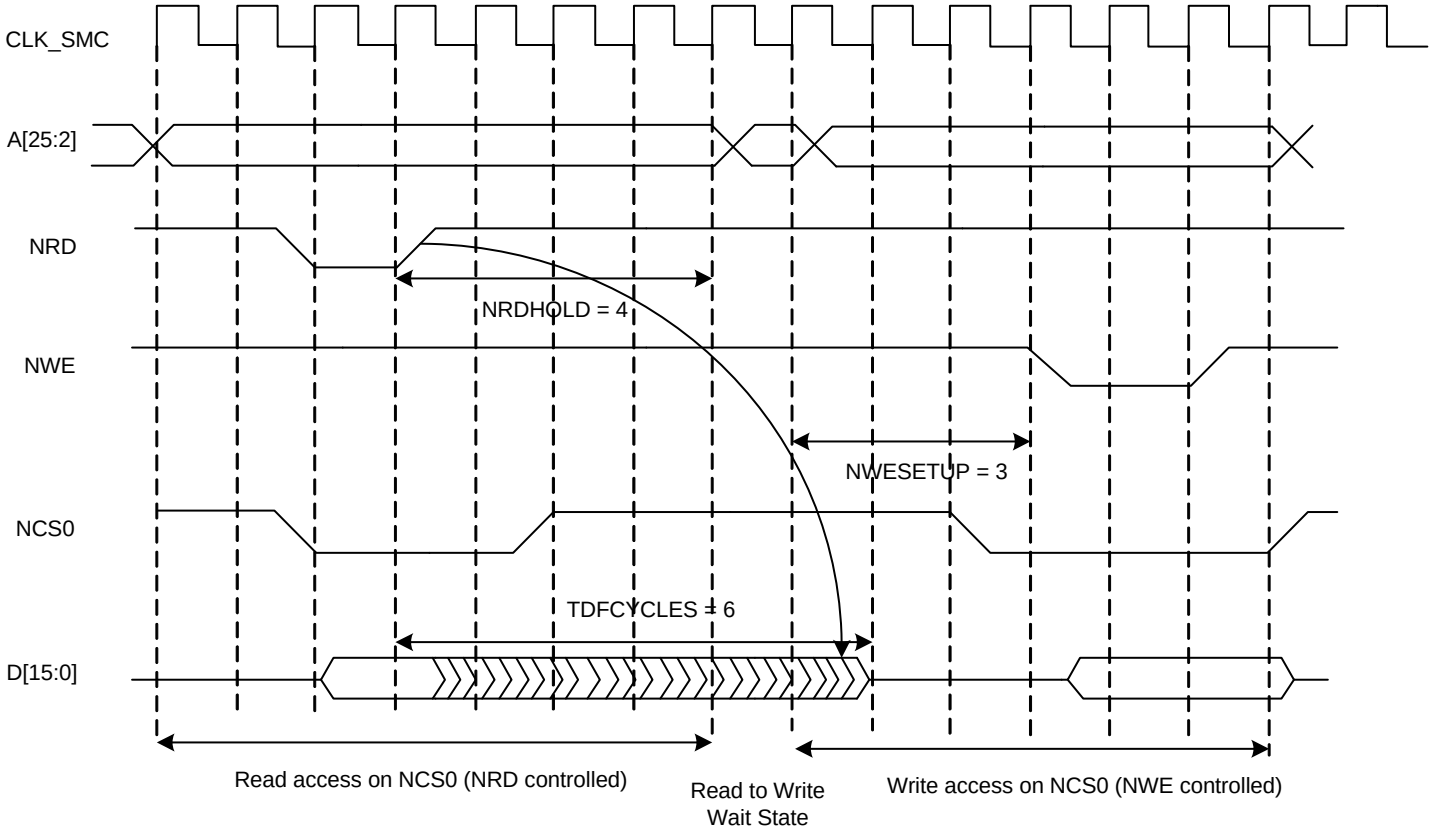
Figure 16-21 on page 190 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRDHOLD = 4; READMODE = 1 (NRD controlled)

NWESETUP = 3; WRITEMODE = 1 (NWE controlled)

TDFCYCLES = 6; TDFMODE = 1 (optimization enabled).

Figure 16-21. TDF Optimization: No TDF Wait States Are Inserted if the TDF Period Is over when the Next Access Begins



16.6.6.3 *TDF optimization disabled (MODE.TDFMODE = 0)*

When optimization is disabled, data float wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional data float wait states will be inserted.

Figure 16-22 on page 191, Figure 16-23 on page 191 and Figure 16-24 on page 192 illustrate the cases:

- read access followed by a read access on another chip select.
- read access followed by a write access on another chip select.

- read access followed by a write access on the same chip select.

with no TDF optimization.

Figure 16-22. TDF Optimization Disabled (MODE.TDFMODE = 0). TDF Wait States between Two Read Accesses on Different Chip Selects.

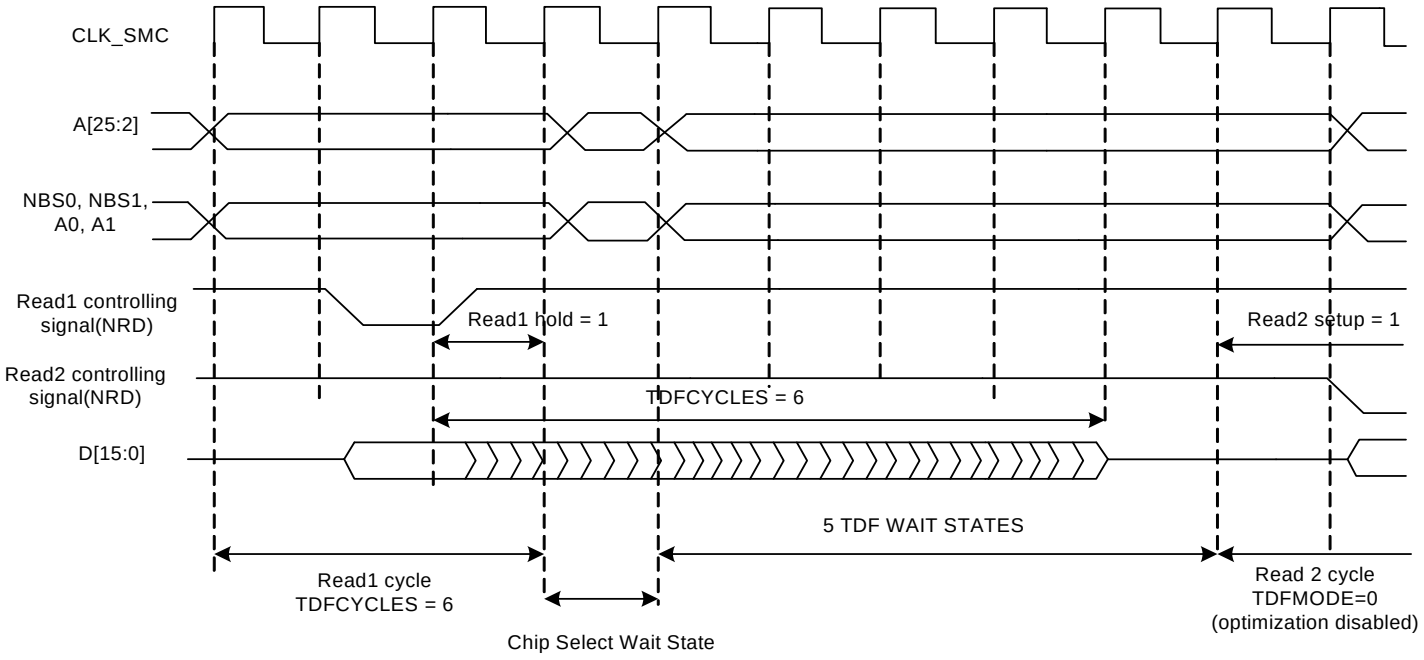


Figure 16-23. TDF Optimization Disabled (MODE.TDFMODE= 0). TDF Wait States between a Read and a Write Access on Different Chip Selects.

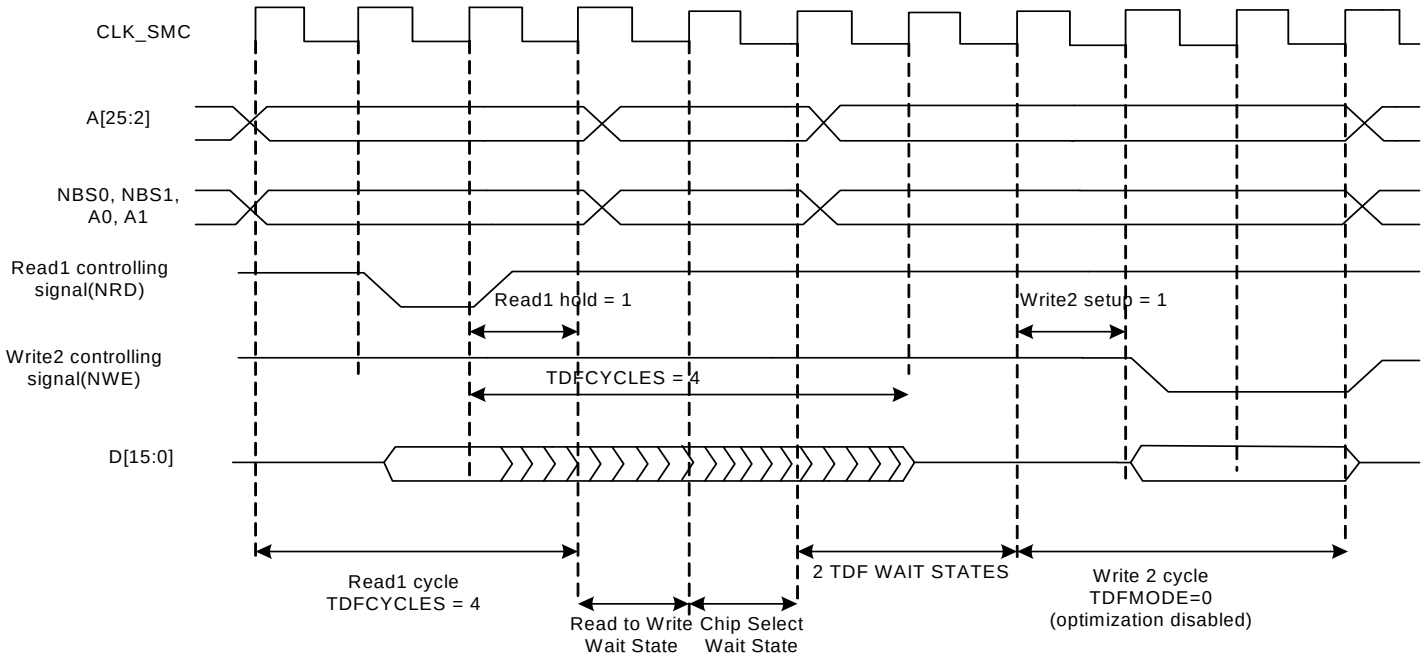
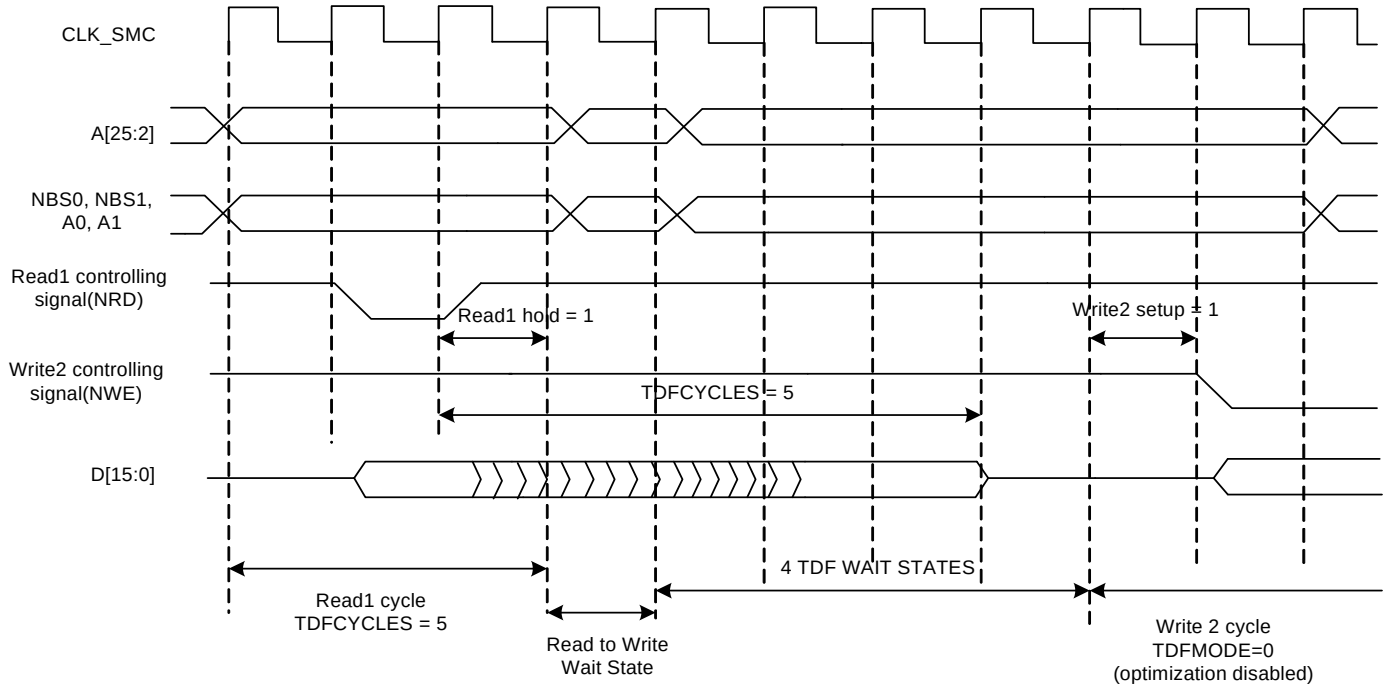


Figure 16-24. TDF Optimization Disabled (MODE.TDFMODE = 0). TDF Wait States between Read and Write accesses on the Same Chip Select.



16.6.7 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The External Wait Mode field of the MODE register (MODE.EXNWMODE) on the corresponding chip select must be written to either two (frozen mode) or three (ready mode). When the MODE.EXNWMODE field is written to zero (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the read and write modes of the corresponding chip select.

16.6.7.1 Restriction

When one of the MODE.EXNWMODE is enabled, it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page Mode (Section 16.6.9), or in Slow Clock Mode (Section 16.6.8).

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

16.6.7.2 Frozen mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the synchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 16-25 on page 193. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 16-26 on page 194.

Figure 16-25. Write Access with NWAIT Assertion in Frozen Mode (MODE.EXNWMODE = 2).

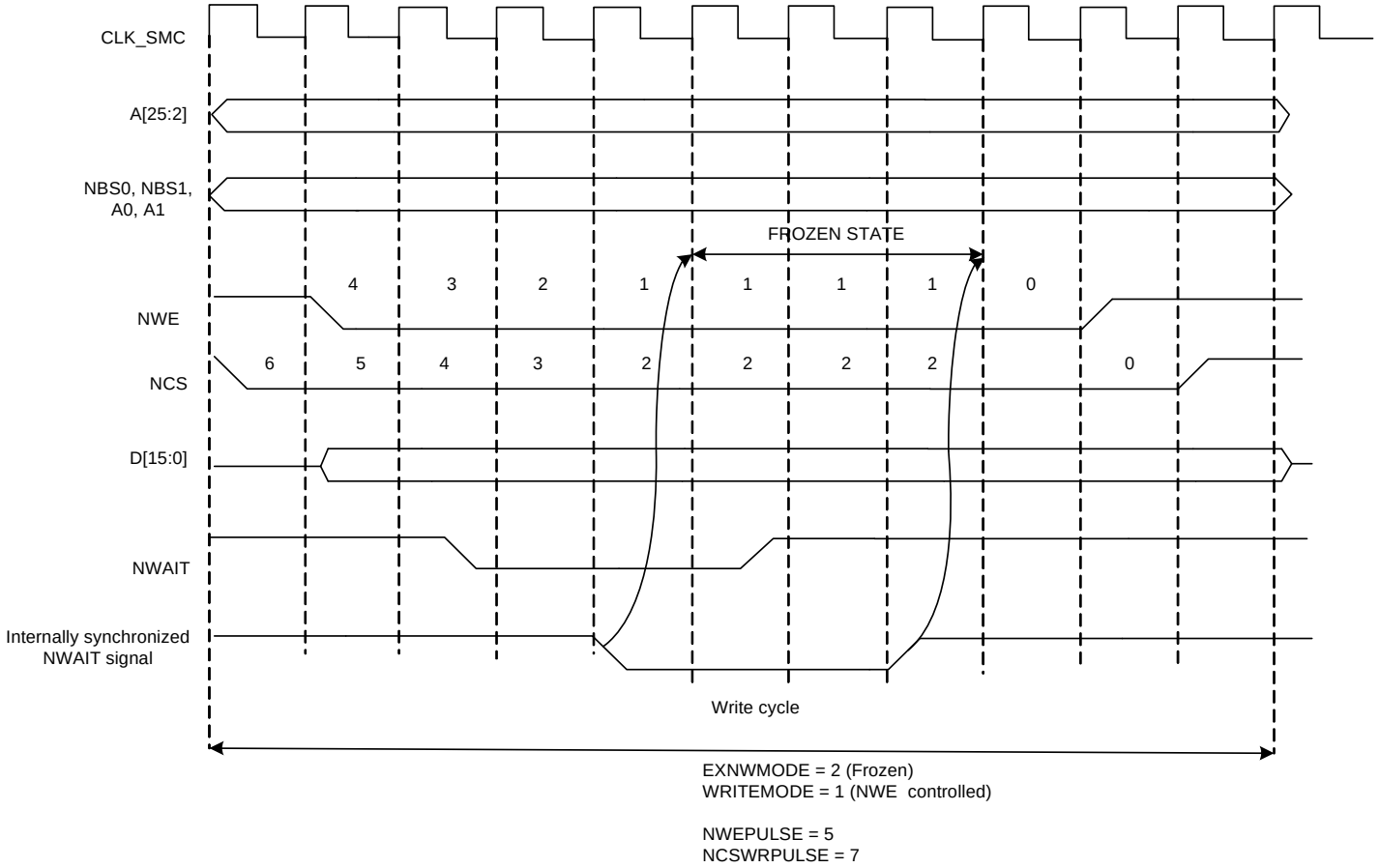
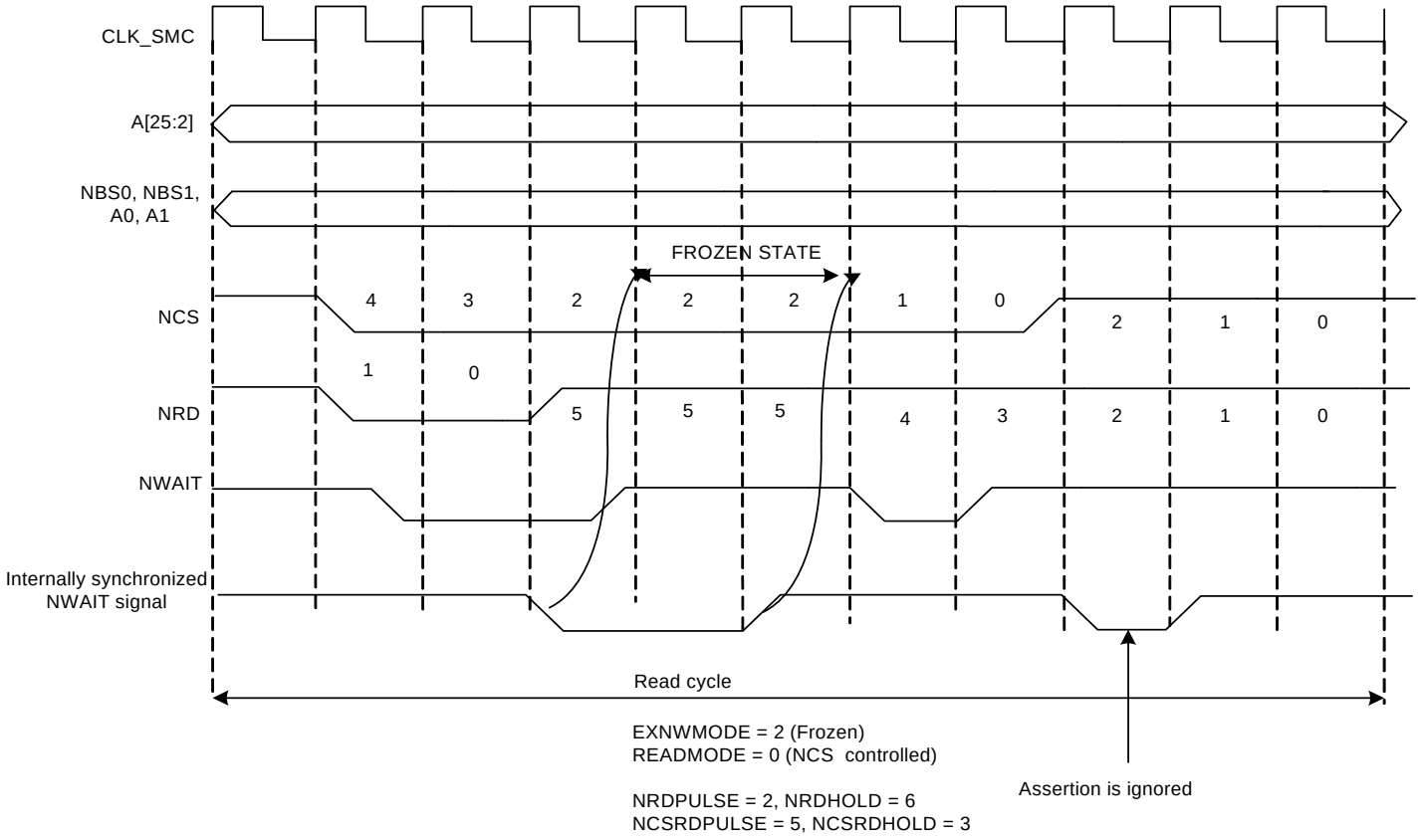


Figure 16-26. Read Access with NWAIT Assertion in Frozen Mode (MODE.EXNWMODE = 2).



16.6.7.3 Ready mode

In Ready mode (MODE.EXNWMODE = 3), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 16-27 on page 195 and Figure 16-28 on page 196. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 16-28 on page 196.

Figure 16-27. NWAIT Assertion in Write Access: Ready Mode (MODE.EXNWMODE = 3).

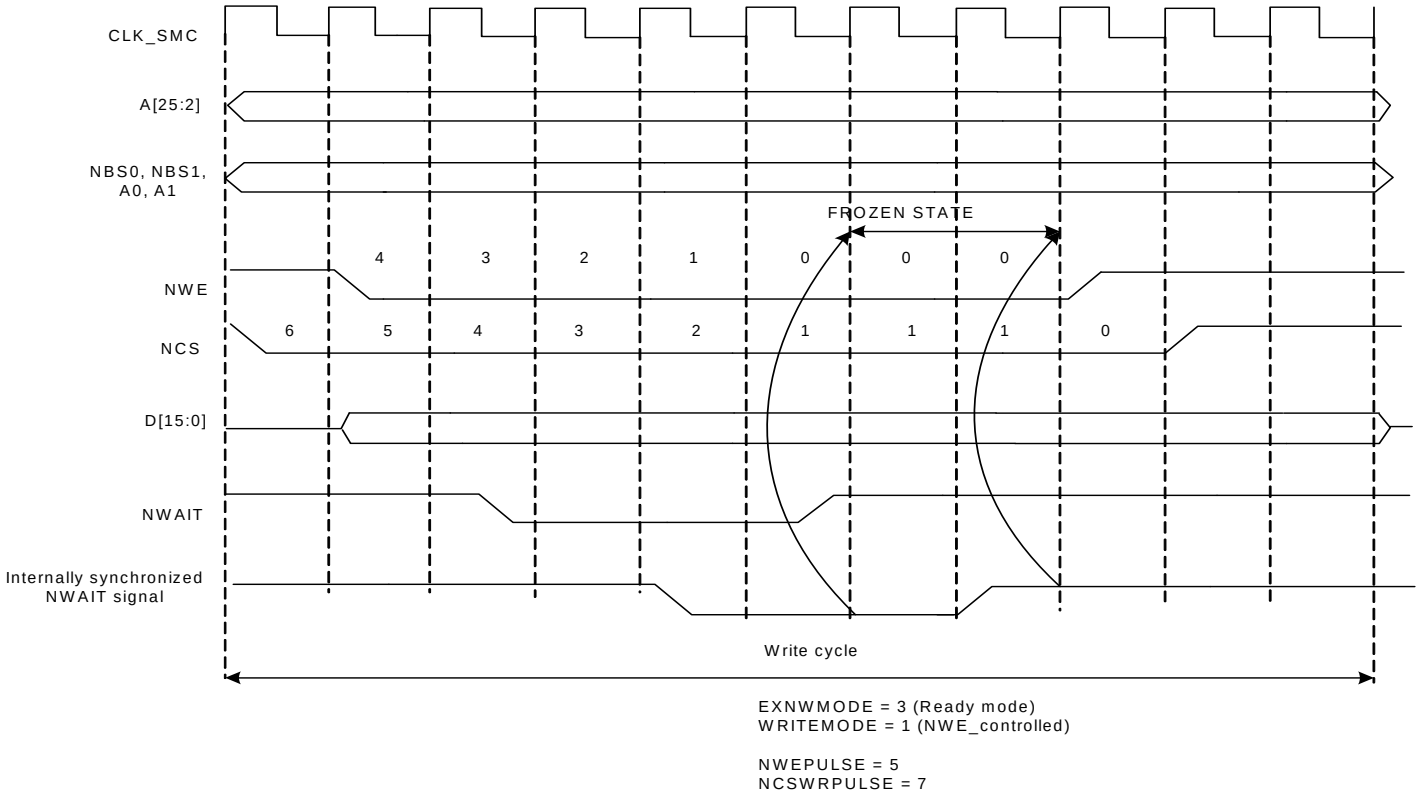
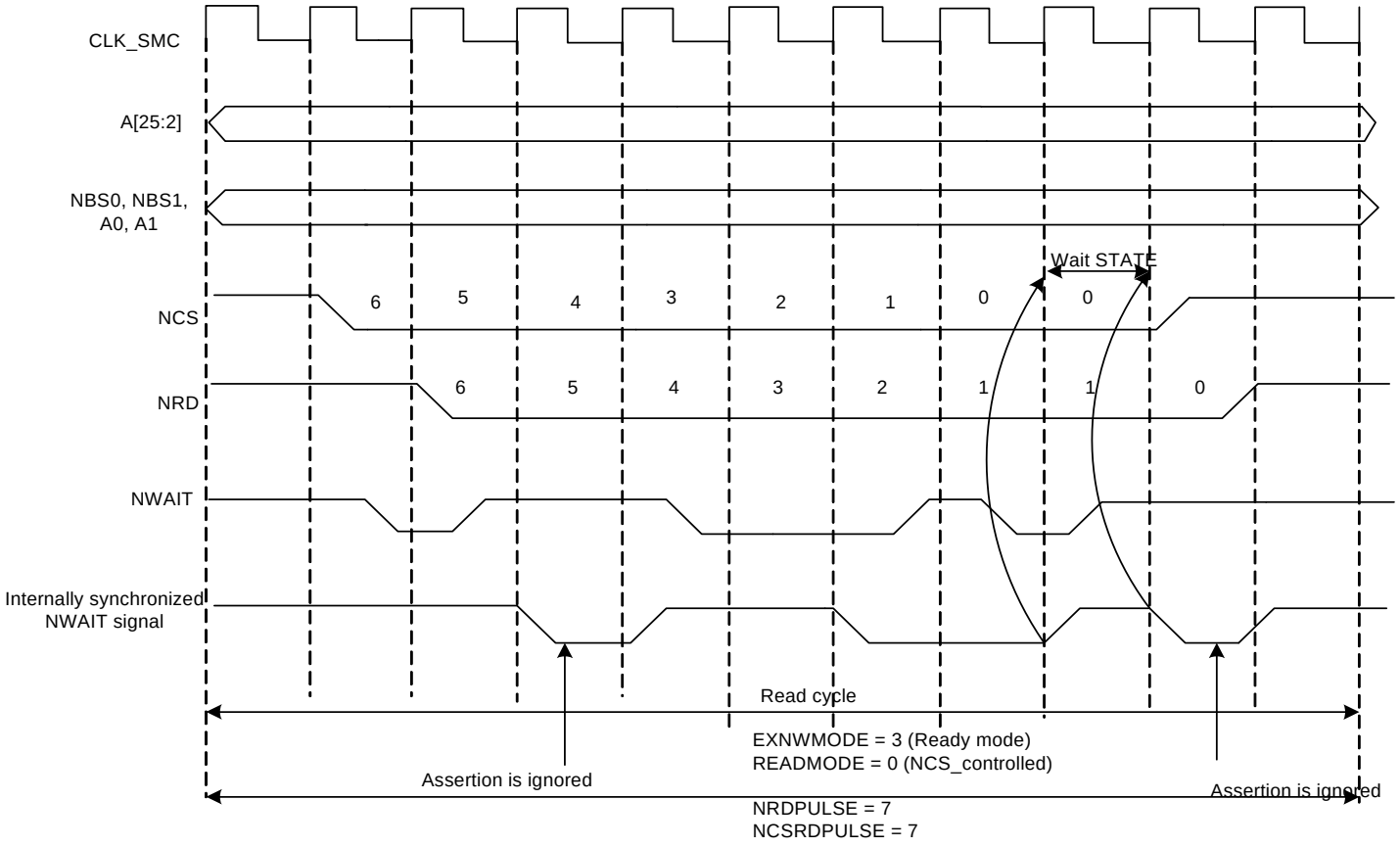


Figure 16-28. NWAIT Assertion in Read Access: Ready Mode (EXNWMODE = 3).



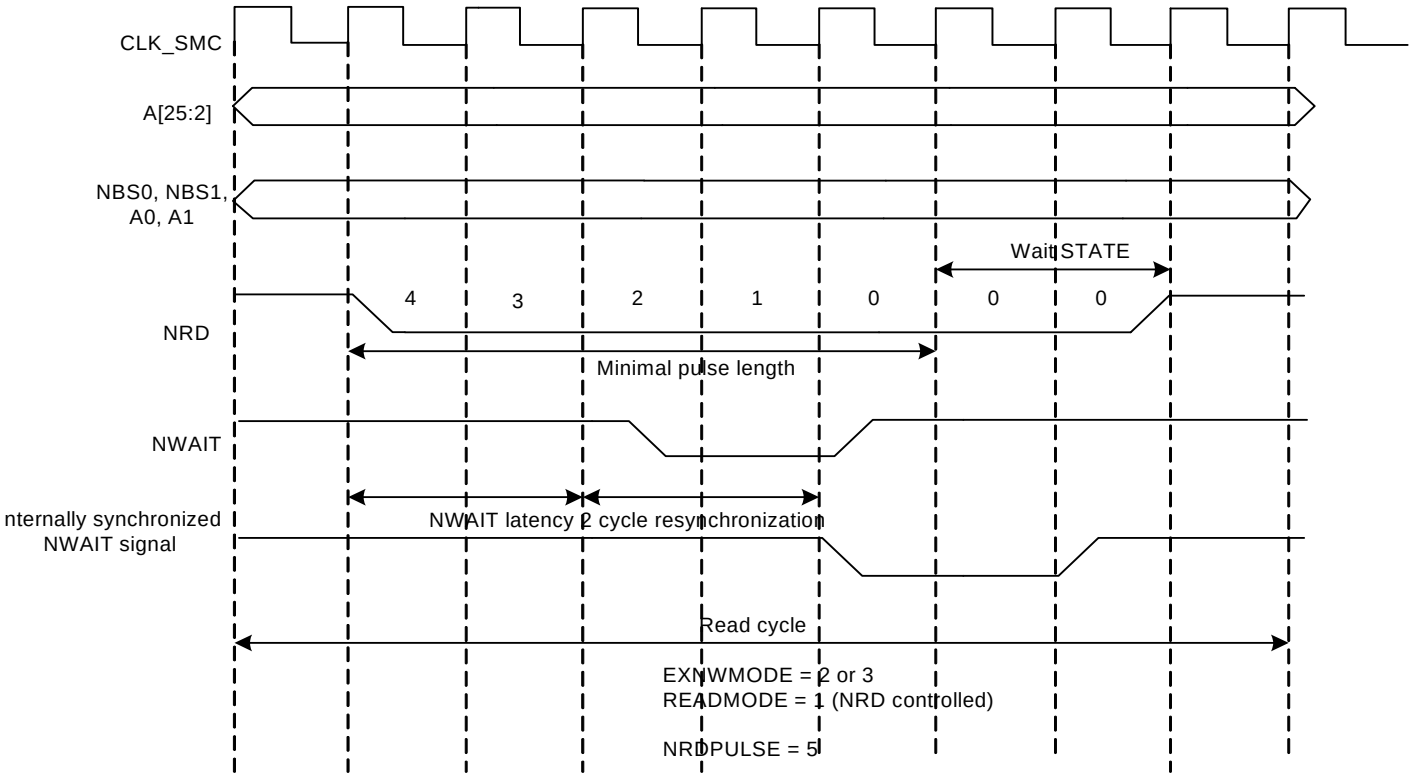
16.6.7.4 NWAIT latency and read/write timings

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the two cycles of resynchronization plus one cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in frozen mode as well as in ready mode. This is illustrated on Figure 16-29 on page 197.

When the MODE.EXNWMODE field is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

$$\text{minimal pulse length} = \text{NWAIT latency} + 2 \text{ synchronization cycles} + 1 \text{ cycle}$$

Figure 16-29. NWAIT Latency



16.6.8 Slow Clock Mode

The SMC is able to automatically apply a set of “slow clock mode” read/write waveforms when an internal signal driven by the SMC’s Power Management Controller is asserted because CLK_SMC has been turned to a very slow clock rate (typically 32 kHz clock rate). In this mode, the user-programmed waveforms are ignored and the slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the slow mode is active on all chip selects.

16.6.8.1 Slow clock mode waveforms

Figure 16-30 on page 198 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 16-5 on page 198 indicates the value of read and write parameters in slow clock mode.

Figure 16-30. Read and Write Cycles in Slow Clock Mode

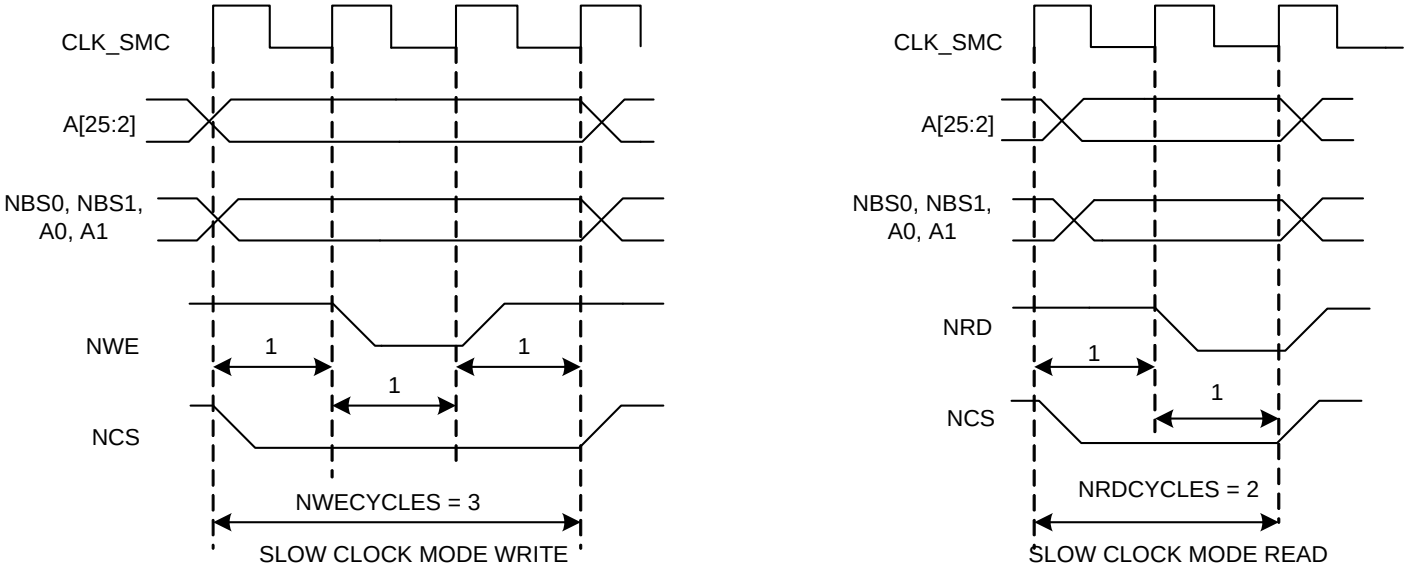


Table 16-5. Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRDSETUP	1	NWESETUP	1
NRDPULSE	1	NWEPULSE	1
NCSRDSETUP	0	NCSWRSETUP	0
NCSRDPULSE	2	NCSWRPULSE	3
NRDCYCLE	2	NWECYCLE	3

16.6.8.2 *Switching from (to) slow clock mode to (from) normal mode*

When switching from slow clock mode to the normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters. See Figure 16-31 on page 199. The external device may not be fast enough to support such timings.

Figure 16-32 on page 200 illustrates the recommended procedure to properly switch from one mode to the other.

Figure 16-31. Clock Rate Transition Occurs while the SMC is Performing a Write Operation

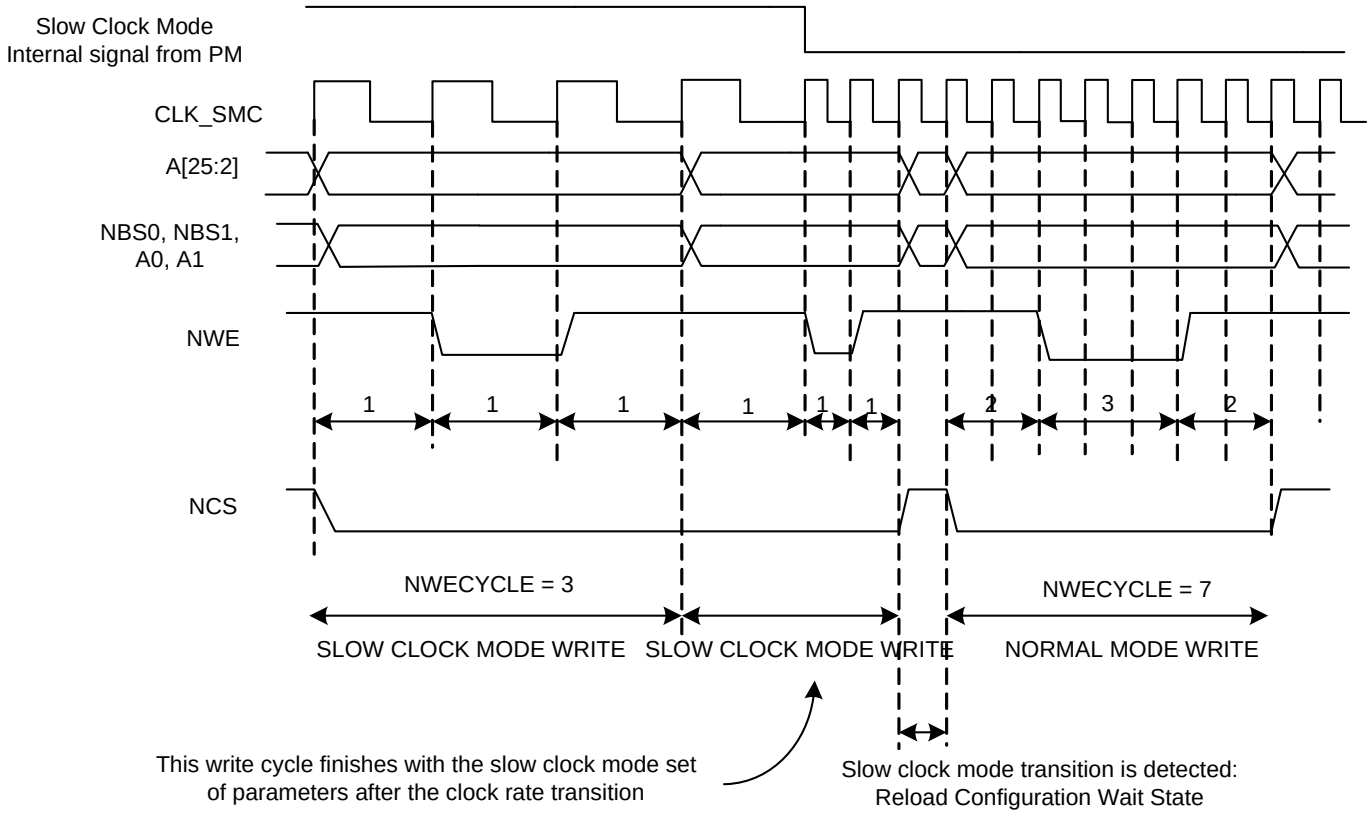
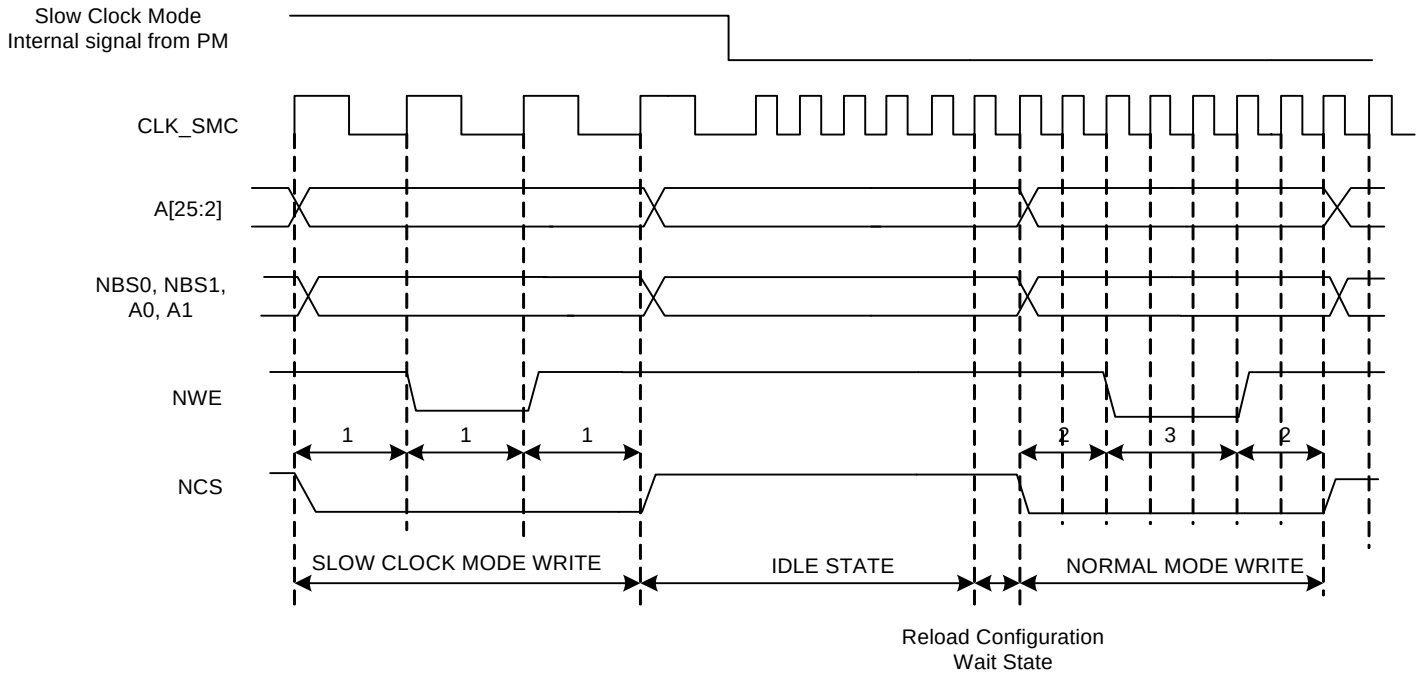


Figure 16-32. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



16.6.9 Asynchronous Page Mode

The SMC supports asynchronous burst reads in page mode, providing that the Page Mode Enabled bit is written to one in the MODE register (MODE.PMEN). The page size must be configured in the Page Size field in the MODE register (MODE.PS) to 4, 8, 16, or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 16-6 on page 200.

With page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 16-33 on page 201. When in page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 16-6. Page Address and Data Address within a Page

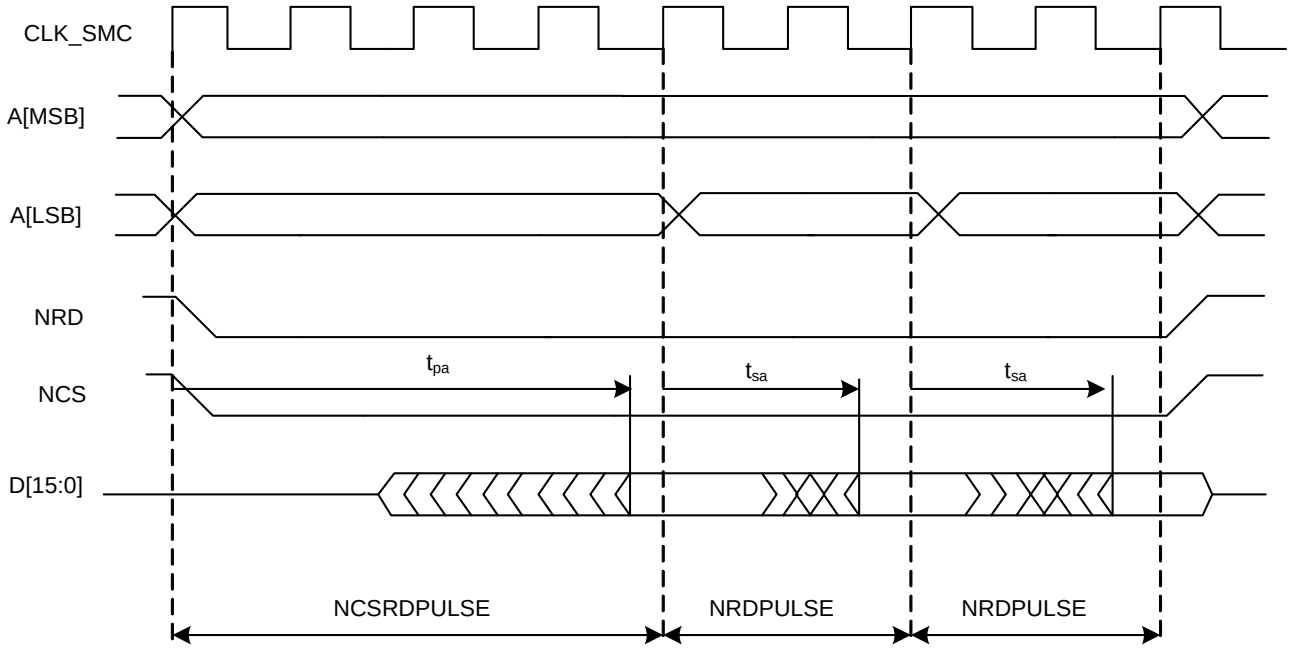
Page Size	Page Address(1)	Data Address in the Page(2)
4 bytes	A[25:2]	A[1:0]
8 bytes	A[25:3]	A[2:0]
16 bytes	A[25:4]	A[3:0]
32 bytes	A[25:5]	A[4:0]

Notes: 1. A denotes the address bus of the memory device
2. For 16-bit devices, the bit 0 of address is ignored.

16.6.9.1 Protocol and timings in page mode

Figure 16-33 on page 201 shows the NRD and NCS timings in page mode access.

Figure 16-33. Page Mode Read Protocol (Address MSB and LSB Are Defined in Table 16-6 on page 200)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the PULSE.NCSRDPULSE field value. The pulse length of subsequent accesses within the page are defined using the PULSE.NRDPULSE field value.

In page mode, the programming of the read timings is described in Table 16-7 on page 201:

Table 16-7. Programming of Read Timings in Page Mode

Parameter	Value	Definition
READMODE	'x'	No impact
NCSRDSETUP	'x'	No impact
NCSRDPULSE	t_{pa}	Access time of first access to the page
NRDSETUP	'x'	No impact
NRDPULSE	t_{sa}	Access time of subsequent accesses in the page
NRDCYCLE	'x'	No impact

The SMC does not check the coherency of timings. It will always apply the NCSRDPULSE timings as page access timing (t_{pa}) and the NRDPULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

16.6.9.2 Byte access type in page mode

The byte access type configuration remains active in page mode. For 16-bit or 32-bit page mode devices that require byte selection signals, configure the MODE.BAT bit to zero (byte select access type).

16.6.9.3 *Page mode restriction*

The page mode is not compatible with the use of the NWAIT signal. Using the page mode and the NWAIT signal may lead to unpredictable behavior.

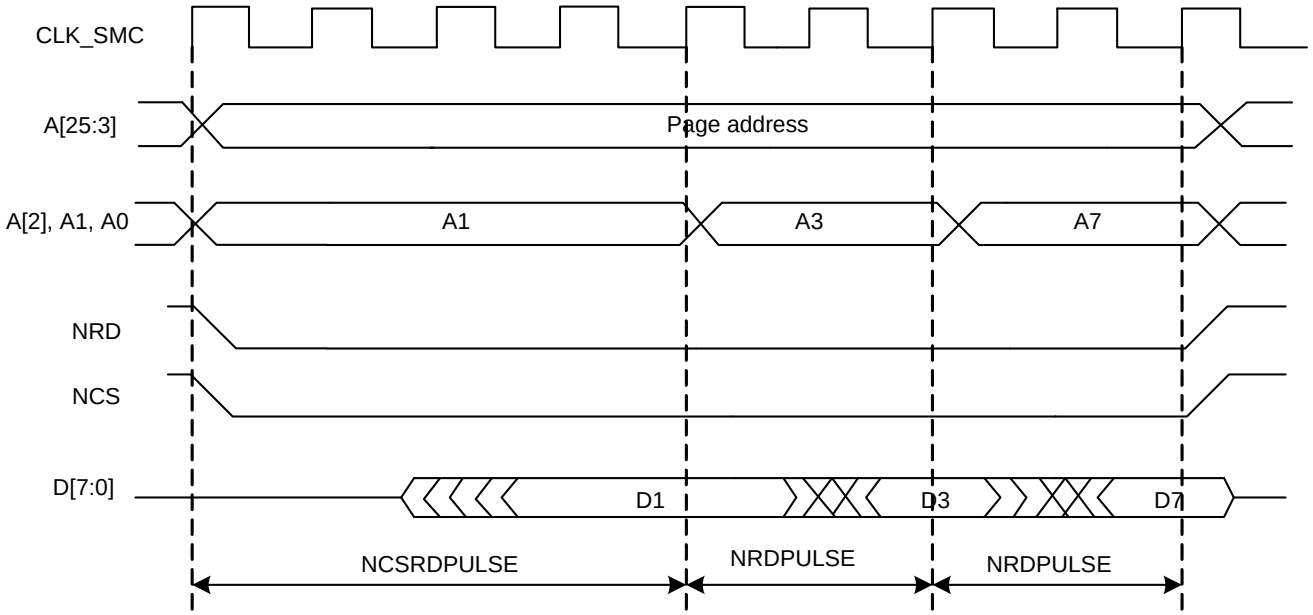
16.6.9.4 *Sequential and non-sequential accesses*

If the chip select and the MSB of addresses as defined in Table 16-6 on page 200 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 16-34 on page 202 illustrates access to an 8-bit memory device in page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

Figure 16-34. Access to Non-sequential Data within the Same Page



16.7 User Interface

The SMC is programmed using the registers listed in Table 16-8 on page 203. For each chip select, a set of four registers is used to program the parameters of the external device connected on it. In Table 16-8 on page 203, “CS_number” denotes the chip select number. Sixteen bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing anyone of the Mode Registers.

Table 16-8. SMC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00 + CS_number*0x10	Setup Register	SETUP	Read/Write	0x01010101
0x04 + CS_number*0x10	Pulse Register	PULSE	Read/Write	0x01010101
0x08 + CS_number*0x10	Cycle Register	CYCLE	Read/Write	0x00030003
0x0C + CS_number*0x10	Mode Register	MODE	Read/Write	0x10002103

16.7.1 Setup Register

Register Name: SETUP

Access Type: Read/Write

Offset: 0x00 + CS_number*0x10

Reset Value: 0x01010101

31	30	29	28	27	26	25	24
–	–	NCSRDSETUP					

23	22	21	20	19	18	17	16
–	–	NRDSETUP					

15	14	13	12	11	10	9	8
–	–	NCSWRSETUP					

7	6	5	4	3	2	1	0
–	–	NWESETUP					

- **NCSRDSETUP: NCS Setup Length in READ Access**
  In read access, the NCS signal setup length is defined as:

$$\text{NCS Setup Length in read access} = (128 \times NCSRDSETUP[5] + NCSRDSETUP[4:0]) \text{ clock cycles}$$

- **NRDSETUP: NRD Setup Length**
  The NRD signal setup length is defined in clock cycles as:

$$\text{NRD Setup Length} = (128 \times NRDSETUP[5] + NRDSETUP[4:0]) \text{ clock cycles}$$

- **NCSWRSETUP: NCS Setup Length in WRITE Access**
  In write access, the NCS signal setup length is defined as:

$$\text{NCS Setup Length in write access} = (128 \times NCSWRSETUP[5] + NCSWRSETUP[4:0]) \text{ clock cycles}$$

- **NWESETUP: NWE Setup Length**
  The NWE signal setup length is defined as:

$$\text{NWE Setup Length} = (128 \times NWESETUP[5] + NWESETUP[4:0]) \text{ clock cycles}$$

16.7.2 Pulse Register

Register Name: PULSE

Access Type: Read/Write

Offset: 0x04 + CS_number*0x10

Reset Value: 0x01010101

31	30	29	28	27	26	25	24
–	NCSRDPULSE						

23	22	21	20	19	18	17	16
–	NRDPULSE						

15	14	13	12	11	10	9	8
–	NCSWRPULSE						

7	6	5	4	3	2	1	0
–	NWEPULSE						

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

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

  $$\text{NCS Pulse Length in read access} = (256 \times NCSRDPULSE[6] + NCSRDPULSE[5:0]) \text{ clock cycles}$$

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

  In page mode read access, the NCSRDPULSE field defines the duration of the first access to one page.

- **NRDPULSE: NRD Pulse Length**

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

  $$\text{NRD Pulse Length} = (256 \times NRDPULSE[6] + NRDPULSE[5:0]) \text{ clock cycles}$$

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

  In page mode read access, the NRDPULSE field defines the duration of the subsequent accesses in the page.

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

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

  $$\text{NCS Pulse Length in write access} = (256 \times NCSWRPULSE[6] + NCSWRPULSE[5:0]) \text{ clock cycles}$$

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

- **NWEPULSE: NWE Pulse Length**

  The NWE signal pulse length is defined as:

  $$\text{NWE Pulse Length} = (256 \times NWEPULSE[6] + NWEPULSE[5:0]) \text{ clock cycles}$$

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

16.7.3 Cycle Register

Register Name: CYCLE

Access Type: Read/Write

Offset: 0x08 + CS_number*0x10

Reset Value: 0x00030003

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NRDCYCLE[8]

23	22	21	20	19	18	17	16
NRDCYCLE[7:0]							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWECYCLE[8]

7	6	5	4	3	2	1	0
NWECYCLE[7:0]							

- **NRDCYCLE[8:0]: Total Read Cycle Length**

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

$$\text{Read Cycle Length} = (256 \times NRDCYCLE[8:7] + NRDCYCLE[6:0]) \text{ clock cycles}$$

- **NWECYCLE[8:0]: Total Write Cycle Length**

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

$$\text{Write Cycle Length} = (256 \times NWECYCLE[8:7] + NWECYCLE[6:0]) \text{ clock cycles}$$

16.7.4 Mode Register

Register Name: MODE

Access Type: Read/Write

Offset: 0x0C + CS_number*0x10

Reset Value: 0x10002103

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

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

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

7	6	5	4	3	2	1	0
–	–	EXNWMODE		–	–	WRITEMODE	READMODE

- **PS: Page Size**

  If page mode is enabled, this field indicates the size of the page in bytes.

PS	Page Size
0	4-byte page
1	8-byte page
2	16-byte page
3	32-byte page

- **PMEN: Page Mode Enabled**

  1: Asynchronous burst read in page mode is applied on the corresponding chip select.

  0: Standard read is applied.

- **TDFMODE: TDF Optimization**

  1: TDF optimization is enabled. The number of TDF wait states is optimized using the setup period of the next read/write access.

  0: TDF optimization is disabled.The number of TDF wait states is inserted before the next access begins.

- **TDFCYCLES: Data Float Time**

  This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDFCYCLES period. The external bus cannot be used by another chip select during TDFCYCLES plus one cycles. From 0 up to 15 TDFCYCLES can be set.

- **DBW: Data Bus Width**

DBW	Data Bus Width
0	8-bit bus
1	16-bit bus
2	Reserved
3	Reserved

- **BAT: Byte Access Type**

  This field is used only if DBW defines a 16-bit data bus.

BAT	Byte Access Type
0	Byte select access type: Write operation is controlled using NCS, NWE, NBS0, NBS1 Read operation is controlled using NCS, NRD, NBS0, NBS1
1	Byte write access type: Write operation is controlled using NCS, NWR0, NWR1 Read operation is controlled using NCS and NRD

- **EXNWMODE: External WAIT Mode**

  The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNWMODE	External NWAIT Mode
0	Disabled: the NWAIT input signal is ignored on the corresponding chip select.
1	Reserved
2	Frozen Mode: if asserted, the NWAIT signal freezes the current read or write cycle. after deassertion, the read or write cycle is resumed from the point where it was stopped.
3	Ready Mode: the NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

- **WRITEMODE: Write Mode**

  1: The write operation is controlled by the NWE signal. If TDF optimization is enabled (TDFMODE =1), TDF wait states will be inserted after the setup of NWE.

  0: The write operation is controlled by the NCS signal. If TDF optimization is enabled (TDFMODE =1), TDF wait states will be inserted after the setup of NCS.

- **READMODE: Read Mode**

READMODE	Read Access Mode
0	The read operation is controlled by the NCS signal. If TDF are programmed, the external bus is marked busy after the rising edge of NCS. If TDF optimization is enabled (TDFMODE = 1), TDF wait states are inserted after the setup of NCS.
1	The read operation is controlled by the NRD signal. If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD. If TDF optimization is enabled (TDFMODE =1), TDF wait states are inserted after the setup of NRD.

17. SDRAM Controller (SDRAMC)

Rev: 2.2.0.3

17.1 Features

- 128-Mbytes address space
- Numerous configurations supported
  - 2K, 4K, 8K row address memory parts
  - SDRAM with two or four internal banks
  - SDRAM with 16-bit data path
- Programming facilities
  - Word, halfword, 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
  - Automatic update of DS, TCR and PASR parameters (mobile SDRAM devices)
- Energy-saving capabilities
  - Self-refresh, power-down, and deep power-down modes supported
  - Supports mobile SDRAM devices
- Error detection
  - Refresh error interrupt
- SDRAM power-up initialization by software
- CAS latency of one, two, and three supported
- Auto Precharge command not used

17.2 Overview

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-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) and halfword (16-bit) accesses.

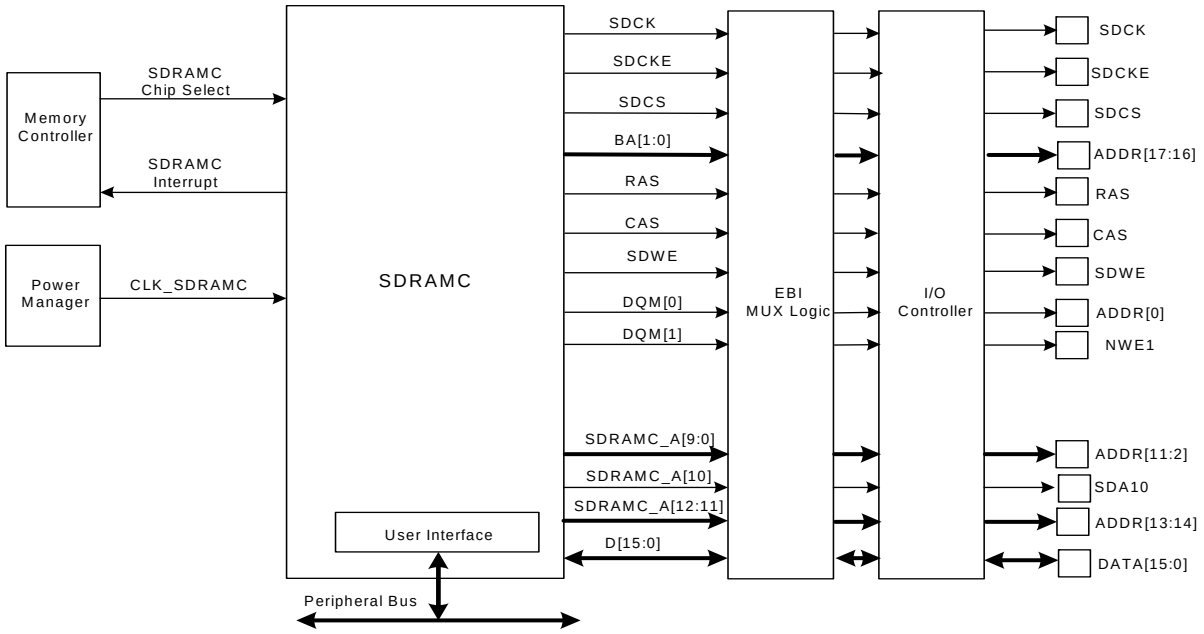
The SDRAMC supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAMC supports a CAS latency of one, two, or three and optimizes the read access depending on the frequency.

The different modes available (self refresh, power-down, and deep power-down modes) minimize power consumption on the SDRAM device.

17.3 Block Diagram

Figure 17-1. SDRAM Controller Block Diagram



17.4 I/O Lines Description

Table 17-1. I/O Lines Description

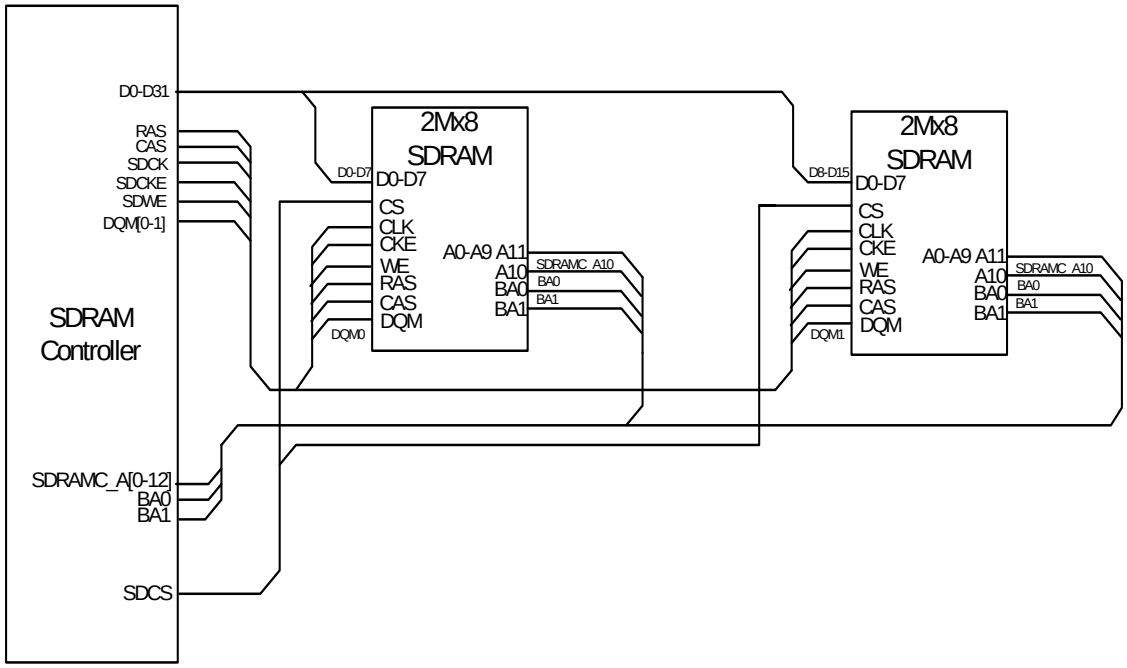
Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
DQM[1:0]	Data Mask Enable Signals	Output	High
SDRAMC_A[12:0]	Address Bus	Output	
D[15:0]	Data Bus	Input/Output	

17.5 Application Example

17.5.1 Hardware Interface

Figure 17-2 on page 212 shows an example of SDRAM device connection using a 16-bit data bus width. It is important to note that this example is given for a direct connection of the devices to the SDRAMC, without External Bus Interface or I/O Controller multiplexing.

Figure 17-2. SDRAM Controller Connections to SDRAM Devices: 16-bit Data Bus Width



17.5.2 Software Interface

The SDRAM address space is organized into banks, rows, and columns. The SDRAMC allows mapping different memory types according to the values set in the SDRAMC Configuration Register (CR).

The SDRAMC's function is to make the SDRAM device access protocol transparent to the user. Table 17-2 on page 213 to Table 17-4 on page 213 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

17.5.2.1 16-bit memory data bus width

Table 17-2. 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
						BA[1:0]		Row[10:0]											Column[7:0]								M0
					BA[1:0]			Row[10:0]											Column[8:0]								M0
				BA[1:0]				Row[10:0]											Column[9:0]								M0
			BA[1:0]					Row[10:0]											Column[10:0]								M0

Table 17-3. 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
					BA[1:0]			Row[11:0]											Column[7:0]								M0
				BA[1:0]				Row[11:0]											Column[8:0]								M0
			BA[1:0]					Row[11:0]											Column[9:0]								M0
		BA[1:0]						Row[11:0]											Column[10:0]								M0

Table 17-4. 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
				BA[1:0]				Row[12:0]											Column[7:0]								M0
			BA[1:0]					Row[12:0]											Column[8:0]								M0
		BA[1:0]						Row[12:0]											Column[9:0]								M0
	BA[1:0]							Row[12:0]											Column[10:0]								M0

Notes: 1. M0 is the byte address inside a 16-bit halfword.

17.6 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

17.6.1 I/O Lines

The SDRAMC module signals pass through the External Bus Interface (EBI) module where they are multiplexed. The user must first configure the I/O controller to assign the EBI pins corresponding to SDRAMC signals to their peripheral function. If I/O lines of the EBI corresponding to SDRAMC signals are not used by the application, they can be used for other purposes by the I/O Controller.

17.6.2 Power Management

The SDRAMC must be properly stopped before entering in reset mode, i.e., the user must issue a Deep power mode command in the Mode (MD) register and wait for the command to be completed.

17.6.3 Clocks

The clock for the SDRAMC bus interface (CLK_SDRAMC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SDRAMC before disabling the clock, to avoid freezing the SDRAMC in an undefined state.

17.6.4 Interrupts

The SDRAMC interrupt request line is connected to the interrupt controller. Using the SDRAMC interrupt requires the interrupt controller to be programmed first.

17.7 Functional Description

17.7.1 SDRAM Device Initialization

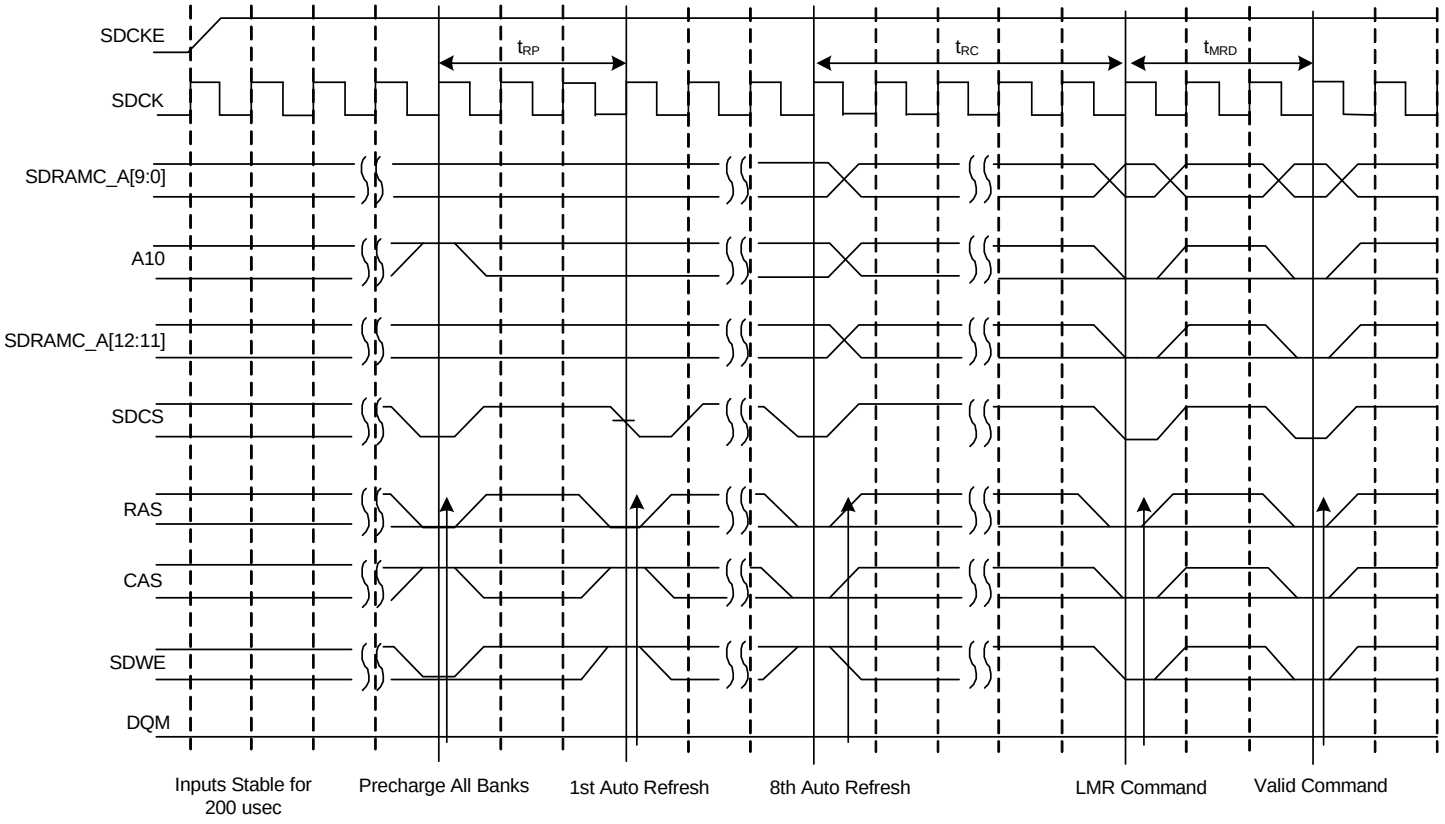
The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

1. SDRAM features must be defined in the CR register by writing the following fields with the desired value: asynchronous timings (TXSR, TRAS, TRCD, TRP, TRC, and TWR), Number of Columns (NC), Number of Rows (NR), Number of Banks (NB), CAS Latency (CAS), and the Data Bus Width (DBW).
2. For mobile SDRAM devices, Temperature Compensated Self Refresh (TCSR), Drive Strength (DS) and Partial Array Self Refresh (PASR) fields must be defined in the Low Power Register (LPR).
3. The Memory Device Type field must be defined in the Memory Device Register (MDR.MD).
4. A No Operation (NOP) command must be issued to the SDRAM devices to start the SDRAM clock. The user must write the value one to the Command Mode field in the SDRAMC Mode Register (MR.MODE) and perform a write access to any SDRAM address.
5. A minimum pause of 200µs is provided to precede any signal toggle.
6. An All Banks Precharge command must be issued to the SDRAM devices. The user must write the value two to the MR.MODE field and perform a write access to any SDRAM address.
7. Eight Auto Refresh commands are provided. The user must write the value four to the MR.MODE field and performs a write access to any SDRAM location eight times.
8. A Load Mode Register command must be issued to program the parameters of the SDRAM devices in its Mode Register, in particular CAS latency, burst type, and burst length. The user must write the value three to the MR.MODE field and perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to zero. See Section 17.8.1 for details about Load Mode Register command.
9. For mobile SDRAM initialization, an Extended Load Mode Register command must be issued to program the SDRAM devices parameters (TCSR, PASR, DS). The user must write the value five to the MR.MODE field and perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are equal to one. See Section 17.8.1 for details about Extended Load Mode Register command.

10. The user must go into Normal Mode, writing the value 0 to the MR.MODE field and performing a write access at any location in the SDRAM.
11. Write the refresh rate into the Refresh Timer Count field in the Refresh Timer Register (TR.COUNT). The refresh rate is the delay between two successive refresh cycles. The SDRAM device requires a refresh every 15.625µs or 7.81µs. With a 100MHz frequency, the TR register must be written with the value 1562 (15.625 µs x 100 MHz) or 781 (7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

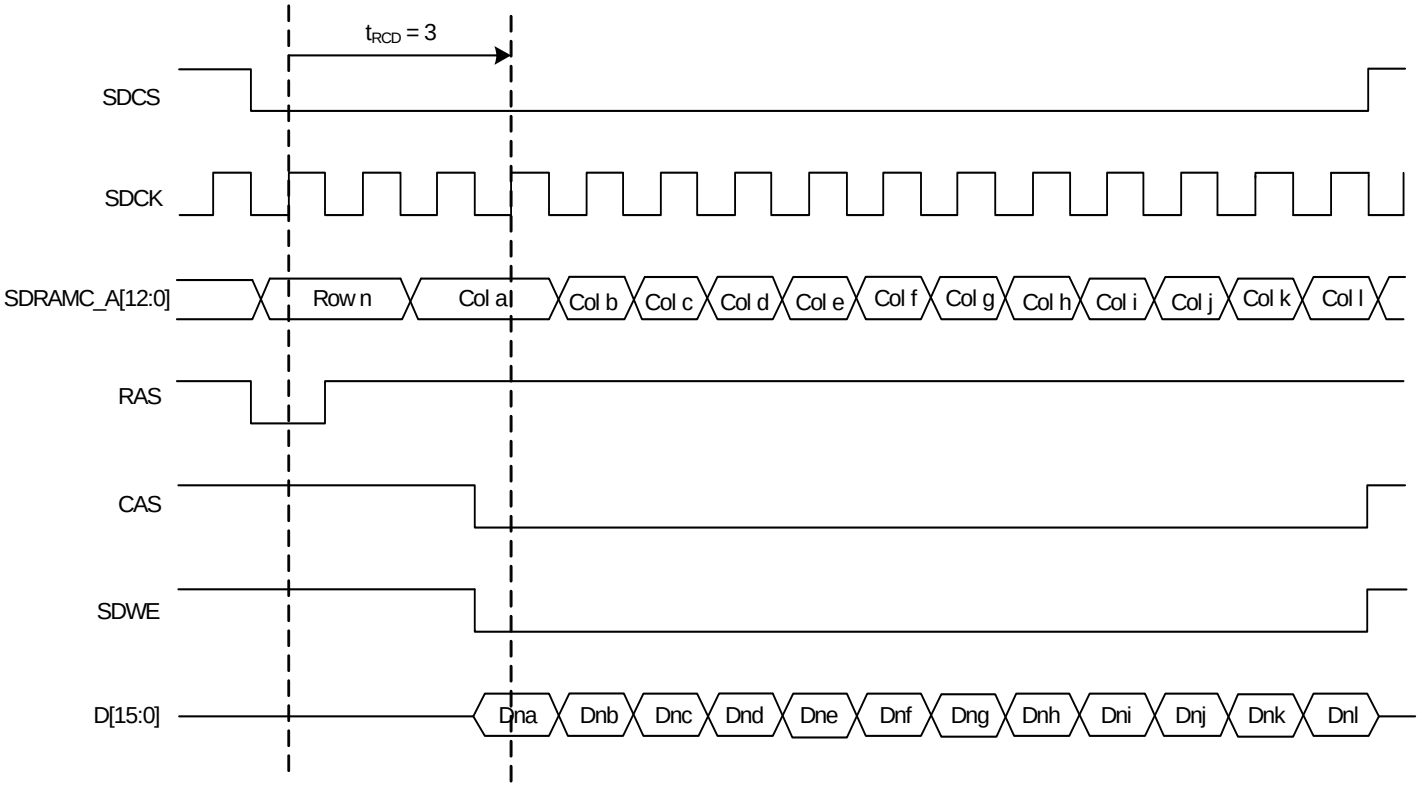
Figure 17-3. SDRAM Device Initialization Sequence



17.7.2 SDRAM Controller Write Cycle

The SDRAMC allows burst access or single access. In both cases, the SDRAMC keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAMC 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 SDRAMC 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 and active (t_{RP}) commands and between active and write (t_{RCD}) commands. For definition of these timing parameters, refer to the Section 17.8.3. This is described in Figure 17-4 on page 216.

Figure 17-4. Write Burst, 16-bit SDRAM Access



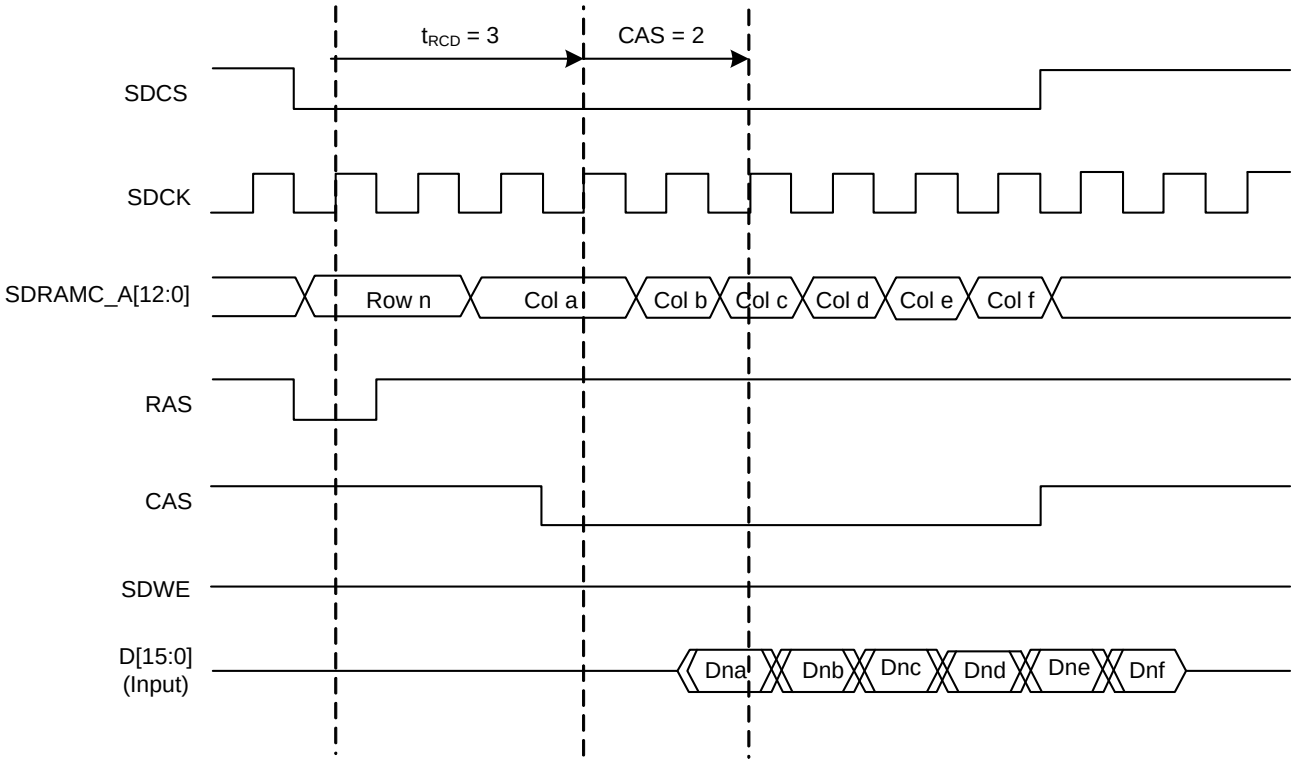
17.7.3 SDRAM Controller Read Cycle

The SDRAMC allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAMC keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAMC automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active (t_{RP}) commands and between active and read (t_{RCD}) commands. These two parameters are set in the CR register of the SDRAMC. After a read command, additional wait states are generated to comply with the CAS latency (one, two, or three clock delays specified in the CR register).

For a single access or an incremented burst of unspecified length, the SDRAMC anticipates the next access. While the last value of the column is returned by the SDRAMC on the bus, the SDRAMC anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

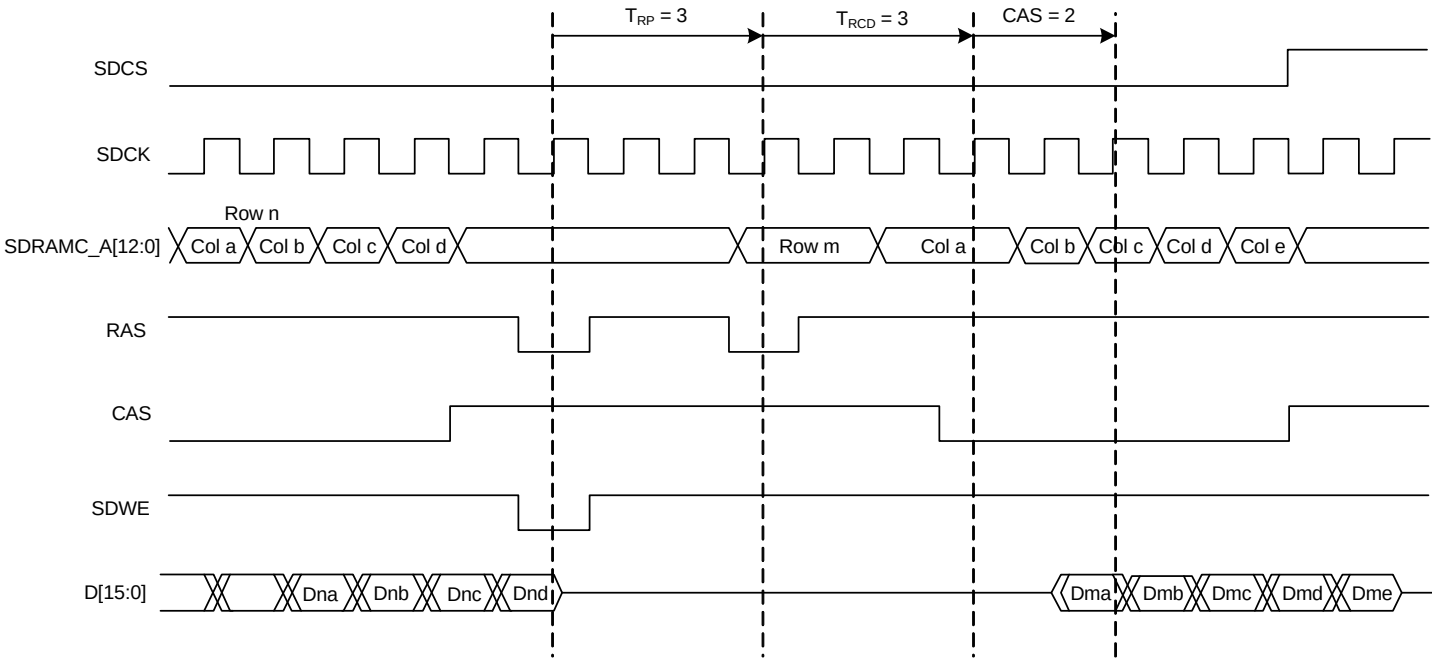
Figure 17-5. Read Burst, 16-bit SDRAM Access



17.7.4 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAMC 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 and active (t_{RP}) commands and between the active and read (t_{RCD}) commands. This is described in Figure 17-6 on page 218.

Figure 17-6. Read Burst with Boundary Row Access



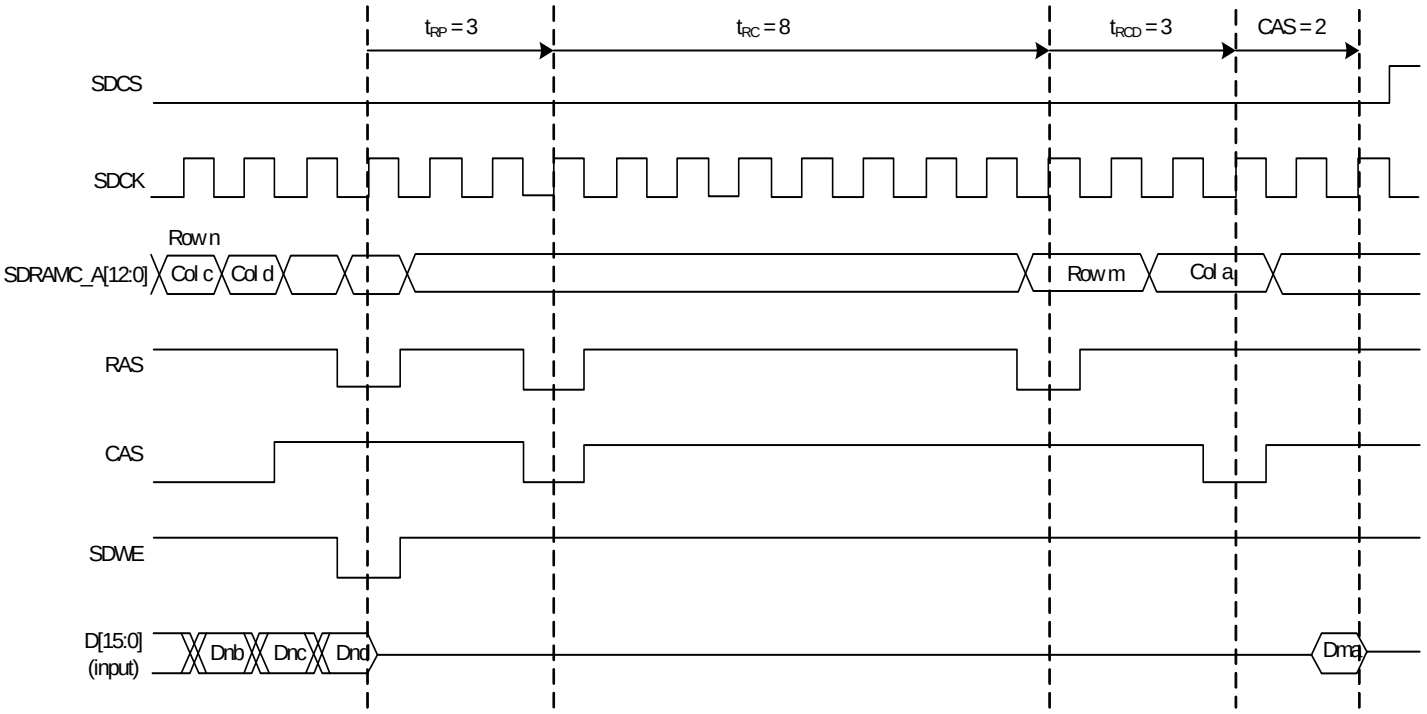
17.7.5 SDRAM Controller Refresh Cycles

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 SDRAMC generates these auto refresh commands periodically. An internal timer is loaded with the value in the Refresh Timer Register (TR) that indicates the number of clock cycles between successive refresh cycles.

A refresh error interrupt is generated when the previous auto refresh command did not perform. In this case a Refresh Error Status bit is set in the Interrupt Status Register (ISR.RES). It is cleared by reading the ISR register.

When the SDRAMC initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. See Figure 17-7 on page 219.

Figure 17-7. Refresh Cycle Followed by a Read Access



17.7.6 Power Management

Three low power modes are available:

- Self refresh mode: the SDRAM executes its own auto refresh cycles without control of the SDRAMC. Current drained by the SDRAM is very low.
- Power-down mode: auto refresh cycles are controlled by the SDRAMC. Between auto refresh cycles, the SDRAM is in power-down. Current drained in power-down mode is higher than in self refresh mode.
- Deep power-down mode (only available with mobile SDRAM): the SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAMC activates one low power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in self refresh and power-down mode after the last access by configuring the Timeout field in the Low Power Register (LPR.TIMEOUT).

17.7.6.1 Self refresh mode

This mode is selected by writing the value one to the Low Power Configuration Bits field in the SDRAMC Low Power Register (LPR.LPCB). 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 SDRAMC provides a sequence of commands and exits self refresh mode.

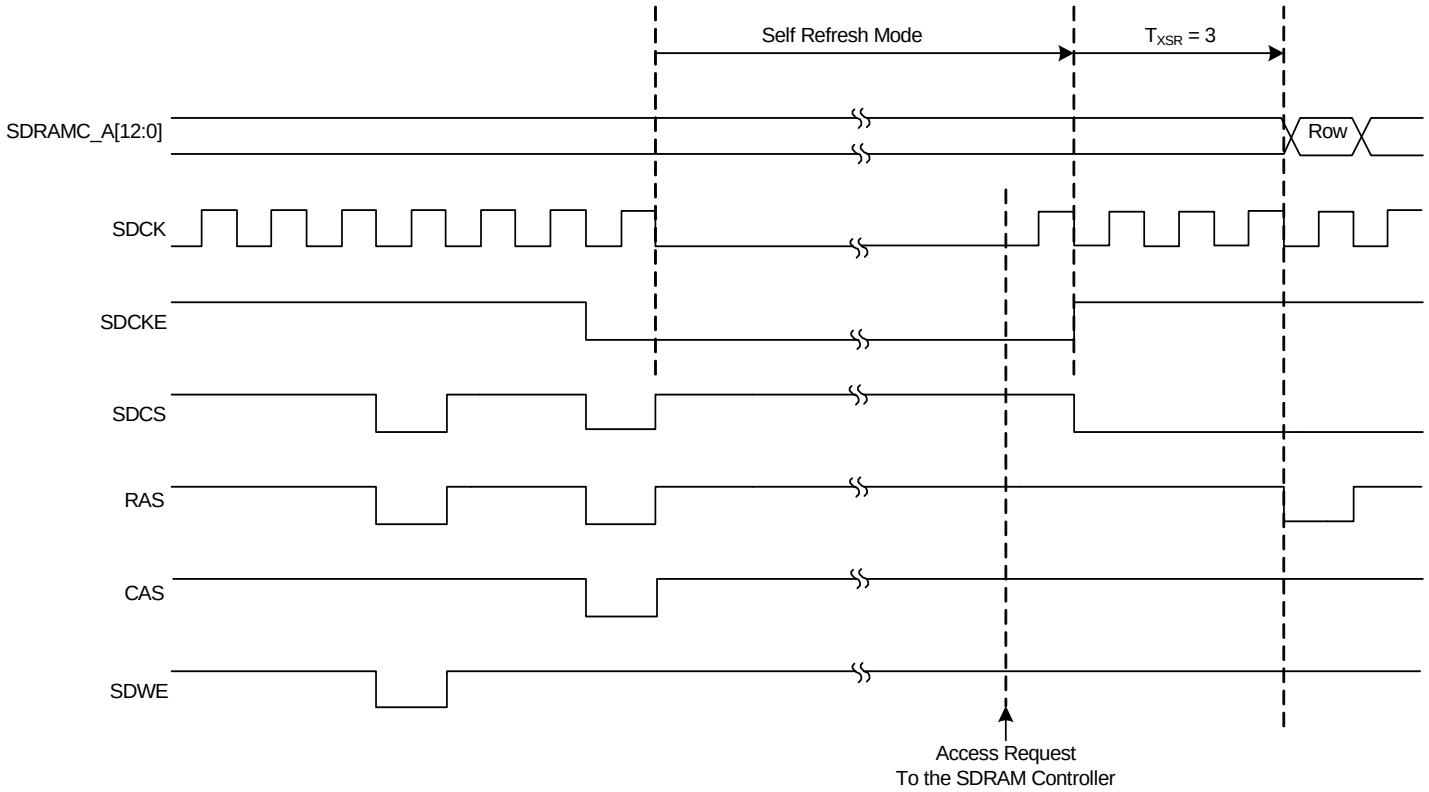
Some low power SDRAMs (e.g., mobile SDRAM) can refresh only one quarter or a half quarter or all banks of the SDRAM array. This feature reduces the self refresh current. To configure this feature, Temperature Compensated Self Refresh (TCSR), Partial Array Self Refresh (PASR)

and Drive Strength (DS) parameters must be set by writing the corresponding fields in the LPR register, and transmitted to the low power SDRAM device during initialization.

After initialization, as soon as the LPR.PASR, LPR.DS, or LPR.TCSR fields are modified and self refresh mode is activated, the SDRAMC issues an Extended Load Mode Register command to the SDRAM and the Extended Mode Register of the SDRAM device is accessed automatically. The PASR/DS/TCSR parameters values are therefore updated before entry into self refresh mode.

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 17-8 on page 220.

Figure 17-8. Self Refresh Mode Behavior

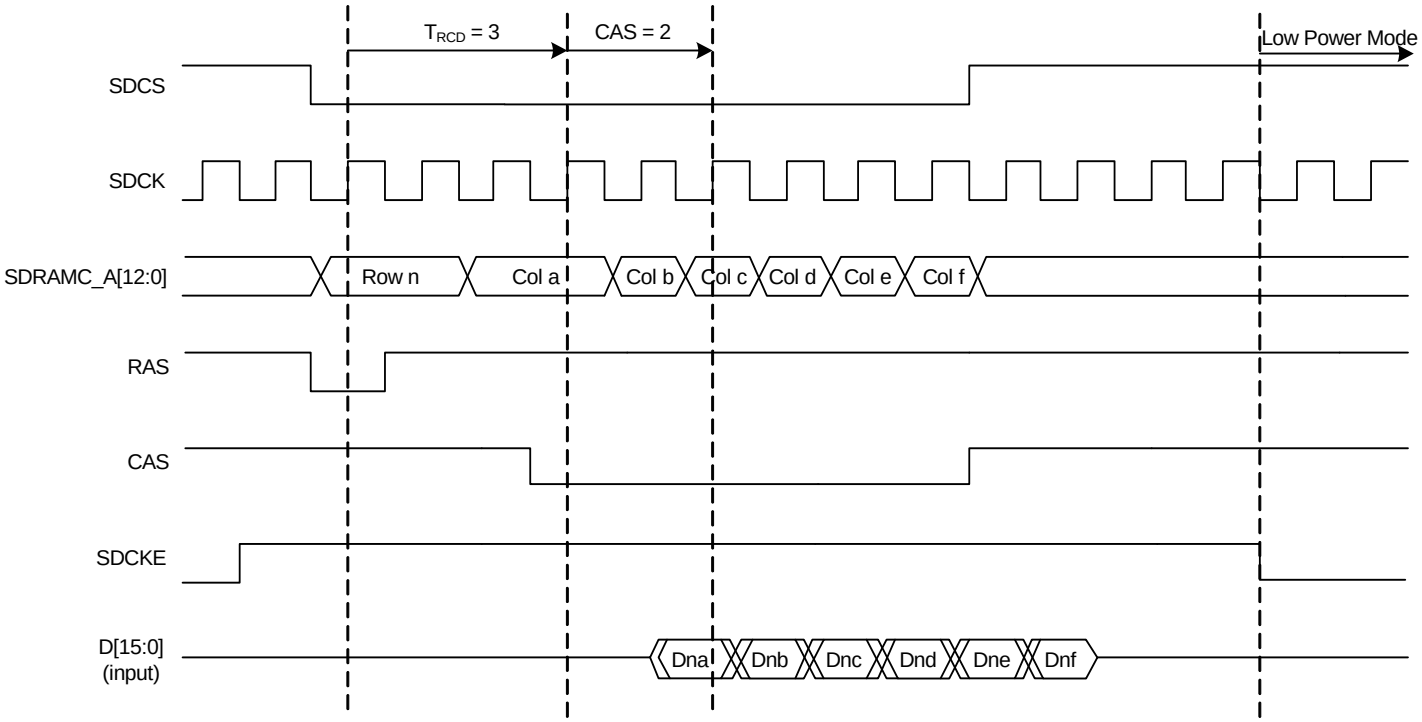


17.7.6.2 Low power mode

This mode is selected by writing the value two to the LPR.LPCB field. Power consumption is greater than in self refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to self refresh mode, the SDRAM device cannot remain in low power mode longer than the refresh period (64ms for a whole device refresh operation). As no auto refresh operations are performed by the SDRAM itself, the SDRAMC carries out the refresh operation. The exit procedure is faster than in self refresh mode.

This is described in Figure 17-9 on page 221.

Figure 17-9. Low Power Mode Behavior



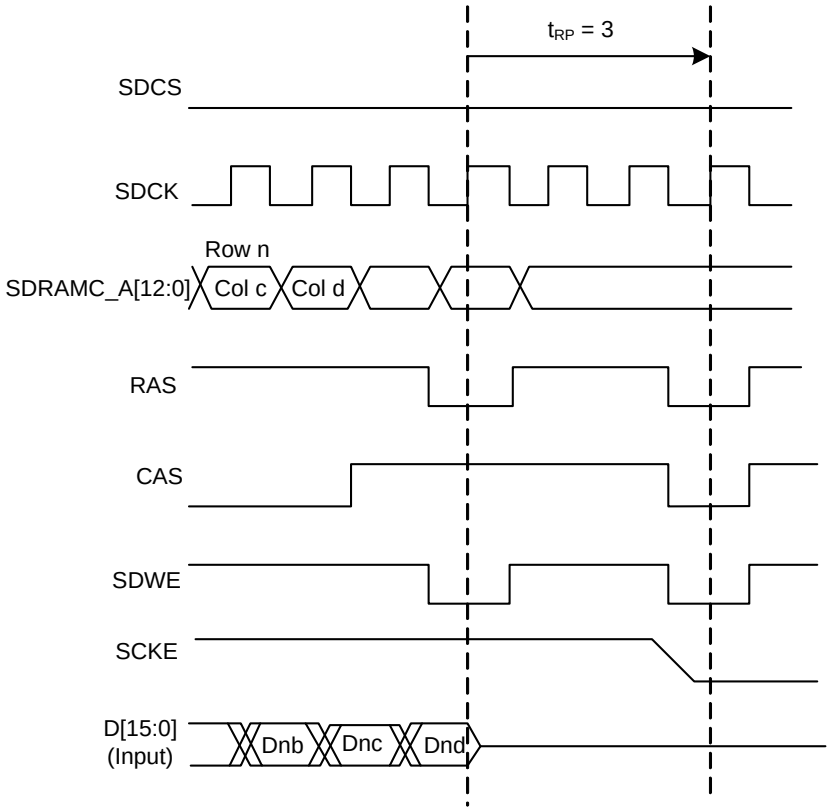
17.7.6.3 Deep power-down mode

This mode is selected by writing the value three to the LPR.LPCB field. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the user must not access to the SDRAM until a new initialization sequence is done (See Section 17.7.1).

This is described in Figure 17-10 on page 222.

Figure 17-10. Deep Power-down Mode Behavior



17.8 User Interface

Table 17-5. SDRAMC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Mode Register	MR	Read/Write	0x00000000
0x04	Refresh Timer Register	TR	Read/Write	0x00000000
0x08	Configuration Register	CR	Read/Write	0x852372C0
0x0C	High Speed Register	HSR	Read/Write	0x00000000
0x10	Low Power Register	LPR	Read/Write	0x00000000
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x20	Interrupt Status Register	ISR	Read-only	0x00000000
0x24	Memory Device Register	MDR	Read/Write	0x00000000

17.8.1 Mode Register

Register Name: MR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	MODE		

- **MODE: Command Mode**
  This field defines the command issued by the SDRAMC when the SDRAM device is accessed.

MODE	Description
0	Normal mode. Any access to the SDRAM is decoded normally.
1	The SDRAMC issues a “NOP” command when the SDRAM device is accessed regardless of the cycle.
2	The SDRAMC issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle.
3	The SDRAMC issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. This command will load the CR.CAS field into the SDRAM device Mode Register. All the other parameters of the SDRAM device Mode Register will be set to zero (burst length, burst type, operating mode, write burst mode...).
4	The SDRAMC issues an “Auto Refresh” command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued.
5	The SDRAMC issues an “Extended Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. This command will load the LPR.PASR, LPR.DS, and LPR.TCR fields into the SDRAM device Extended Mode Register. All the other bits of the SDRAM device Extended Mode Register will be set to zero.
6	Deep power-down mode. Enters deep power-down mode.

17.8.2 Refresh Timer Register

Register Name: TR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
-	-	-	-	COUNT[11:8]			

7	6	5	4	3	2	1	0
COUNT[7:0]							

- **COUNT[11:0]: 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 (CLK_SDRAMC), the refresh rate of the SDRAM device and the refresh burst length where 15.6µs per row is a typical value for a burst of length one.

  To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

17.8.3 Configuration Register

Register Name: CR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x852372C0

31	30	29	28	27	26	25	24
TXSR				TRAS			
23	22	21	20	19	18	17	16
TRCD				TRP			
15	14	13	12	11	10	9	8
TRC				TWR			
7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

- **TXSR: Exit Self Refresh to Active Delay**
  Reset value is eight cycles.
  This field defines the delay between SCKE set high and an Activate command in number of cycles. Number of cycles is between 0 and 15.
- **TRAS: Active to Precharge Delay**
  Reset value is five cycles.
  This field defines the delay between an Activate command and a Precharge command in number of cycles. Number of cycles is between 0 and 15.
- **TRCD: Row to Column Delay**
  Reset value is two cycles.
  This field defines the delay between an Activate command and a Read/Write command in number of cycles. Number of cycles is between 0 and 15.
- **TRP: Row Precharge Delay**
  Reset value is three cycles.
  This field defines the delay between a Precharge command and another command in number of cycles. Number of cycles is between 0 and 15.
- **TRC: Row Cycle Delay**
  Reset value is seven cycles.
  This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 0 and 15.
- **TWR: Write Recovery Delay**
  Reset value is two cycles.
  This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.
- **DBW: Data Bus Width**
  Reset value is 16 bits.
  0: Reserved.
  1: Data bus width is 16 bits.

- **CAS: CAS Latency**
  Reset value is two cycles.
  In the SDRAMC, only a CAS latency of one, two and three cycles is managed.

CAS	CAS Latency (Cycles)
0	Reserved
1	1
2	2
3	3

- **NB: Number of Banks**
  Reset value is two banks.

NB	Number of Banks
0	2
1	4

- **NR: Number of Row Bits**
  Reset value is 11 row bits.

NR	Row Bits
0	11
1	12
2	13
3	Reserved

- **NC: Number of Column Bits**
  Reset value is 8 column bits.

NC	Column Bits
0	8
1	9
2	10
3	11

17.8.4 High Speed Register

Register Name: HSR

Access Type: Read/Write

Offset: 0x0C

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
-	-	-	-	-	-	-	DA

- **DA: Decode Cycle Enable**

  A decode cycle can be added on the addresses as soon as a non-sequential access is performed on the HSB bus.

  The addition of the decode cycle allows the SDRAMC to gain time to access the SDRAM memory.

  1: Decode cycle is enabled.

  0: Decode cycle is disabled.

17.8.5 Low Power Register

Register Name: LPR

Access Type: Read/Write

Offset: 0x10

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
-	-	TIMEOUT		DS		TCSR	

7	6	5	4	3	2	1	0
-	PASR			-	-	LPCB	

- **TIMEOUT: Time to Define when Low Power Mode Is Enabled**

TIMEOUT	Time to Define when Low Power Mode Is Enabled
0	The SDRAMC activates the SDRAM low power mode immediately after the end of the last transfer.
1	The SDRAMC activates the SDRAM low power mode 64 clock cycles after the end of the last transfer.
2	The SDRAMC activates the SDRAM low power mode 128 clock cycles after the end of the last transfer.
3	Reserved.

- **DS: Drive Strength (only for low power SDRAM)**
  This field is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.
  After initialization, as soon as this field is modified and self refresh mode is activated, the Extended Mode Register of the SDRAM device is accessed automatically and its DS parameter value is updated before entry in self refresh mode.
- **TCSR: Temperature Compensated Self Refresh (only for low power SDRAM)**
  This field is transmitted to the SDRAM during initialization to set the refresh interval during self refresh mode depending on the temperature of the low power SDRAM. This parameter must be set according to the SDRAM device specification.
  After initialization, as soon as this field is modified and self refresh mode is activated, the Extended Mode Register of the SDRAM device is accessed automatically and its TCSR parameter value is updated before entry in self refresh mode.
- **PASR: Partial Array Self Refresh (only for low power SDRAM)**
  This field is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self refresh mode. This parameter must be set according to the SDRAM device specification.
  After initialization, as soon as this field is modified and self refresh mode is activated, the Extended Mode Register of the SDRAM device is accessed automatically and its PASR parameter value is updated before entry in self refresh mode.

- **LPCB: Low Power Configuration Bits**

LPCB	Low Power Configuration
0	Low power feature is inhibited: no power-down, self refresh or deep power-down command is issued to the SDRAM device.
1	The SDRAMC issues a self refresh command to the SDRAM device, the SDCLK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the self refresh mode when accessed and enters it after the access.
2	The SDRAMC issues a power-down command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the power-down mode when accessed and enters it after the access.
3	The SDRAMC issues a deep power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

17.8.6 Interrupt Enable Register

Register Name: IER

Access Type: Write-only

Offset: 0x14

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
-	-	-	-	-	-	-	RES

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

17.8.7 Interrupt Disable Register

Register Name: IDR

Access Type: Write-only

Offset: 0x18

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
-	-	-	-	-	-	-	RES

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

17.8.8 Interrupt Mask Register

Register Name: IMR

Access Type: Read-only

Offset: 0x1C

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
-	-	-	-	-	-	-	RES

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

17.8.9 Interrupt Status Register

Register Name: ISR

Access Type: Read-only

Offset: 0x20

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
-	-	-	-	-	-	-	RES

- **RES: Refresh Error Status**
  This bit is set when a refresh error is detected.
  This bit is cleared when the register is read.

17.8.10 Memory Device Register

Register Name: MDR

Access Type: Read/Write

Offset: 0x24

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
-	-	-	-	-	-	MD	

- **MD: Memory Device Type**

MD	Device Type
0	SDRAM
1	Low power SDRAM
Other	Reserved

18. Error Corrected Code Controller (ECCHRS)

Rev. 1.0.0.0

18.1 Features

- **Hardware Error Corrected Code Generation with two methods :**
  - **Hamming code detection and correction by software (ECC-H)**
  - **Reed-Solomon code detection by hardware, correction by hardware or software (ECC-RS)**
- **Supports NAND Flash and SmartMedia™ devices with 8- or 16-bit data path for ECC-H, and with 8-bit data path for ECC-RS**
- **Supports NAND Flash and SmartMedia™ with page sizes of 528, 1056, 2112, and 4224 bytes (specified by software)**
- **ECC_H supports :**
  - **One bit correction per page of 512,1024,2048, or 4096 bytes**
  - **One bit correction per sector of 512 bytes of data for a page size of 512, 1024, 2048, or 4096 bytes**
  - **One bit correction per sector of 256 bytes of data for a page size of 512, 1024, 2048, or 4096 bytes**
- **ECC_RS supports :**
  - **4 errors correction per sector of 512 bytes of data for a page size of 512, 1024, 2048, and 4096 bytes with 8-bit data path**

18.2 Overview

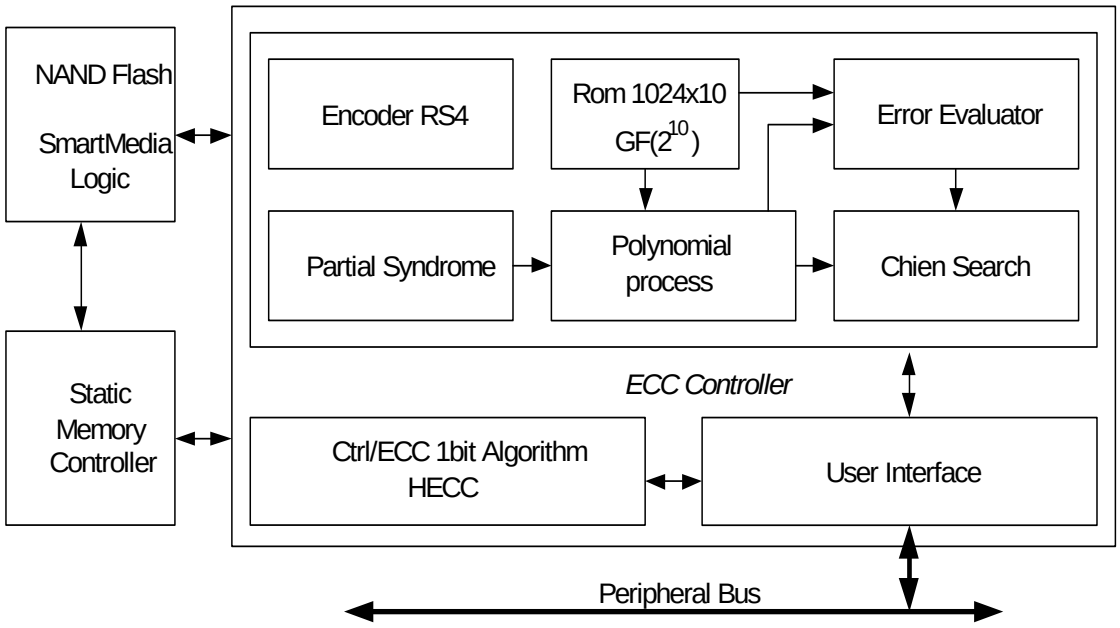
NAND Flash and SmartMedia™ devices contain by default invalid blocks which have one or more invalid bits. Over the NAND Flash and SmartMedia™ lifetime, additional invalid blocks may occur which can be detected and corrected by an Error Corrected Code (ECC).

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 two-bit random detection when using the Hamming code (ECC-H) and four-error correction whatever the number of erroneous bit in the byte error when using the Reed-Solomon code (ECC-RS).

When NAND Flash/SmartMedia™ have more than two erroneous bits when using the Hamming code (ECC-H) or more than four bits in error when using the Reed-Solomon code (ECC-RS), the data cannot be corrected.

18.3 Block Diagram

Figure 18-1. ECCHRS Block Diagram



18.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

18.4.1 I/O Lines

The ECCHRS signals pass through the External Bus Interface module (EBI) where they are multiplexed.

The programmer must first configure the I/O Controller to assign the EBI pins corresponding to the Static Memory Controller (SMC) signals to their peripheral function. If I/O lines of the EBI corresponding to SMC signals are not used by the application, they can be used for other purposes by the I/O Controller.

18.4.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the ECCHRS, the ECCHRS will stop functioning and resume operation after the system wakes up from sleep mode.

18.4.3 Clocks

The clock for the ECCHRS bus interface (CLK_ECCHRS) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the ECCHRS before disabling the clock, to avoid freezing the ECCHRS in an undefined state.

18.4.4 Interrupts

The ECCHRS interrupt request line is connected to the interrupt controller. Using the ECCHRS interrupt requires the interrupt controller to be programmed first.

18.5 Functional Description

A page in NAND Flash and SmartMedia™ memories contains an area for main data and an additional area used for redundancy (ECC). The page is organized in 8-bit or 16-bit words. The page size corresponds to the number of words in the main area plus the number of words in the extra area used for redundancy.

Over time, some memory locations may fail to program or erase properly. In order to ensure that data is stored properly over the life of the NAND Flash device, NAND Flash providers recommend to utilize either one ECC per 256 bytes of data, one ECC per 512 bytes of data, or one ECC for all of the page. For the next generation of deep micron SLC NAND Flash and with the new MLC NAND Flash, it is also recommended to ensure at least a four-error ECC per 512 bytes whatever is the page size.

The only configurations required for ECC are the NAND Flash or the SmartMedia™ page size (528/1056/2112/4224) and the type of correction wanted (one ECC-H for all the page, one ECC-H per 256 bytes of data, one ECC-H per 512 bytes of data, or four-error ECC-RS per 512 bytes of data). The page size is configured by writing in the Page Size field in the Mode Register (MD.PAGESIZE). Type of correction is configured by writing the Type of Correction field in the Mode Register (MD.TYPECORREC).

The ECC is automatically computed as soon as a read (0x00) or a write (0x80) command to the NAND Flash or the SmartMedia™ is detected. Read and write access must start at a page boundary.

The ECC results are available as soon as the counter reaches the end of the main area. The values in the Parity Registers (PR0 to PR15) for ECC-H and in the Codeword Parity registers (CWPS00 to CWPS79) for ECC-RS are then valid and locked until a new start condition occurs (read/write command followed by address cycles).

18.5.1 Write Access

Once the Flash memory page is written, the computed ECC codes are available in PR0 to PR15 registers for ECC-H and in CWPS00 to CWPS79 registers for ECC-RS. The ECC code values must be written by the software application in the extra area used for redundancy. The number of write access in the extra area depends on the value of the MD.TYPECORREC field.

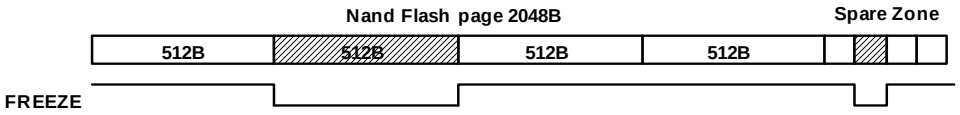
For example, for one ECC per 256 bytes of data for a page of 512 bytes, only the values of PR0 and PR1 must be written by the software application in the extra area. For ECC-RS, a NAND Flash with page of 512 bytes, the software application will have to write the ten registers CWPS00 to CWPS09 in the extra area, and would have to write 40 registers (CWPS00 to CWPS39) for a NAND Flash with page of 2048 bytes.

Other registers are meaningless.

18.5.2 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-H controller or the ECC-RS controller. In ECC-RS, writing a one to the Halt of Computation bit in the ECC Mode Register (MD.FREEZE) allows to stop error detection when software is jumping to the correct parity area.

Figure 18-2. FREEZE signal waveform



The application can check the ECC Status Registers (SR1/SR2) for any detected errors. It is up to the application to correct any detected error for ECC-H. The application can correct any detected error or let the hardware do the correction by writing a one to the Correction Enable bit in the MD register (MD.CORRS4) for ECC-RS.

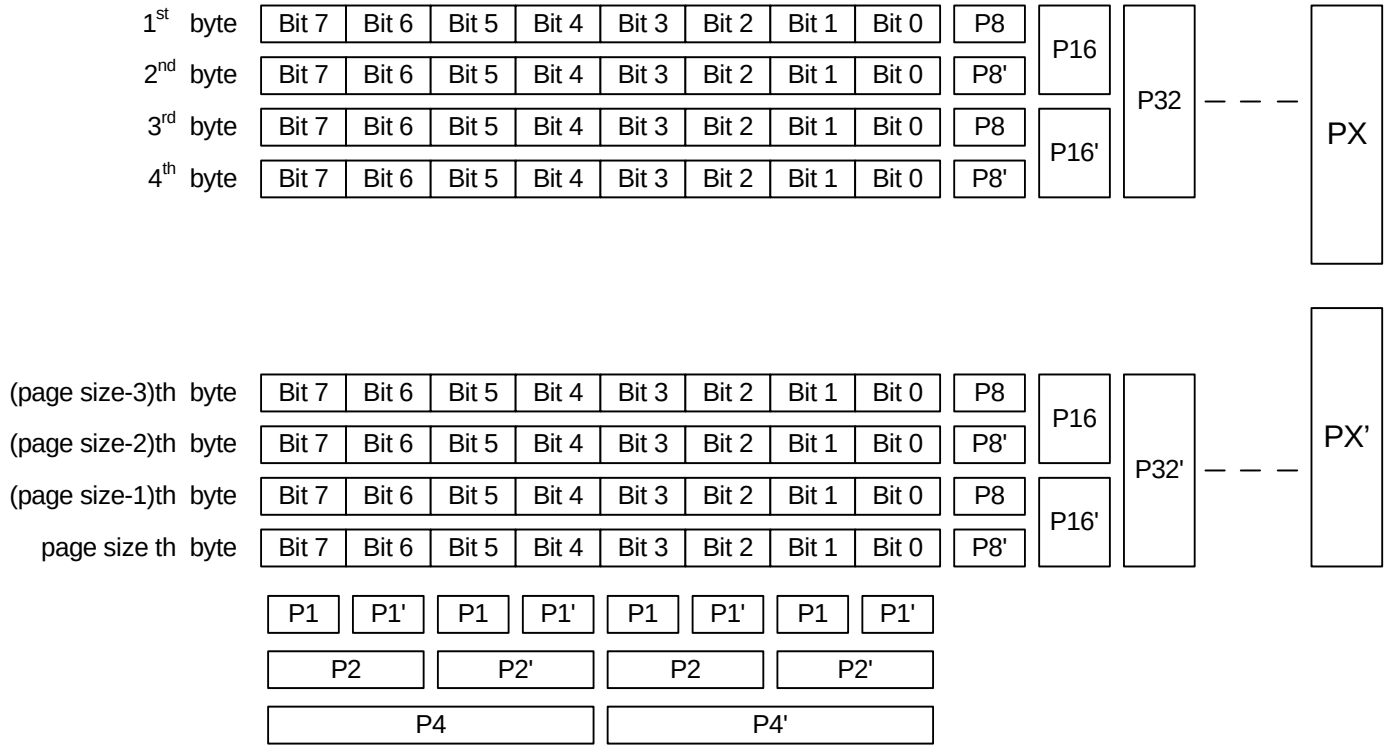
ECC computation can detect four different circumstances:

- 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 zero. All bits in the SR1 and SR2 registers will be cleared.
- Recoverable error: Only the Recoverable Error bits in the ECC Status registers (SR1.RECERRn and/or SR2.RECERRn) are set. The corrupted word offset in the read page is defined by the Word Address field (WORDADDR) in the PR0 to PR15 registers. The corrupted bit position in the concerned word is defined in the Bit Address field (BITADDR) in the PR0 to PR15 registers.
- ECC error: The ECC Error bits in the ECC Status Registers (SR1.ECCERRn / SR2.ECCERRn) are set. An error has been detected in the ECC code stored in the Flash memory. The position of the corrupted bit can be found by the application performing an XOR between the Parity and the NParity contained in the ECC code stored in the Flash memory. For ECC-RS it is the responsibility of the software to determine where the error is located on ECC code stored in the spare zone flash area and not on user data area.
- Non correctable error: The Multiple Error bits (MULERRn) in the SR1 and SR2 registers are set. Several unrecoverable errors have been detected in the Flash memory page.

ECC Status Registers, ECC Parity Registers are cleared when a read/write command is detected or a software reset is performed.

For Single-bit Error Correction and Double-bit Error Detection (SEC-DED) Hsiao code is used. 24-bit ECC is generated in order to perform one bit correction per 256 or 512 bytes for pages of 512/2048/4096 8-bit words. 32-bit ECC is generated in order to perform one bit correction per 512/1024/2048/4096 8- or 16-bit words.They are generated according to the schemes shown in Figure 18-3 on page 240 and Figure 18-4 on page 241.

Figure 18-3. Parity Generation for 512/1024/2048/4096 8-bit Words



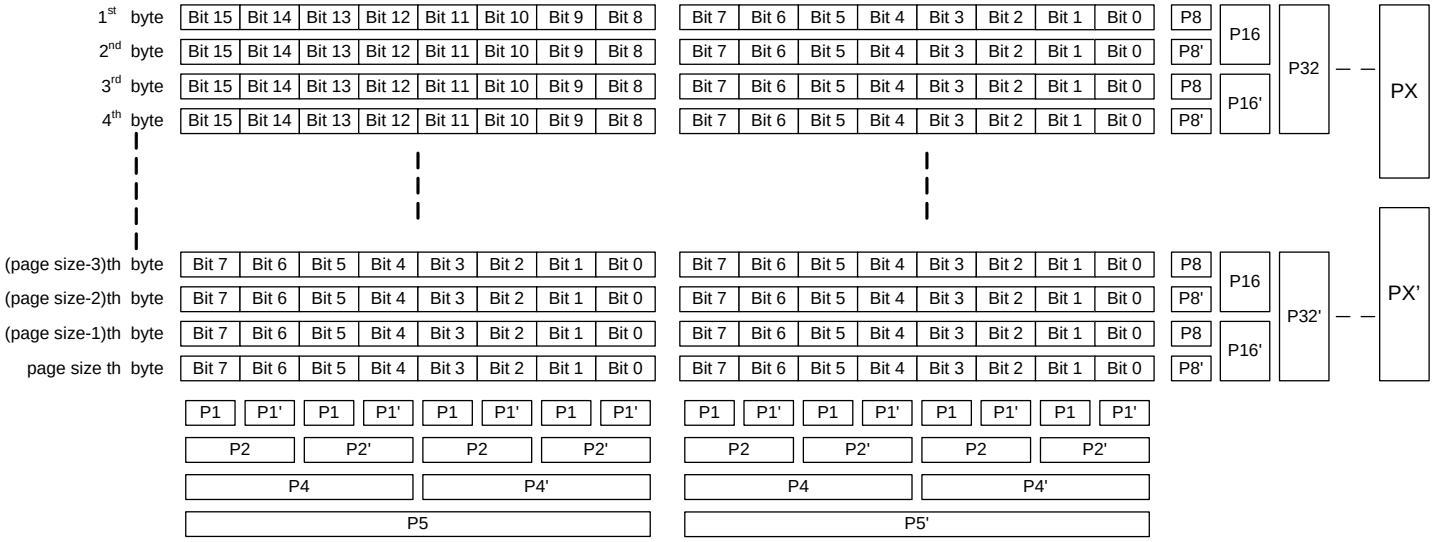
Page size = 512	Px = 2048	P1=bit7(+)bit5(+)bit3(+)bit1(+)P1
Page size = 1024	Px = 4096	P2=bit7(+)bit6(+)bit3(+)bit2(+)P2
Page size = 2048	Px = 8192	P4=bit7(+)bit6(+)bit5(+)bit4(+)P4
Page size = 4096	Px = 16384	P1'=bit6(+)bit4(+)bit2(+)bit0(+)P1'
		P2'=bit5(+)bit4(+)bit1(+)bit0(+)P2'
		P4'=bit7(+)bit6(+)bit5(+)bit4(+)P4'

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 18-4. Parity Generation for 512/1024/2048/4096 16-bit Words



Page size = 512	Px = 2048	P1=bit15(+)bit13(+)bit11(+)bit9(+)bit7(+)bit5(+)bit3(+)bit1(+)P1
Page size = 1024	Px = 4096	P2=bit15(+)bit14(+)bit11(+)bit10(+)bit7(+)bit6(+)bit3(+)bit2(+)P2
Page size = 2048	Px = 8192	P4=bit15(+)bit14(+)bit13(+)bit12(+)bit7(+)bit6(+)bit5(+)bit4(+)P4
Page size = 4096	Px = 16384	P5=bit15(+)bit14(+)bit13(+)bit12(+)bit11(+)bit10(+)bit9(+)bit8(+)P5

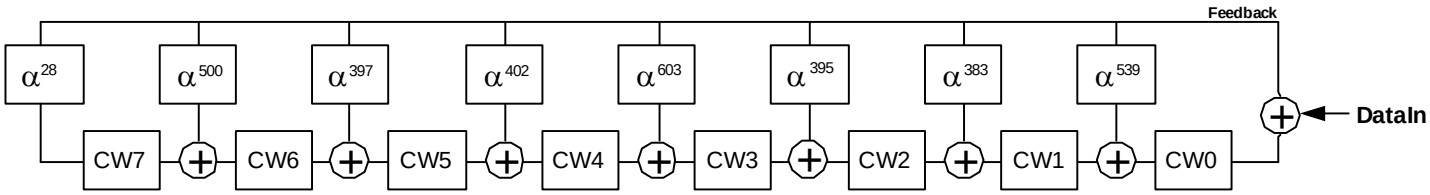
To calculate P8' to PX' and P8 to PX, apply the algorithm that follows.

```
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
```

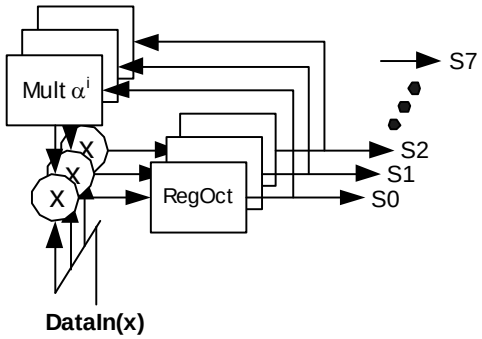
For ECC-RS, in order to perform 4-error correction per 512 bytes of 8-bit words, the codeword have to be generated by the RS4 Encoder module and stored into the NAND Flash extra area, according to the scheme shown in Figure 18-5 on page 242

Figure 18-5. RS Codeword Generation



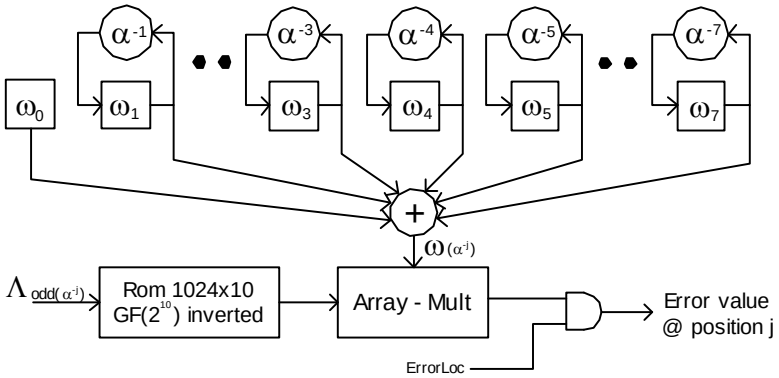
In read mode, firstly, the detection for any error is done with the partial syndrome module. It is the responsibility of the ECC-RS Controller to determine after receiving the old codeword stored in the extra area if there is any error on data and /or on the old codeword. If all syndromes (Si) are equal to zero, there is no error, otherwise a polynomial representation is written into CWPS00 to CWPS79 registers. The Partial Syndrome module performs an algorithm according to the scheme in Figure 18-6 on page 242

Figure 18-6. Partial Syndrome Block Diagram



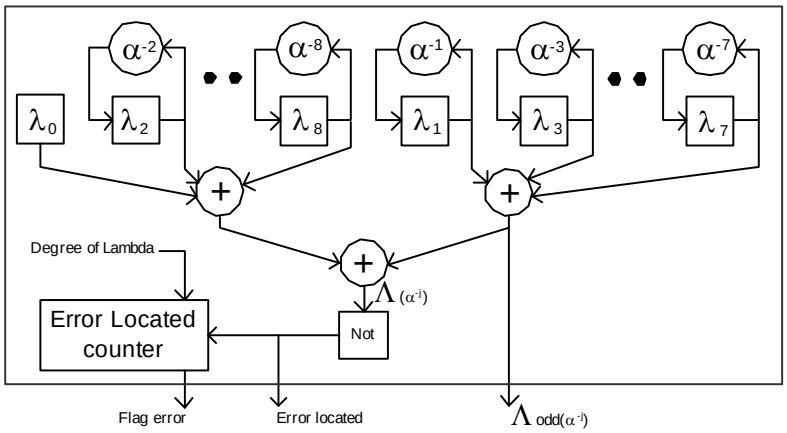
If the Correction Enable bit is set in the ECC Mode Register (MD.CORRS4) then the polynomial representation of error are sent to the polynomial processor. The aim of this module is to perform the polynomial division in order to calculate two polynomials, Omega (Z) and Lambda (Z), which are necessary for the two following modules (Chien Search and Error Evaluator). In order to perform addition, multiplication, and division a Read Only Memory (ROM) has been added containing the 1024 elements of the Galois field. Both Chien Search and Error Evaluator work in parallel. The Error Evaluator has the responsibility to determine the Nth error value in the data and in the old codeword according to the scheme in Figure 18-7 on page 243

Figure 18-7. Error Evaluator Block Diagram



The Chien Search takes charge of determining if an error has occurred at symbol N according to the scheme in Figure 18-8 on page 243

Figure 18-8. Chien Search Block Diagram



18.6 User Interface

Table 18-1. ECCHRS Register Memory Map

Offset	Register	Name	Access	Reset
0x000	Control Register	CTRL	Write-only	0x00000000
0x004	Mode Register	MD	Read/write	0x00000000
0x008	Status Register 1	SR1	Read-only	0x00000000
0x00C	Parity Register 0	PR0	Read-only	0x00000000
0x010	Parity Register 1	PR1	Read-only	0x00000000
0x014	Status Register 2	SR2	Read-only	0x00000000
0x018	Parity Register 2	PR2	Read-only	0x00000000
0x01C	Parity Register 3	PR3	Read-only	0x00000000
0x020	Parity Register 4	PR4	Read-only	0x00000000
0x024	Parity Register 5	PR5	Read-only	0x00000000
0x028	Parity Register 6	PR6	Read-only	0x00000000
0x02C	Parity Register 7	PR7	Read-only	0x00000000
0x030	Parity Register 8	PR8	Read-only	0x00000000
0x034	Parity Register 9	PR9	Read-only	0x00000000
0x038	Parity Register 10	PR10	Read-only	0x00000000
0x03C	Parity Register 11	PR11	Read-only	0x00000000
0x040	Parity Register 12	PR12	Read-only	0x00000000
0x044	Parity Register 13	PR13	Read-only	0x00000000
0x048	Parity Register 14	PR14	Read-only	0x00000000
0x04C	Parity Register 15	PR15	Read-only	0x00000000
0x050 - 0x18C	Codeword and Syndrome 00 - Codeword and Syndrome 79	CWPS00 - CWPS79	Read-only	0x00000000
0x190 - 0x19C	MaskData 0 - Mask Data 3	MDATA0 - MDATA3	Read-only	0x00000000
0x1A0 - 0x1AC	Address Offset 0 - Address Offset 3	ADOFF0 - ADOFF3	Read-only	0x00000000
0x1B0	Interrupt Enable Register	IER	Write-only	0x00000000
0x1B4	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1B8	Interrupt Mask Register	MR	Read-only	0x00000000
0x1BC	Interrupt Status Register	ISR	Read-only	0x00000000
0x1C0	Interrupt Status Clear Register	ISCR	Write-only	0x00000000
0x1FC	Version Register	VERSION	Read-only	-(1)

Note: 1. The reset value is device specific. Please refer to the Module Configuration section at the end of this chapter.

18.6.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x000

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
-	-	-	-	-	-	-	RST

- **RST: RESET Parity**

  Writing a one to this bit will reset the ECC Parity registers.

  Writing a zero to this bit has no effect.

  This bit always reads as zero.

18.6.2 Mode Register

Name: MD

Access Type: Read/Write

Offset: 0x004

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
-	-	-	-	-	CORRS4	-	FREEZE

7	6	5	4	3	2	1	0
-	TYPECORREC			-	PAGESIZE		

- **CORRS4: Correction Enable**
  Writing a one to this bit will enable the correction to be done after the Partial Syndrome process and allow interrupt to be sent to CPU.
  Writing a zero to this bit will stop the correction after the Partial Syndrome process.
  1: The correction will continue after the Partial Syndrome process.
  0: The correction will stop after the Partial Syndrome process.
- **FREEZE: Halt of Computation**
  Writing a one to this bit will stop the computation.
  Writing a zero to this bit will allow the computation as soon as read/write command to the NAND Flash or the SmartMedia™ is detected.
  1: The computation will stop until a zero is written to this bit.
  0: The computation is allowed.
- **TYPECORREC: Type of Correction**

ECC code	TYPECORREC	Description
ECC-H	0b000	One bit correction per page
	0b001	One bit correction per sector of 256 bytes
	0b010	One bit correction per sector of 512 bytes
ECC-RS	0b100	Four bits correction per sector of 512 bytes
-	Others	Reserved

- **PAGESIZE: Page Size**

This table defines the page size of the NAND Flash device when using the ECC-H code (TYPECORREC = 0b0xx).

Page Size	Description
0	528 words
1	1056 words
2	2112 words
3	4224 words
Others	Reserved

A word has a value of 8 bits or 16 bits, depending on the NAND Flash or SmartMedia™ memory organization.

This table defines the page size of the NAND Flash device when using the ECC-RS code (TYPECORREC = 0b1xx)

Page Size	Description	Comment
0	528 bytes	1 page of 512 bytes
1	1056 bytes	2 pages of 512 bytes
2	1584 bytes	3 pages of 512 bytes
3	2112 bytes	4 pages of 512 bytes
4	2640 bytes	5 pages of 512 bytes
5	3168 bytes	6 pages of 512 bytes
6	3696 bytes	7 pages of 512 bytes
7	4224 bytes	8 pages of 512 bytes

i.e.: for NAND Flash device with page size of 4096 bytes and 128 bytes extra area ECC-RS can manage any sub page of 512 bytes up to 8.

18.6.3 Status Register 1

Name: SR1

Access Type: Read-only

Offset: 0x008

Reset Value: 0x000000000

MD.TYPECORREC=0b0xx, using ECC-H code

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

- **MULERRn: Multiple Error in the sector number n of 256/512 bytes in the page**
  1: Multiple errors are detected.
  0: No multiple error is detected.

TYPECORREC	Sector Size	Comments
0	page size	Only MULERR0 is used
1	256	MULERR0 to MULERR7 are used depending on the page size
2	512	MULERR0 to MULERR7 are used depending on the page size
Others	Reserved	

- **ECCERRn: ECC Error in the packet number n of 256/512 bytes in the page**
  1: A single bit error has occurred.
  0: No error have been detected.

TYPECORREC	Sector Size	Comments
0	page size	Only ECCERR0 is used The user should read PR0 and PR1 to know where the error occurs in the page.
1	256	ECCERR0 to ECCERR7 are used depending on the page size
2	512	ECCERR0 to ECCERR7 are used depending on the page size
Others	Reserved	

- **RECERRn: Recoverable Error in the packet number n of 256/512 Bytes in the page**
  1: Errors detected. If MULERRn is zero, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.
  0: No errors have been detected.

TYPECORREC	sector size	Comments
0	page size	Only RECERR0 is used
1	256	RECERR0 to RECERR7 are used depending on the page size
2	512	RECERR0 to RECERR7 are used depending on the page size
Others	Reserved	

MD.TYPECORREC=0b1xx, using ECC-RS code

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
SYNVEC							

- **SYNVEC: Syndrome Vector**
  After reading a page made of n sector of 512 bytes, this field returns which sector contains error detected after the syndrome analysis.
  The SYNVEC[n] bit is set when there is at least one error in the corresponding sector.
  The SYNVEC[n] bit is cleared when a read/write command is detected or a software reset is performed.
  1: At least one error has occurred in the corresponding sector.
  0: No error has been detected.

Bit Index (n)	Sector Boundaries
0	0-511
1	512-1023
2	1023-1535
3	1536-2047
4	2048-2559

Bit Index (n)	Sector Boundaries
5	2560-3071
6	3072-3583
7	3584-4095

18.6.4 Parity Register 0

Name: PR0

Access Type: Read-only

Offset: 0x00C

Reset Value: 0x00000000

Using ECC-H code, one bit correction per page (MD.TYPECORREC=0b000)

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[11:4]							

7	6	5	4	3	2	1	0
WORDADDR[3:0]				BITADDR			

Once the entire main area of a page is written with data, this register content must be stored at any free location of the spare area.

- **WORDADDR: Word Address**
  During a page read, this field 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 field is meaningless.
- **BITADDR: Bit Address**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

Using ECC-H code, one bit correction per sector of 256 bytes (MD.TYPECORREC=0b001)

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

23	22	21	20	19	18	17	16
-	NPARITY0[10:4]						

15	14	13	12	11	10	9	8

NPARITY0[3:0]	0	WORDADD0[7:5]

7	6	5	4	3	2	1	0
WORDADD0[4:0]					BITADDR0		

Once the entire main area of a page is written with data, this register content must be stored at any free location of the spare area.

- **NPARITY0: Parity N**
  Parity calculated by the ECC-H.
- **WORDADDR0: Corrupted Word Address in the page between the first byte and the 255th byte**
  During a page read, this field contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.
- **BITADDR0: Corrupted Bit Address in the page between the first byte and the 255th byte**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

Using ECC-H code, one bit correction per sector of 512 bytes (MD.TYPECORREC=0b010)

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

23	22	21	20	19	18	17	16
NPARITY0[11:4]							

15	14	13	12	11	10	9	8
NPARITY0[3:0]				WORDADD0[8:5]			

7	6	5	4	3	2	1	0
WORDADD0[4:0]					BITADDR0		

Once the entire main area of a page is written with data, this register content must be stored at any free location of the spare area.

- **NPARITY0: Parity N**
  Parity calculated by the ECC-H.
- **WORDADDR0: Corrupted Word Address in the page between the first byte and the 511th byte**
  During a page read, this field contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.
- **BITADDR0: Corrupted Bit Address in the page between the first byte and the 511th byte**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

18.6.5 Parity Register 1

Name: PR1

Access Type: Read-only

Offset: 0x010

Reset Value: 0x00000000

Using ECC-H code, one bit correction per page (MD.TYPECORREC=0b000)

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[15:8]							

7	6	5	4	3	2	1	0
NPARITY[7:0]							

- **NPARITY: Parity N**
  During a write, the field of this register must be written in the extra area used for redundancy (for a 512-byte page size: address 514-515).

Using ECC-H code, one bit correction per sector of 256 bytes (MD.TYPECORREC=0b001)

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

23	22	21	20	19	18	17	16
-	NPARITY1[10:0]						

15	14	13	12	11	10	9	8
NPARITY1[3:0]				0	WORDADD1[7:5]		

7	6	5	4	3	2	1	0
WORDADD1[4:0]					BITADDR1		

Once the entire main area of a page is written with data, this register content must be stored at any free location of the spare area.

- **NPARITY1: Parity N**
  Parity alculated by the ECC-H.
- **WORDADDR1: corrupted Word Address in the page between the 256th and the 511th byte**
  During a page read, this field contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.
- **BITADDR1: corrupted Bit Address in the page between the 256th and the 511th byte**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

Using ECC-H code, one bit correction per sector of 512 bytes (MD.TYPECORREC=0b010)

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

23	22	21	20	19	18	17	16
NPARITY1[11:4]							

15	14	13	12	11	10	9	8
NPARITY1[3:0]				WORDADD1[8:5]			

7	6	5	4	3	2	1	0
WORDADD1[4:0]					BITADDR1		

Once the entire main area of a page is written with data, this register content must be stored at any free location of the spare area.

- **NPARITY1: Parity N**
  Parity calculated by the ECC-H.
- **WORDADDR1: Corrupted Word Address in the page between the 512th and the 1023th byte**
  During a page read, this field contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.
- **BITADDR1: Corrupted Bit Address in the page between the 512th and the 1023th byte**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

18.6.6 Status Register 2

Name: SR2

Access Type: Read-only

Offset: 0x014

Reset Value: 0x00000000

MD.TYPECORREC=0b0xx, using ECC-H code

31	30	29	28	27	26	25	24
-	MULERR15	ECCERR15	RECERR15	-	MULERR14	ECCERR14	RECERR14
23	22	21	20	19	18	17	16
-	MULERR13	ECCERR13	RECERR13	-	MULERR12	ECCERR12	RECERR12
15	14	13	12	11	10	9	8
-	MULERR11	ECCERR11	RECERR11	-	MULERR10	ECCERR10	RECERR10
7	6	5	4	3	2	1	0
-	MULERR9	ECCERR9	RECERR9	-	MULERR8	ECCERR8	RECERR8

- **MULERRn: Multiple Error in the sector number n of 256/512 bytes in the page**
  1: Multiple errors are detected.
  0: No multiple error is detected.

TYPECORREC	Sector Size	Comments
0	page size	Only MULERR0 is used
1	256	MULERR0 to MULERR7 are used depending on the page size
2	512	MULERR0 to MULERR7 are used depending on the page size
Others	Reserved	

- **ECCERRn: ECC Error in the packet number n of 256/512 bytes in the page**
  1: A single bit error has occurred.
  0: No error is detected.

TYPECORREC	sector size	Comments
0	page size	Only ECCERR0 is used The user should read PR0 and PR1 to know where the error occurs in the page.
1	256	ECCERR0 to ECCERR7 are used depending on the page size
2	512	ECCERR0 to ECCERR7 are used depending on the page size
Others	Reserved	

MD.TYPECORREC=0b1xx, using ECC-RS code

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

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	RECERR		

Only one sub page of 512 bytes is corrected at a time. If several sub page are on error then it is necessary to do several time the correction process.

- **MULERR: Multiple error**

  This bit is set to one when a multiple error have been detected by the ECC-RS.

  This bit is cleared when a read/write command is detected or a software reset is performed.

  1: Multiple errors detected: more than four errors.Registers for one ECC for a page of 512/1024/2048/4096 bytes

  0: No multiple error detected

- **RECERR: Number of recoverable errors if MULERR is zero**

RECERR	Comments
000	no error
001	one single error detected
010	two errors detected
011	three errors detected
100	four errors detected

18.6.7 Parity Register 2 - 15

Name: PR2 - PR15

Access Type: Read-only

Offset: 0x018 - 0x04C

Reset Value: 0x00000000

Using ECC-H code, one bit correction per sector of 256 bytes (MD.TYPECORREC=0b001)

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

23	22	21	20	19	18	17	16
-	NPARITYn[10:4]						

15	14	13	12	11	10	9	8
NPARITYn[3:0]				0	WORDADDn[7:5]		

7	6	5	4	3	2	1	0
WORDADDn[4:0]					BITADDRn		

Once the entire main area of a page is written with data, this register content must be stored at any free location of the spare area.

- **NPARITYn: Parity N**
  Parity calculated by the ECC-H.
- **WORDADDRn: corrupted Word Address in the packet number n of 256 bytes in the page**
  During a page read, this field contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.
- **BITADDRn: corrupted Bit Address in the packet number n of 256 bytes in the page**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

Using ECC-H code, one bit correction per sector of 512 bytes (MD.TYPECORREC=0b010)

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

23	22	21	20	19	18	17	16
NPARITYn[11:4]							

15	14	13	12	11	10	9	8
NPARITYn[3:0]				WORDADDn[8:5]			

7	6	5	4	3	2	1	0
WORDADDn[4:0]					BITADDRn		

Once the entire main area of a page is written with data, this register content must be stored to any free location of the spare area.
Only PR2 to PR7 registers are available in this case.

- **NPARITYn: Parity N**
  Parity calculated by the ECC-H.
- **WORDADDRn: corrupted Word Address in the packet number n of 512 bytes in the page**
  During a page read, this field contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.
- **BITADDRn: corrupted Bit Address in the packet number n of 512 bytes in the page**
  During a page read, this field contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this field is meaningless.

18.6.8 Codeword 00 - Codeword79

Name: CWPS00 - CWPS79

Access Type: Read-only

Offset: 0x050 - 0x18C

Reset Value: 0x00000000

Page Write:

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
CODEWORD							

- **CODEWORD:**
  Once the 512 bytes of a page is written with data, this register content must be stored to any free location of the spare area. For a page of 512 bytes the entire redundancy words are made of 8 words of 10 bits. All those redundancies words are concatenated to a word of 80 bits and then cut to 10 words of 8 bits to facilitate their writing in the extra area.
  At the end of a page write, this field contains the redundancy word to be stored to the extra area.

Page Read:

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
PARSYND							

- **PARSYND:**

  At the end of a page read, this field contains the Partial Syndrome S.

  PARSYND00-PARSYND09: this conclude all the codeword and partial syndrome word for the sub page 1

  PARSYND10-PARSYND19: this conclude all the codeword and partial syndrome word for the sub page 2

  PARSYND20-PARSYND29: this conclude all the codeword and partial syndrome word for the sub page 3

  PARSYND30-PARSYND39: this conclude all the codeword and partial syndrome word for the sub page 4

  PARSYND40-PARSYND49: this conclude all the codeword and partial syndrome word for the sub page 5

  PARSYND50-PARSYND59: this conclude all the codeword and partial syndrome word for the sub page 6

  PARSYND60-PARSYND69: this conclude all the codeword and partial syndrome word for the sub page 7

  PARSYND70-PARSYND79: this conclude all the codeword and partial syndrome word for the sub page 8

18.6.9 Mask Data 0 - Mask Data 3

Name: MDATA0 -MDATA3

Access Type: Read-only

Offset: 0x190 - 0x19C

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
-	-	-	-	-	-	MASKDATA[9:8]	

7	6	5	4	3	2	1	0
MASKDATA[7:0]							

- **MASKDATA:**
  At the end of the correction process, this field contains the mask to be XORed with the data read to perform the final correction.This XORed is under the responsibility of the software.
  This field is meaningless if MD.CORRS4 is zero.

18.6.10 Address Offset 0 - Address Offset 3

Name: ADOFF0 - ADOFF3

Access Type: Read-only

Offset: 0x1A0 - 0x1AC

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
-	-	-	-	-	-	OFFSET[9:8]	

7	6	5	4	3	2	1	0
OFFSET[7:0]							

- **OFFSET:**
  At the end of correction process, this field contains the offset address of the data read to be corrected.
  This field is meaningless if MD.CORRS4 is zero.

18.6.11 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x1B0

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
-	-	-	-	-	-	-	ENDCOR

- **ENDCOR:**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in IMR.

18.6.12 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x1B4

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
-	-	-	-	-	-	-	ENDCOR

- **ENDCOR:**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.

18.6.13 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x1B8

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
-	-	-	-	-	-	-	ENDCOR

- **ENDCOR:**
  0: The corresponding interrupt is disabled.
  1: The corresponding interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.

18.6.14 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x1BC

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
-	-	-	-	-	-	-	ENDCOR

- **ENDCOR:**
  This bit is cleared when the corresponding bit in ISCR is written to one.
  This bit is set when a correction process has ended.

18.6.15 Interrupt Status Clear Register

Name: ISCR

Access Type: Write-only

Offset: 0x1C0

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
-	-	-	-	-	-	-	ENDCOR

- **ENDCOR:**
  Writing a zero to this bit has no effect
  Writing a one to this bit will clear the corresponding bit in ISR and the corresponding interrupt request.

18.6.16 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x1FC

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	VARIANT		

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

18.7 Module Configuration

The specific configuration for the ECCHRS instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 18-2. Module clock name

Module name	Clock name
ECCHRS	CLK_ECCHRS

Table 18-3. Register Reset Values

Register	Reset Value
VERSION	0x00000100

19. Peripheral DMA Controller (PDCA)

Rev: 1.1.0.0

19.1 Features

- **8 channels**
- **Generates transfers to/from peripherals such as USART, SSC and SPI**
- **Two address pointers/counters per channel allowing double buffering**
- **Performance monitors to measure average and maximum transfer latency**

19.2 Overview

The Peripheral DMA controller (PDCA) transfers data between on-chip peripheral modules such as USART, SPI, SSC and memories (those memories may be on- and off-chip memories). Using the PDCA avoids CPU intervention for data transfers, improving the performance of the microcontroller. The PDCA can transfer data from memory to a peripheral or from a peripheral to memory.

The PDCA consists of 8 DMA channels. Each channel has:

- A 32-bit memory pointer
- A 16-bit transfer counter
- A 32-bit memory pointer reload value
- A 16-bit transfer counter reload value

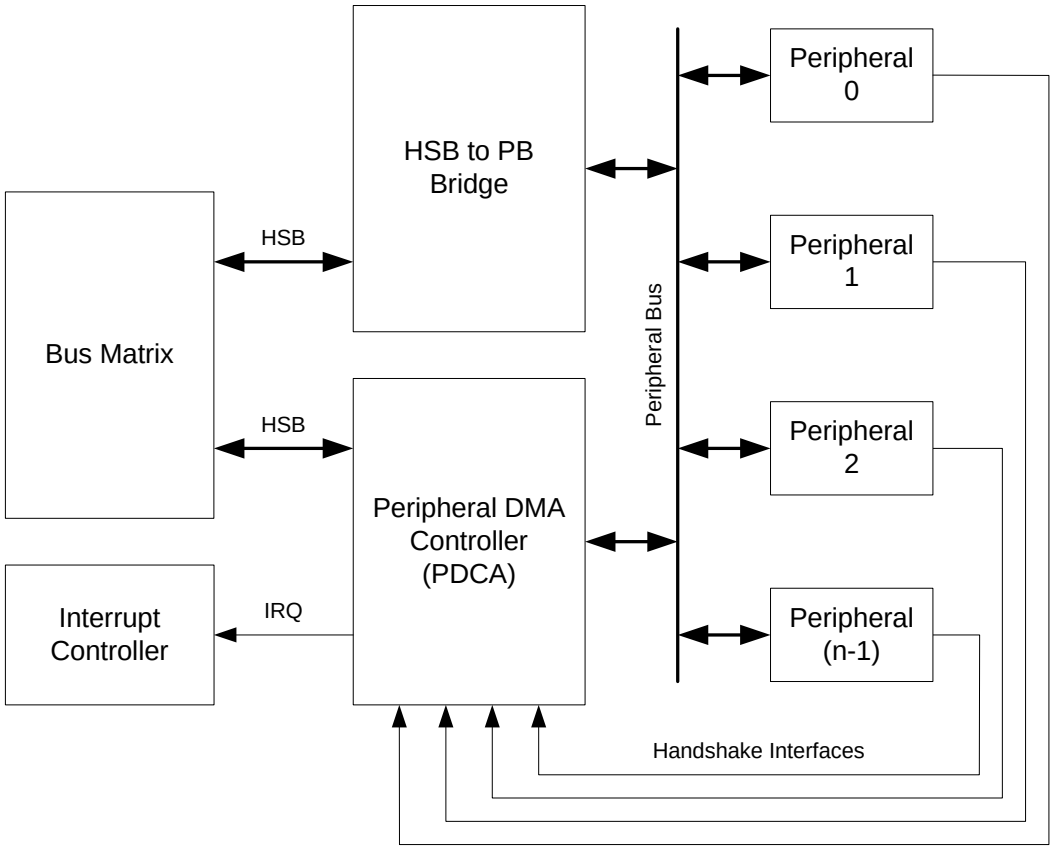
The PDCA communicates with the peripheral modules over a number of handshake interfaces. The peripheral signals to the PDCA when it is ready to receive or transmit data. The PDCA acknowledges the request when the transmission has started.

The number of handshake-interfaces may be higher than the number of DMA channels. If this is the case, the DMA channel must be programmed to use the desired interface.

When a transmit buffer is empty or a receive buffer is full, an interrupt request can be signalled.

19.3 Block Diagram

Figure 19-1. PDCA Block Diagram



19.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

19.4.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the PDCA, the PDCA will stop functioning and resume operation after the system wakes up from sleep mode. To avoid unexpected results after waking up from sleep mode, it requires to be checked that all the data transfers through PDCA are completed before entering a sleep mode.

19.4.2 Clocks

The PDCA has two bus clocks connected: One High Speed Bus clock (CLK_PDCA_HSB) and one Peripheral Bus clock (CLK_PDCA_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by writing to the Power Manager. It is recommended to disable the PDCA before disabling the clocks, to avoid freezing the PDCA in an undefined state.

19.4.3 Interrupts

The PDCA interrupt request line is connected to the interrupt controller. Using the PDCA interrupt requires the interrupt controller to be programmed first.

19.4.4 Peripheral Events

The PDCA peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

19.5 Functional Description

19.5.1 Channel Configuration

Each channel in the PDCA has a set of configuration registers. Among these are the Memory Address Register (MAR), the Peripheral Select Register (PSR) and the Transfer Counter Register (TCR). The 32-bit MAR must be programmed with the start address of the memory buffer. The register will be automatically updated after each transfer to point to the next location in memory. The PSR must be programmed to select the desired peripheral/handshake interface. The TCR determines the number of data items to be transferred. The counter will be decreased by one for each data item that has been transferred.

Both the MAR and the TCR registers can be read at any time to check the progress of the transfer.

Each channel has reload registers for the MAR and the TCR: the Memory Address Reload Register (MARR) and the Transfer Counter Reload Register (TCRR). When the TCR reaches zero, the values in the reload registers are loaded into MAR and TCR. In this way, the PDCA can operate on two buffers for each channel.

Since a new transfer starts immediately when TCR gets a non-zero value, TCR and TCRR should be the last registers to be written.

19.5.2 Memory Pointer

Each channel has a 32-bit Memory Address Register. This register holds the memory address for the next transfer to be performed. The register is automatically updated after each transfer. The address will be increased by either one, two or four depending on the size of the DMA transfer (byte, halfword or word). The MAR can be read at any time during transfer.

19.5.3 Transfer Counter

Each channel has a 16-bit Transfer Counter Register. This register must be programmed with the number of transfers to be performed. The TCR register should contain the number of data items to be transferred independently of the transfer size. The TCR can be read at any time during transfer to see the number of remaining transfers.

19.5.4 Reload Registers

Both the MAR and the TCR have a reload register, respectively Memory Address Reload Register and Transfer Counter Reload Register. These registers provide the possibility for the PDCA to work on two memory buffers for each channel. When one buffer has completed, MAR and TCR will be reloaded with the values in MARR and TCRR. The reload logic is always enabled and will trigger if the TCR reaches zero while TCRR holds a non-zero value.

If TCR is zero when writing to TCRR and MARR, the TCR and MAR are automatically updated with the value written in TCRR and MARR.

19.5.5 Peripheral Selection

The Peripheral Select Register decides which peripheral should be connected to the PDCA channel. Configuring PSR will both select the direction of the transfer (memory to peripheral or

peripheral to memory), which handshake interface to use, and the address of the peripheral holding register.

19.5.6 Transfer Size

The transfer size can be set individually for each channel to be either byte, halfword or word (8-bit, 16-bit or 32-bit respectively). Transfer size is set by writing the desired value to the Transfer Size field in the Mode Register (MR.SIZE).

19.5.7 Enabling and Disabling

Each DMA channel is enabled by writing a one to the Transfer Enable bit in the Control Register (CR.TEN) and disabled by writing a one to the Transfer Disable bit (CR.TDIS). The current status can be read from the Status Register (SR).

19.5.8 Interrupts

Interrupts can be enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER) and disabled by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The Interrupt Mask Register (IMR) can be read to see whether an interrupt is enabled or not. The current status of an interrupt source can be read through the Interrupt Status Register (ISR).

The PDCA has three interrupt sources:

- Reload Counter Zero - The TCRR register is zero.
- Transfer Finished - Both the TCR and TCRR registers are zero.
- Transfer Error - An error has occurred in accessing memory.

19.5.9 Priority

If more than one PDCA channel is requesting transfer at a given time, the PDCA channels are prioritized by their channel number. Channels with lower numbers have priority over channels with higher numbers, giving channel zero the highest priority.

19.5.10 Error Handling

If the memory address is set to point to an invalid location in memory, an error will occur when the PDCA tries to perform a transfer. When an error occurs, the Transfer Error bit in the Interrupt Status Register (ISR.TERR) will be set and the DMA channel that caused the error will be stopped. In order to restart the channel, the user must program the Memory Address Register to a valid address and then write a one to the Error Clear bit in the Control Register (CR.ECLR). An interrupt can optionally be triggered on errors by writing a one to the Transfer Error bit in the Interrupt Enable Register (IER.TERR).

19.5.11 Performance Monitors

The performance monitor hardware allows the user to measure the activity and stall cycles for PDCA transfers. Performance monitoring is implemented for two PDCA channels, the other channels cannot be monitored. This reduces the hardware cost of the feature. The selection of the two PDCA channels to monitor is done through the PDCA Channel to Monitor with Counter0/1 fields in the Performance Control Register (PCONTROL.MON1CH and PCONTROL.MON0CH). Due to performance monitor hardware resource sharing, the two monitor channels should NOT be programmed to monitor the same PDCA channel. This may result in UNDEFINED performance monitor behavior.

19.5.11.1 Measuring mechanisms

Three different parameters can be measured by each channel:

- The number of data transfer cycles since last channel reset, both for read and write
- The number of stall cycles since last channel reset, both for read and write
- The maximum latency since last channel reset, both for read and write

These measurements can be extracted by software and used to generate indicators for bus latency, bus load and maximum bus latency.

Each of the counters has a fixed width, and may therefore overflow. When overflow is encountered in either the Performance Channel Data Read/Write Cycle registers (PRDATAn and PWDATAn) or Performance Channel Read/Write Stall Cycles registers (PRSTALLn and PWSTALLn) of a channel, all registers in the channel are reset. This behavior is altered if the Channel Overflow Freeze bit is written to one in the Performance Control register (PCONTROL.CHnOVF). If this bit is set, the channel registers are frozen when either DATA or STALL reaches its maximum value. This simplifies one-shot readout of the counter values.

The registers can also be manually reset by writing a one to the Channel Reset bit in PCONTROL register (PCONTROL.CHnRES). The Performance Channel Read/Write Latency registers (PRLATn and PWLATn) are saturating when their maximum count value is reached. The PRLATn and PWLATn registers are reset only by the user writing the reset bits in PCONTROL (PCONTROL.CHnRES).

A counter must manually be enabled by writing a one to the Channel Enable bit in the Performance Control Register (PCONTROL.CHnEN).

19.6 User Interface

19.6.1 Memory Map Overview

Table 19-1. PDCA Register Memory Map

Address Range	Contents
0x000 - 0x03F	DMA channel 0 configuration registers
0x040 - 0x07F	DMA channel 1 configuration registers
...	...
0x1C0 - 0x1FF	DMA channel 7 configuration registers
0x800-0x830	Performance Monitor registers
0x834	Version register

19.6.2 Channel Memory Map

Table 19-2. PDCA Channel Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000 + n*0x040	Memory Address Register	MAR	Read/Write	0x00000000
0x004 + n*0x040	Peripheral Select Register	PSR	Read/Write	- (1)
0x008 + n*0x040	Transfer Counter Register	TCR	Read/Write	0x00000000
0x00C + n*0x040	Memory Address Reload Register	MARR	Read/Write	0x00000000
0x010 + n*0x040	Transfer Counter Reload Register	TCRR	Read/Write	0x00000000
0x014 + n*0x040	Control Register	CR	Write-only	0x00000000
0x018 + n*0x040	Mode Register	MR	Read/Write	0x00000000
0x01C + n*0x040	Status Register	SR	Read-only	0x00000000
0x020 + n*0x040	Interrupt Enable Register	IER	Write-only	0x00000000
0x024 + n*0x040	Interrupt Disable Register	IDR	Write-only	0x00000000
0x028 + n*0x040	Interrupt Mask Register	IMR	Read-only	0x00000000
0x02C + n*0x040	Interrupt Status Register	ISR	Read-only	0x00000000

19.6.3 Performance Monitor Memory Map

Table 19-3. PDCA Performance Monitor Register Memory Map

Offset	Register	Register Name	Access	Reset
0x800	Control	PCONTROL	Read/Write	0x00000000
0x804	Channel0 Read Data Cycles	PRDATA0	Read-only	0x00000000
0x808	Channel0 Read Stall Cycles	PRSTALL0	Read-only	0x00000000
0x80C	Channel0 Read Max Latency	PRLAT0	Read-only	0x00000000
0x810	Channel0 Write Data Cycles	PWDATA0	Read-only	0x00000000
0x814	Channel0 Write Stall Cycles	PWSTALL0	Read-only	0x00000000
0x818	Channel0 Write Max Latency	PWLAT0	Read-only	0x00000000

Table 19-3. PDCA Performance Monitor Register Memory Map

Offset	Register	Register Name	Access	Reset
0x81C	Channel1 Read Data Cycles	PRDATA1	Read-only	0x00000000
0x820	Channel1 Read Stall Cycles	PRSTALL1	Read-only	0x00000000
0x824	Channel1 Read Max Latency	PRLAT1	Read-only	0x00000000
0x828	Channel1 Write Data Cycles	PWDATA1	Read-only	0x00000000
0x82C	Channel1 Write Stall Cycles	PWSTALL1	Read-only	0x00000000
0x830	Channel1 Write Max Latency	PWLAT1	Read-only	0x00000000

19.6.4 Version Register Memory Map

Table 19-4. PDCA Version Register Memory Map

Offset	Register	Register Name	Access	Reset
0x834	Version Register	VERSION	Read-only	- (1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

19.6.5 Memory Address Register

Name: MAR

Access Type: Read/Write

Offset: 0x000 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MADDR[31:24]							

23	22	21	20	19	18	17	16
MADDR[23:16]							

15	14	13	12	11	10	9	8
MADDR[15:8]							

7	6	5	4	3	2	1	0
MADDR[7:0]							

- **MADDR: Memory Address**
  Address of memory buffer. MADDR should be programmed to point to the start of the memory buffer when configuring the PDCA. During transfer, MADDR will point to the next memory location to be read/written.

19.6.6 Peripheral Select Register

Name: PSR

Access Type: Read/Write

Offset: 0x004 + n*0x040

Reset Value: -

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
PID							

- **PID: Peripheral Identifier**

  The Peripheral Identifier selects which peripheral should be connected to the DMA channel. Programming PID will select both which handshake interface to use, the direction of the transfer and also the address of the Receive/Transfer Holding Register for the peripheral. See the module configuration section of PDCA for details. The width of the PID field is implementation specific and dependent on the number of peripheral modules in the microcontroller.

19.6.7 Transfer Counter Register

Name: TCR

Access Type: Read/Write

Offset: 0x008 + n*0x040

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
TCV[15:8]							

7	6	5	4	3	2	1	0
TCV[7:0]							

- **TCV: Transfer Counter Value**
  Number of data items to be transferred by PDCA. TCV must be programmed with the total number of transfers to be made. During transfer, TCV contains the number of remaining transfers to be done.

19.6.8 Memory Address Reload Register

Name: MARR

Access Type: Read/Write

Offset: 0x00C + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MARV[31:24]							

23	22	21	20	19	18	17	16
MARV[23:16]							

15	14	13	12	11	10	9	8
MARV[15:8]							

7	6	5	4	3	2	1	0
MARV[7:0]							

- **MARV: Memory Address Reload Value**
  Reload Value for the MAR register. This value will be loaded into MAR when TCR reaches zero if the TCRR register has a non-zero value.

19.6.9 Transfer Counter Reload Register

Name: TCRR

Access Type: Read/Write

Offset: 0x010 + n*0x040

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
TCRV[15:8]							

7	6	5	4	3	2	1	0
TCRV[7:0]							

- **TCRV: Transfer Counter Reload Value**
  Reload value for the TCR register. When TCR reaches zero, it will be reloaded with TCRV if TCRV has a positive value. If TCRV is zero, no more transfers will be performed for the channel. When TCR is reloaded, the TCRR register is cleared.

19.6.10 Control Register

Name: CR

Access Type: Write-only

Offset: 0x014 + n*0x040

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
-	-	-	-	-	-	-	ECLR

7	6	5	4	3	2	1	0
-	-	-	-	-	-	TDIS	TEN

- **ECLR: Transfer Error Clear**
  Writing a one to this bit will clear the Transfer Error bit in the Status Register (SR.TERR). Clearing the SR.TERR bit will allow the channel to transmit data. The memory address must first be set to point to a valid location.
  Writing a zero to this bit has no effect.
- **TDIS: Transfer Disable**
  Writing a one to this bit will disable transfer for DMA channel.
  Writing a zero to this bit has no effect.
- **TEN: Transfer Enable**
  Writing a one to this bit will enable transfer for DMA channel.
  Writing a zero to this bit has no effect.

19.6.11 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x018 + n*0x040

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
-	-	-	-	-	-	SIZE	

- **SIZE: Size of Transfer**

Table 19-5. Size of Transfer

SIZE	Size of Transfer
0	byte
1	halfword
2	word
3	Reserved

19.6.12 Status Register

Name: SR

Access Type: Read-only

Offset: 0x01C + n*0x040

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
-	-	-	-	-	-	-	TEN

- **TEN: Transfer Enabled**

  This bit is set when the TEN bit in CR register is written to one.

  This bit is cleared when the TDIS bit in CR register is written to one.

  1: Transfer is enabled for the DMA channel.

  0: Transfer is disabled for the DMA channel.

19.6.13 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x020 + n*0x040

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
-	-	-	-	-	TERR	TRC	RCZ

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

19.6.14 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x024 + n*0x040

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
-	-	-	-	-	TERR	TRC	RCZ

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

19.6.15 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x028 + n*0x040

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
-	-	-	-	-	TERR	TRC	RCZ

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

19.6.16 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x02C + n*0x040

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
-	-	-	-	-	TERR	TRC	RCZ

- **TERR: Transfer Error**
  1: A transfer error has occurred.
  0: No transfer errors have occurred.
- **TRC: Transfer Complete**
  1: Both the TCR and the TCRR are zero.
  0: The TCR and/or the TCRR hold a non-zero value.
- **RCZ: Reload Counter Zero**
  1: The TCRR is zero.
  0: The TCRR holds a non-zero value.

19.6.17 Performance Control Register

Name: PCONTROL

Access Type: Read/Write

Offset: 0x800

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	MON1CH					

23	22	21	20	19	18	17	16
-	-	MON0CH					

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CH1RES	CH0RES

7	6	5	4	3	2	1	0
-	-	CH1OF	CH0OF	-	-	CH1EN	CH0EN

- **MON1CH: PDCA Channel to Monitor with Counter 1**
- **MON0CH: PDCA Channel to Monitor with Counter 0**

  Due to performance monitor hardware resource sharing, the two monitor channels should NOT be programmed to monitor the same PDCA channel. This may result in UNDEFINED performance monitor behavior.
- **CH1RES: Channel 1 Counter Reset**

  Writing a one to this bit will reset the counter in the channel 1.

  Writing a zero to this bit has no effect.

  Always read as 0.
- **CH0RES: Channel 0 Counter Reset**

  Writing a one to this bit will reset the counter in the channel 0.

  Writing a zero to this bit has no effect.

  Always read as 0.
- **CH1OF: Channel Overflow Freeze**

  1: All channel registers are frozen just before DATA or STALL overflows.

  0: The channel registers are reset if DATA or STALL overflows.
- **CH1OF: Channel Overflow Freeze**

  1: All channel registers are frozen just before DATA or STALL overflows.

  0: The channel registers are reset if DATA or STALL overflows.
- **CH1EN: Channel 1 Enable**

  1: Channel 1 is enabled.

  0: Channel 1 is disabled.
- **CH0EN: Channel 0 Enable**

  1: Channel 0 is enabled.

  0: Channel 0 is disabled.

19.6.18 Performance Channel 0 Read Data Cycles

Name: PRDATA0

Access Type: Read-only

Offset: 0x804

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**

19.6.19 Performance Channel 0 Read Stall Cycles

Name: PRSTALL0

Access Type: Read-only

Offset: 0x808

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**

19.6.20 Performance Channel 0 Read Max Latency

Name: PRLAT0

Access Type: Read/Write

Offset: 0x80C

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
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  This counter is saturating. The register is reset only when the reset bits in PCONTROL are written.

19.6.21 Performance Channel 0 Write Data Cycles

Name: PWDATA0

Access Type: Read-only

Offset: 0x810

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**

19.6.22 Performance Channel 0 Write Stall Cycles

Name: PWSTALL0

Access Type: Read-only

Offset: 0x814

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**

19.6.23 Performance Channel 0 Write Max Latency

Name: PWLAT0

Access Type: Read/Write

Offset: 0x818

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
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  This counter is saturating. The register is reset only when the reset bits in PCONTROL are written.

19.6.24 Performance Channel 1 Read Data Cycles

Name: PRDATA1

Access Type: Read-only

Offset: 0x81C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**

19.6.25 Performance Channel 1 Read Stall Cycles

Name: PRSTALL1

Access Type: Read-only

Offset: 0x820

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**

19.6.26 Performance Channel 1 Read Max Latency

Name: PLATR1

Access Type: Read/Write

Offset: 0x824

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
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  This counter is saturating. The register is reset only when the reset bits in PCONTROL are written.

19.6.27 Performance Channel 1 Write Data Cycles

Name: PWDATA1

Access Type: Read-only

Offset: 0x828

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**

19.6.28 Performance Channel 1 Write Stall Cycles

Name: PWSTALL1

Access Type: Read-only

Offset: 0x82C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**

19.6.29 Performance Channel 1 Write Max Latency

Name: PWLAT1

Access Type: Read/Write

Offset: 0x830

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
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  This counter is saturating. The register is reset only when the reset bits in PCONTROL are written.

19.6.30 PDCA Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x834

Reset Value: -

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

19.7 Module Configuration

The specific configuration for the PDCA instance is listed in the following tables.

Table 19-6. Register Reset Values

Register	Reset Value
PSRn	n
VERSION	0x00000110

19.7.1 DMA Handshake Signals

The following table defines the valid settings for the Peripheral Identifier (PID) in the PDCA Peripheral Select Register (PSR).).

Table 19-7. PDCA Handshake Signals

PID Value	Peripheral module & direction
0	ADC - RX
1	SSC - RX
2	USART0 - RX
3	USART1 - RX
4	USART2 - RX
5	USART3 - RX
6	TWIM0 - RX
7	TWIM1 - RX
8	TWIS0 - RX
9	TWIS1 - RX
10	SPI0 - RX
11	SPI1 - RX
12	SSC - TX
13	USART0 - TX
14	USART1 - TX
15	USART2 - TX
16	USART3 - TX
17	TWIM0 - TX
18	TWIM1 - TX
19	TWIS0 - TX
20	TWIS1 - TX
21	SPI0 - TX
22	SPI1 - TX
23	DAC - TX

20. DMA Controller (DMACA)

Rev: 2.0.6a.6

20.1 Features

- 2 HSB Master Interfaces
- 4 Channels
- Software and Hardware Handshaking Interfaces
  - 11 Hardware Handshaking Interfaces
- Memory/Non-Memory Peripherals to Memory/Non-Memory Peripherals Transfer
- Single-block DMA Transfer
- Multi-block DMA Transfer
  - Linked Lists
  - Auto-Reloading
  - Contiguous Blocks
- DMA Controller is Always the Flow Controller
- Additional Features
  - Scatter and Gather Operations
  - Channel Locking
  - Bus Locking
  - FIFO Mode
  - Pseudo Fly-by Operation

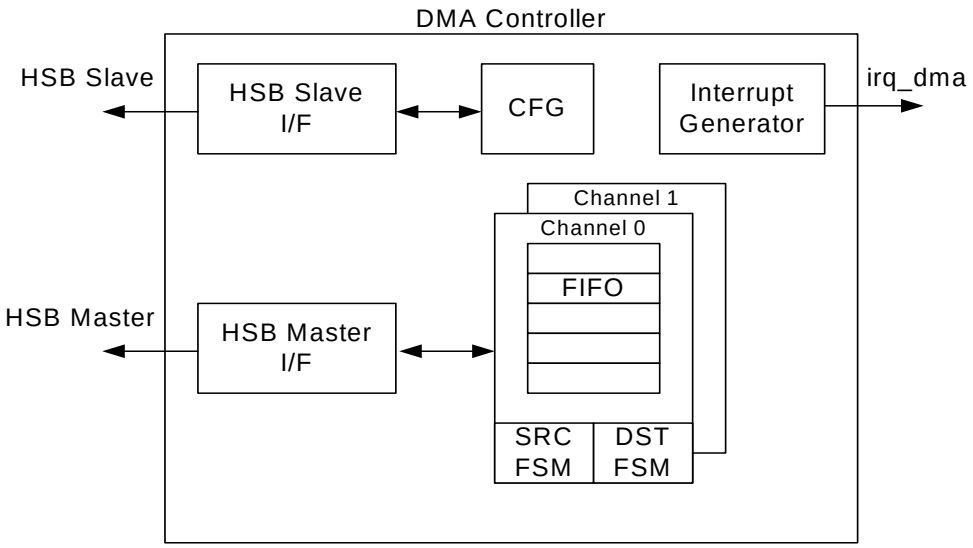
20.2 Overview

The DMA Controller (DMACA) is an HSB-central DMA controller core that transfers data from a source peripheral to a destination peripheral over one or more System Bus. One channel is required for each source/destination pair. In the most basic configuration, the DMACA has one master interface and one channel. The master interface reads the data from a source and writes it to a destination. Two System Bus transfers are required for each DMA data transfer. This is also known as a dual-access transfer.

The DMACA is programmed via the HSB slave interface.

20.3 Block Diagram

Figure 20-1. DMA Controller (DMACA) Block Diagram



20.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

20.4.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with GPIO lines. The user must first program the GPIO controller to assign the DMACA pins to their peripheral functions.

20.4.2 Power Management

To prevent bus errors the DMACA operation must be terminated before entering sleep mode.

20.4.3 Clocks

The CLK_DMACA to the DMACA is generated by the Power Manager (PM). Before using the DMACA, the user must ensure that the DMACA clock is enabled in the power manager.

20.4.4 Interrupts

The DMACA interface has an interrupt line connected to the Interrupt Controller. Handling the DMACA interrupt requires programming the interrupt controller before configuring the DMACA.

20.4.5 Peripherals

Both the source peripheral and the destination peripheral must be set up correctly prior to the DMA transfer.

20.5 Functional Description

20.5.1 Basic Definitions

Source peripheral: Device on a System Bus layer from where the DMACA reads data, which is then stored in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel.

Destination peripheral: Device to which the DMACA writes the stored data from the FIFO (previously read from the source peripheral).

Memory: Source or destination that is always “ready” for a DMA transfer and does not require a handshaking interface to interact with the DMACA. A peripheral should be assigned as memory only if it does not insert more than 16 wait states. If more than 16 wait states are required, then the peripheral should use a handshaking interface (the default if the peripheral is not programmed to be memory) in order to signal when it is ready to accept or supply data.

Channel: Read/write datapath between a source peripheral on one configured System Bus layer and a destination peripheral on the same or different System Bus layer that occurs through the channel FIFO. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.

Master interface: DMACA is a master on the HSB bus reading data from the source and writing it to the destination over the HSB bus.

Slave interface: The HSB interface over which the DMACA is programmed. The slave interface in practice could be on the same layer as any of the master interfaces or on a separate layer.

Handshaking interface: A set of signal registers that conform to a protocol and *handshake* between the DMACA and source or destination peripheral to control the transfer of a single or burst transaction between them. This interface is used to request, acknowledge, and control a DMACA transaction. A channel can receive a request through one of three types of handshaking interface: hardware, software, or peripheral interrupt.

Hardware handshaking interface: Uses hardware signals to control the transfer of a single or burst transaction between the DMACA and the source or destination peripheral.

Software handshaking interface: Uses software registers to control the transfer of a single or burst transaction between the DMACA and the source or destination peripheral. No special DMACA handshaking signals are needed on the I/O of the peripheral. This mode is useful for interfacing an existing peripheral to the DMACA without modifying it.

Peripheral interrupt handshaking interface: A simple use of the hardware handshaking interface. In this mode, the interrupt line from the peripheral is tied to the dma_req input of the hardware handshaking interface. Other interface signals are ignored.

Flow controller: The device (either the DMACA or source/destination peripheral) that determines the length of and terminates a DMA block transfer. If the length of a block is known before enabling the channel, then the DMACA should be programmed as the flow controller. If the length of a block is not known prior to enabling the channel, the source or destination peripheral needs to terminate a block transfer. In this mode, the peripheral is the flow controller.

Flow control mode (CFGx.FCMODE): Special mode that only applies when the destination peripheral is the flow controller. It controls the pre-fetching of data from the source peripheral.

Transfer hierarchy: Figure 20-2 on page 309 illustrates the hierarchy between DMACA transfers, block transfers, transactions (single or burst), and System Bus transfers (single or burst) for non-memory peripherals. Figure 20-3 on page 309 shows the transfer hierarchy for memory.

Figure 20-2. DMACA Transfer Hierarchy for Non-Memory Peripheral

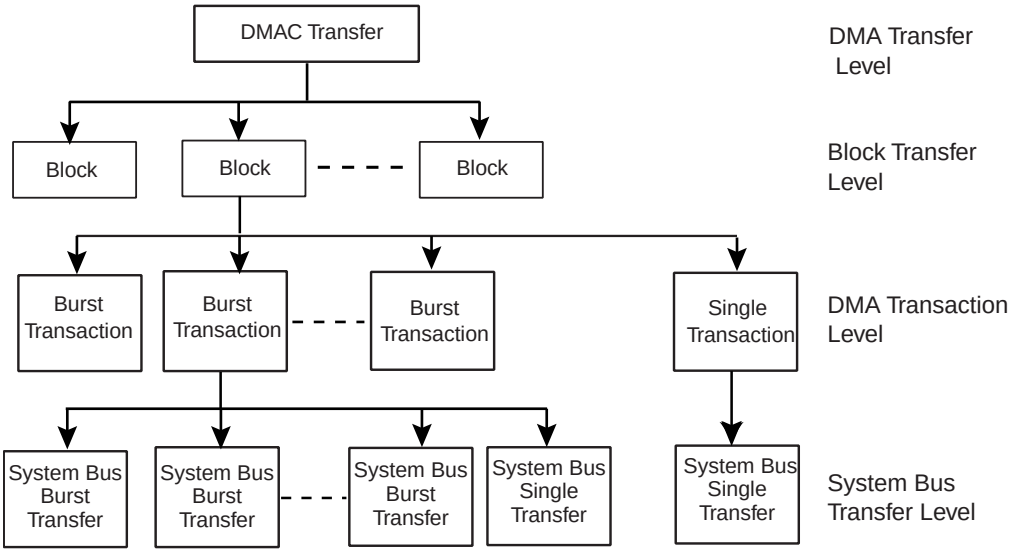
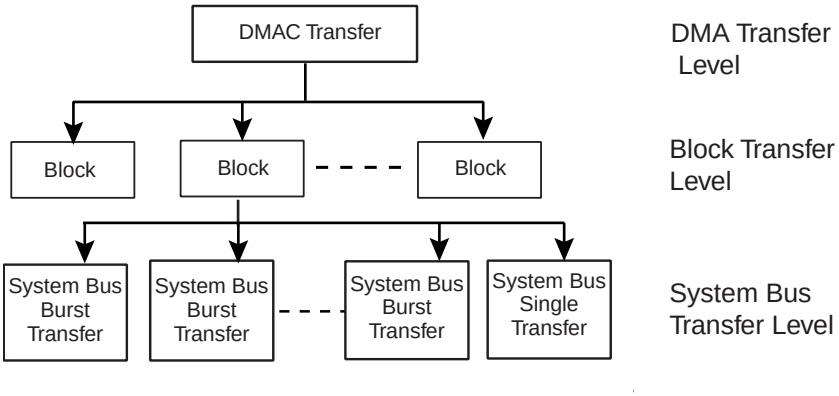


Figure 20-3. DMACA Transfer Hierarchy for Memory



Block: A block of DMACA data. The amount of data (block length) is determined by the flow controller. For transfers between the DMACA and memory, a block is broken directly into a sequence of System Bus bursts and single transfers. For transfers between the DMACA and a non-memory peripheral, a block is broken into a sequence of DMACA transactions (single and bursts). These are in turn broken into a sequence of System Bus transfers.

Transaction: A basic unit of a DMACA transfer as determined by either the hardware or software handshaking interface. A transaction is only relevant for transfers between the DMACA and a source or destination peripheral if the source or destination peripheral is a non-memory device. There are two types of transactions: single and burst.

– **Single transaction:** The length of a single transaction is always 1 and is converted to a single System Bus transfer.

– **Burst transaction:** The length of a burst transaction is programmed into the DMACA. The burst transaction is converted into a sequence of System Bus bursts and single transfers. DMACA executes each burst transfer by performing incremental bursts that are no longer than the maximum System Bus burst size set. The burst transaction length is under program control and normally bears some relationship to the FIFO sizes in the DMACA and in the source and destination peripherals.

DMA transfer: Software controls the number of blocks in a DMACA transfer. Once the DMA transfer has completed, then hardware within the DMACA disables the channel and can generate an interrupt to signal the completion of the DMA transfer. You can then re-program the channel for a new DMA transfer.

Single-block DMA transfer: Consists of a single block.

Multi-block DMA transfer: A DMA transfer may consist of multiple DMACA blocks. Multi-block DMA transfers are supported through block chaining (linked list pointers), auto-reloading of channel registers, and contiguous blocks. The source and destination can independently select which method to use.

– **Linked lists (block chaining) –** A linked list pointer (LLP) points to the location in system memory where the next linked list item (LLI) exists. The LLI is a set of registers that describe the next block (block descriptor) and an LLP register. The DMACA fetches the LLI at the beginning of every block when block chaining is enabled.

– **Auto-reloading –** The DMACA automatically reloads the channel registers at the end of each block to the value when the channel was first enabled.

– **Contiguous blocks –** Where the address between successive blocks is selected to be a continuation from the end of the previous block.

Scatter: Relevant to destination transfers within a block. The destination System Bus address is incremented or decremented by a programmed amount -the scatter increment- when a scatter boundary is reached. The destination System Bus address is incremented or decremented by the value stored in the destination scatter increment (DSRx.DSI) field, multiplied by the number of bytes in a single HSB transfer to the destination (decoded value of CTLx.DST_TR_WIDTH)/8. The number of destination transfers between successive scatter boundaries is programmed into the Destination Scatter Count (DSC) field of the DSRx register.

Scatter is enabled by writing a ‘1’ to the CTLx.DST_SCATTER_EN bit. The CTLx.DINC field determines if the address is incremented, decremented or remains fixed when a scatter boundary is reached. If the CTLx.DINC field indicates a fixed-address control throughout a DMA transfer, then the CTLx.DST_SCATTER_EN bit is ignored, and the scatter feature is automatically disabled.

Gather: Relevant to source transfers within a block. The source System Bus address is incremented or decremented by a programmed amount when a gather boundary is reached. The number of System Bus transfers between successive gather boundaries is programmed into the Source Gather Count (SGRx.SGC) field. The source address is incremented or decremented by the value stored in the source gather increment (SGRx.SGI) field multiplied by the number of bytes in a single HSB transfer from the source -(decoded value of CTLx.SRC_TR_WIDTH)/8 - when a gather boundary is reached.

Gather is enabled by writing a ‘1’ to the CTLx.SRC_GATHER_EN bit. The CTLx.SINC field determines if the address is incremented, decremented or remains fixed when a gather boundary is reached. If the CTLx.SINC field indicates a fixed-address control throughout a DMA transfer, then the CTLx.SRC_GATHER_EN bit is ignored and the gather feature is automatically disabled.

Note: For multi-block transfers, the counters that keep track of the number of transfer left to reach a gather/scatter boundary are re-initialized to the source gather count (SGRx.SGC) and destination scatter count (DSRx.DSC), respectively, at the start of each block transfer.

Figure 20-4. Destination Scatter Transfer

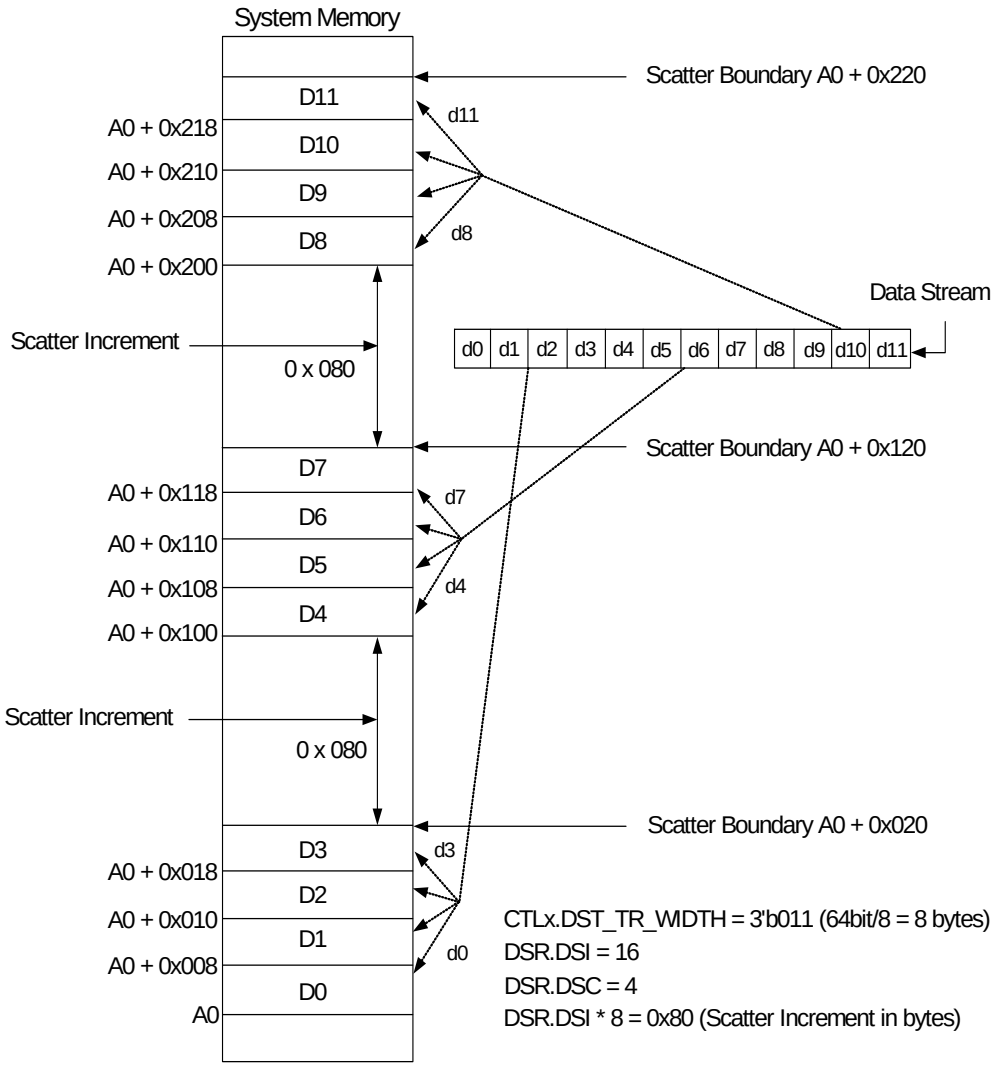
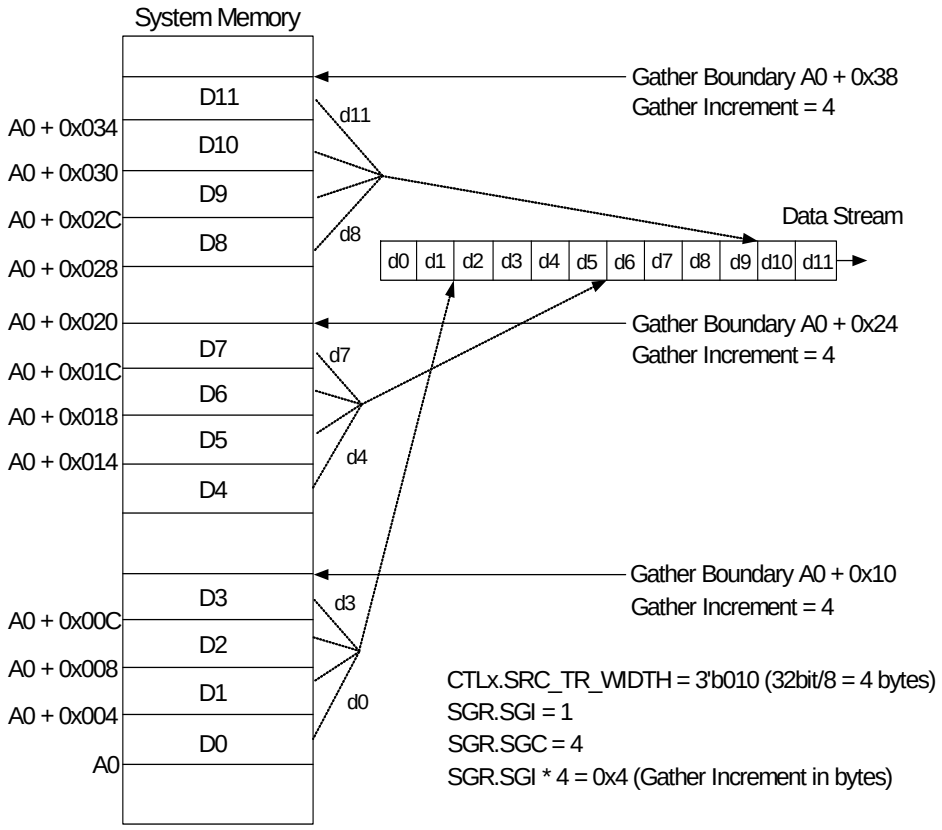


Figure 20-5. Source Gather Transfer



Channel locking: Software can program a channel to keep the HSB master interface by locking the arbitration for the master bus interface for the duration of a DMA transfer, block, or transaction (single or burst).

Bus locking: Software can program a channel to maintain control of the System Bus bus by asserting hlock for the duration of a DMA transfer, block, or transaction (single or burst). Channel locking is asserted for the duration of bus locking at a minimum.

FIFO mode: Special mode to improve bandwidth. When enabled, the channel waits until the FIFO is less than half full to fetch the data from the source peripheral and waits until the FIFO is greater than or equal to half full to send data to the destination peripheral. Thus, the channel can transfer the data using System Bus bursts, eliminating the need to arbitrate for the HSB master interface for each single System Bus transfer. When this mode is not enabled, the channel only waits until the FIFO can transmit/accept a single System Bus transfer before requesting the master bus interface.

Pseudo fly-by operation: Typically, it takes two System Bus cycles to complete a transfer, one for reading the source and one for writing to the destination. However, when the source and destination peripherals of a DMA transfer are on different System Bus layers, it is possible for the DMACA to fetch data from the source and store it in the channel FIFO at the same time as the DMACA extracts data from the channel FIFO and writes it to the destination peripheral. This activity is known as *pseudo fly-by operation*. For this to occur, the master interface for both source and destination layers must win arbitration of their HSB layer. Similarly, the source and destination peripherals must win ownership of their respective master interfaces.

20.6 Arbitration for HSB Master Interface

Each DMACA channel has two request lines that request ownership of a particular master bus interface: channel source and channel destination request lines.

Source and destination arbitrate separately for the bus. Once a source/destination state machine gains ownership of the master bus interface and the master bus interface has ownership of the HSB bus, then HSB transfers can proceed between the peripheral and the DMACA.

An arbitration scheme decides which of the request lines (2 * DMAH_NUM_CHANNELS) is granted the particular master bus interface. Each channel has a programmable priority. A request for the master bus interface can be made at any time, but is granted only after the current HSB transfer (burst or single) has completed. Therefore, if the master interface is transferring data for a lower priority channel and a higher priority channel requests service, then the master interface will complete the current burst for the lower priority channel before switching to transfer data for the higher priority channel.

If only one request line is active at the highest priority level, then the request with the highest priority wins ownership of the HSB master bus interface; it is not necessary for the priority levels to be unique.

If more than one request is active at the highest requesting priority, then these competing requests proceed to a second tier of arbitration:

If equal priority requests occur, then the lower-numbered channel is granted.

In other words, if a peripheral request attached to Channel 7 and a peripheral request attached to Channel 8 have the same priority, then the peripheral attached to Channel 7 is granted first.

20.7 Memory Peripherals

Figure 20-3 on page 309 shows the DMA transfer hierarchy of the DMACA for a memory peripheral. There is no handshaking interface with the DMACA, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request. The alternative to not having a transaction-level handshaking interface is to allow the DMACA to attempt System Bus transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these System Bus transfers, it inserts wait states onto the bus until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking interface, the peripheral can signal to the DMACA that it is ready to transmit/receive data, and then the DMACA can access the peripheral without the peripheral inserting wait states onto the bus.

20.8 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or burst transactions. The operation of the handshaking interface is different and depends on whether the peripheral or the DMACA is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMACA that it is ready to transfer/accept data over the System Bus. A non-memory peripheral can request a DMA transfer through the DMACA using one of two handshaking interfaces:

- Hardware handshaking
- Software handshaking

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

20.8.1 Software Handshaking

When the slave peripheral requires the DMACA to perform a DMA transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller.

The interrupt service routine then uses the software registers to initiate and control a DMA transaction. These software registers are used to implement the software handshaking interface.

The HS_SEL_SRC/HS_SEL_DST bit in the CFGx channel configuration register must be set to enable software handshaking.

When the peripheral is not the flow controller, then the last transaction registers LstSrcReg and LstDstReg are not used, and the values in these registers are ignored.

20.8.1.1 Burst Transactions

Writing a 1 to the ReqSrcReg[x]/ReqDstReg[x] register is always interpreted as a burst transaction request, where *x* is the channel number. However, in order for a burst transaction request to start, software must write a 1 to the SglReqSrcReg[x]/SglReqDstReg[x] register.

You can write a 1 to the SglReqSrcReg[*x*]/SglReqDstReg[*x*] and ReqSrcReg[*x*]/ReqDstReg[*x*] registers in any order, but both registers must be asserted in order to initiate a burst transaction. Upon completion of the burst transaction, the hardware clears the SglReqSrcReg[*x*]/SglReqDstReg[*x*] and ReqSrcReg[*x*]/ReqDstReg[*x*] registers.

20.8.1.2 Single Transactions

Writing a 1 to the SglReqSrcReg/SglReqDstReg initiates a single transaction. Upon completion of the single transaction, both the SglReqSrcReg/SglReqDstReg and ReqSrcReg/ReqDstReg bits are cleared by hardware. Therefore, writing a 1 to the ReqSrcReg/ReqDstReg is ignored while a single transaction has been initiated, and the requested burst transaction is not serviced.

Again, writing a 1 to the ReqSrcReg/ReqDstReg register is always a burst transaction request. However, in order for a burst transaction request to start, the corresponding channel bit in the SglReqSrcReg/SglReqDstReg must be asserted. Therefore, to ensure that a burst transaction is serviced, you must write a 1 to the ReqSrcReg/ReqDstReg before writing a 1 to the SglReqSrcReg/SglReqDstReg register.

Software can poll the relevant channel bit in the SglReqSrcReg/ SglReqDstReg and ReqSrcReg/ReqDstReg registers. When both are 0, then either the requested burst or single transaction has completed. Alternatively, the IntSrcTran or IntDstTran interrupts can be enabled and unmasked in order to generate an interrupt when the requested source or destination transaction has completed.

Note: The transaction-complete interrupts are triggered when both single and burst transactions are complete. The same transaction-complete interrupt is used for both single and burst transactions.

20.8.2 Hardware Handshaking

There are 11 hardware handshaking interfaces between the DMACA and peripherals. Refer to the module configuration chapter for the device-specific mapping of these interfaces.

20.8.2.1 External DMA Request Definition

When an external slave peripheral requires the DMACA to perform DMA transactions, it communicates its request by asserting the external nDMAREQx signal. This signal is resynchronized to ensure a proper functionality (see “External DMA Request Timing” on page 315).

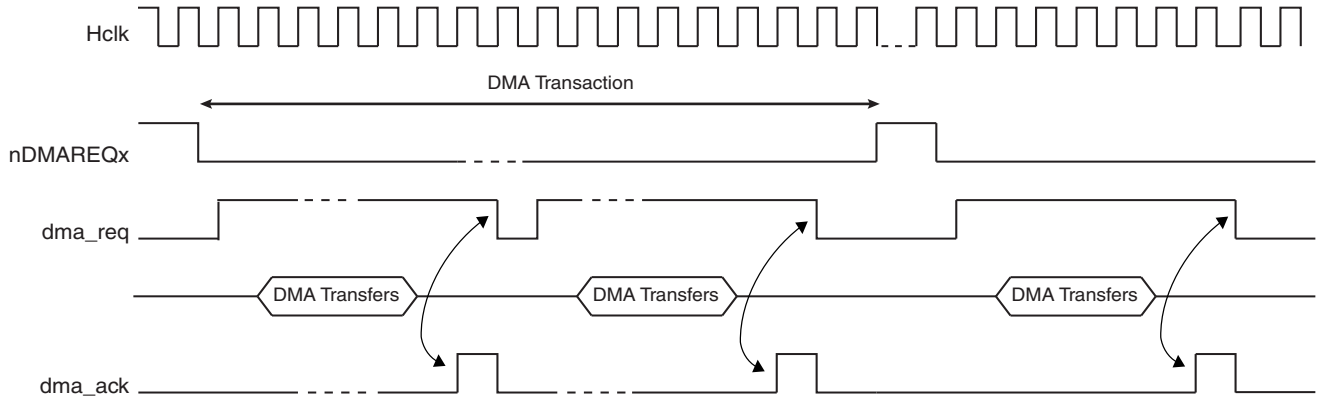
The external nDMAREQx signal should be asserted when the source threshold level is reached. After resynchronization, the rising edge of dma_req starts the transfer. An external DMAACKx acknowledge signal is also provided to indicate when the DMA transfer has completed. The peripheral should de-assert the DMA request signal when DMAACKx is asserted.

The external nDMAREQx signal must be de-asserted after the last transfer and re-asserted again before a new transaction starts.

For a source FIFO, an active edge should be triggered on nDMAREQx when the source FIFO exceeds a watermark level. For a destination FIFO, an active edge should be triggered on nDMAREQx when the destination FIFO drops below the watermark level.

The source transaction length, CTLx.SRC_MSIZE, and destination transaction length, CTLx.DEST_MSIZE, must be set according to watermark levels on the source/destination peripherals.

Figure 20-6. External DMA Request Timing



20.9 DMACA Transfer Types

A DMA transfer may consist of single or multi-block transfers. On successive blocks of a multi-block transfer, the SARx/DARx register in the DMACA is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the CTLx register in the DMACA is re-programmed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using linked lists is the multi-block method of choice, and on successive blocks, the LLPx register in the DMACA is re-programmed using the following method:

- Block chaining using linked lists

A block descriptor (LLI) consists of following registers, SARx, DARx, LLPx, CTL. These registers, along with the CFGx register, are used by the DMACA to set up and describe the block transfer.

20.9.1 Multi-block Transfers

20.9.1.1 Block Chaining Using Linked Lists

In this case, the DMACA re-programs the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

DMACA block chaining is supported by using a Linked List Pointer register (LLPx) that stores the address in memory of the next linked list item. Each LLI (block descriptor) contains the corresponding block descriptor (SARx, DARx, LLPx, CTLx).

To set up block chaining, a sequence of linked lists must be programmed in memory.

The SARx, DARx, LLPx and CTLx registers are fetched from system memory on an LLI update. The updated contents of the CTLx register are written back to memory on block completion. Figure 20-7 on page 316 shows how to use chained linked lists in memory to define multi-block transfers using block chaining.

The Linked List multi-block transfers is initiated by programming LLPx with LLPx(0) (LLI(0) base address) and CTLx with CTLx.LLP_S_EN and CTLx.LLP_D_EN.

Figure 20-7. Multi-block Transfer Using Linked Lists

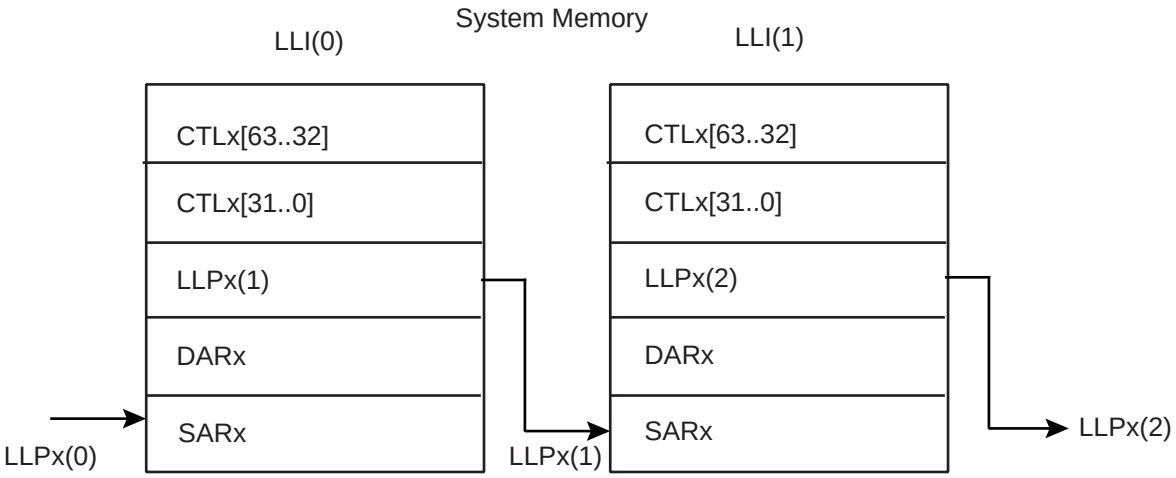


Table 20-1. Programming of Transfer Types and Channel Register Update Method (DMACA State Machine Table)

Transfer Type	LLP.LOC = 0	LLP_S_EN (CTLx)	RELOAD_SR (CFGx)	LLP_D_EN (CTLx)	RELOAD_DS (CFGx)	CTLx, LLPx Update Method	SARx Update Method	DARx Update Method	Write Back
1) Single Block or last transfer of multi-Block	Yes	0	0	0	0	None, user reprograms	None (single)	None (single)	No
2) Auto Reload multi-block transfer with contiguous SAR	Yes	0	0	0	1	CTLx,LLPx are reloaded from initial values.	Contiguous	Auto-Reload	No
3) Auto Reload multi-block transfer with contiguous DAR	Yes	0	1	0	0	CTLx,LLPx are reloaded from initial values.	Auto-Reload	Con-tiguous	No
4) Auto Reload multi-block transfer	Yes	0	1	0	1	CTLx,LLPx are reloaded from initial values.	Auto-Reload	Auto-Reload	No
5) Single Block or last transfer of multi-block	No	0	0	0	0	None, user reprograms	None (single)	None (single)	Yes
6) Linked List multi-block transfer with contiguous SAR	No	0	0	1	0	CTLx,LLPx loaded from next Linked List item	Contiguous	Linked List	Yes
7) Linked List multi-block transfer with auto-reload SAR	No	0	1	1	0	CTLx,LLPx loaded from next Linked List item	Auto-Reload	Linked List	Yes
8) Linked List multi-block transfer with contiguous DAR	No	1	0	0	0	CTLx,LLPx loaded from next Linked List item	Linked List	Con-tiguous	Yes
9) Linked List multi-block transfer with auto-reload DAR	No	1	0	0	1	CTLx,LLPx loaded from next Linked List item	Linked List	Auto-Reload	Yes
10) Linked List multi-block transfer	No	1	0	1	0	CTLx,LLPx loaded from next Linked List item	Linked List	Linked List	Yes

20.9.1.2 Auto-reloading of Channel Registers

During auto-reloading, the channel registers are reloaded with their initial values at the completion of each block and the new values used for the new block. Depending on the row number in Table 20-1 on page 317, some or all of the SARx, DARx and CTLx channel registers are reloaded from their initial value at the start of a block transfer.

20.9.1.3 Contiguous Address Between Blocks

In this case, the address between successive blocks is selected to be a continuation from the end of the previous block. Enabling the source or destination address to be contiguous between

blocks is a function of CTLx.LLP_S_EN, CFGx.RELOAD_SR, CTLx.LLP_D_EN, and CFGx.RELOAD_DS registers (see Figure 20-1 on page 307).

Note: Both SARx and DARx updates cannot be selected to be contiguous. If this functionality is required, the size of the Block Transfer (CTLx.BLOCK_TS) must be increased. If this is at the maximum value, use Row 10 of Table 20-1 on page 317 and setup the LLI.SARx address of the block descriptor to be equal to the end SARx address of the previous block. Similarly, setup the LLI.DARx address of the block descriptor to be equal to the end DARx address of the previous block.

20.9.1.4 Suspension of Transfers Between Blocks

At the end of every block transfer, an end of block interrupt is asserted if:

- interrupts are enabled, CTLx.INT_EN = 1
- the channel block interrupt is unmasked, MaskBlock[n] = 0, where n is the channel number.

Note: The block complete interrupt is generated at the completion of the block transfer to the destination.

For rows 6, 8, and 10 of Table 20-1 on page 317, the DMA transfer does not stall between block transfers. For example, at the end of block N, the DMACA automatically proceeds to block N + 1.

For rows 2, 3, 4, 7, and 9 of Table 20-1 on page 317 (SARx and/or DARx auto-reloaded between block transfers), the DMA transfer automatically stalls after the end of block. Interrupt is asserted if the end of block interrupt is enabled and unmasked.

The DMACA does not proceed to the next block transfer until a write to the block interrupt clear register, ClearBlock[n], is performed by software. This clears the channel block complete interrupt.

For rows 2, 3, 4, 7, and 9 of Table 20-1 on page 317 (SARx and/or DARx auto-reloaded between block transfers), the DMA transfer does not stall if either:

- interrupts are disabled, CTLx.INT_EN = 0, or
- the channel block interrupt is masked, MaskBlock[n] = 1, where n is the channel number.

Channel suspension between blocks is used to ensure that the end of block ISR (interrupt service routine) of the next-to-last block is serviced before the start of the final block commences. This ensures that the ISR has cleared the CFGx.RELOAD_SR and/or CFGx.RELOAD_DS bits before completion of the final block. The reload bits CFGx.RELOAD_SR and/or CFGx.RELOAD_DS should be cleared in the ‘end of block ISR’ for the next-to-last block transfer.

20.9.2 Ending Multi-block Transfers

All multi-block transfers must end as shown in either Row 1 or Row 5 of Table 20-1 on page 317. At the end of every block transfer, the DMACA samples the row number, and if the DMACA is in Row 1 or Row 5 state, then the previous block transferred was the last block and the DMA transfer is terminated.

Note: Row 1 and Row 5 are used for single block transfers or terminating multiblock transfers. Ending in Row 5 state enables status fetch for the last block. Ending in Row 1 state disables status fetch for the last block.

For rows 2,3 and 4 of Table 20-1 on page 317, (LLPx = 0 and CFGx.RELOAD_SR and/or CFGx.RELOAD_DS is set), multi-block DMA transfers continue until both the CFGx.RELOAD_SR and CFGx.RELOAD_DS registers are cleared by software. They should be

programmed to zero in the end of block interrupt service routine that services the next-to-last block transfer. This puts the DMACA into Row 1 state.

For rows 6, 8, and 10 (both CFGx.RELOAD_SR and CFGx.RELOAD_DS cleared) the user must setup the last block descriptor in memory such that both LLI.CTLx.LLP_S_EN and LLI.CTLx.LLP_D_EN are zero. If the LLI.LLPx register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the LLI.LLPx register of the last block descriptor in memory is zero, then the DMA transfer is terminated in Row 1.

For rows 7 and 9, the end-of-block interrupt service routine that services the next-to-last block transfer should clear the CFGx.RELOAD_SR and CFGx.RELOAD_DS reload bits. The last block descriptor in memory should be set up so that both the LLI.CTLx.LLP_S_EN and LLI.CTLx.LLP_D_EN are zero. If the LLI.LLPx register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the LLI.LLPx register of the last block descriptor in memory is zero, then the DMA transfer is terminated in Row 1.

Note: The only allowed transitions between the rows of Table 20-1 on page 317are from any row into row 1 or row 5. As already stated, a transition into row 1 or row 5 is used to terminate the DMA transfer. All other transitions between rows are not allowed. Software must ensure that illegal transitions between rows do not occur between blocks of a multi-block transfer. For example, if block N is in row 10 then the only allowed rows for block N + 1 are rows 10, 5 or 1.

20.10 Programming a Channel

Three registers, the LLPx, the CTLx and CFGx, need to be programmed to set up whether single or multi-block transfers take place, and which type of multi-block transfer is used. The different transfer types are shown in Table 20-1 on page 317.

The “Update Method” column indicates where the values of SARx, DARx, CTLx, and LLPx are obtained for the next block transfer when multi-block DMACA transfers are enabled.

Note: In Table 20-1 on page 317, all other combinations of LLPx.LOC = 0, CTLx.LLP_S_EN, CFGx.RELOAD_SR, CTLx.LLP_D_EN, and CFGx.RELOAD_DS are illegal, and causes indeterminate or erroneous behavior.

20.10.1 Programming Examples

20.10.1.1 Single-block Transfer (Row 1)

Row 5 in Table 20-1 on page 317 is also a single block transfer.

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:
   a. Write the starting source address in the SARx register for channel x.
   b. Write the starting destination address in the DARx register for channel x.
   c. Program CTLx and CFGx according to Row 1 as shown in Table 20-1 on page 317. Program the LLPx register with ‘0’.
   d. Write the control information for the DMA transfer in the CTLx register for channel x. For example, in the register, you can program the following:
   –i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register.

–ii. Set up the transfer characteristics, such as:

– Transfer width for the source in the SRC_TR_WIDTH field.

– Transfer width for the destination in the DST_TR_WIDTH field.

– Source master layer in the SMS field where source resides.

– Destination master layer in the DMS field where destination resides.

– Incrementing/decrementing or fixed address for source in SINC field.

– Incrementing/decrementing or fixed address for destination in DINC field.

e. Write the channel configuration information into the CFGx register for channel x.

–i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests. Writing a '1' activates the software handshaking interface to handle source/destination requests.

–ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.

4. After the DMACA selected channel has been programmed, enable the channel by writing a '1' to the ChEnReg.CH_EN bit. Make sure that bit 0 of the DmaCfgReg register is enabled.

5. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripherals). The DMACA acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.

6. Once the transfer completes, hardware sets the interrupts and disables the channel. At this time you can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (ChEnReg.CH_EN) bit until it is cleared by hardware, to detect when the transfer is complete.

20.10.1.2 Multi-block Transfer with Linked List for Source and Linked List for Destination (Row 10)

1. Read the Channel Enable register to choose a free (disabled) channel.

2. Set up the chain of Linked List Items (otherwise known as block descriptors) in memory. Write the control information in the LLI.CTLx register location of the block descriptor for each LLI in memory (see Figure 20-7 on page 316) for channel x. For example, in the register, you can program the following:

a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register.

b. Set up the transfer characteristics, such as:

–i. Transfer width for the source in the SRC_TR_WIDTH field.

–ii. Transfer width for the destination in the DST_TR_WIDTH field.

–iii. Source master layer in the SMS field where source resides.

–iv. Destination master layer in the DMS field where destination resides.

–v. Incrementing/decrementing or fixed address for source in SINC field.

–vi. Incrementing/decrementing or fixed address for destination DINC field.

3. Write the channel configuration information into the CFGx register for channel x.

a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires pro-

gramming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a ‘0’ activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a ‘1’ activates the software handshaking interface to handle source/destination requests.

b. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.

4. Make sure that the LLI.CTLx register locations of all LLI entries in memory (except the last) are set as shown in Row 10 of Table 20-1 on page 317. The LLI.CTLx register of the last Linked List Item must be set as described in Row 1 or Row 5 of Table 20-1 on page 317. Figure 20-9 on page 323 shows a Linked List example with two list items.
5. Make sure that the LLI.LLPx register locations of all LLI entries in memory (except the last) are non-zero and point to the base address of the next Linked List Item.
6. Make sure that the LLI.SARx/LLI.DARx register locations of all LLI entries in memory point to the start source/destination block address preceding that LLI fetch.
7. Make sure that the LLI.CTLx.DONE field of the LLI.CTLx register locations of all LLI entries in memory are cleared.
8. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
9. Program the CTLx, CFGx registers according to Row 10 as shown in Table 20-1 on page 317.
10. Program the LLPx register with LLPx(0), the pointer to the first Linked List item.
11. Finally, enable the channel by writing a ‘1’ to the ChEnReg.CH_EN bit. The transfer is performed.
12. The DMACA fetches the first LLI from the location pointed to by LLPx(0).

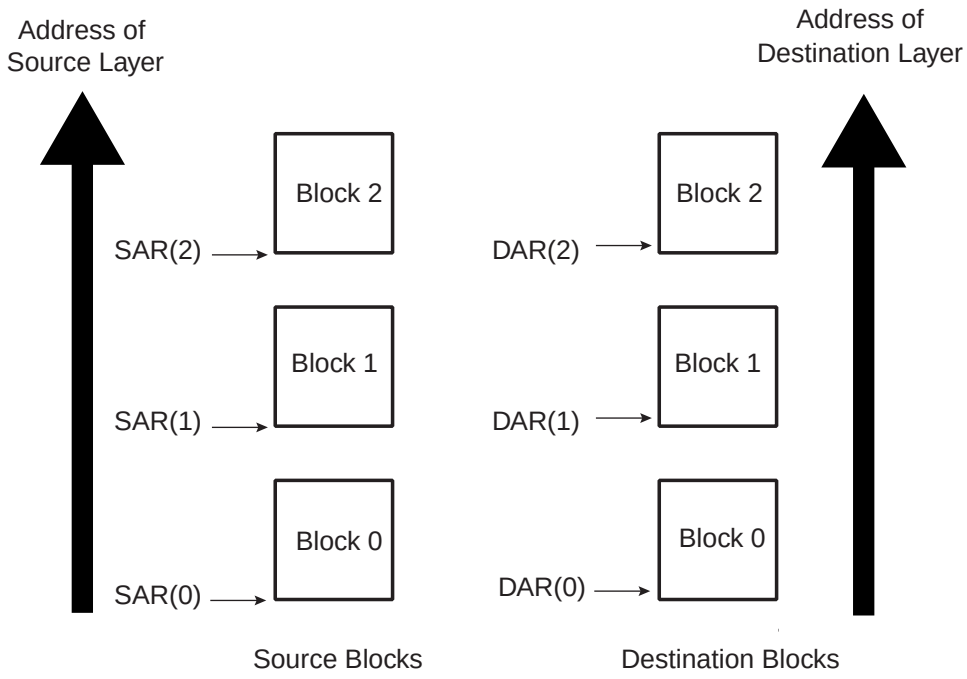
Note: The LLI.SARx, LLI. DARx, LLI.LLPx and LLI.CTLx registers are fetched. The DMACA automatically reprograms the SARx, DARx, LLPx and CTLx channel registers from the LLPx(0).

13. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripheral). The DMACA acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.

Note: Table 20-1 on page 317

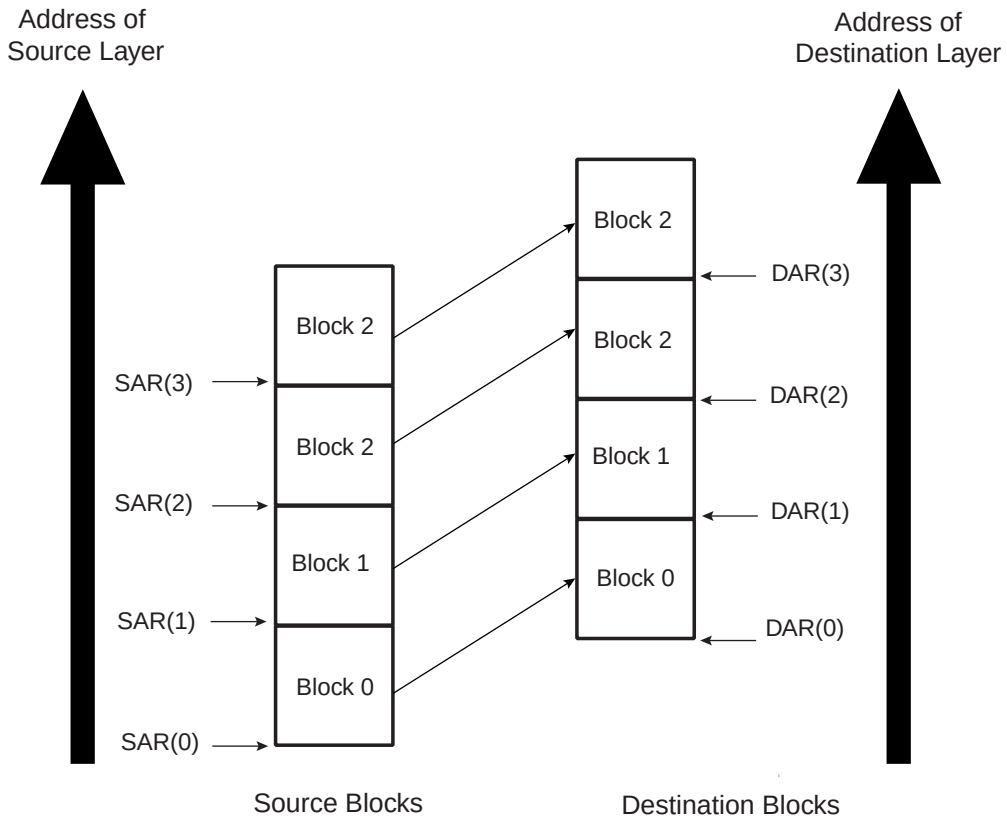
14. The DMACA does not wait for the block interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current LLPx register and automatically reprograms the SARx, DARx, LLPx and CTLx channel registers. The DMA transfer continues until the DMACA determines that the CTLx and LLPx registers at the end of a block transfer match that described in Row 1 or Row 5 of Table 20-1 on page 317. The DMACA then knows that the previous block transferred was the last block in the DMA transfer. The DMA transfer might look like that shown in Figure 20-8 on page 322.

Figure 20-8. Multi-Block with Linked List Address for Source and Destination



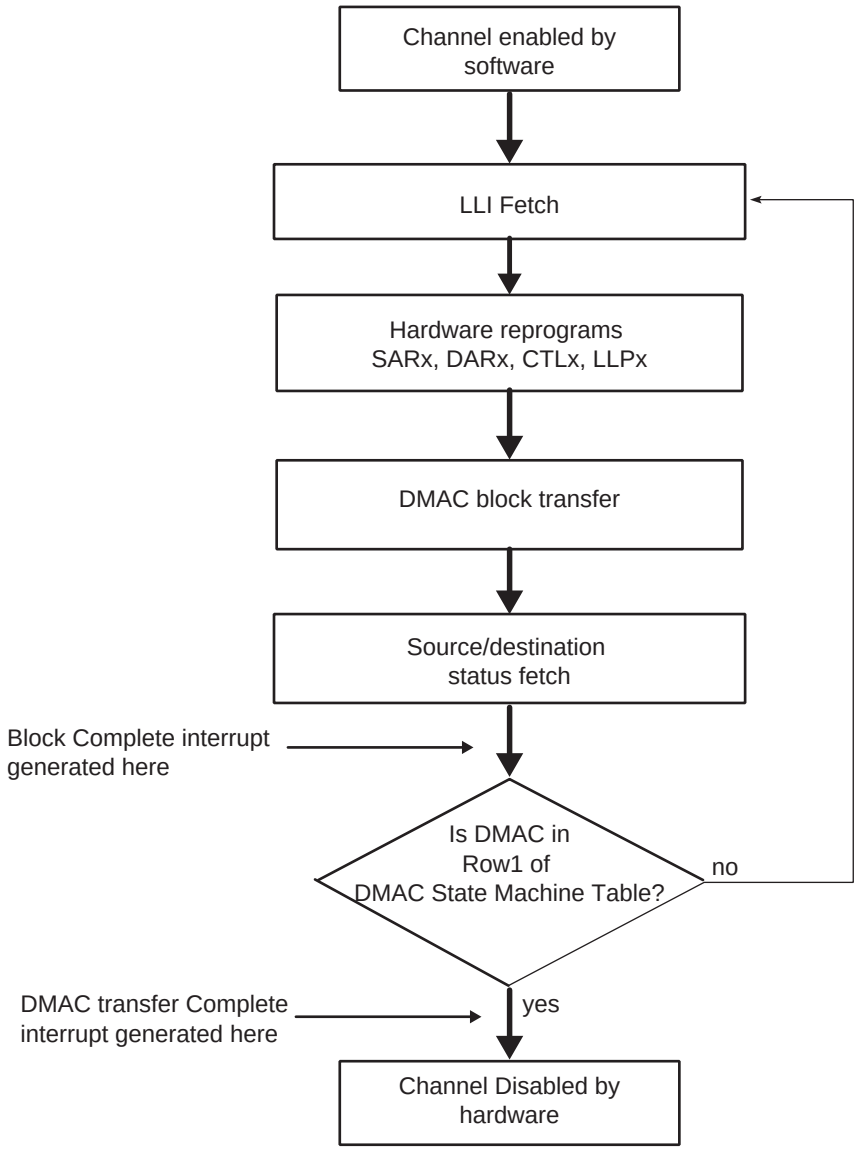
If the user needs to execute a DMA transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum block size CTLx.BLOCK_TS, then this can be achieved using the type of multi-block transfer as shown in Figure 20-9 on page 323.

Figure 20-9. Multi-Block with Linked Address for Source and Destination Blocks are Contiguous



The DMA transfer flow is shown in Figure 20-11 on page 326.

Figure 20-10. DMA Transfer Flow for Source and Destination Linked List Address



20.10.1.3 Multi-block Transfer with Source Address Auto-reloaded and Destination Address Auto-reloaded (Row 4)

1. Read the Channel Enable register to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:

a. Write the starting source address in the SARx register for channel x.

b. Write the starting destination address in the DARx register for channel x.

c. Program CTLx and CFGx according to Row 4 as shown in Table 20-1 on page 317. Program the LLPx register with ‘0’.

d. Write the control information for the DMA transfer in the CTLx register for channel x. For example, in the register, you can program the following:

–i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register.

–ii. Set up the transfer characteristics, such as:

– Transfer width for the source in the SRC_TR_WIDTH field.

– Transfer width for the destination in the DST_TR_WIDTH field.

– Source master layer in the SMS field where source resides.

– Destination master layer in the DMS field where destination resides.

– Incrementing/decrementing or fixed address for source in SINC field.

– Incrementing/decrementing or fixed address for destination in DINC field.

e. Write the channel configuration information into the CFGx register for channel x. Ensure that the reload bits, CFGx. RELOAD_SR and CFGx.RELOAD_DS are enabled.

–i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a ‘0’ activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a ‘1’ activates the software handshaking interface to handle source/destination requests.

–ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.

4. After the DMACA selected channel has been programmed, enable the channel by writing a ‘1’ to the ChEnReg.CH_EN bit. Make sure that bit 0 of the DmaCfgReg register is enabled.

5. Source and destination request single and burst DMACA transactions to transfer the block of data (assuming non-memory peripherals). The DMACA acknowledges on completion of each burst/single transaction and carry out the block transfer.

6. When the block transfer has completed, the DMACA reloads the SARx, DARx and CTLx registers. Hardware sets the Block Complete interrupt. The DMACA then samples the row number as shown in Table 20-1 on page 317. If the DMACA is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. So you can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (ChEnReg.CH_EN) bit until it is disabled, to detect when the transfer is complete. If the DMACA is not in Row 1, the next step is performed.

7. The DMA transfer proceeds as follows:

a. If interrupts are enabled (CTLx.INT_EN = 1) and the block complete interrupt is unmasked (MaskBlock[x] = 1’b1, where x is the channel number) hardware sets the block complete interrupt when the block transfer has completed. It then stalls until the block complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block complete ISR (interrupt service routine)

should clear the reload bits in the CFGx.RELOAD_SR and CFGx.RELOAD_DS registers. This put the DMACA into Row 1 as shown in Table 20-1 on page 317. If the next block is not the last block in the DMA transfer, then the reload bits should remain enabled to keep the DMACA in Row 4.

b. If interrupts are disabled (CTLx.INT_EN = 0) or the block complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number), then hardware does not stall until it detects a write to the block complete interrupt clear register but starts the next block transfer immediately. In this case software must clear the reload bits in the CFGx.RELOAD_SR and CFGx.RELOAD_DS registers to put the DMACA into ROW 1 of Table 20-1 on page 317 before the last block of the DMA transfer has completed. The transfer is similar to that shown in Figure 20-11 on page 326. The DMA transfer flow is shown in Figure 20-12 on page 327.

Figure 20-11. Multi-Block DMA Transfer with Source and Destination Address Auto-reloaded

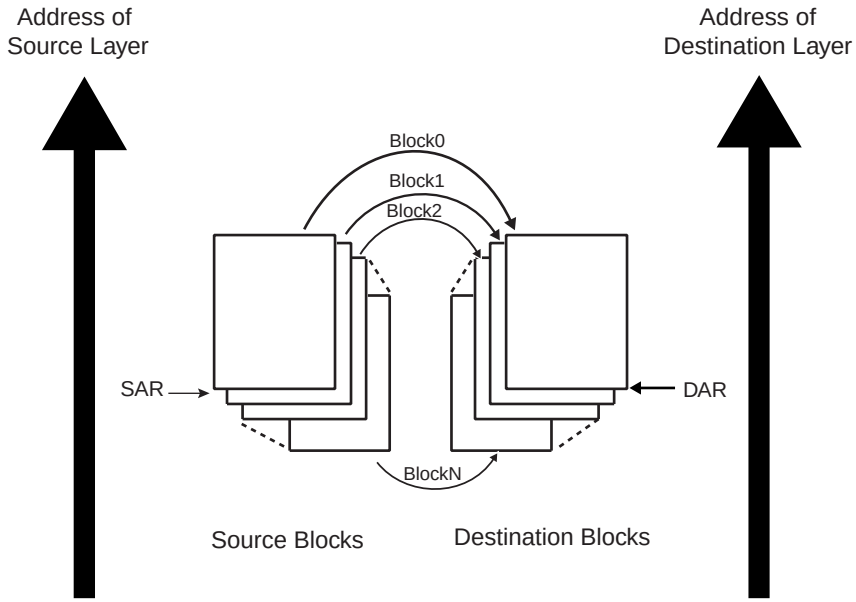
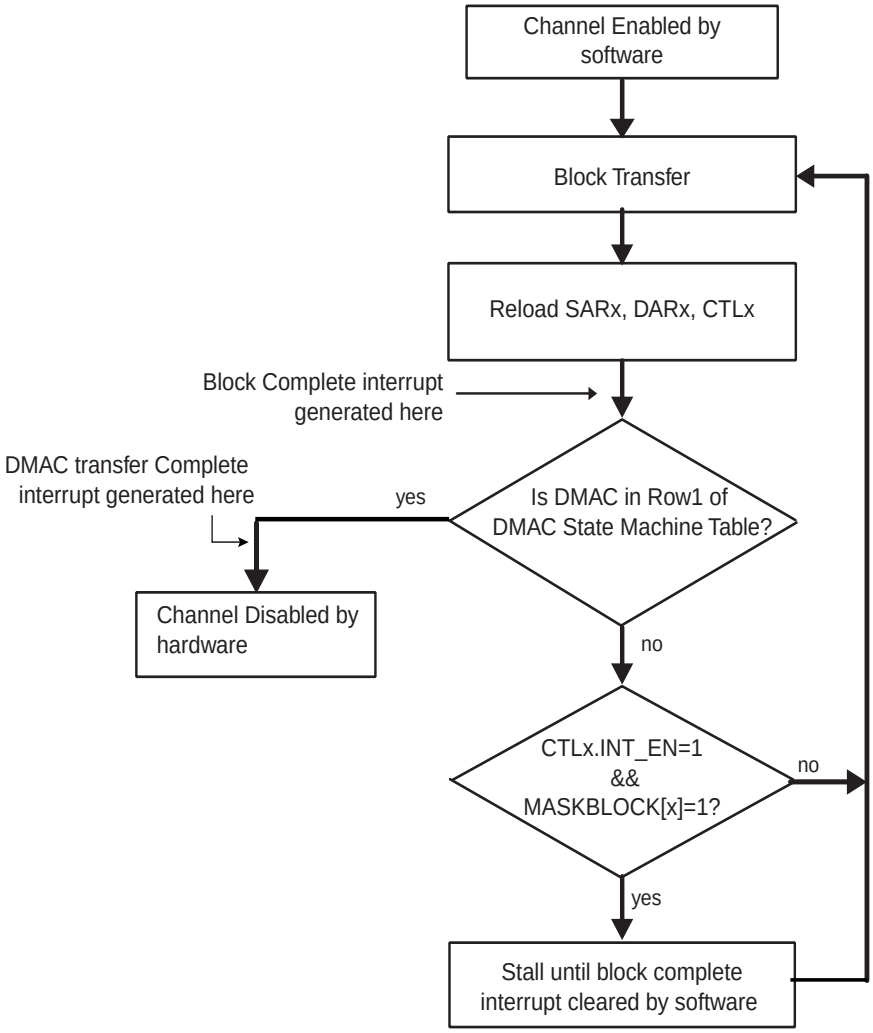


Figure 20-12. DMA Transfer Flow for Source and Destination Address Auto-reloaded



20.10.1.4 Multi-block Transfer with Source Address Auto-reloaded and Linked List Destination Address (Row7)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as block descriptors) in memory. Write the control information in the LLI.CTLx register location of the block descriptor for each LLI in memory for channel x. For example, in the register you can program the following:
   a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control peripheral by programming the TT_FC of the CTLx register.
   b. Set up the transfer characteristics, such as:
   –i. Transfer width for the source in the SRC_TR_WIDTH field.
   –ii. Transfer width for the destination in the DST_TR_WIDTH field.
   –iii. Source master layer in the SMS field where source resides.
   –iv. Destination master layer in the DMS field where destination resides.
   –v. Incrementing/decrementing or fixed address for source in SINC field.
   –vi. Incrementing/decrementing or fixed address for destination DINC field.

3. Write the starting source address in the SARx register for channel x.

Note: The values in the LLI.SARx register locations of each of the Linked List Items (LLIs) setup up in memory, although fetched during a LLI fetch, are not used.

4. Write the channel configuration information into the CFGx register for channel x.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' activates the software handshaking interface source/destination requests.
   b. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
5. Make sure that the LLI.CTLx register locations of all LLIs in memory (except the last) are set as shown in Row 7 of Table 20-1 on page 317 while the LLI.CTLx register of the last Linked List item must be set as described in Row 1 or Row 5 of Table 20-1 on page 317. Figure 20-7 on page 316 shows a Linked List example with two list items.
6. Make sure that the LLI.LLPx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Make sure that the LLI.DARx register location of all LLIs in memory point to the start destination block address proceeding that LLI fetch.
8. Make sure that the LLI.CTLx.DONE field of the LLI.CTLx register locations of all LLIs in memory is cleared.
9. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
10. Program the CTLx, CFGx registers according to Row 7 as shown in Table 20-1 on page 317.
11. Program the LLPx register with LLPx(0), the pointer to the first Linked List item.
12. Finally, enable the channel by writing a '1' to the ChEnReg.CH_EN bit. The transfer is performed. Make sure that bit 0 of the DmaCfgReg register is enabled.
13. The DMACA fetches the first LLI from the location pointed to by LLPx(0).

Note: The LLI.SARx, LLI.DARx, LLI. LLPx and LLI.CTLx registers are fetched. The LLI.SARx register although fetched is not used.

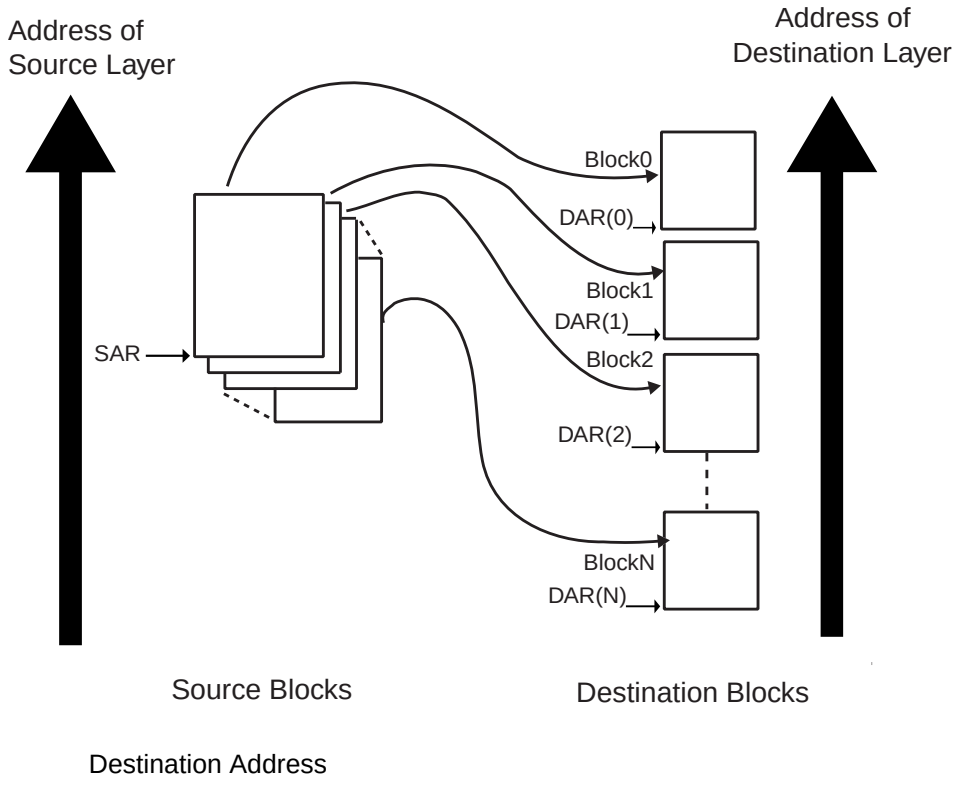
14. Source and destination request single and burst DMACA transactions to transfer the block of data (assuming non-memory peripherals). DMACA acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.
15. Table 20-1 on page 317The DMACA reloads the SARx register from the initial value. Hardware sets the block complete interrupt. The DMACA samples the row number as shown in Table 20-1 on page 317. If the DMACA is in Row 1 or 5, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (ChEnReg.CH_EN) bit until it is cleared by hardware, to detect when the transfer is complete. If the DMACA is not in Row 1 or 5 as shown in Table 20-1 on page 317 the following steps are performed.
16. The DMA transfer proceeds as follows:
   a. If interrupts are enabled (CTLx.INT_EN = 1) and the block complete interrupt is unmasked (MaskBlock[x] = 1'b1, where x is the channel number) hardware sets the

block complete interrupt when the block transfer has completed. It then stalls until the block complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block complete ISR (interrupt service routine) should clear the CFGx.RELOAD_SR source reload bit. This puts the DMACA into Row1 as shown in Table 20-1 on page 317. If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the DMACA in Row 7 as shown in Table 20-1 on page 317.

b. If interrupts are disabled (CTLx.INT_EN = 0) or the block complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number) then hardware does not stall until it detects a write to the block complete interrupt clear register but starts the next block transfer immediately. In this case, software must clear the source reload bit, CFGx.RELOAD_SR, to put the device into Row 1 of Table 20-1 on page 317 before the last block of the DMA transfer has completed.

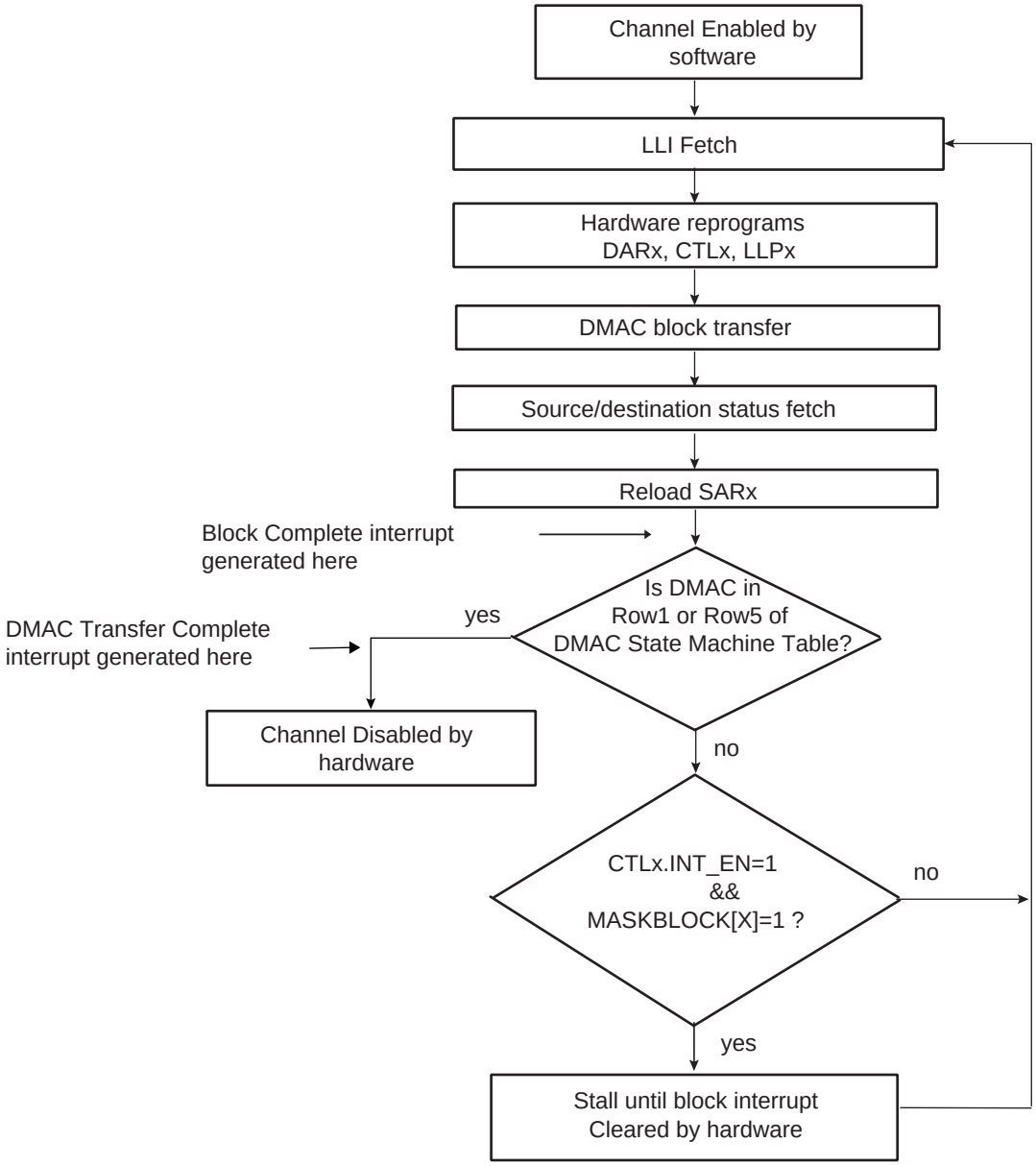
17. The DMACA fetches the next LLI from memory location pointed to by the current LLPx register, and automatically reprograms the DARx, CTLx and LLPx channel registers. Note that the SARx is not re-programmed as the reloaded value is used for the next DMA block transfer. If the next block is the last block of the DMA transfer then the CTLx and LLPx registers just fetched from the LLI should match Row 1 or Row 5 of Table 20-1 on page 317. The DMA transfer might look like that shown in Figure 20-13 on page 329.

Figure 20-13. Multi-Block DMA Transfer with Source Address Auto-reloaded and Linked List



The DMA Transfer flow is shown in Figure 20-14 on page 330.

Figure 20-14. DMA Transfer Flow for Source Address Auto-reloaded and Linked List Destination Address



20.10.1.5 Multi-block Transfer with Source Address Auto-reloaded and Contiguous Destination Address (Row 3)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing a '1' to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:
   a. Write the starting source address in the SARx register for channel x.
   b. Write the starting destination address in the DARx register for channel x.
   c. Program CTLx and CFGx according to Row 3 as shown in Table 20-1 on page 317. Program the LLPx register with '0'.
   d. Write the control information for the DMA transfer in the CTLx register for channel x. For example, in this register, you can program the following:
   –i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register.
   –ii. Set up the transfer characteristics, such as:
      – Transfer width for the source in the SRC_TR_WIDTH field.
      – Transfer width for the destination in the DST_TR_WIDTH field.
      – Source master layer in the SMS field where source resides.
      – Destination master layer in the DMS field where destination resides.
      – Incrementing/decrementing or fixed address for source in SINC field.
      – Incrementing/decrementing or fixed address for destination in DINC field.
   e. Write the channel configuration information into the CFGx register for channel x.
   –i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' activates the software handshaking interface to handle source/destination requests.
   –ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
4. After the DMACA channel has been programmed, enable the channel by writing a '1' to the ChEnReg.CH_EN bit. Make sure that bit 0 of the DmaCfgReg register is enabled.
5. Source and destination request single and burst DMACA transactions to transfer the block of data (assuming non-memory peripherals). The DMACA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
6. When the block transfer has completed, the DMACA reloads the SARx register. The DARx register remains unchanged. Hardware sets the block complete interrupt. The DMACA then samples the row number as shown in Table 20-1 on page 317. If the DMACA is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. So you can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (ChEn-

Reg.CH_EN) bit until it is cleared by hardware, to detect when the transfer is complete. If the DMACA is not in Row 1, the next step is performed.

7. The DMA transfer proceeds as follows:
   a. If interrupts are enabled (CTLx.INT_EN = 1) and the block complete interrupt is unmasked (MaskBlock[x] = 1'b1, where x is the channel number) hardware sets the block complete interrupt when the block transfer has completed. It then stalls until the block complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block complete ISR (interrupt service routine) should clear the source reload bit, CFGx.RELOAD_SR. This puts the DMACA into Row1 as shown in Table 20-1 on page 317. If the next block is not the last block in the DMA transfer then the source reload bit should remain enabled to keep the DMACA in Row3 as shown in Table 20-1 on page 317.
   b. If interrupts are disabled (CTLx.INT_EN = 0) or the block complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number) then hardware does not stall until it detects a write to the block complete interrupt clear register but starts the next block transfer immediately. In this case software must clear the source reload bit, CFGx.RELOAD_SR, to put the device into ROW 1 of Table 20-1 on page 317 before the last block of the DMA transfer has completed.

The transfer is similar to that shown in Figure 20-15 on page 332.

The DMA Transfer flow is shown in Figure 20-16 on page 333.

Figure 20-15. Multi-block Transfer with Source Address Auto-reloaded and Contiguous Destination Address

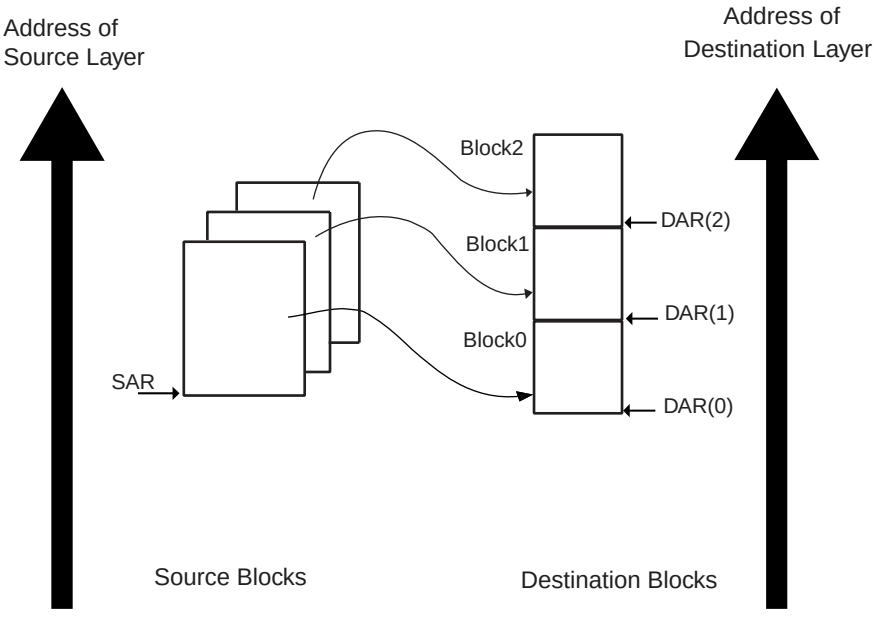
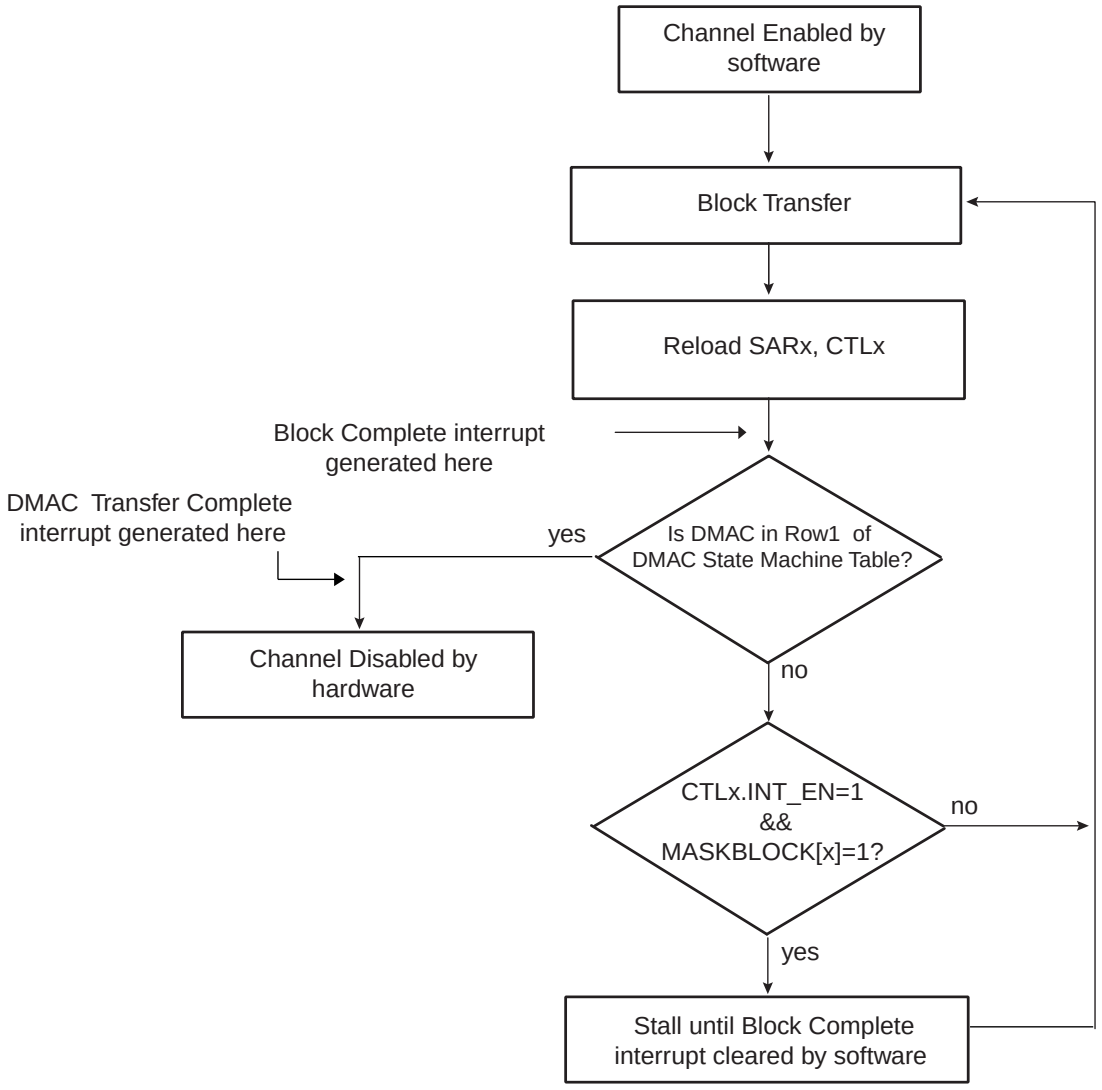


Figure 20-16. DMA Transfer for Source Address Auto-reloaded and Contiguous Destination Address



20.10.1.6 Multi-block DMA Transfer with Linked List for Source and Contiguous Destination Address (Row 8)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the linked list in memory. Write the control information in the LLI. CTLx register location of the block descriptor for each LLI in memory for channel x. For example, in the register, you can program the following:
   a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register.
   b. Set up the transfer characteristics, such as:
   –i. Transfer width for the source in the SRC_TR_WIDTH field.
   –ii. Transfer width for the destination in the DST_TR_WIDTH field.
   –iii. Source master layer in the SMS field where source resides.
   –iv. Destination master layer in the DMS field where destination resides.

–v. Incrementing/decrementing or fixed address for source in SINC field.

–vi. Incrementing/decrementing or fixed address for destination DINC field.

3. Write the starting destination address in the DARx register for channel x.

Note: The values in the LLI.DARx register location of each Linked List Item (LLI) in memory, although fetched during an LLI fetch, are not used.

4. Write the channel configuration information into the CFGx register for channel x.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' activates the software handshaking interface to handle source/destination requests.
   b. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripherals. This requires programming the SRC_PER and DEST_PER bits, respectively.
5. Make sure that all LLI.CTLx register locations of the LLI (except the last) are set as shown in Row 8 of Table 20-1 on page 317, while the LLI.CTLx register of the last Linked List item must be set as described in Row 1 or Row 5 of Table 20-1 on page 317. Figure 20-7 on page 316 shows a Linked List example with two list items.
6. Make sure that the LLI.LLPx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Make sure that the LLI.SARx register location of all LLIs in memory point to the start source block address proceeding that LLI fetch.
8. Make sure that the LLI.CTLx.DONE field of the LLI.CTLx register locations of all LLIs in memory is cleared.
9. Clear any pending interrupts on the channel from the previous DMA transfer by writing a '1' to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
10. Program the CTLx, CFGx registers according to Row 8 as shown in Table 20-1 on page 317
11. Program the LLPx register with LLPx(0), the pointer to the first Linked List item.
12. Finally, enable the channel by writing a '1' to the ChEnReg.CH_EN bit. The transfer is performed. Make sure that bit 0 of the DmaCfgReg register is enabled.
13. The DMACA fetches the first LLI from the location pointed to by LLPx(0).

Note: The LLI.SARx, LLI.DARx, LLI.LLPx and LLI.CTLx registers are fetched. The LLI.DARx register location of the LLI although fetched is not used. The DARx register in the DMACA remains unchanged.

14. Source and destination requests single and burst DMACA transactions to transfer the block of data (assuming non-memory peripherals). The DMACA acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.

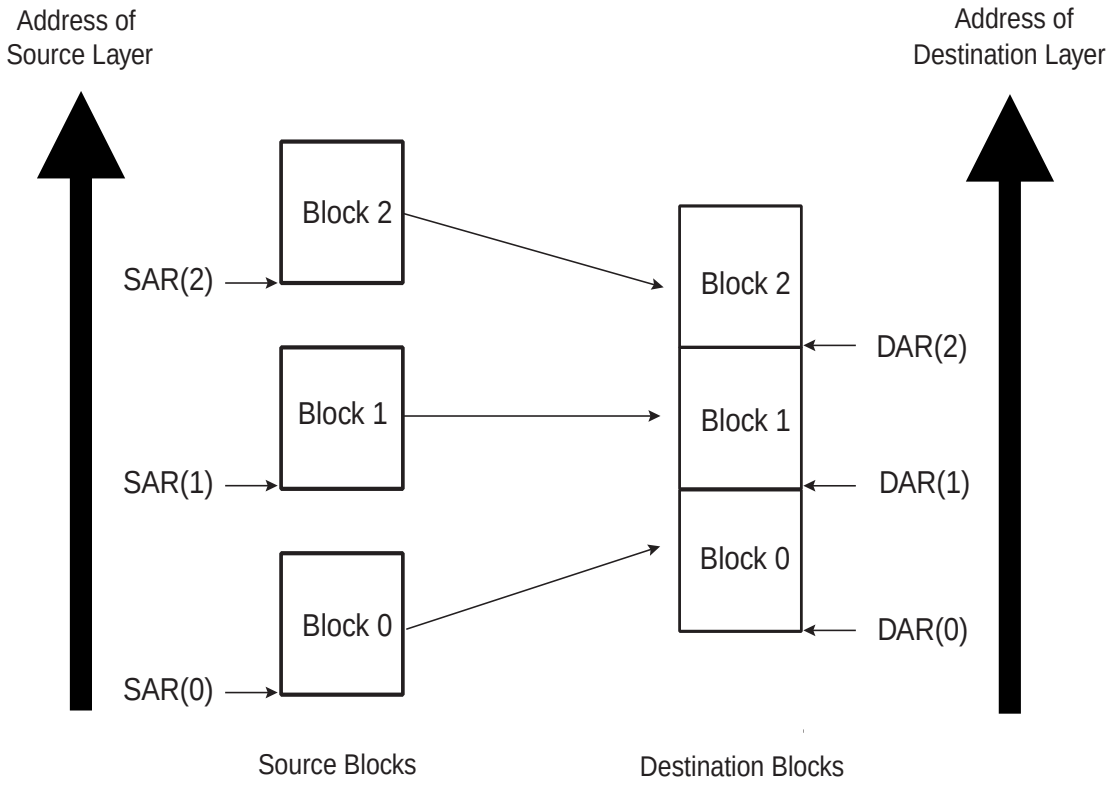
Note:

15. The DMACA does not wait for the block interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by current LLPx register and automatically reprograms the SARx, CTLx and LLPx channel registers. The DARx register is left unchanged. The DMA transfer continues until the DMACA samples the CTLx and LLPx registers at the end of a block transfer match that described in Row 1 or Row

5 of Table 20-1 on page 317. The DMACA then knows that the previous block transferred was the last block in the DMA transfer.

The DMACA transfer might look like that shown in Figure 20-17 on page 335 Note that the destination address is decrementing.

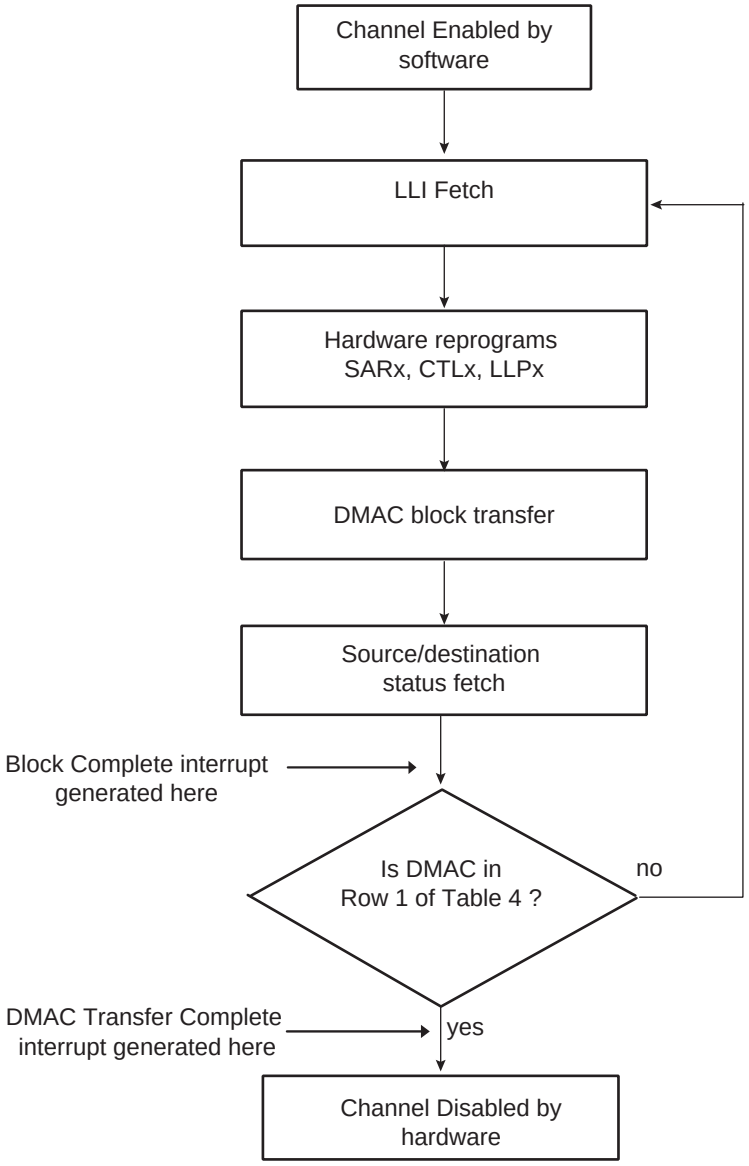
Figure 20-17. DMA Transfer with Linked List Source Address and Contiguous Destination Address



The DMA transfer flow is shown in Figure 20-19 on page 336.

Figure 20-18.

Figure 20-19. DMA Transfer Flow for Source Address Auto-reloaded and Contiguous Destination Address



20.11 Disabling a Channel Prior to Transfer Completion

Under normal operation, software enables a channel by writing a '1' to the Channel Enable Register, ChEnReg.CH_EN, and hardware disables a channel on transfer completion by clearing the ChEnReg.CH_EN register bit.

The recommended way for software to disable a channel without losing data is to use the CH_SUSP bit in conjunction with the FIFO_EMPTY bit in the Channel Configuration Register (CFGx) register.

1. If software wishes to disable a channel prior to the DMA transfer completion, then it can set the CFGx.CH_SUSP bit to tell the DMACA to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
2. Software can now poll the CFGx.FIFO_EMPTY bit until it indicates that the channel FIFO is empty.

3. The ChEnReg.CH_EN bit can then be cleared by software once the channel FIFO is empty.

When CTLx.SRC_TR_WIDTH is less than CTLx.DST_TR_WIDTH and the CFGx.CH_SUSP bit is high, the CFGx.FIFO_EMPTY is asserted once the contents of the FIFO do not permit a single word of CTLx.DST_TR_WIDTH to be formed. However, there may still be data in the channel FIFO but not enough to form a single transfer of CTLx.DST_TR_WIDTH width. In this configuration, once the channel is disabled, the remaining data in the channel FIFO are not transferred to the destination peripheral. It is permitted to remove the channel from the suspension state by writing a '0' to the CFGx.CH_SUSP register. The DMA transfer completes in the normal manner.

Note: If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.

20.11.1 Abnormal Transfer Termination

A DMACA DMA transfer may be terminated abruptly by software by clearing the channel enable bit, ChEnReg.CH_EN. This does not mean that the channel is disabled immediately after the ChEnReg.CH_EN bit is cleared over the HSB slave interface. Consider this as a request to disable the channel. The ChEnReg.CH_EN must be polled and then it must be confirmed that the channel is disabled by reading back 0. A case where the channel is not be disabled after a channel disable request is where either the source or destination has received a split or retry response. The DMACA must keep re-attempting the transfer to the system HADDR that originally received the split or retry response until an OKAY response is returned. To do otherwise is an System Bus protocol violation.

Software may terminate all channels abruptly by clearing the global enable bit in the DMACA Configuration Register (DmaCfgReg[0]). Again, this does not mean that all channels are disabled immediately after the DmaCfgReg[0] is cleared over the HSB slave interface. Consider this as a request to disable all channels. The ChEnReg must be polled and then it must be confirmed that all channels are disabled by reading back '0'.

Note: If the channel enable bit is cleared while there is data in the channel FIFO, this data is not sent to the destination peripheral and is not present when the channel is re-enabled. For read sensitive source peripherals such as a source FIFO this data is therefore lost. When the source is not a read sensitive device (i.e., memory), disabling a channel without waiting for the channel FIFO to empty may be acceptable as the data is available from the source peripheral upon request and is not lost.

Note: If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.

20.12 User Interface

Table 20-2. DMA Controller Memory Map

Offset	Register	Register Name	Access	Reset Value
0x000	Channel 0 Source Address Register	SAR0	Read/Write	0x00000000
0x008	Channel 0 Destination Address Register	DAR0	Read/Write	0x00000000
0x010	Channel 0 Linked List Pointer Register	LLP0	Read/Write	0x00000000
0x018	Channel 0 Control Register Low	CTL0L	Read/Write	0x00304801
0x01C	Channel 0 Control Register High	CTL0H	Read/Write	0x00000002
0x040	Channel 0 Configuration Register Low	CFG0L	Read/Write	0x00000c00
0x044	Channel 0 Configuration Register High	CFG0H	Read/Write	0x00000004
0x048	Channel 0 Source Gather Register	SGR0	Read/Write	0x00000000
0x050	Channel 0 Destination Scatter Register	DSR0	Read/Write	0x00000000
0x058	Channel 1 Source Address Register	SAR1	Read/Write	0x00000000
0x060	Channel 1 Destination Address Register	DAR1	Read/Write	0x00000000
0x068	Channel 1 Linked List Pointer Register	LLP1	Read/Write	0x00000000
0x070	Channel 1 Control Register Low	CTL1L	Read/Write	0x00304801
0x074	Channel 1 Control Register High	CTL1H	Read/Write	0x00000002
0x098	Channel 1 Configuration Register Low	CFG1L	Read/Write	0x00000c20
0x09C	Channel 1 Configuration Register High	CFG1H	Read/Write	0x00000004
0x0A0	Channel 1Source Gather Register	SGR1	Read/Write	0x00000000
0x0A8	Channel 1 Destination Scatter Register	DSR1	Read/Write	0x00000000
0x0B0	Channel 2 Source Address Register	SAR2	Read/Write	0x00000000
0x0B8	Channel 2 Destination Address Register	DAR2	Read/Write	0x00000000
0x0C0	Channel 2 Linked List Pointer Register	LLP2	Read/Write	0x00000000
0x0C8	Channel 2 Control Register Low	CTL2L	Read/Write	0x00304801
0x0CC	Channel 2 Control Register High	CTL2H	Read/Write	0x00000002
0x0F0	Channel 2 Configuration Register Low	CFG2L	Read/Write	0x00000c40
0x0F4	Channel 2 Configuration Register High	CFG2H	Read/Write	0x00000004
0x0F8	Channel 2 Source Gather Register	SGR2	Read/Write	0x00000000
0x100	Channel 2 Destination Scatter Register	DSR2	Read/Write	0x00000000
0x108	Channel 3 Source Address Register	SAR3	Read/Write	0x00000000
0x110	Channel 3 Destination Address Register	DAR3	Read/Write	0x00000000
0x118	Channel 3 Linked List Pointer Register	LLP3	Read/Write	0x00000000
0x120	Channel 3 Control Register Low	CTL3L	Read/Write	0x00304801
0x124	Channel 3 Control Register High	CTL3H	Read/Write	0x00000002
0x148	Channel 3 Configuration Register Low	CFG3L	Read/Write	0x00000c60
0x14c	Channel 3 Configuration Register High	CFG3H	Read/Write	0x00000004
0x150	Channel 3 Source Gather Register	SGR3	Read/Write	0x00000000

Table 20-2. DMA Controller Memory Map (Continued)

Offset	Register	Register Name	Access	Reset Value
0x158	Channel 3Destination Scatter Register	DSR3	Read/Write	0x00000000
0x2C0	Raw Status for IntTfr Interrupt	RawTfr	Read-only	0x00000000
0x2C8	Raw Status for IntBlock Interrupt	RawBlock	Read-only	0x00000000
0x2D0	Raw Status for IntSrcTran Interrupt	RawSrcTran	Read-only	0x00000000
0x2D8	Raw Status for IntDstTran Interrupt	RawDstTran	Read-only	0x00000000
0x2E0	Raw Status for IntErr Interrupt	RawErr	Read-only	0x00000000
0x2E8	Status for IntTfr Interrupt	StatusTfr	Read-only	0x00000000
0x2F0	Status for IntBlock Interrupt	StatusBlock	Read-only	0x00000000
0x2F8	Status for IntSrcTran Interrupt	StatusSrcTran	Read-only	0x00000000
0x300	Status for IntDstTran Interrupt	StatusDstTran	Read-only	0x00000000
0x308	Status for IntErr Interrupt	StatusErr	Read-only	0x00000000
0x310	Mask for IntTfr Interrupt	MaskTfr	Read/Write	0x00000000
0x318	Mask for IntBlock Interrupt	MaskBlock	Read/Write	0x00000000
0x320	Mask for IntSrcTran Interrupt	MaskSrcTran	Read/Write	0x00000000
0x328	Mask for IntDstTran Interrupt	MaskDstTran	Read/Write	0x00000000
0x330	Mask for IntErr Interrupt	MaskErr	Read/Write	0x00000000
0x338	Clear for IntTfr Interrupt	ClearTfr	Write-only	0x00000000
0x340	Clear for IntBlock Interrupt	ClearBlock	Write-only	0x00000000
0x348	Clear for IntSrcTran Interrupt	ClearSrcTran	Write-only	0x00000000
0x350	Clear for IntDstTran Interrupt	ClearDstTran	Write-only	0x00000000
0x358	Clear for IntErr Interrupt	ClearErr	Write-only	0x00000000
0x360	Status for each interrupt type	StatusInt	Read-only	0x00000000
0x368	Source Software Transaction Request Register	ReqSrcReg	Read/Write	0x00000000
0x370	Destination Software Transaction Request Register	ReqDstReg	Read/Write	0x00000000
0x378	Single Source Transaction Request Register	SglReqSrcReg	Read/Write	0x00000000
0x380	Single Destination Transaction Request Register	SglReqDstReg	Read/Write	0x00000000
0x388	Last Source Transaction Request Register	LstSrcReg	Read/Write	0x00000000
0x390	Last Destination Transaction Request Register	LstDstReg	Read/Write	0x00000000
0x398	DMA Configuration Register	DmaCfgReg	Read/Write	0x00000000
0x3A0	DMA Channel Enable Register	ChEnReg	Read/Write	0x00000000
0x3F8	DMA Component ID Register Low	DmaCompIdRegL	Read-only	0x44571110
0x3FC	DMA Component ID Register High	DmaCompIdRegH	Read-only	0x3230362A

20.12.1 Channel x Source Address Register

Name: SARx

Access Type: Read/Write

Offset: 0x000 + [x * 0x58]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SADD[31:24]							

23	22	21	20	19	18	17	16
SADD[23:16]							

15	14	13	12	11	10	9	8
SADD[15:8]							

7	6	5	4	3	2	1	0
SADD[7:0]							

- **SADD: Source Address of DMA transfer**

The starting System Bus source address is programmed by software before the DMA channel is enabled or by a LLI update before the start of the DMA transfer. As the DMA transfer is in progress, this register is updated to reflect the source address of the current System Bus transfer.

Updated after each source System Bus transfer. The SINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source System Bus transfer throughout the block transfer.

20.12.2 Channel x Destination Address Register

Name: DARx

Access Type: Read/Write

Offset: 0x008 + [x * 0x58]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DADD[31:24]							

23	22	21	20	19	18	17	16
DADD[23:16]							

15	14	13	12	11	10	9	8
DADD[15:8]							

7	6	5	4	3	2	1	0
DADD[7:0]							

- **DADD: Destination Address of DMA transfer**

The starting System Bus destination address is programmed by software before the DMA channel is enabled or by a LLI update before the start of the DMA transfer. As the DMA transfer is in progress, this register is updated to reflect the destination address of the current System Bus transfer.

Updated after each destination System Bus transfer. The DINC field in the CTLx register determines whether the address increments, decrements or is left unchanged on every destination System Bus transfer throughout the block transfer.

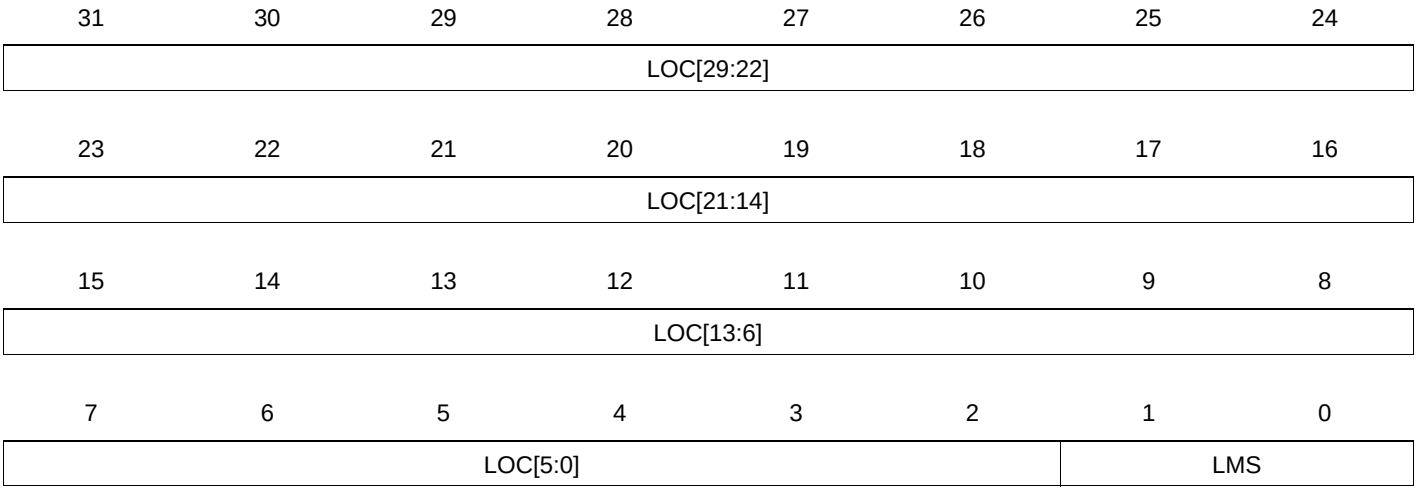
20.12.3 Linked List Pointer Register for Channel x

Name: LLPx

Access Type: Read/Write

Offset: 0x010 + [x * 0x58]

Reset Value: 0x00000000



- **LOC: Address of the next LLI**

Starting address in memory of next LLI if block chaining is enabled.

The user need to program this register to point to the first Linked List Item (LLI) in memory prior to enabling the channel if block chaining is enabled.

The LLP register has two functions:

The logical result of the equation LLP.LOC != 0 is used to set up the type of DMA transfer (single or multi-block).

If LLP.LOC is set to 0x0, then transfers using linked lists are NOT enabled. This register must be programmed prior to enabling the channel in order to set up the transfer type.

It (LLP.LOC != 0) contains the pointer to the next Linked Listed Item for block chaining using linked lists.

The LLPx register is also used to point to the address where write back of the control and source/destination status information occurs after block completion.

- **LMS: List Master Select**

Identifies the High speed bus interface for the device that stores the next linked list item:

Table 20-3. List Master Select

LMS	HSB Master
0	HSB master 1
1	HSB master 2
Other	Reserved

20.12.4 Control Register for Channel x Low

Name: CTLxL

Access Type: Read/Write

Offset: 0x018 + [x * 0x58]

Reset Value: 0x00304801

31	30	29	28	27	26	25	24
			LLP_SRC_EN	LLP_DST_EN	SMS		DMS[1]
23	22	21	20	19	18	17	16
DMS[0]	TT_FC				DST_GATHER_EN	SRC_GATHER_EN	SRC_MSIZE[2]
15	14	13	12	11	10	9	8
SRC_MSIZE[1:0]		DEST_MSIZE			SINC		DINC[1]
7	6	5	4	3	2	1	0
DINC[0]	SRC_TR_WIDTH			DST_TR_WIDTH			INT_EN

This register contains fields that control the DMA transfer. The CTLxL register is part of the block descriptor (linked list item) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled.

- **LLP_SRC_EN**

Block chaining is only enabled on the source side if the *LLP_SRC_EN* field is high and LLPx.LOC is non-zero.

- **LLP_DST_EN**

Block chaining is only enabled on the destination side if the *LLP_DST_EN* field is high and LLPx.LOC is non-zero.

- **SMS: Source Master Select**

Identifies the Master Interface layer where the source device (peripheral or memory) is accessed from

Table 20-4. Source Master Select

SMS	HSB Master
0	HSB master 1
1	HSB master 2
Other	Reserved

- **DMS: Destination Master Select**

Identifies the Master Interface layer where the destination device (peripheral or memory) resides

Table 20-5. Destination Master Select

DMS	HSB Master
0	HSB master 1
1	HSB master 2
Other	Reserved

- **TT_FC: Transfer Type and Flow Control**

The four following transfer types are supported:

• Memory to Memory, Memory to Peripheral, Peripheral to Memory and Peripheral to Peripheral.

The DMACA is always the Flow Controller.

TT_FC	Transfer Type	Flow Controller
000	Memory to Memory	DMACA
001	Memory to Peripheral	DMACA
010	Peripheral to Memory	DMACA
011	Peripheral to Peripheral	DMACA
Other	Reserved	Reserved

- **DST_SCATTER_EN: Destination Scatter Enable**

  0 = Scatter disabled

  1 = Scatter enabled

Scatter on the destination side is applicable only when the CTL*x*.DINC bit indicates an incrementing or decrementing address control.

- **SRC_GATHER_EN: Source Gather Enable**

  0 = Gather disabled

  1 = Gather enabled

Gather on the source side is applicable only when the CTL*x*.SINC bit indicates an incrementing or decrementing address control.

- **SRC_MSIZE: Source Burst Transaction Length**

Number of data items, each of width *CTLx.SRC_TR_WIDTH*, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface.

SRC_MSIZE	Size (items number)
0	1
1	4
2	8

SRC_MSIZE	Size (items number)
3	16
4	32
Other	Reserved

- **DST_MSIZE: Destination Burst Transaction Length**

Number of data items, each of width *CTLx.DST_TR_WIDTH*, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface.

DST_MSIZE	Size (items number)
0	1
1	4
2	8
3	16
4	32
Other	Reserved

- **SINC: Source Address Increment**

Indicates whether to increment or decrement the source address on every source System Bus transfer. If your device is fetching data from a source peripheral FIFO with a fixed address, then set this field to “No change”

SINC	Source Address Increment
0	Increment
1	Decrement
Other	No change

- **DINC: Destination Address Increment**

Indicates whether to increment or decrement the destination address on every destination System Bus transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to “No change”

DINC	Destination Address Increment
0	Increment
1	Decrement
Other	No change

- **SRT_TR_WIDTH: Source Transfer Width**
- **DSC_TR_WIDTH: Destination Transfer Width**

SRC_TR_WIDTH/DST_TR_WIDTH	Size (bits)
0	8
1	16
2	32
Other	Reserved

- **INT_EN: Interrupt Enable Bit**

If set, then all five interrupt generating sources are enabled.

20.12.5 Control Register for Channel x High

Name: CTLxH

Access Type: Read/Write

Offset: 0x01C + [x * 0x58]

Reset Value: 0x00000002

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

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

15	14	13	12	11	10	9	8
-	-	-	DONE	BLOCK_TS[11:8]			

7	6	5	4	3	2	1	0
BLOCK_TS[7:0]							

- **DONE: Done Bit**

Software can poll this bit to see when a block transfer is complete

- **BLOCK_TS: Block Transfer Size**

When the DMACA is flow controller, this field is written by the user before the channel is enabled to indicate the block size.

The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer, unless the transfer is already in progress, in which case the value of BLOCK_TS indicates the number of single transactions that have been performed so far.

The width of the single transaction is determined by CTLx.SRC_TR_WIDTH.

20.12.6 Configuration Register for Channel x Low

Name: CFGxL

Access Type: Read/Write

Offset: 0x040 + [x * 0x58]

- **Reset Value: 0x00000C00 + [x * 0x20]**

31	30	29	28	27	26	25	24
RELOAD_DST	RELOAD_SRC	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	SRC_HS_POL	DST_HS_POL	-	-
15	14	13	12	11	10	9	8
-		-		HS_SEL_SRC	HS_SEL_DST	FIFO_EMPTY	CH_SUSP
7	6	5	4	3	2	1	0
CH_PRIOR			-	-	-	-	-

- **RELOAD_DST: Automatic Destination Reload**

The DARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.

- **RELOAD_SRC: Automatic Source Reload**

The SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.

- **SRC_HS_POL: Source Handshaking Interface Polarity**
  0 = Active high
  1 = Active low
- **DST_HS_POL: Destination Handshaking Interface Polarity**
  0 = Active high
  1 = Active low
- **HS_SEL_SRC: Source Software or Hardware Handshaking Select**

This register selects which of the handshaking interfaces, hardware or software, is active for source requests on this channel.

0 = Hardware handshaking interface. Software-initiated transaction requests are ignored.
1 = Software handshaking interface. Hardware-initiated transaction requests are ignored.

If the source peripheral is memory, then this bit is ignored.

- **HS_SEL_DST: Destination Software or Hardware Handshaking Select**

This register selects which of the handshaking interfaces, hardware or software, is active for destination requests on this channel.

0 = Hardware handshaking interface. Software-initiated transaction requests are ignored.

1 = Software handshaking interface. Hardware Initiated transaction requests are ignored.

If the destination peripheral is memory, then this bit is ignored.

- **FIFO_EMPTY**

Indicates if there is data left in the channel's FIFO. Can be used in conjunction with CFGx.CH_SUSP to cleanly disable a channel.

1 = Channel's FIFO empty

0 = Channel's FIFO not empty

- **CH_SUSP: Channel Suspend**

Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data.

0 = Not Suspended.

1 = Suspend. Suspend DMA transfer from the source.

- **CH_PRIOR: Channel priority**

A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range [0, x-1].

A programmed value outside this range causes erroneous behavior.

20.12.7 Configuration Register for Channel x High

Name: CFGxH

Access Type: Read/Write

Offset: 0x044 + [x * 0x58]

Reset Value: 0x00000004

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

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

15	14	13	12	11	10	9	8
-	DEST_PER				SRC_PER[3:1]		

7	6	5	4	3	2	1	0
SRC_PER[0]	-	-	PROTCTL			FIFO_MODE	FCMODE

- **DEST_PER: Destination Hardware Handshaking Interface**

Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the destination of channel *x* if the CFGx.HS_SEL_DST field is 0. Otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface via the assigned hardware handshaking interface.

For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.

- **SRC_PER: Source Hardware Handshaking Interface**

Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the source of channel *x* if the CFGx.HS_SEL_SRC field is 0. Otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface via the assigned hardware handshaking interface.

For correct DMACA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.

- **PROTCTL: Protection Control**

Bits used to drive the System Bus HPROT[3:1] bus. The *System Bus Specification* recommends that the default value of HPROT indicates a non-cached, nonbuffered, privileged data access. The reset value is used to indicate such an access.

HPROT[0] is tied high as all transfers are data accesses as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals.

- **FIFO_MODE: R/W 0x0 FIFO Mode Select**

Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.

0 = Space/data available for single System Bus transfer of the specified transfer width.
1 = Space/data available is greater than or equal to half the FIFO depth for destination transfers and less than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.

- **FCMODE: Flow Control Mode**

Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.

0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled.

1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode the amount of data transferred from the source is limited such that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

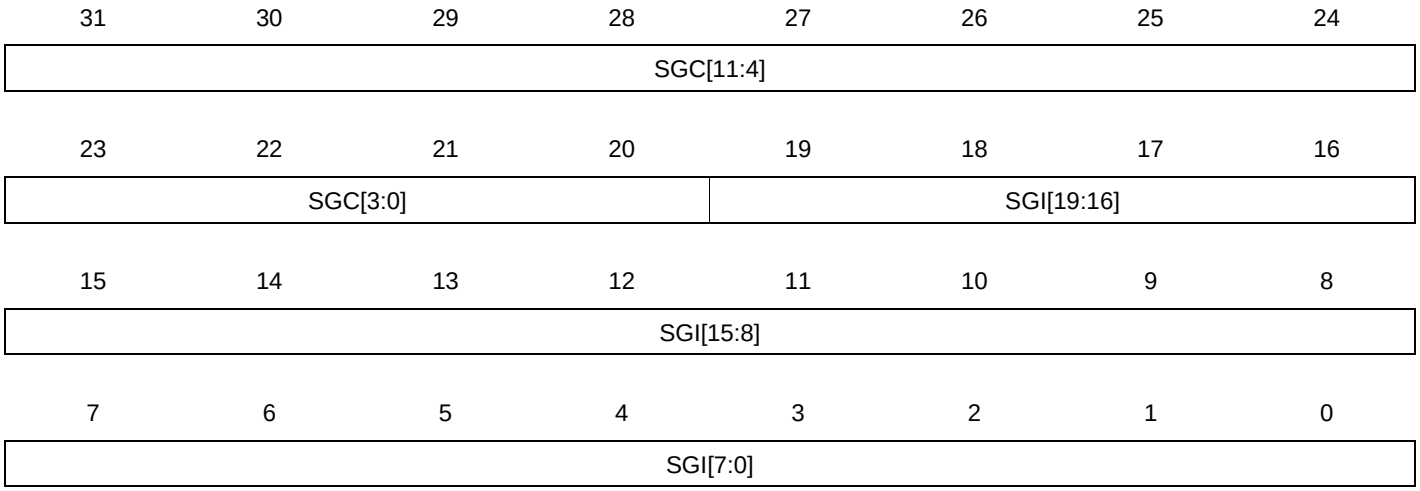
20.12.8 Source Gather Register for Channel x

Name: SGRx

Access Type: Read/Write

Offset: 0x048 + [x * 0x58]

Reset Value: 0x00000000



- **SGC: Source Gather Count**

Specifies the number of contiguous source transfers of CTL*x*.SRC_TR_WIDTH between successive gather intervals. This is defined as a gather boundary.

- **SGI: Source Gather Interval**

Specifies the source address increment/decrement in multiples of CTL*x*.SRC_TR_WIDTH on a gather boundary when gather mode is enabled for the source transfer.

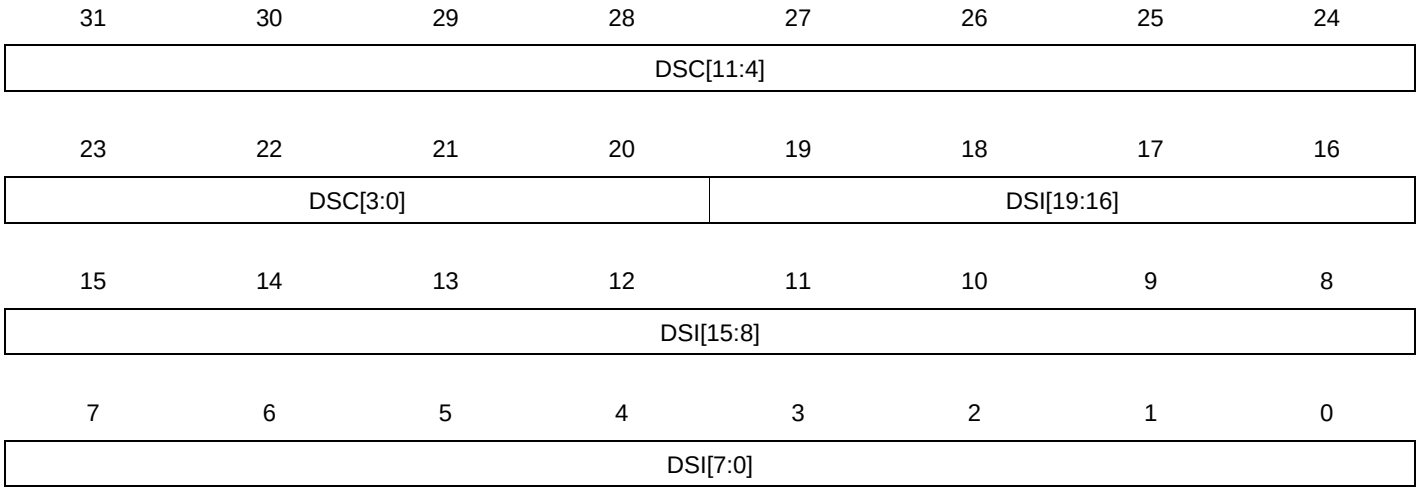
20.12.9 Destination Scatter Register for Channel x

Name: DSRx

Access Type: Read/Write

Offset: 0x050 + [x * 0x58]

Reset Value: 0x00000000



- **DSC: Destination Scatter Count**

Specifies the number of contiguous destination transfers of CTL*x*.DST_TR_WIDTH between successive scatter boundaries.

- **DSI: Destination Scatter Interval**

Specifies the destination address increment/decrement in multiples of CTL*x*.DST_TR_WIDTH on a scatter boundary when scatter mode is enabled for the destination transfer.

20.12.10 Interrupt Registers

The following sections describe the registers pertaining to interrupts, their status, and how to clear them. For each channel, there are five types of interrupt sources:

- IntTfr: DMA Transfer Complete Interrupt

This interrupt is generated on DMA transfer completion to the destination peripheral.

- IntBlock: Block Transfer Complete Interrupt

This interrupt is generated on DMA block transfer completion to the destination peripheral.

- IntSrcTran: Source Transaction Complete Interrupt

This interrupt is generated after completion of the last System Bus transfer of the requested single/burst transaction from the handshaking interface on the source side.

If the source for a channel is memory, then that channel never generates a IntSrcTran interrupt and hence the corresponding bit in this field is not set.

- IntDstTran: Destination Transaction Complete Interrupt

This interrupt is generated after completion of the last System Bus transfer of the requested single/burst transaction from the handshaking interface on the destination side.

If the destination for a channel is memory, then that channel never generates the IntDstTran interrupt and hence the corresponding bit in this field is not set.

- IntErr: Error Interrupt

This interrupt is generated when an ERROR response is received from an HSB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.

20.12.11 Interrupt Raw Status Registers

Name: RawTfr, RawBlock, RawSrcTran, RawDstTran, RawErr

Access Type: Read-only

Offset: 0x2C0, 0x2C8, 0x2D0, 0x2D8, 0x2E0

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
-	-	-	-	RAW3	RAW2	RAW1	RAW0

- **RAW[3:0]Raw interrupt for each channel**

Interrupt events are stored in these Raw Interrupt Status Registers before masking: RawTfr, RawBlock, RawSrcTran, RawDstTran, RawErr. Each Raw Interrupt Status register has a bit allocated per channel, for example, RawTfr[2] is Channel 2's raw transfer complete interrupt. Each bit in these registers is cleared by writing a 1 to the corresponding location in the ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr registers.

20.12.12 Interrupt Status Registers

Name: StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran, StatusErr

Access Type: Read-only

Offset: 0x2E8, 0x2F0, 0x2F8, 0x300, 0x308

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
-	-	-	-	STATUS3	STATUS2	STATUS1	STATUS0

- **STATUS[3:0]**

All interrupt events from all channels are stored in these Interrupt Status Registers after masking: StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran, StatusErr. Each Interrupt Status register has a bit allocated per channel, for example, StatusTfr[2] is Channel 2's status transfer complete interrupt.The contents of these registers are used to generate the interrupt signals leaving the DMACA.

20.12.13 Interrupt Mask Registers

Name: MaskTfr, MaskBlock, MaskSrcTran, MaskDstTran, MaskErr

Access Type: Read/Write

Offset: 0x310, 0x318, 0x320, 0x328, 0x330

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
-	-	-	-	INT_M_WE3	INT_M_WE2	INT_M_WE1	INT_M_WE0
7	6	5	4	3	2	1	0
-	-	-	-	INT_MASK3	INT_MASK2	INT_MASK1	INT_MASK0

The contents of the Raw Status Registers are masked with the contents of the Mask Registers: MaskTfr, MaskBlock, MaskSrcTran, MaskDstTran, MaskErr. Each Interrupt Mask register has a bit allocated per channel, for example, MaskTfr[2] is the mask bit for Channel 2's transfer complete interrupt.

A channel's INT_MASK bit is only written if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same System Bus write transfer. This allows software to set a mask bit without performing a read-modified write operation.

For example, writing hex 01x1 to the MaskTfr register writes a 1 into MaskTfr[0], while MaskTfr[7:1] remains unchanged. Writing hex 00*xx* leaves MaskTfr[7:0] unchanged.

Writing a 1 to any bit in these registers unmasks the corresponding interrupt, thus allowing the DMACA to set the appropriate bit in the Status Registers.

- **INT_M_WE[11:8]: Interrupt Mask Write Enable**
  0 = Write disabled
  1 = Write enabled
- **INT_MASK[3:0]: Interrupt Mask**
  0= Masked
  1 = Unmasked

20.12.14 Interrupt Clear Registers

Name: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr

Access Type: Write-only

Offset: 0x338, 0x340, 0x348, 0x350, 0x358

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
-	-	-	-	CLEAR3	CLEAR2	CLEAR1	CLEAR0

- **CLEAR[3:0]: Interrupt Clear**
  0 = No effect
  1 = Clear interrupt

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr. Each Interrupt Clear register has a bit allocated per channel, for example, ClearTfr[2] is the clear bit for Channel 2's transfer complete interrupt. Writing a 0 has no effect. These registers are not readable.

20.12.15 Combined Interrupt Status Registers

Name: StatusInt

Access Type: Read-only

Offset: 0x360

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
-	-	-	ERR	DSTT	SRCT	BLOCK	TFR

The contents of each of the five Status Registers (StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran, StatusErr) is OR'ed to produce a single bit per interrupt type in the Combined Status Register (StatusInt).

- **ERR**

OR of the contents of StatusErr Register.

- **DSTT**

OR of the contents of StatusDstTran Register.

- **SRCT**

OR of the contents of StatusSrcTran Register.

- **BLOCK**

OR of the contents of StatusBlock Register.

- **TFR**

OR of the contents of StatusTfr Register.

20.12.16 Source Software Transaction Request Register

Name: ReqSrcReg

Access Type: Read/write

Offset: 0x368

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
-	-	-	-	REQ_WE3	REQ_WE2	REQ_WE1	REQ_WE0
7	6	5	4	3	2	1	0
-	-	-	-	SRC_REQ3	SRC_REQ2	SRC_REQ1	SRC_REQ0

A bit is assigned for each channel in this register. ReqSrcReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel SRC_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same System Bus write transfer.

For example, writing 0x101 writes a 1 into ReqSrcReg[0], while ReqSrcReg[4:1] remains unchanged. Writing hex 0x0*yy* leaves ReqSrcReg[4:0] unchanged. This allows software to set a bit in the ReqSrcReg register without performing a read-modified write

- **REQ_WE[11:8]: Request write enable**
  0 = Write disabled
  1 = Write enabled
- **SRC_REQ[3:0]: Source request**

20.12.17 Destination Software Transaction Request Register

Name: ReqDstReg

Access Type: Read/write

Offset: 0x370

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
-	-	-	-	REQ_WE3	REQ_WE2	REQ_WE1	REQ_WE0

7	6	5	4	3	2	1	0
-	-	-	-	DST_REQ3	DST_REQ2	DST_REQ1	DST_REQ0

A bit is assigned for each channel in this register. ReqDstReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel DST_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same System Bus write transfer.

- **REQ_WE[11:8]: Request write enable**
  0 = Write disabled
  1 = Write enabled
- **DST_REQ[3:0]: Destination request**

20.12.18 Single Source Transaction Request Register

Name: SglReqSrcReg

Access Type: Read/write

Offset: 0x378

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
-	-	-	-	REQ_WE3	REQ_WE2	REQ_WE1	REQ_WE0
7	6	5	4	3	2	1	0
-	-	-	-	S_SG_REQ3	S_SG_REQ2	S_SG_REQ1	S_SG_REQ0

A bit is assigned for each channel in this register. SglReqSrcReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel S_SG_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same System Bus write transfer.

- **REQ_WE[11:8]: Request write enable**
  0 = Write disabled
  1 = Write enabled
- **S_SG_REQ[3:0]: Source single request**

20.12.19 Single Destination Transaction Request Register

Name: SglReqDstReg

Access Type: Read/write

Offset: 0x380

Reset Value: 0x0000000

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

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

15	14	13	12	11	10	9	8
-	-	-	-	REQ_WE3	REQ_WE2	REQ_WE1	REQ_WE0

7	6	5	4	3	2	1	0
-	-	-	-	D_SG_REQ3	D_SG_REQ2	D_SG_REQ1	D_SG_REQ0

A bit is assigned for each channel in this register. SglReqDstReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel D_SG_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same System Bus write transfer.

- **REQ_WE[11:8]: Request write enable**
  0 = Write disabled
  1 = Write enabled
- **D_SG_REQ[3:0]: Destination single request**

20.12.20 Last Source Transaction Request Register

Name: LstSrcReg

Access Type: Read/write

Offset: 0x388

Reset Value: 0x0000000

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

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

15	14	13	12	11	10	9	8
-	-	-	-	LSTSRC_WE3	LSTSRC_WE2	LSTSRC_WE1	LSTSRC_WE0

7	6	5	4	3	2	1	0
-	-	-	-	LSTSRC3	LSTSRC2	LSTSRC1	LSTSRC0

A bit is assigned for each channel in this register. LstSrcReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel LSTSRC bit is written only if the corresponding channel write enable bit in the LSTSRC_WE field is asserted on the same System Bus write transfer.

- **LSTSRC_WE[11:8]: Source Last Transaction request write enable**
  0 = Write disabled
  1 = Write enabled
- **LSTSRC[3:0]: Source Last Transaction request**

20.12.21 Last Destination Transaction Request Register

Name: LstDstReg

Access Type: Read/write

Offset: 0x390

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
-	-	-	-	LSTDST_WE3	LSTDST_WE2	LSTDST_WE1	LSTDST_WE0

7	6	5	4	3	2	1	0
-	-	-	-	LSTDST3	LSTDST2	LSTDST1	LSTDST0

A bit is assigned for each channel in this register. LstDstReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel LSTDST bit is written only if the corresponding channel write enable bit in the LSTDST_WE field is asserted on the same System Bus write transfer.

- **LSTDST_WE[11:8]: Destination Last Transaction request write enable**
  0 = Write disabled
  1 = Write enabled
- **LSTDST[3:0]: Destination Last Transaction request**

20.12.22 DMA Configuration Register

Name: DmaCfgReg

Access Type: Read/Write

Offset: 0x398

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
-	-	-	-	-	-	-	DMA_EN

- **DMA_EN: DMA Controller Enable**
  0 = DMACA Disabled
  1 = DMACA Enabled.

This register is used to enable the DMACA, which must be done before any channel activity can begin.

If the global channel enable bit is cleared while any channel is still active, then DmaCfgReg.DMA_EN still returns ‘1’ to indicate that there are channels still active until hardware has terminated all activity on all channels, at which point the DmaCfgReg.DMA_EN bit returns ‘0’.

20.12.23 DMA Channel Enable Register

Name: ChEnReg

Access Type: Read/Write

Offset: 0x3A0

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
-	-	-	-	CH_EN_WE3	CH_EN_WE2	CH_EN_WE1	CH_EN_WE0
7	6	5	4	3	2	1	0
-	-	-	-	CH_EN3	CH_EN2	CH_EN1	CH_EN0

- **CH_EN_WE[11:8]: Channel Enable Write Enable**

The channel enable bit, CH_EN, is only written if the corresponding channel write enable bit, CH_EN_WE, is asserted on the same System Bus write transfer.

For example, writing 0x101 writes a 1 into ChEnReg[0], while ChEnReg[7:1] remains unchanged.

- **CH_EN[3:0]**

  0 = Disable the Channel

  1 = Enable the Channel

Enables/Disables the channel. Setting this bit enables a channel, clearing this bit disables the channel.

The ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last System Bus transfer of the DMA transfer to the destination has completed.Software can therefore poll this bit to determine when a DMA transfer has completed.

20.12.24 DMACA Component Id Register Low

Name: DmaCompIdRegL

Access Type: Read-only

Offset: 0x3F8

Reset Value: 0x44571110

31	30	29	28	27	26	25	24
DMA_COMP_TYPE[31:24]							

23	22	21	20	19	18	17	16
DMA_COMP_TYPE[23:16]							

15	14	13	12	11	10	9	8
DMA_COMP_TYPE[15:8]							

7	6	5	4	3	2	1	0
DMA_COMP_TYPE[7:0]							

- **DMA_COMP_TYPE**

DesignWare component type number = 0x44571110.

This assigned unique hex value is constant and is derived from the two ASCII letters “DW” followed by a 32-bit unsigned number

20.12.25 DMACA Component Id Register High

Name: DmaCompIdRegH

Access Type: Read-only

Offset: 0x3FC

Reset Value: 0x3230362A

31	30	29	28	27	26	25	24
DMA_COMP_VERSION[31:24]							

23	22	21	20	19	18	17	16
DMA_COMP_VERSION[23:16]							

15	14	13	12	11	10	9	8
DMA_COMP_VERSION[15:8]							

7	6	5	4	3	2	1	0
DMA_COMP_VERSION[7:0]							

- **DMA_COMP_VERSION: Version of the component**

20.13 Module Configuration

The following table defines the valid settings for the DEST_PER and SRC_PER fields in the CFGxH register.

Table 20-6. DMACA Handshake Interfaces

PER Value	Hardware Handshaking Interface
0	AES - RX
1	AES - TX
2	MCI - RX
3	MCI -TX
4	MSI - RX
5	MSI - TX
6	EXTRQ0
7	EXTRQ1

21. General-Purpose Input/Output Controller (GPIO)

Rev: 1.1.0.4

21.1 Features

Each I/O line of the GPIO features:

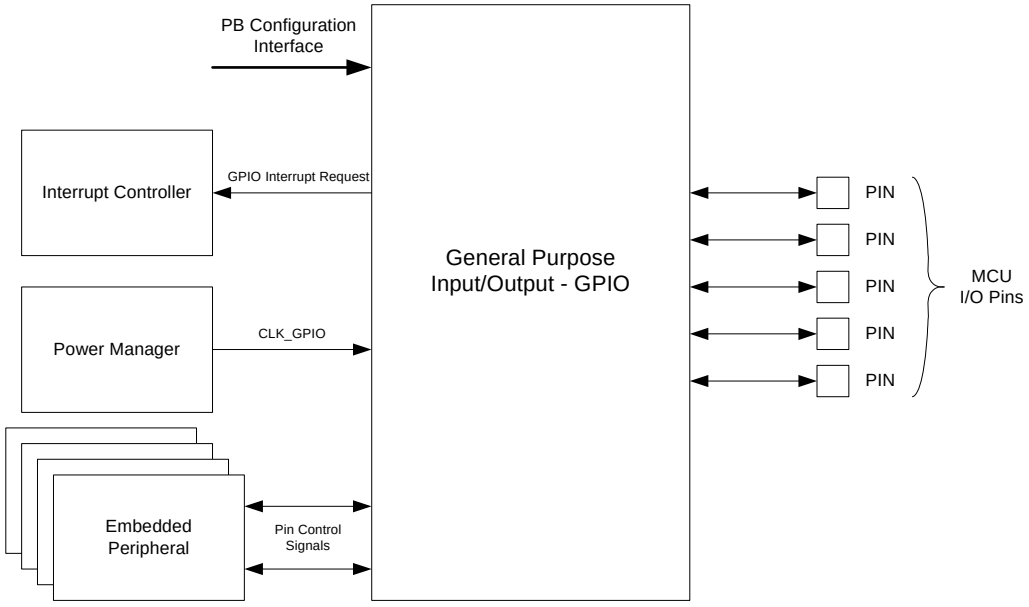
- **Configurable pin-change, rising-edge or falling-edge interrupt on any I/O line**
- **A glitch filter providing rejection of pulses shorter than one clock cycle**
- **Input visibility and output control**
- **Multiplexing of up to four peripheral functions per I/O line**
- **Programmable internal pull-up resistor**

21.2 Overview

The General Purpose Input/Output Controller manages the I/O pins of the microcontroller. 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.

21.3 Block Diagram

Figure 21-1. GPIO Block Diagram



21.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

21.4.1 Module Configuration

Most of the features of the GPIO are configurable for each product. The user must refer to the Package and Pinout chapter for these settings.

Product specific settings includes:

- Number of I/O pins.
- Functions implemented on each pin

- Peripheral function(s) multiplexed on each I/O pin
- Reset state of registers

21.4.2 Clocks

The clock for the GPIO bus interface (CLK_GPIO) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager.

The CLK_GPIO must be enabled in order to access the configuration registers of the GPIO or to use the GPIO interrupts. After configuring the GPIO, the CLK_GPIO can be disabled if interrupts are not used.

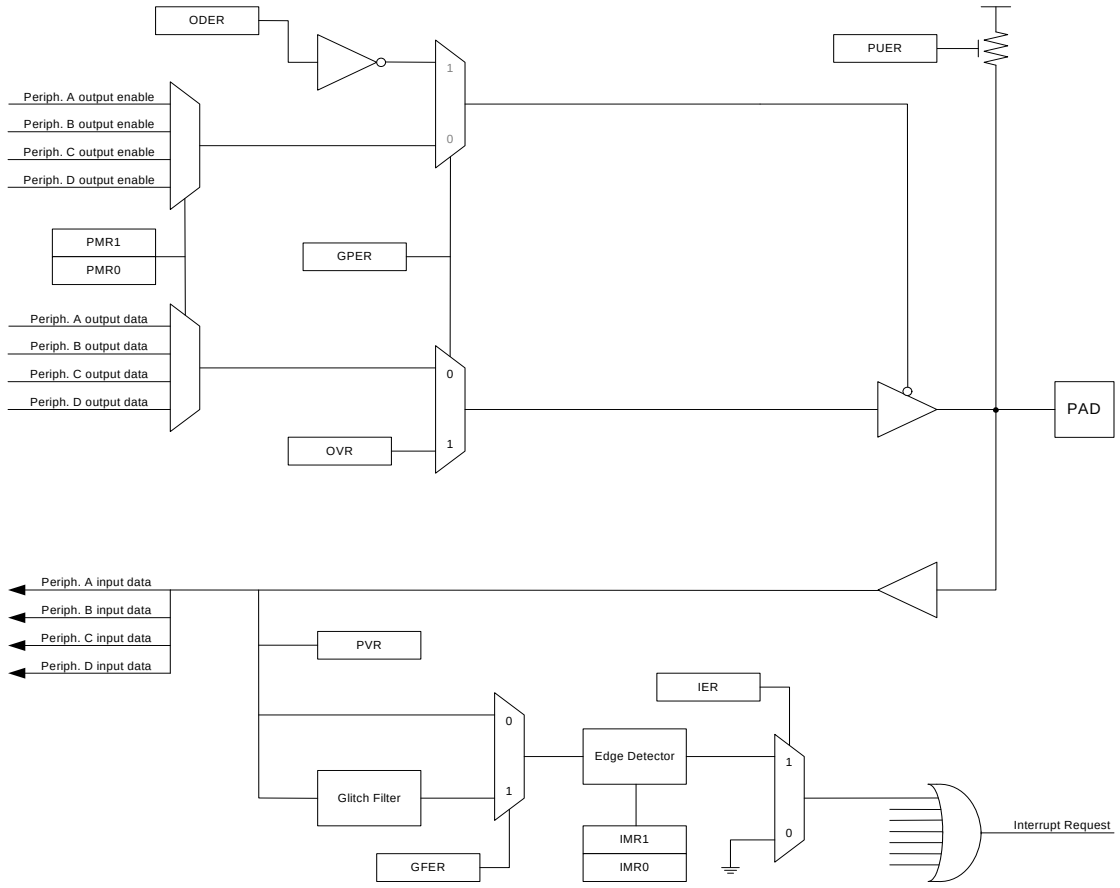
21.4.3 Interrupts

The GPIO interrupt lines are connected to the interrupt controller. Using the GPIO interrupt requires the interrupt controller to be configured first.

21.5 Functional Description

The GPIO controls the I/O lines of the microcontroller. The control logic associated with each pin is represented in the figure below:

Figure 21-2. Overview of the GPIO Pad Connections



21.5.1 Basic Operation

21.5.1.1 I/O Line or peripheral function selection

When a pin is multiplexed with one or more peripheral functions, the selection is controlled with the GPIO Enable Register (GPER). If a bit in GPER is written to one, the corresponding pin is controlled by the GPIO. If a bit is written to zero, the corresponding pin is controlled by a peripheral function.

21.5.1.2 Peripheral selection

The GPIO provides multiplexing of up to four peripheral functions on a single pin. The selection is performed by accessing Peripheral Mux Register 0 (PMR0) and Peripheral Mux Register 1 (PMR1).

21.5.1.3 Output control

When the I/O line is assigned to a peripheral function, i.e. the corresponding bit in GPER is written to zero, the drive of the I/O line is controlled by the peripheral. The peripheral, depending on the value in PMR0 and PMR1, determines whether the pin is driven or not.

When the I/O line is controlled by the GPIO, the value of the Output Driver Enable Register (ODER) determines if the pin is driven or not. When a bit in this register is written to one, the corresponding I/O line is driven by the GPIO. When the bit is written to zero, the GPIO does not drive the line.

The level driven on an I/O line can be determined by writing to the Output Value Register (OVR).

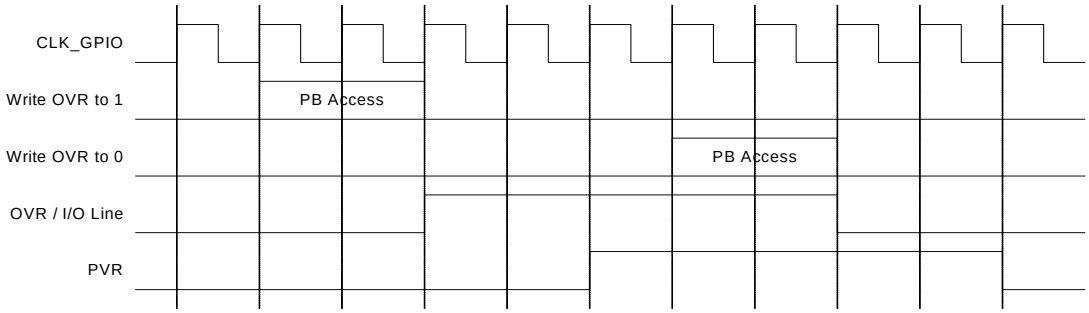
21.5.1.4 Inputs

The level on each I/O line can be read through the Pin Value Register (PVR). This register indicates the level of the I/O lines regardless of whether the lines are driven by the GPIO or by an external component. Note that due to power saving measures, the PVR register can only be read when GPER is written to one for the corresponding pin or if interrupt is enabled for the pin.

21.5.1.5 Output line timings

The figure below shows the timing of the I/O line when writing a one and a zero to OVR. The same timing applies when performing a ‘set’ or ‘clear’ access, i.e., writing a one to the Output Value Set Register (OVRS) or the Output Value Clear Register (OVRC). The timing of PVR is also shown.

Figure 21-3. Output Line Timings



21.5.2 Advanced Operation

21.5.2.1 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 a one or a zero to the corresponding bit in the Pull-up Enable Register (PUER). Control of the pull-up resistor is possible whether an I/O line is controlled by a peripheral or the GPIO.

21.5.2.2 Input glitch filter

Optional input glitch filters can be enabled on each I/O line. When the glitch filter is enabled, a glitch with duration of less than 1 clock cycle is automatically rejected, while a pulse with duration of 2 clock cycles or more is accepted. For pulse durations between 1 clock cycle and 2 clock cycles, the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be guaranteed visible it must exceed 2 clock cycles, whereas for a glitch to be reliably filtered out, its duration must not exceed 1 clock cycle. The filter introduces 2 clock cycles of latency.

The glitch filters are controlled by the Glitch Filter Enable Register (GFER). When a bit is written to one in GFER, the glitch filter on the corresponding pin is enabled. The glitch filter affects only interrupt inputs. Inputs to peripherals or the value read through PVR are not affected by the glitch filters.

21.5.3 Interrupts

The GPIO can be configured to generate an interrupt when it detects an input change on an I/O line. The module can be configured to signal an interrupt whenever a pin changes value or only to trigger on rising edges or falling edges. Interrupts are enabled on a pin by writing a one to the corresponding bit in the Interrupt Enable Register (IER). The interrupt mode is set by writing to the Interrupt Mode Register 0 (IMR0) and the Interrupt Mode Register 1(IMR1). Interrupts can be enabled on a pin, regardless of the configuration of the I/O line, i.e. whether it is controlled by the GPIO or assigned to a peripheral function.

In every port there are four interrupt lines connected to the interrupt controller. Groups of eight interrupts in the port are ORed together to form an interrupt line.

When an interrupt event is detected on an I/O line, and the corresponding bit in IER is written to one, the GPIO interrupt request line is asserted. A number of interrupt signals are ORed-wired together to generate a single interrupt signal to the interrupt controller.

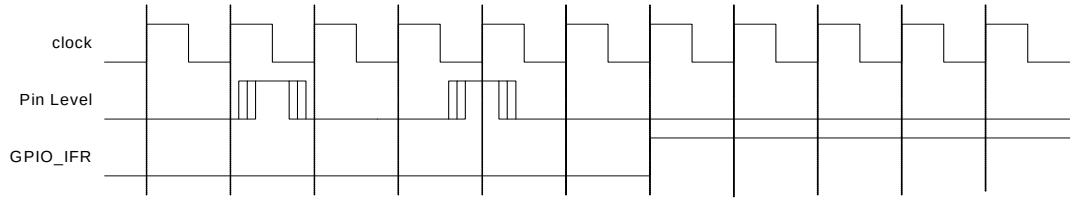
The Interrupt Flag Register (IFR) can by read to determine which pin(s) caused the interrupt. The interrupt bit must be cleared by writing a one to the Interrupt Flag Clear Register (IFRC).

GPIO interrupts can only be triggered when the CLK_GPIO is enabled.

21.5.4 Interrupt Timings

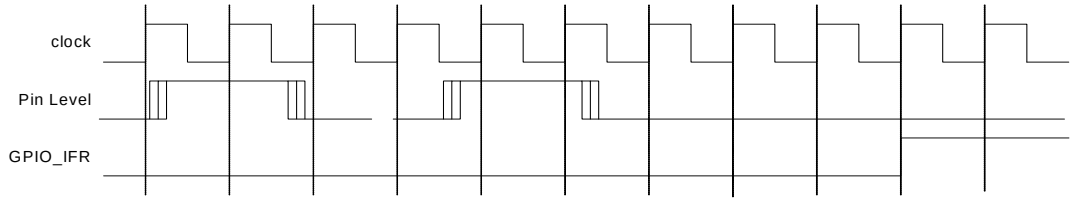
The figure below shows the timing for rising edge (or pin-change) interrupts when the glitch filter is disabled. For the pulse to be registered, it must be sampled at the rising edge of the clock. In this example, this is not the case for the first pulse. The second pulse is however sampled on a rising edge and will trigger an interrupt request.

Figure 21-4. Interrupt Timing With Glitch Filter Disabled



The figure below shows the timing for rising edge (or pin-change) interrupts when the glitch filter is enabled. For the pulse to be registered, it must be sampled on two subsequent rising edges. In the example, the first pulse is rejected while the second pulse is accepted and causes an interrupt request.

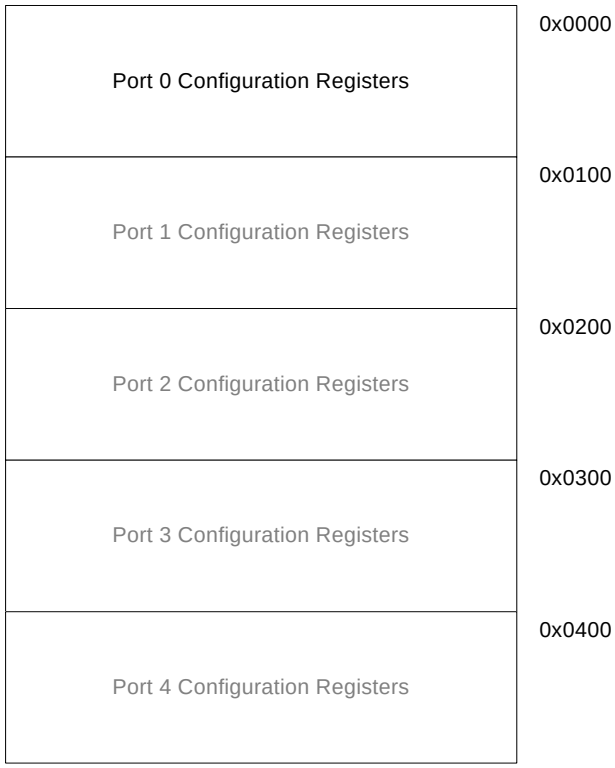
Figure 21-5. Interrupt Timing With Glitch Filter Enabled



21.6 User Interface

The GPIO controls all the I/O pins on the AVR32 microcontroller. The pins are managed as 32-bit ports that are configurable through a PB interface. Each port has a set of configuration registers. The overall memory map of the GPIO is shown below. The number of pins and hence the number of ports are product specific.

Figure 21-6. Overall Mermory Map



In the GPIO Controller Function Multiplexingtable in the Package and Pinout chapter, each GPIO line has a unique number. Note that the PA, PB, PC and PX ports do not directly correspond to the GPIO ports. To find the corresponding port and pin the following formula can be used:

GPIO port = floor((GPIO number) / 32), example: floor((36)/32) = 1

GPIO pin = GPIO number mod 32, example: 36 mod 32 = 4

The table below shows the configuration registers for one port. Addresses shown are relative to the port address offset. The specific address of a configuration register is found by adding the

register offset and the port offset to the GPIO start address. One bit in each of the configuration registers corresponds to an I/O pin.

Table 21-1. GPIO Register Memory Map

Offset	Register	Function	Name	Access	Reset value
0x00	GPIO Enable Register	Read/Write	GPER	Read/Write	0x00000000
0x04	GPIO Enable Register	Set	GPERS	Write-Only	0x00000000
0x08	GPIO Enable Register	Clear	GPERC	Write-Only	0x00000000
0x0C	GPIO Enable Register	Toggle	GPERT	Write-Only	0x00000000
0x10	Peripheral Mux Register 0	Read/Write	PMR0	Read/Write	0x00000000
0x14	Peripheral Mux Register 0	Set	PMR0S	Write-Only	0x00000000
0x18	Peripheral Mux Register 0	Clear	PMR0C	Write-Only	0x00000000
0x1C	Peripheral Mux Register 0	Toggle	PMR0T	Write-Only	0x00000000
0x20	Peripheral Mux Register 1	Read/Write	PMR1	Read/Write	0x00000000
0x24	Peripheral Mux Register 1	Set	PMR1S	Write-Only	0x00000000
0x28	Peripheral Mux Register 1	Clear	PMR1C	Write-Only	0x00000000
0x2C	Peripheral Mux Register 1	Toggle	PMR1T	Write-Only	0x00000000
0x40	Output Driver Enable Register	Read/Write	ODER	Read/Write	0x00000000
0x44	Output Driver Enable Register	Set	ODERS	Write-Only	0x00000000
0x48	Output Driver Enable Register	Clear	ODERC	Write-Only	0x00000000
0x4C	Output Driver Enable Register	Toggle	ODERT	Write-Only	0x00000000
0x50	Output Value Register	Read/Write	OVR	Read/Write	0x00000000
0x54	Output Value Register	Set	OVRS	Write-Only	0x00000000
0x58	Output Value Register	Clear	OVRC	Write-Only	0x00000000
0x5c	Output Value Register	Toggle	OVRT	Write-Only	0x00000000
0x60	Pin Value Register	Read	PVR	Read-Only	-(1)
0x70	Pull-up Enable Register	Read/Write	PUER	Read/Write	0x00000000
0x74	Pull-up Enable Register	Set	PUERS	Write-Only	0x00000000
0x78	Pull-up Enable Register	Clear	PUERC	Write-Only	0x00000000
0x7C	Pull-up Enable Register	Toggle	PUERT	Write-Only	0x00000000
0x90	Interrupt Enable Register	Read/Write	IER	Read/Write	0x00000000
0x94	Interrupt Enable Register	Set	IERS	Write-Only	0x00000000
0x98	Interrupt Enable Register	Clear	IERC	Write-Only	0x00000000
0x9C	Interrupt Enable Register	Toggle	IERT	Write-Only	0x00000000
0xA0	Interrupt Mode Register 0	Read/Write	IMR0	Read/Write	0x00000000
0xA4	Interrupt Mode Register 0	Set	IMR0S	Write-Only	0x00000000
0xA8	Interrupt Mode Register 0	Clear	IMR0C	Write-Only	0x00000000
0xAC	Interrupt Mode Register 0	Toggle	IMR0T	Write-Only	0x00000000
0xB0	Interrupt Mode Register 1	Read/Write	IMR1	Read/Write	0x00000000

Table 21-1. GPIO Register Memory Map

Offset	Register	Function	Name	Access	Reset value
0xB4	Interrupt Mode Register 1	Set	IMR1S	Write-Only	0x00000000
0xB8	Interrupt Mode Register 1	Clear	IMR1C	Write-Only	0x00000000
0xBC	Interrupt Mode Register 1	Toggle	IMR1T	Write-Only	0x00000000
0xC0	Glitch Filter Enable Register	Read/Write	GFER	Read/Write	0x00000000
0xC4	Glitch Filter Enable Register	Set	GFERS	Write-Only	0x00000000
0xC8	Glitch Filter Enable Register	Clear	GFERC	Write-Only	0x00000000
0xCC	Glitch Filter Enable Register	Toggle	GFERT	Write-Only	0x00000000
0xD0	Interrupt Flag Register	Read	IFR	Read-Only	0x00000000
0xD4	Interrupt Flag Register	-	-	-	0x00000000
0xD8	Interrupt Flag Register	Clear	IFRC	Write-Only	0x00000000
0xDC	Interrupt Flag Register	-	-	-	0x00000000

Note: 1. The reset value is undefined depending on the pin states.

21.6.1 Access Types

Each configuration register can be accessed in four different ways. The first address location can be used to write the register directly. This address can also be used to read the register value. The following addresses facilitate three different types of write access to the register. Performing a “set” access, all bits written to one will be set. Bits written to zero will be unchanged by the operation. Performing a “clear” access, all bits written to one will be cleared. Bits written to zero will be unchanged by the operation. Finally, a toggle access will toggle the value of all bits written to one. Again all bits written to zero remain unchanged. Note that for some registers (e.g. IFR), not all access methods are permitted.

Note that for ports with less than 32 bits, the corresponding control registers will have unused bits. This is also the case for features that are not implemented for a specific pin. Writing to an unused bit will have no effect. Reading unused bits will always return 0.

21.6.2 Enable Register

Name: GPER

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x00, 0x04, 0x08, 0x0C

Reset Value: 0x00000000

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: Pin Enable**
  0: A peripheral function controls the corresponding pin.
  1: The GPIO controls the corresponding pin.

21.6.3 Peripheral Mux Register 0

Name: PMR0

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x10, 0x14, 0x18, 0x1C

Reset Value: 0x00000000

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

- **P0-31: Peripheral Multiplexer Select bit 0**

21.6.4 Peripheral Mux Register 1

Name: PMR1

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x20, 0x24, 0x28, 0x2C

Reset Value: 0x00000000

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

- **P0-31: Peripheral Multiplexer Select bit 1**

{PMR1, PMR0}	Selected Peripheral Function
00	A
01	B
10	C
11	D

21.6.5 Output Driver Enable Register

Name: ODER

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x40, 0x44, 0x48, 0x4C

Reset Value: 0x00000000

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

- **P0-31: Output Driver Enable**
  0: The output driver is disabled for the corresponding pin.
  1: The output driver is enabled for the corresponding pin.

21.6.6 Output Value Register

Name: OVR

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x50, 0x54, 0x58, 0x5C

Reset Value: 0x00000000

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

- **P0-31: Output Value**
  0: The value to be driven on the I/O line is 0.
  1: The value to be driven on the I/O line is 1.

21.6.7 Pin Value Register

Name: PVR

Access Type: Read

Offset: 0x60, 0x64, 0x68, 0x6C

Reset Value: -

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

- **P0-31: Pin Value**

  0: The I/O line is at level ‘0’.

  1: The I/O line is at level ‘1’.

  Note that the level of a pin can only be read when GPER is set or interrupt is enabled for the pin.

21.6.8 Pull-up Enable Register

Name: PUER

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x70, 0x74, 0x78, 0x7C

Reset Value: 0x00000000

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

- **P0-31: Pull-up Enable**
  0: The internal pull-up resistor is disabled for the corresponding pin.
  1: The internal pull-up resistor is enabled for the corresponding pin.

21.6.9 Interrupt Enable Register

Name: IER

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x90, 0x94, 0x98, 0x9C

Reset Value: 0x00000000

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

- **P0-31: Interrupt Enable**
  0: Interrupt is disabled for the corresponding pin.
  1: Interrupt is enabled for the corresponding pin.

21.6.10 Interrupt Mode Register 0

Name: IMR0

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0xA0, 0xA4, 0xA8, 0xAC

Reset Value: 0x00000000

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

- **P0-31: Interrupt Mode Bit 0**

21.6.11 Interrupt Mode Register 1

Name: IMR1

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0xB0, 0xB4, 0xB8, 0xBC

Reset Value: 0x00000000

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

- **P0-31: Interrupt Mode Bit 1**

{IMR1, IMR0}	Interrupt Mode
00	Pin Change
01	Rising Edge
10	Falling Edge
11	Reserved

21.6.12 Glitch Filter Enable Register

Name: GFER

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0xC0, 0xC4, 0xC8, 0xCC

Reset Value: 0x00000000

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

- **P0-31: Glitch Filter Enable**
  0: Glitch filter is disabled for the corresponding pin.
  1: Glitch filter is enabled for the corresponding pin.
  NOTE! The value of this register should only be changed when IER is ‘0’. Updating this GFER while interrupt on the corresponding pin is enabled can cause an unintentional interrupt to be triggered.

21.6.13 Interrupt Flag Register

Name: IFR

Access Type: Read, Clear

Offset: 0xD0, 0xD8

Reset Value: 0x00000000

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

- **P0-31: Interrupt Flag**

  1: An interrupt condition has been detected on the corresponding pin.

  0: No interrupt condition has beedn detected on the corresponding pin since reset or the last time it was cleared.

  The number of interrupt request lines is dependant on the number of I/O pins on the MCU. Refer to the product specific data for details. Note also that a bit in the Interrupt Flag register is only valid if the corresponding bit in IER is set.

21.7 Programming Examples

21.7.1 8-bit LED-Chaser

```
        // Set R0 to GPIO base address
        mov    R0, LO(AVR32_GPIO_ADDRESS)
        orh    R0, HI(AVR32_GPIO_ADDRESS)

        // Enable GPIO control of pin 0-8
        mov    R1, 0xFF
        st.w   R0[AVR32_GPIO_GPERS], R1

        // Set initial value of port
        mov    R2, 0x01
        st.w   R0[AVR32_GPIO_OVRS], R2

        // Set up toggle value. Two pins are toggled
        // in each round. The bit that is currently set,
        // and the next bit to be set.
        mov    R2, 0x0303
        orh    R2, 0x0303


loop:
        // Only change 8 LSB
        mov    R3, 0x00FF
        and    R3, R2
        st.w   R0[AVR32_GPIO_OVRT], R3
        rol    R2
        rcall  delay
        rjmp   loop
```

It is assumed in this example that a subroutine "delay" exists that returns after a given time.

21.7.2 Configuration of USART pins

The example below shows how to configure a peripheral module to control I/O pins. It assumed in this example that the USART receive pin (RXD) is connected to PC16 and that the USART transmit pin (TXD) is connected to PC17. For both pins, the USART is peripheral B. In this example, the state of the GPIO registers is assumed to be unknown. The two USART pins are therefore first set to be controlled by the GPIO with output drivers disabled. The pins can then be assured to be tri-stated while changing the Peripheral Mux Registers.

```
        // Set up pointer to GPIO, PORTC
        mov    R0, LO(AVR32_GPIO_ADDRESS + PORTC_OFFSET)
        orh    R0, HI(AVR32_GPIO_ADDRESS + PORTC_OFFSET)

        // Disable output drivers
```

```
mov     R1, 0x0000
orh     R1, 0x0003
st.w    R0[AVR32_GPIO_ODERC], R1

// Make the GPIO control the pins
st.w    R0[AVR32_GPIO_GPERS], R1

// Select peripheral B on PC16-PC17
st.w    R0[AVR32_GPIO_PMR0S], R1
st.w    R0[AVR32_GPIO_PMR1C], R1

// Enable peripheral control
st.w    R0[AVR32_GPIO_GPERC], R1
```

22. Serial Peripheral Interface (SPI)

Rev. 2.1.0.1

22.1 Features

- **Compatible with an embedded 32-bit microcontroller**
- **Supports communication with serial external devices**
  - **Four chip selects with external decoder support allow communication with up to 15 peripherals**
  - **Serial memories, such as DataFlash and 3-wire EEPROMs**
  - **Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and Sensors**
  - **External co-processors**
- **Master or Slave Serial Peripheral Bus Interface**
  - **4 - to 16-bit programmable data length per chip select**
  - **Programmable phase and polarity per chip select**
  - **Programmable transfer delays between consecutive transfers and between clock and data per chip select**
  - **Programmable delay between consecutive transfers**
  - **Selectable mode fault detection**
- **Connection to Peripheral DMA Controller channel capabilities optimizes data transfers**
  - **One channel for the receiver, one channel for the transmitter**
  - **Next buffer support**
  - **Four character FIFO in reception**

22.2 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.

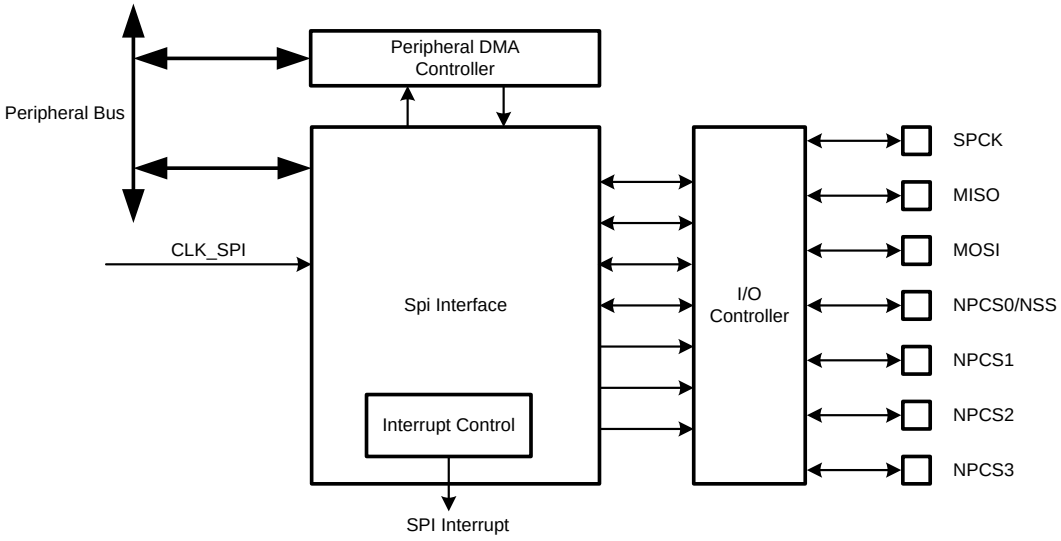
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

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.

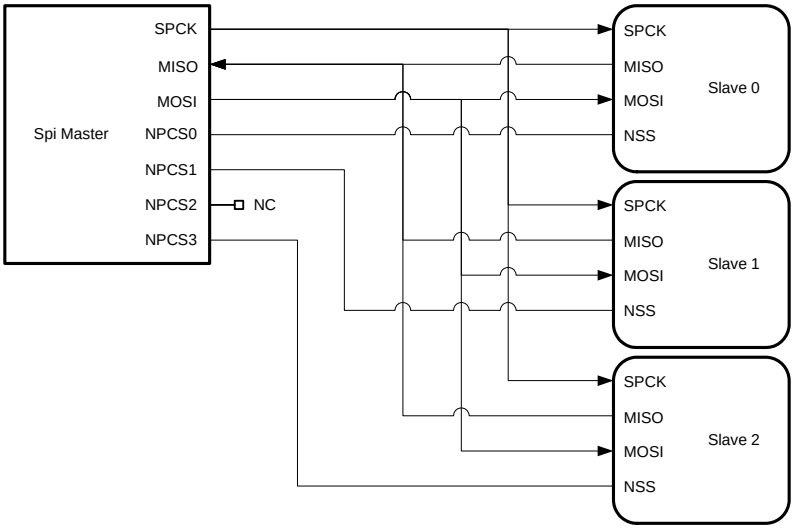
22.3 Block Diagram

Figure 22-1. SPI Block Diagram



22.4 Application Block Diagram

Figure 22-2. Application Block Diagram: Single Master/Multiple Slave Implementation



22.5 I/O Lines Description

Table 22-1. I/O Lines 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

22.6 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

22.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O lines. The user must first configure the I/O Controller to assign the SPI pins to their peripheral functions.

22.6.2 Clocks

The clock for the SPI bus interface (CLK_SPI) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SPI before disabling the clock, to avoid freezing the SPI in an undefined state.

22.6.3 Interrupts

The SPI interrupt request line is connected to the interrupt controller. Using the SPI interrupt requires the interrupt controller to be programmed first.

22.7 Functional Description

22.7.1 Modes of Operation

The SPI operates in master mode or in slave mode.

Operation in master mode is configured by writing a one to the Master/Slave Mode bit in the Mode Register (MR.MSTR). 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 MR.MSTR bit is written to zero, 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.

22.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is configured with the Clock Polarity bit in the Chip Select Registers (CSRn.CPOL). The clock phase is configured with the Clock Phase bit in the CSRn registers (CSRn.NCPHA). These two bits determine the edges of the clock signal on which data is driven and sampled. Each of the two bits 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.

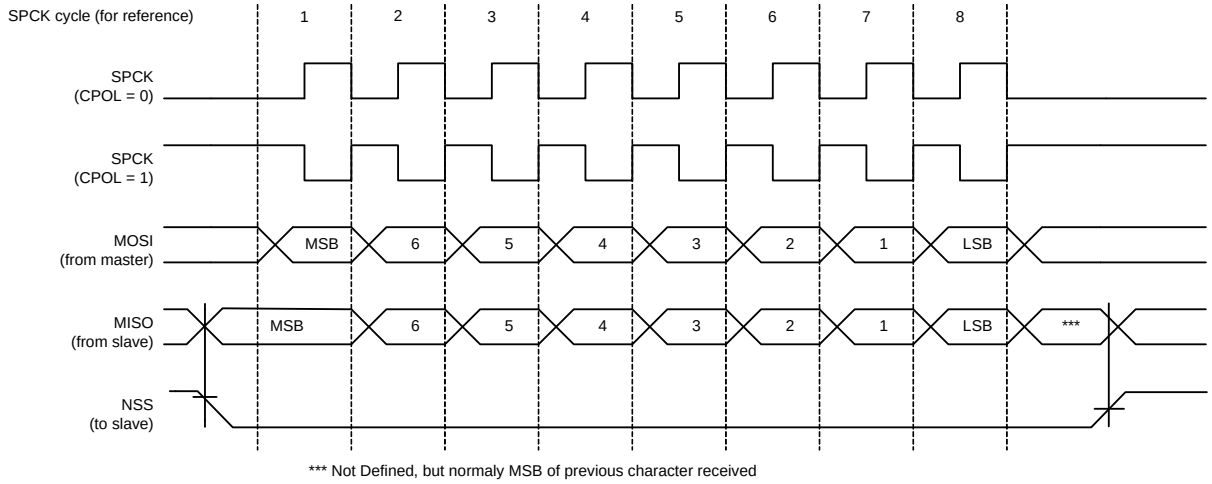
Table 22-2 on page 396 shows the four modes and corresponding parameter settings.

Table 22-2. SPI modes

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

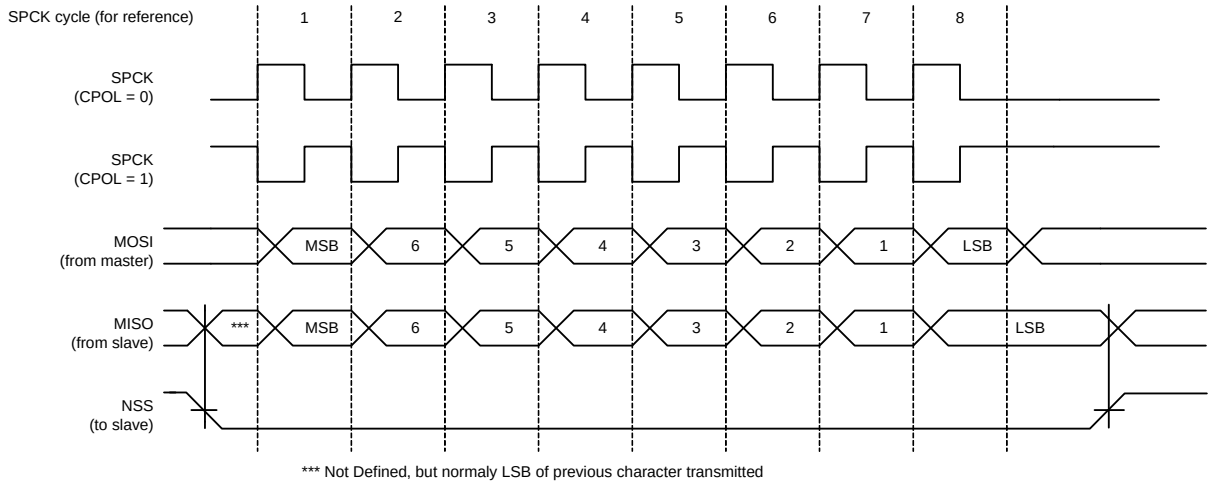
Figure 22-3 on page 396 and Figure 22-4 on page 397 show examples of data transfers.

Figure 22-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



*** Not Defined, but normaly MSB of previous character received

Figure 22-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



22.7.3 Master Mode Operations

When configured in master mode, the SPI uses the internal programmable baud rate generator as clock source. 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 (TDR) and the Receive Data Register (RDR), 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 TDR 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 to the TDR, the Peripheral Chip Select field in TDR (TDR.PCS) must be written in order to select a slave.

If new data is written to 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 RDR, the data in TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in TDR in the Shift Register is indicated by the Transmit Data Register Empty bit in the Status Register (SR.TDRE). When new data is written in TDR, this bit is cleared. The SR.TDRE bit is used to trigger the Transmit Peripheral DMA Controller channel.

The end of transfer is indicated by the Transmission Registers Empty bit in the SR register (SR.TXEMPTY). If a transfer delay (CSRn.DLYBCT) is greater than zero for the last transfer, SR.TXEMPTY is set after the completion of said delay. The CLK_SPI can be switched off at this time.

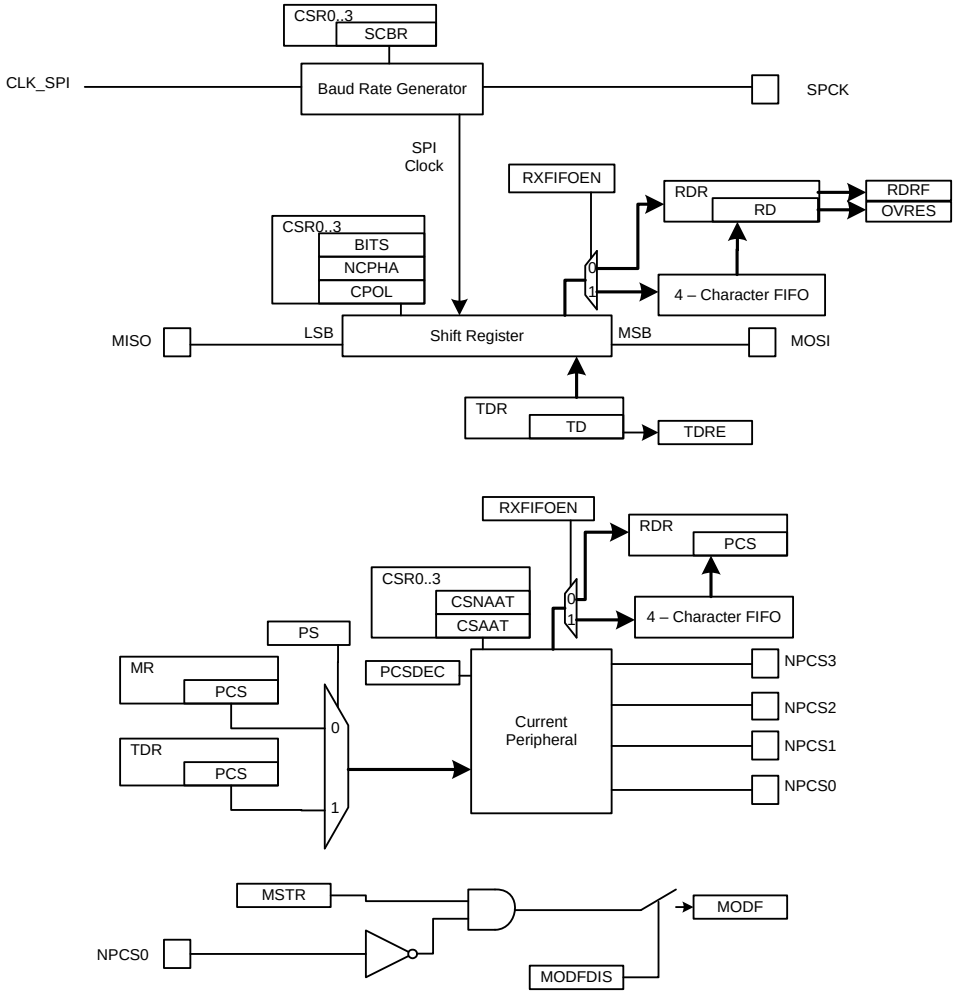
During reception, received data are transferred from the Shift Register to the reception FIFO. The FIFO can contain up to 4 characters (both Receive Data and Peripheral Chip Select fields). While a character of the FIFO is unread, the Receive Data Register Full bit in SR remains high (SR.RDRF). Characters are read through the RDR register. If the four characters stored in the FIFO are not read and if a new character is stored, this sets the Overrun Error Status bit in the SR register (SR.OVRES). The procedure to follow in such a case is described in Section 22.7.3.8.

In master mode, if the received data is not read fast enough compared to the transfer rhythm imposed by the write accesses in the TDR, some overrun errors may occur, even if the FIFO is enabled. To insure a perfect data integrity of received data (especially at high data rate), the mode Wait Data Read Before Transfer can be enabled in the MR register (MR.WDRBT). When this mode is activated, no transfer starts while received data remains unread in the RDR. When data is written to the TDR and if unread received data is stored in the RDR, the transfer is paused until the RDR is read. In this mode no overrun error can occur. Please note that if this mode is enabled, it is useless to activate the FIFO in reception.

Figure 22-5 on page 398shows a block diagram of the SPI when operating in master mode. Figure 22-6 on page 399 shows a flow chart describing how transfers are handled.

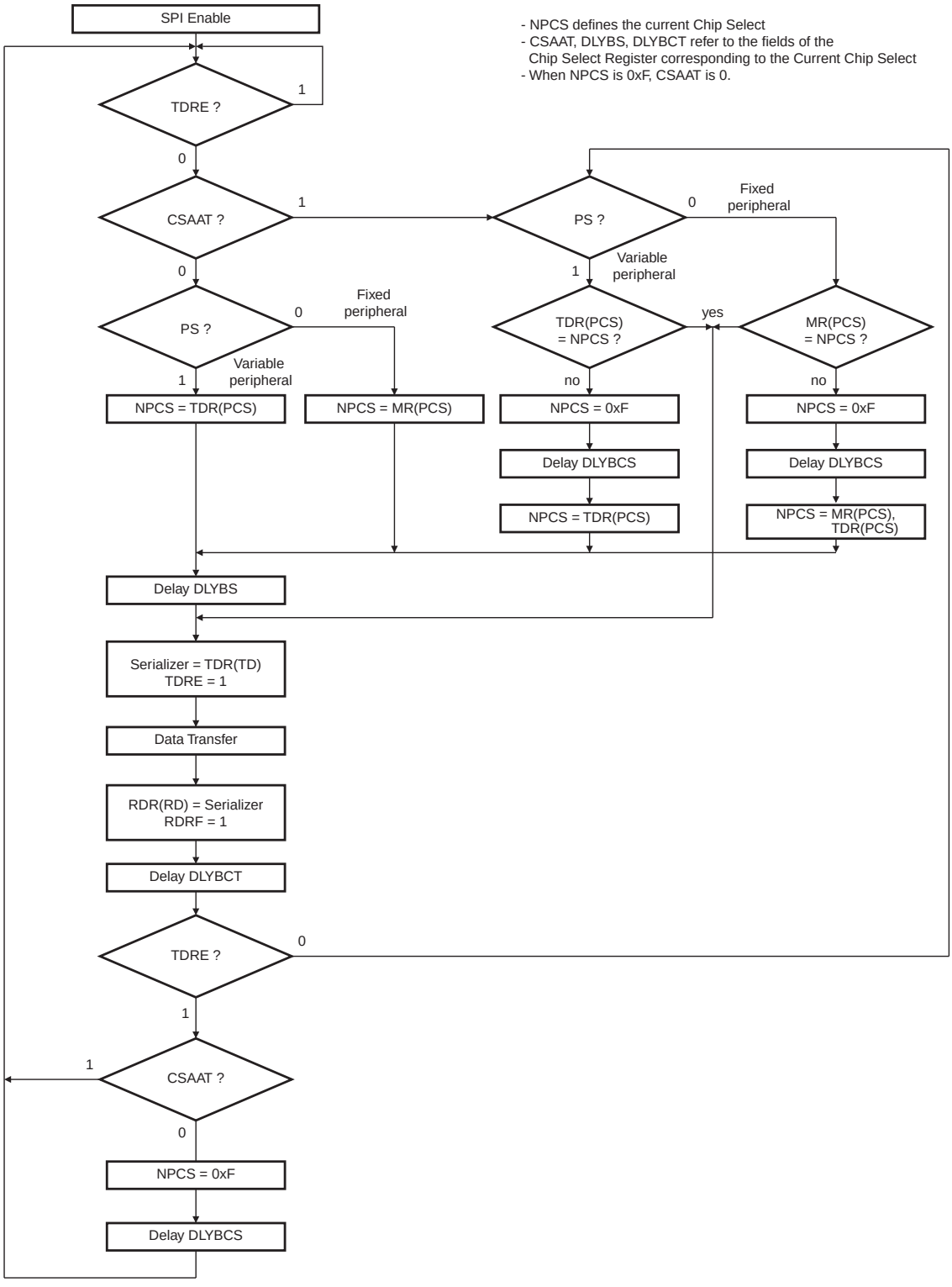
22.7.3.1 Master mode block diagram

Figure 22-5. Master Mode Block Diagram



22.7.3.2 *Master mode flow diagram*

Figure 22-6. Master Mode Flow Diagram



22.7.3.3 Clock generation

The SPI Baud rate clock is generated by dividing the CLK_SPI , by a value between 1 and 255.

This allows a maximum operating baud rate at up to CLK_SPI and a minimum operating baud rate of CLK_SPI divided by 255.

Writing the Serial Clock Baud Rate field in the CSRn registers (CSRn.SCBR) to zero is forbidden. Triggering a transfer while CSRn.SCBR is zero can lead to unpredictable results.

At reset, CSRn.SCBR is zero and the user has to configure 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 configured in the CSRn.SCBR field. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

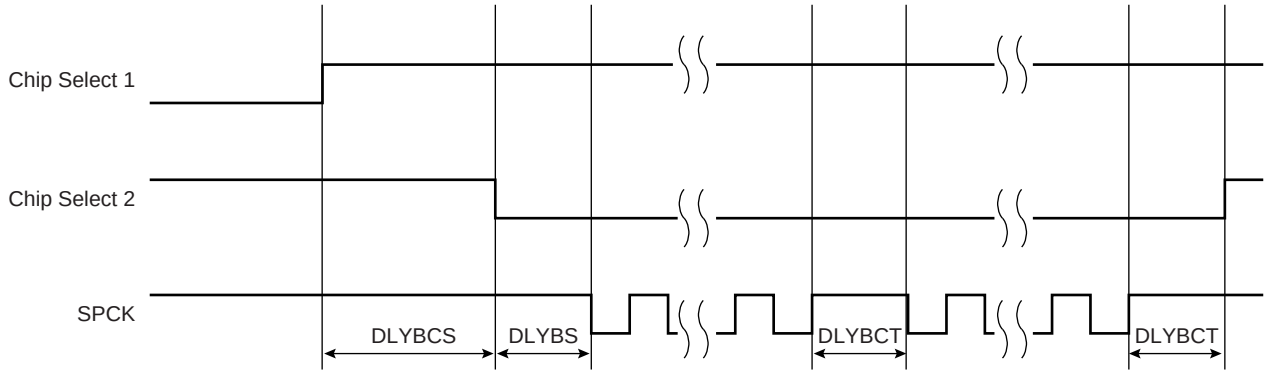
22.7.3.4 Transfer delays

Figure 22-7 on page 400 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be configured to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing to the Delay Between Chip Selects field in the MR register (MR.DLYBCS). 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 Delay Before SPCK field in the CSRn registers (CSRn.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 Delay Between Consecutive Transfers field in the CSRn registers (CSRn.DLYBCT). 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 22-7. Programmable Delays



22.7.3.5 Peripheral selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

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 a zero to the Peripheral Select bit in MR (MR.PS). In this case, the current peripheral is defined by the MR.PCS field and the TDR.PCS field has no effect.

Variable Peripheral Select is activated by writing a one to the MR.PS bit . The TDR.PCS field 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 Peripheral DMA Controller is an optimal means, as the size of the data transfer between the memory and the SPI is either 4 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 MR register. Data written to TDR is 32-bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the Peripheral DMA Controller 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 CSRn 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.

22.7.3.6 Peripheral chip select decoding

The user can configure 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 a one to the Chip Select Decode bit in the MR register (MR.PCSDEC).

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 MR register or the TDR 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 one) 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, the CRS0 register 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.

22.7.3.7 Peripheral deselection

When operating normally, as soon as the transfer of the last data written in 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 CSRn registers can be configured with the Chip Select Active After Transfer bit written to one (CSRn.CSAAT) . This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

When the CSRn.CSAAT bit is written to qero, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a Chip Select, the SR.TDRE bit rises as soon as the content of the TDR is transferred into the internal shifter. When this bit is detected the TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. This might lead to difficulties for interfacing with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate interfacing with such devices, the CSRn registers can be configured with the Chip Select Not Active After Transfer bit (CSRn.CSNAAT) written to one. This allows to de-assert systematically the chip select lines during a time DLYBCS. (The value of the CSRn.CSNAAT bit is taken into account only if the CSRn.CSAAT bit is written to zero for the same Chip Select).

Figure 22-8 on page 403 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

22.7.3.8 FIFO management

A FIFO has been implemented in Reception FIFO (both in master and in slave mode), in order to be able to store up to 4 characters without causing an overrun error. If an attempt is made to store a fifth character, an overrun error rises. If such an event occurs, the FIFO must be flushed. There are two ways to Flush the FIFO:

- By performing four read accesses of the RDR (the data read must be ignored)
- By writing a one to the Flush Fifo Command bit in the CR register (CR.FLUSHFIFO).

After that, the SPI is able to receive new data.

Figure 22-8. Peripheral Deselection

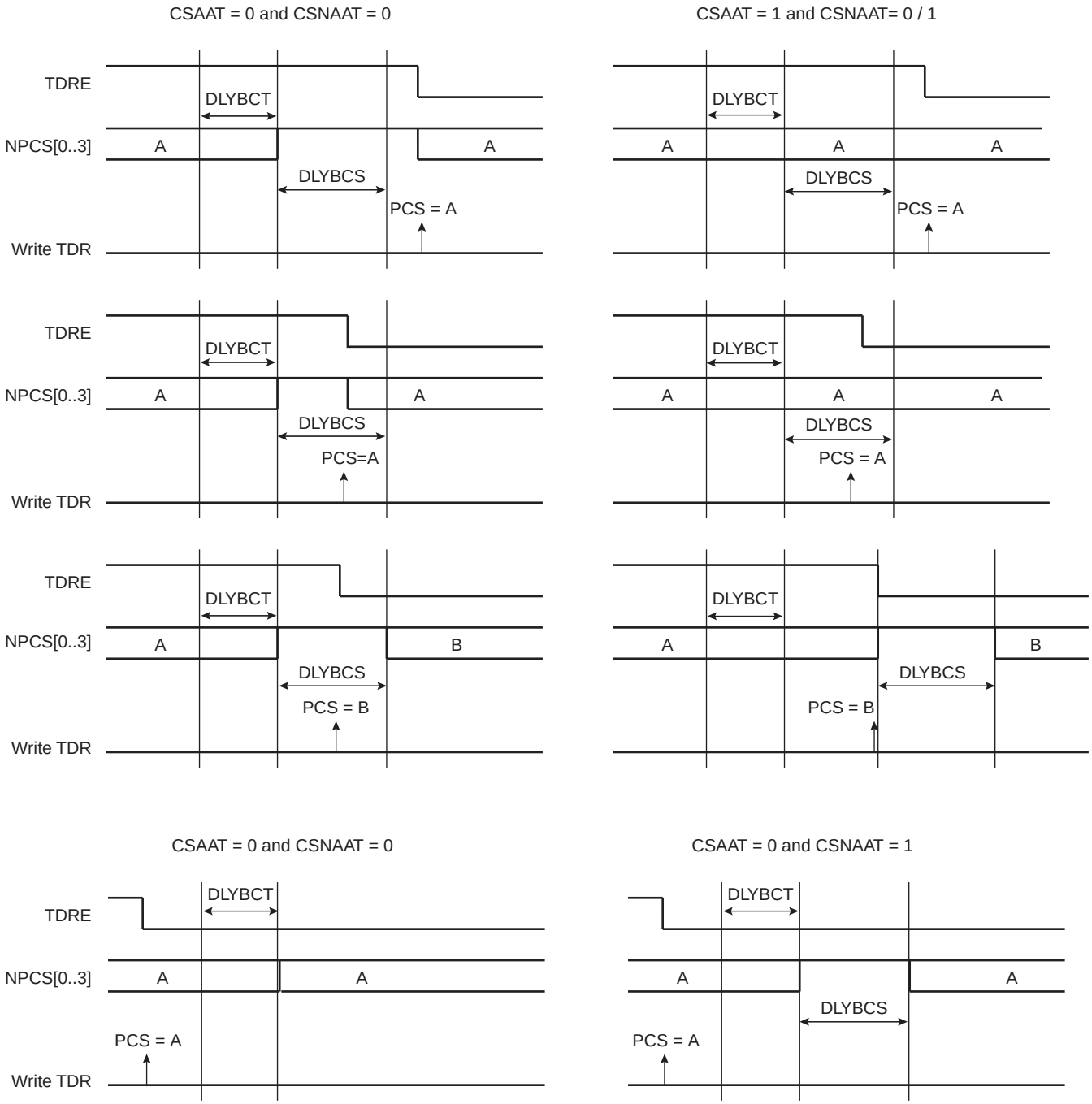


Figure 22-8 on page 403 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

22.7.3.9 Mode fault detection

A mode fault is detected when the SPI is configured 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 I/O Controller, so that external pull up resistors are needed to guarantee high level.

When a mode fault is detected, the Mode Fault Error bit in the SR (SR.MODF) is set until the SR is read and the SPI is automatically disabled until re-enabled by writing a one to the Spi Enable bit in the CR register (CR.SPIEN).

By default, the mode fault detection circuitry is enabled. The user can disable mode fault detection by writing a one to the Mode Fault Detection bit in the MR register (MR.MODFDIS).

22.7.4 SPI Slave Mode

When operating in slave mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the Bits Per Transfer field of the Chip Select Register 0 (CSR0.BITS). These bits are processed following a phase and a polarity defined respectively by the CSR0.NCPHA and CSR0.CPOL bits. Note that the BITS, CPOL, and NCPHA bits of the other Chip Select Registers have no effect when the SPI is configured 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 SR.RDRF bit rises. If the RDR register has not been read before new data is received, the SR.OVRES bit is set. As long as this bit is set, data is loaded in RDR. The user has to read the SR register to clear the SR.OVRES bit.

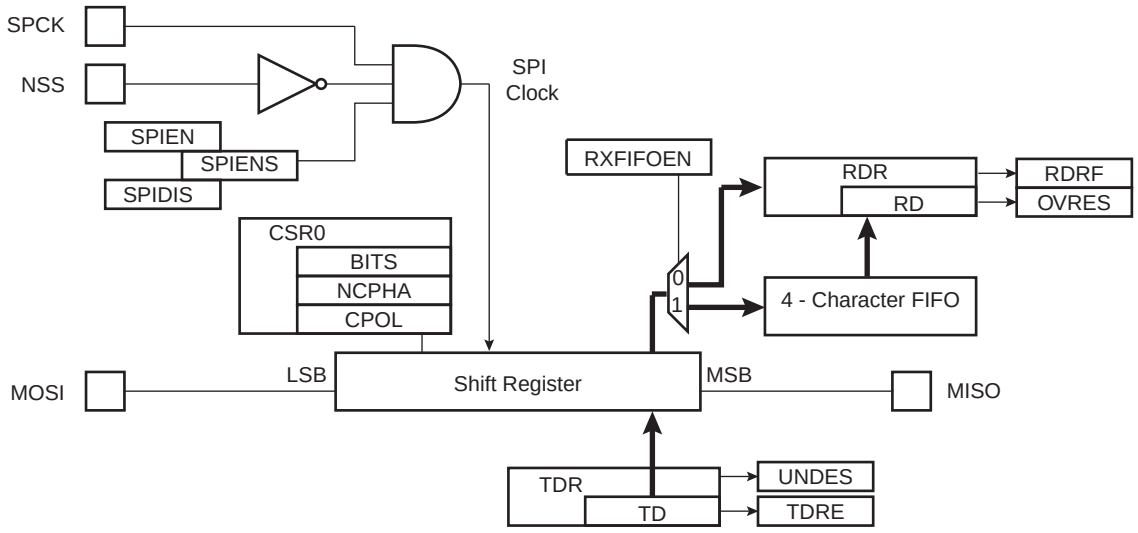
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the TDR register, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets to zero.

When a first data is written in TDR, it is transferred immediately in the Shift Register and the SR.TDRE bit rises. If new data is written, it remains in 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 TDR is transferred in the Shift Register and the SR.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 TDR. In case no character is ready to be transmitted, i.e. no character has been written in TDR since the last load from TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted. In this case the Underrun Error Status bit is set in SR (SR.UNDES).

Figure 22-9 on page 405 shows a block diagram of the SPI when operating in slave mode.

Figure 22-9. Slave Mode Functional Block Diagram



22.8 User Interface

Table 22-3. SPI Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Mode Register	MR	Read/Write	0x00000000
0x08	Receive Data Register	RDR	Read-only	0x00000000
0x0C	Transmit Data Register	TDR	Write-only	0x00000000
0x10	Status Register	SR	Read-only	0x000000F0
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x30	Chip Select Register 0	CSR0	Read/Write	0x00000000
0x34	Chip Select Register 1	CSR1	Read/Write	0x00000000
0x38	Chip Select Register 2	CSR2	Read/Write	0x00000000
0x3C	Chip Select Register 3	CSR3	Read/Write	0x00000000
0x E4	Write Protection Control Register	WPCR	Read/Write	0X00000000
0xE8	Write Protection Status Register	WPSR	Read-only	0x00000000
0xFC	Version Register	VERSION	Read-only	- (1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

22.8.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

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
-	-	-	-	-	-	-	FLUSHFIFO
7	6	5	4	3	2	1	0
SWRST	-	-	-	-	-	SPIDIS	SPIEN

- **LASTXFER: Last Transfer**
  1: The current NPCS will be deasserted after the character written in TD has been transferred. When CSRn.CSAAT is one, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.
  0: Writing a zero to this bit has no effect.
- **FLUSHFIFO: Flush Fifo Command**
  1: If The FIFO Mode is enabled (MR.FIFOEN written to one) and if an overrun error has been detected, this command allows to empty the FIFO.
  0: Writing a zero to this bit has no effect.
- **SWRST: SPI Software Reset**
  1: Writing a one to this bit will reset the SPI. A software-triggered hardware reset of the SPI interface is performed. The SPI is in slave mode after software reset. Peripheral DMA Controller channels are not affected by software reset.
  0: Writing a zero to this bit has no effect.
- **SPIDIS: SPI Disable**
  1: Writing a one to this bit will disable the SPI. As soon as SPIDIS is written to one, the 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 CR register is written, the SPI is disabled.
  0: Writing a zero to this bit has no effect.
- **SPIEN: SPI Enable**
  1: Writing a one to this bit will enable the SPI to transfer and receive data.
  0: Writing a zero to this bit has no effect.

22.8.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

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	RXFIFOEN	WDRBT	MODFDIS	-	PCSDEC	PS	MSTR

- **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 CLK_SPI periods will be inserted by default.
  Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{CLKSPI}$$

- **PCS: Peripheral Chip Select**
  This field is only used if Fixed Peripheral Select is active (PS = 0).
  If PCSDEC = 0:
  PCS = xxx0NPCS[3:0] = 1110
  PCS = xx01NPCS[3:0] = 1101
  PCS = x011NPCS[3:0] = 1011
  PCS = 0111NPCS[3:0] = 0111
  PCS = 1111forbidden (no peripheral is selected)
  (x = don't care)
  If PCSDEC = 1:
  NPCS[3:0] output signals = PCS.
- **LLB: Local Loopback Enable**
  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).
  0: Local loopback path disabled.
- **RXFIFOEN: FIFO in Reception Enable**
  1: The FIFO is used in reception (four characters can be stored in the SPI).

0: The FIFO is not used in reception (only one character can be stored in the SPI).

- **WDRBT: Wait Data Read Before Transfer**
  1: In master mode, a transfer can start only if the RDR register is empty, i.e. does not contain any unread data. This mode prevents overrun error in reception.
  0: No Effect. In master mode, a transfer can be initiated whatever the state of the RDR register is.
- **MODFDIS: Mode Fault Detection**
  1: Mode fault detection is disabled.
  0: Mode fault detection is enabled.
- **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 CSRn registers define the characteristics of the 15 chip selects according to the following rules:
  CSR0 defines peripheral chip select signals 0 to 3.
  CSR1 defines peripheral chip select signals 4 to 7.
  CSR2 defines peripheral chip select signals 8 to 11.
  CSR3 defines peripheral chip select signals 12 to 14.
- **PS: Peripheral Select**
  1: Variable Peripheral Select.
  0: Fixed Peripheral Select.
- **MSTR: Master/Slave Mode**
  1: SPI is in master mode.
  0: SPI is in slave mode.

22.8.3 Receive Data Register

Name: RDR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

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[15:8]							

7	6	5	4	3	2	1	0
RD[7:0]							

- **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.
- **RD: Receive Data**
  Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

22.8.4 Transmit Data Register

Name: TDR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

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[15:8]							

7	6	5	4	3	2	1	0
TD[7:0]							

- **LASTXFER: Last Transfer**

  1: The current NPCS will be deasserted after the character written in TD has been transferred. When CSRn.CSAAT is one, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.
  0: Writing a zero to this bit has no effect.

  This field is only used if Variable Peripheral Select is active (MR.PS = 1).

- **PCS: Peripheral Chip Select**

  If PCSDEC = 0:
  PCS = xxx0NPCS[3:0] = 1110
  PCS = xx01NPCS[3:0] = 1101
  PCS = x011NPCS[3:0] = 1011
  PCS = 0111NPCS[3:0] = 0111
  PCS = 1111forbidden (no peripheral is selected)
  (x = don’t care)
  If PCSDEC = 1:
  NPCS[3:0] output signals = PCS

  This field is only used if Variable Peripheral Select is active (MR.PS = 1).

- **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 TDR register in a right-justified format.

22.8.5 Status Register

Name: SR

Access Type: Read-only

Offset: 0x10

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	SPIENS

15	14	13	12	11	10	9	8
-	-	-	-	-	UNDES	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

- **SPIENS: SPI Enable Status**
  1: This bit is set when the SPI is enabled.
  0: This bit is cleared when the SPI is disabled.
- **UNDES: Underrun Error Status (Slave Mode Only)**
  1: This bit is set when a transfer begins whereas no data has been loaded in the TDR register.
  0: This bit is cleared when the SR register is read.
- **TXEMPTY: Transmission Registers Empty**
  1: This bit is set when TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.
  0: This bit is cleared as soon as data is written in TDR.
- **NSSR: NSS Rising**
  1: A rising edge occurred on NSS pin since last read.
  0: This bit is cleared when the SR register is read.
- **OVRES: Overrun Error Status**
  1: This bit is set when an overrun has occurred. An overrun occurs when RDR is loaded at least twice from the serializer since the last read of the RDR.
  0: This bit is cleared when the SR register is read.
- **MODF: Mode Fault Error**
  1: This bit is set when a Mode Fault occurred.
  0: This bit is cleared when the SR register is read.
- **TDRE: Transmit Data Register Empty**
  1: This bit is set when the last data written in the TDR register has been transferred to the serializer.
  0: This bit is cleared when data has been written to TDR and not yet transferred to the serializer.
  TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.
- **RDRF: Receive Data Register Full**
  1: Data has been received and the received data has been transferred from the serializer to RDR since the last read of RDR.
  0: No data has been received since the last read of RDR

22.8.6 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x14

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
-	-	-	-	-	UNDES	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

22.8.7 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x18

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
-	-	-	-	-	UNDES	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

22.8.8 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x1C

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
-	-	-	-	-	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

22.8.9 Chip Select Register 0

Name: CSR0

Access Type: Read/Write

Offset: 0x30

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **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:

$$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. 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:

$$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

- **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
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  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.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  0: Data is changed on the leading edge of SPCK and captured 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.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.
  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.

22.8.10 Chip Select Register 1

Name: CSR1

Access Type: Read/Write

Offset: 0x34

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **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:

$$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. 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:

$$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

- **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
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  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.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  0: Data is changed on the leading edge of SPCK and captured 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.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.
  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.

22.8.11 Chip Select Register 2

Name: CSR2

Access Type: Read/Write

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **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:

  $$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **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:

  $$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. 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:

  $$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

- **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
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  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.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  0: Data is changed on the leading edge of SPCK and captured 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.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.
  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.

22.8.12 Chip Select Register 3

Name: CSR3

Access Type: Read/Write

Offset: 0x3C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **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:

  $$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **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:

  $$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. 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:

  $$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

- **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
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  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.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  0: Data is changed on the leading edge of SPCK and captured 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.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.
  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.

22.8.13 Write Protection Control Register

Register Name: WPCR

Access Type: Read-write

Offset: 0xE4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SPIWPKEY[23:16]							

23	22	21	20	19	18	17	16
SPIWPKEY[15:8]							

15	14	13	12	11	10	9	8
SPIWPKEY[7:0]							

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	SPIWPEN

- **SPIWPKEY: SPI Write Protection Key Password**

  If a value is written in SPIWPEN, the value is taken into account only if SPIWPKEY is written with “SPI” (SPI written in ASCII Code, i.e. 0x535049 in hexadecimal).
- **SPIWPEN: SPI Write Protection Enable**

  1: The Write Protection is Enabled

  0: The Write Protection is Disabled

22.8.14 Write Protection Status Register

Register Name: WPSR

Access Type: Read-only

Offset: 0xE8

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
SPIWPVSRC							

7	6	5	4	3	2	1	0
-	-	-	-	-	SPIWPVS		

- **SPIWPVSRC: SPI Write Protection Violation Source**
  This Field indicates the Peripheral Bus Offset of the register concerned by the violation (MR or CSRx)
- **SPIWPVS: SPI Write Protection Violation Status**

SPIWPVS value	Violation Type
1	The Write Protection has blocked a Write access to a protected register (since the last read).
2	Software Reset has been performed while Write Protection was enabled (since the last read or since the last write access on MR, IER, IDR or CSRx).
3	Both Write Protection violation and software reset with Write Protection enabled have occurred since the last read.
4	Write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.
5	The Write Protection has blocked a Write access to a protected register and write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.
6	Software Reset has been performed while Write Protection was enabled (since the last read or since the last write access on MR, IER, IDR or CSRx) and some write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.
7	- The Write Protection has blocked a Write access to a protected register. and - Software Reset has been performed while Write Protection was enabled. and - Write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.

22.8.15 Version Register

Register Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: –

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
				VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT**

  Reserved. No functionality associated.

- **VERSION**

  Version number of the module. No functionality associated.

22.9 Module Configuration

The specific configuration for each SPI instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager section for details.

Table 22-4. Module Clock Name

Module Name	Clock Name
SPI0	CLK_SPI0
SPI1	CLK_SPI1

Table 22-5. Register Reset Values

Register	Reset Value
VERSION	0x00000200

23. Two-Wire Slave Interface (TWIS)

Rev 1.0.0.1

23.1 Features

- **Compatible with I²C standard**
  - **100 and 400 kbit/s transfer speeds**
  - **7 and 10-bit and General Call addressing**
- **Compatible with SMBus standard**
  - **Hardware Packet Error Checking (CRC) generation and verification with ACK response**
  - **SMBALERT interface**
  - **25 ms clock low timeout delay**
  - **25 ms slave cumulative clock low extend time**
- **Compatible with PMBus**
- **DMA interface for reducing CPU load**
- **Arbitrary transfer lengths, including 0 data bytes**
- **Optional clock stretching if transmit or receive buffers not ready for data transfer**
- **32-bit Peripheral Bus interface for configuration of the interface**

23.2 Overview

The Atmel Two-wire Interface Slave (TWIS) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 kbit/s, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus I²C or SMBus compatible master. TWIS is always a bus slave and can transfer sequential or single bytes.

Below, Table 23-1 on page 428 lists the compatibility level of the Atmel Two-wire Slave Interface and a full I²C compatible device.

Table 23-1. Atmel TWIS Compatibility with I²C Standard

I²C Standard	Atmel TWIS
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 NAK Management	Supported
Slope control and input filtering (Fast mode)	Supported
Clock stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

Below, Table 23-2 on page 429 lists the compatibility level of the Atmel Two-wire Slave Interface and a full SMBus compatible device.

Table 23-2. Atmel TWIS Compatibility with SMBus Standard

SMBus Standard	Atmel TWIS
Bus Timeouts	Supported
Address Resolution Protocol	Supported
Alert	Supported
Packet Error Checking	Supported

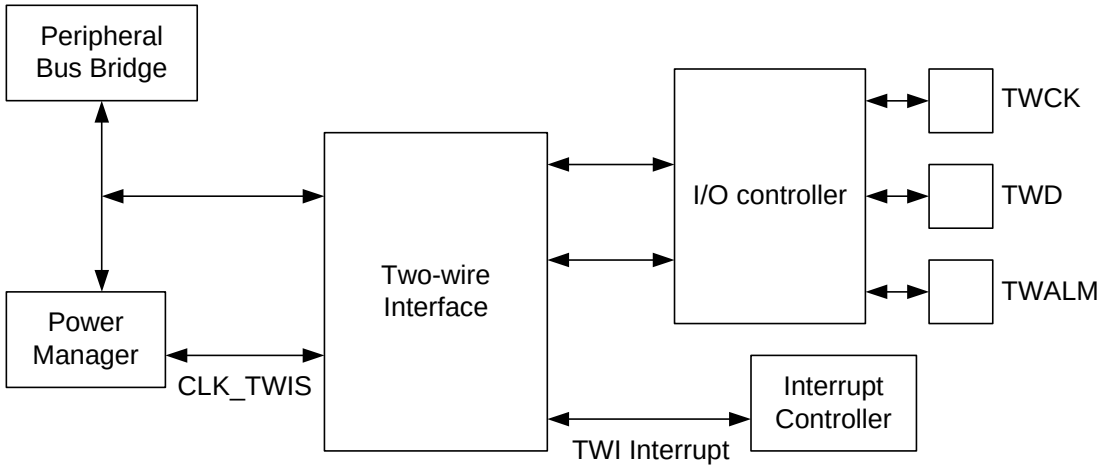
23.3 List of Abbreviations

Table 23-3. 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

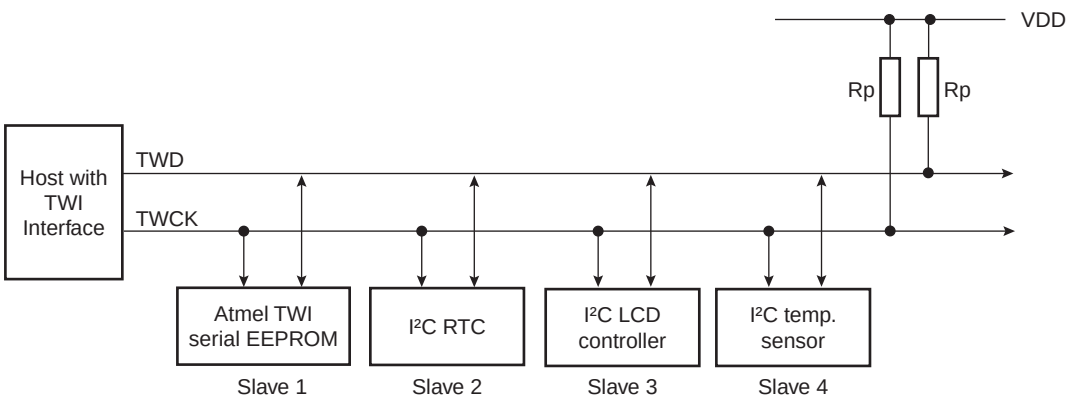
23.4 Block Diagram

Figure 23-1. Block Diagram



23.5 Application Block Diagram

Figure 23-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

23.6 I/O Lines Description

Table 23-4. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output
TWALM	SMBus SMBALERT	Input/Output

23.7 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

23.7.1 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 23-2 on page 430). 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.

TWALM is used to implement the optional SMBus SMBALERT signal.

TWD, TWCK and TWALM pins may be multiplexed with I/O Controller lines. To enable the TWIS, the programmer must perform the following steps:

- Program the I/O Controller to:
  - Dedicate TWD, TWCK and optionally TWALM as peripheral lines.
  - Define TWD, TWCK and optionally TWALM as open-drain.

23.7.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the TWIS, the TWIS will stop functioning and resume operation after the system wakes up from sleep mode.

23.7.3 Clocks

The clock for the TWIS bus interface (CLK_TWIS) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the TWIS before disabling the clock, to avoid freezing the TWIS in an undefined state.

23.7.4 Interrupts

The TWIS interrupt request lines are connected to the interrupt controller. Using the TWIS interrupts requires the interrupt controller to be programmed first.

23.7.5 Debug Operation

When an external debugger forces the CPU into debug mode, the TWIS continues normal operation. If the TWIS is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

23.8 Functional Description

23.8.1 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 23-4 on page 431).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 23-3 on page 431).

- 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 23-3. START and STOP Conditions

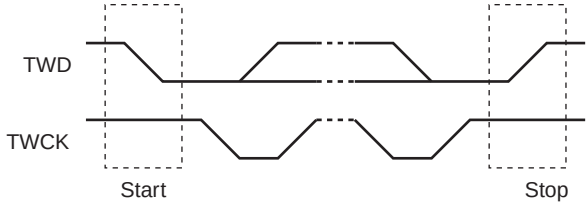
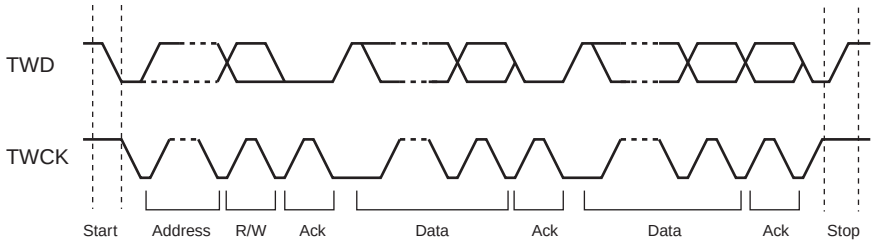


Figure 23-4. Transfer Format



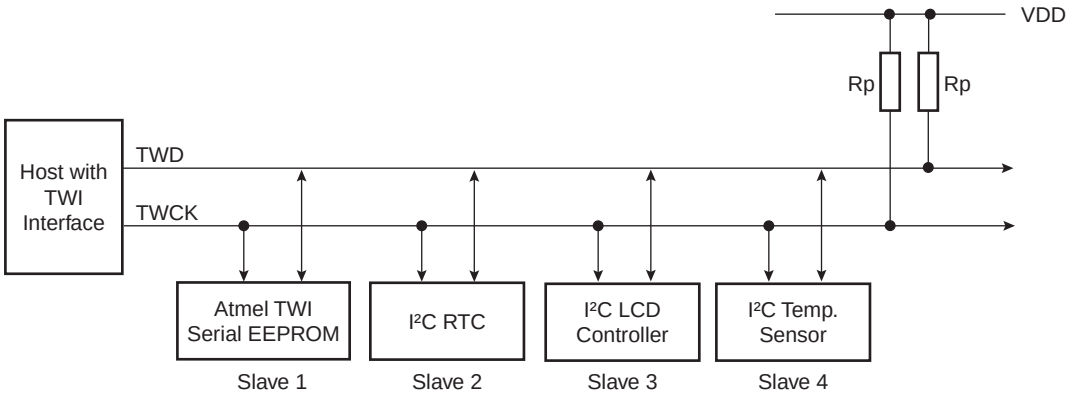
23.8.2 Operation

TWIS has two modes of operation:

- Slave transmitter mode
- Slave receiver mode

A master is a device which starts and stops a transfer and generates the TWCK clock. A slave is assigned an address and responds to requests from the master. These modes are described in the following chapters.

Figure 23-5. Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

23.8.2.1 *Bus Timing*

The Timing Register (TR) is used to control the timing of bus signals driven by TWIS. TR describes bus timings as a function of cycles of the prescaled CLK_TWIS. The clock prescaling can be selected through TR.EXP.

$$f_{prescaled} = \frac{f_{CLK-TWIS}}{2^{(EXP+1))}}$$

TR has the following fields:

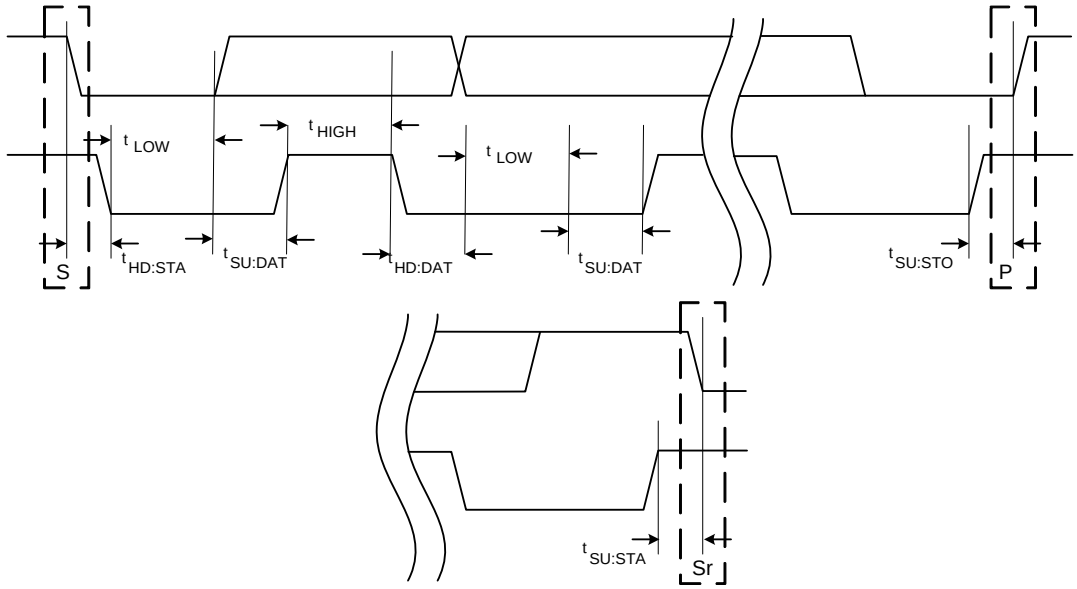
TLOWS: Prescaled clock cycles used to time SMBUS timeout $T_{LOW:SEXT}$.

TTOUT: Prescaled clock cycles used to time SMBUS timeout $T_{TIMEOUT}$.

SUDAT: Non-prescaled clock cycles for data setup and hold count. Used to time T_{SU_DAT}.

EXP: Specifies the clock prescaler setting used for the SMBUS timeouts.

Figure 23-6. Bus Timing Diagram



23.8.2.2 *Setting Up and Performing a Transfer*

Operation of TWIS is mainly controlled by the Control Register (CR). The following list presents the main steps in a typical communication:

1. Before any transfers can be performed, bus timings must be configured by programming the Timing Register (TR).
2. If a DMA controller is to be used for the transfers, it must be set up.
3. The Control Register (CR) must be configured with information such as the slave address, SMBus mode, Packet Error Checking (PEC), number of bytes to transfer, and which addresses to match.

The interrupt system can be set up to give interrupt request on specific events or error conditions, for example when a byte has been received.

The NBYTES register is only used in SMBus mode, when PEC is enabled. In I²C mode or in SMBus mode when PEC is disabled, the NBYTES register is not used, and should be written to 0. NBYTES is updated by hardware, so in order to avoid hazards, software updates of NBYTES can only be done through writes to the NBYTES register.

23.8.2.3 *Address Matching*

TWIS can be set up to match several different addresses. More than one address match may be enabled simultaneously, allowing TWIS to be assigned to several addresses. The address matching phase is initiated after a START or REPEATED START condition. When TWIS receives an address that generates an address match, an ACK is automatically returned to the master.

In I²C mode:

- The address in CR.ADR is checked for address match if CR.SMATCH is set.
- The General Call address is checked for address match if CR.GCMATCH is set.

In SMBus mode:

- The address in CR.ADR is checked for address match if CR.SMATCH is set.
- The Alert Response Address is checked for address match if CR.SMAL is set.
- The Default Address is checked for address match if CR.SMDA is set.
- The Host Header Address is checked for address match if CR.SMHH is set.

23.8.2.4 Clock Stretching

Any slave or bus master taking part in a transfer may extend the TWCK low period at any time. TWIS may extend the TWCK low period after each byte transfer if CR.STREN=1 and:

- Module is in slave transmitter mode, data should be transmitted, but THR is empty, or
- Module is in slave receiver mode, a byte has been received and placed into the internal shifter, but RHR is full, or
- Stretch-on-address-match bit CR.SOAM=1 and slave was addressed. Bus clock remains stretched until all address match bits in SR have been cleared.

If CR.STREN=0 and:

- Module is in slave transmitter mode, data should be transmitted but THR is empty: Transmit the value present in THR (the last transmitted byte or reset value), and set SR.URUN.
- Module is in slave receiver mode, a byte has been received and placed into the internal shifter, but RHR is full: Discard the received byte and set SR.ORUN.

23.8.2.5 Bus Errors

If a bus error (misplaced START or STOP) condition is detected, the SR.BUSERR bit is set and TWIS waits for a new START condition.

23.8.3 Slave Transmitter Mode

If TWIS matches an address in which the R/$\overline{W}$ bit in the TWI address phase transfer is set, it will enter slave transmitter mode and set SR.TRA

After the address phase, the following is done:

1. If SMBus mode and PEC is used, NBYTES must be set up with the number of bytes to transmit. This is necessary in order to know when to transmit PEC byte. NBYTES can also be used to count the number of bytes received if using DMA.
2. Byte to transmit depends on I²C/SMBus mode and CR.PEC:
   – If in I²C mode or CR.PEC=0 or NBYTES!=0: TWIS waits until THR contains a valid data byte, possibly stretching low period of TWCK. SR.TXRDY indicates the state of THR.
   – SMBus mode and CR.PEC=1: If NBYTES=0, the generated PEC byte is automatically transmitted instead of a data byte from THR. TWCK will not be stretched by TWIS.
3. Transmit the correct data byte. Set SR.BTF when done.
4. Update NBYTES. If CR.CUP is set, NBYTES is incremented, otherwise NBYTES is decremented.
5. After each data byte has been transferred, the master transmits an ACK or NAK bit. If a NAK bit is received, transfer is finished, and TWIS will wait for a STOP or REPEATED START. If an ACK bit is received, more data should be transmitted, jump to step 2.
6. If STOP is received, SR.TCOMP and SR.STO will be set.
7. If REPEATED START is received, SR.REP will be set.

The TWI transfers require the receiver to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the slave releases the data line (HIGH), enabling the master to pull it down in order to generate the acknowledge. The slave polls the data line during this clock pulse and sets the Not Acknowledge bit (NAK**)** in the Status Register if the master does not acknowledge the data byte. A NAK means that the master does not wish to receive additional data bytes. As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable Register (IER).

TXRDY is used as Transmit Ready for the Peripheral DMA Controller transmit channel.

The end of the complete transfer is marked by the SR.TCOMP bit set to one. See Figure 23-7 on page 435 and Figure 23-8 on page 435.

Figure 23-7. Slave Transmitter with One Data Byte

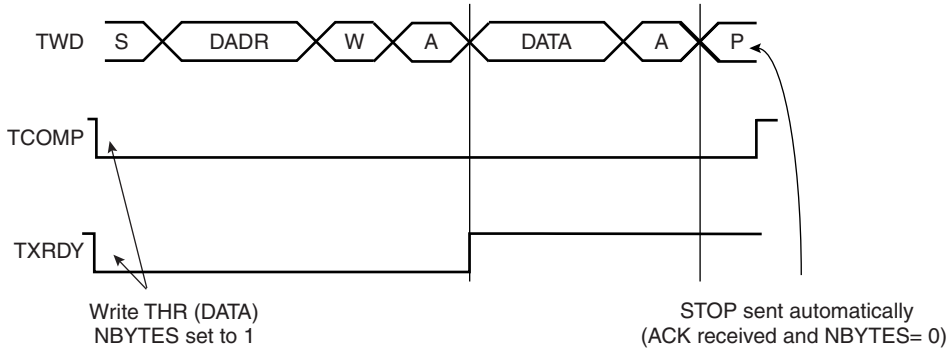
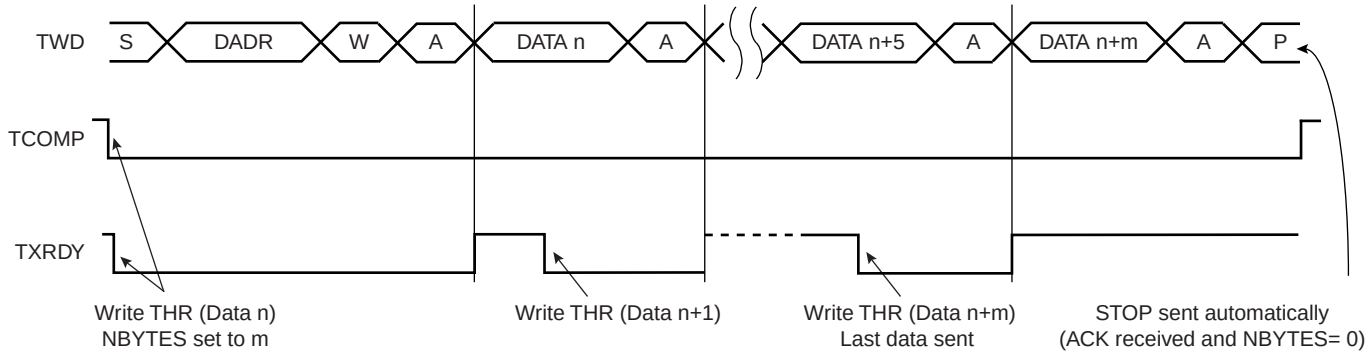


Figure 23-8. Master Write with Multiple Data Bytes



23.8.4 Slave Receiver Mode

If TWIS matches an address in which the R/$\overline{W}$ bit in the TWI address phase transfer is cleared, it will enter slave receiver mode and clear SR.TRA.

After the address phase, the following is repeated:

1. If SMBus mode and PEC is used, NBYTES must be set up with the number of bytes to receive. This is necessary in order to know which of the received bytes is the PEC byte. NBYTES can also be used to count the number of bytes received if using DMA.
2. Receive a byte. Set SR.BTF when done.

3. Update NBYTES. If CR.CUP is written to one, NBYTES is incremented, otherwise NBYTES is decremented. NBYTES is usually configured to count downwards if PEC is used.
4. After a data byte has been received, the slave transmits an ACK or NAK bit. For ordinary data bytes, the CR.ACK field controls if an ACK or NAK should be returned. If PEC is enabled and the last byte received was a PEC byte (indicated by NBYTES=0), TWIS will automatically return an ACK if the PEC value was correct, otherwise a NAK will be returned.
5. If STOP is received, SR.TCOMP will be set.
6. If REPEATED START is received, SR.REP will be set.

The TWI transfers require the receiver to acknowledge each received data 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.

RXRDY is used as Receive Ready for the Peripheral DMA Controller receive channel.

Figure 23-9. Slave Receiver with One Data Byte

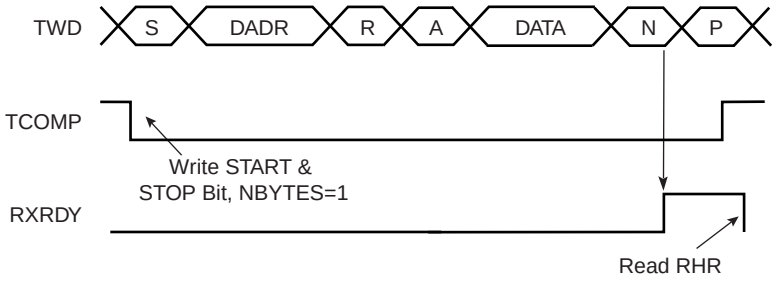
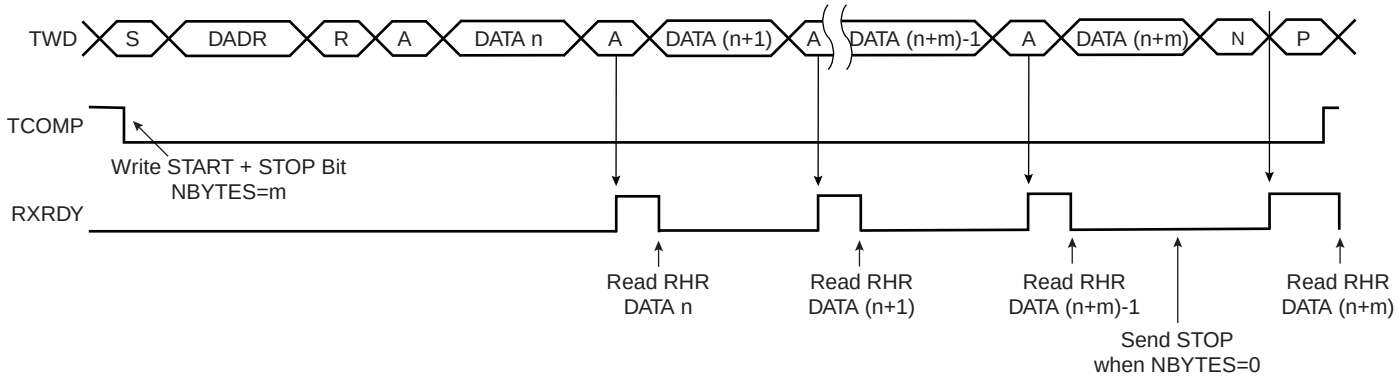


Figure 23-10. Slave Receiver with Multiple Data Bytes



23.8.5 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The programmer can set up ring buffers for the DMA controller, containing data to transmit or free buffer space to place received data. By initializing NBYTES to 0 before a transfer, and setting CR.CUP,

NBYTES is incremented by 1 each time a data has been transmitted or received. This allows the programmer to detect how much data was actually transferred by the DMA system.

To assure correct behavior, respect the following programming sequences:

23.8.5.1 Data Transmit with the Peripheral DMA Controller

1. Initialize the transmit Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIS (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller TXTEN bit.
4. Wait for the Peripheral DMA Controller end TX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller TXDIS bit.

23.8.5.2 Data Receive with the Peripheral DMA Controller

1. Initialize the receive Peripheral DMA Controller (memory pointers, size - 1, etc.).
2. Configure the TWIS (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller RXTEN bit.
4. Wait for the Peripheral DMA Controller end RX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller RXDIS bit.

23.8.6 SMBus Mode

SMBus mode is enabled when CR.SMEN is written to one. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into TR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A dedicated bus line, SMBALERT, allows a slave to get a master's attention.
- A set of addresses have been reserved for protocol handling, such as Alert Response Address (ARA) and Host Header (HH) Address. Address matching on these addresses can be enabled by configuring CR appropriately.

23.8.6.1 Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to CR.PECEN enables automatic PEC handling in the current transfer. The PEC generator is always updated on every bit transmitted or received, so that PEC handling on following linked transfers will be correct.

In slave receiver mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave will compare it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave will return an ACK to the master. If the PEC values differ, data was corrupted, and the slave will return a NAK value. The SR.SMBPECERR bit is set automatically if a PEC error occurred.

In slave transmitter mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master will compare

it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the master must take appropriate action.

The PEC byte is automatically inserted in a slave transmitter transmission if PEC enabled when NBYTES reaches zero. The PEC byte is identified in a slave receiver transmission if PEC enabled when NBYTES reaches zero. NBYTES must therefore be set to the total number of data bytes in the transmission, including the PEC byte.

23.8.6.2 Timeouts

The Timing Register (TR) configures the SMBus timeout values. If a timeout occurs, the slave will leave the bus. The SR.SMBTOUT bit is also set.

23.8.6.3 SMBALERT

A slave can get the master's attention by pulling the SMBALERT line low. This is done by setting the CR.SMBAL bit. This will also enable address match on the Alert Response Address (ARA).

23.8.7 Identifying Bus Events

This chapter lists the different bus events, and how these affects bits in the TWIS registers. This is intended to help writing drivers for the TWIS.

Table 23-5. Bus Events

Event	Effect
Slave transmitter has sent a data byte	SR.THR is cleared. SR.BTF is set. The value of the ACK bit sent immediately after the data byte is given by CR.ACK.
Slave receiver has received a data byte	SR.RHR is set. SR.BTF is set. SR.NAK updated according to value of ACK bit received from master.
Start+Sadr on bus, but address is to another slave	None.
Start+Sadr on bus, current slave is addressed, but address match enable bit in CR is not set	None.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set	Correct address match bit in SR is set. SR.TRA updated according to transfer direction. Slave enters appropriate transfer direction mode and data transfer can commence.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set, SR.STREN and SR.SOAM are set.	Correct address match bit in SR is set. SR.TRA updated according to transfer direction. Slave stretches TWCK immediately after transmitting the address ACK bit. TWCK remains stretched until all address match bits in SR have been cleared. Slave the enters appropriate transfer direction mode and data transfer can commence.
Repeated Start received after being addressed	SR.REP set. SR.TCOMP unchanged.

Table 23-5. Bus Events

Event	Effect
Stop received after being addressed	SR.STO set. SR.TCOMP set.
Start, Repeated Start or Stop received in illegal position on bus	SR.BUSERR set.
Data is to be received in slave receiver mode, SR.STREN is set, and RHR is full	TWCK is stretched until RHR has been read.
Data is to be transmitted in slave receiver mode, SR.STREN is set, and THR is empty	TWCK is stretched until THR has been written.
Data is to be received in slave receiver mode, SR.STREN is cleared, and RHR is full	TWCK is not stretched, read data is discarded. SR.ORUN is set.
Data is to be transmitted in slave receiver mode, SR.STREN is cleared, and THR is empty	TWCK is not stretched, previous contents of THR is written to bus. SR.URUN is set.
SMBus timeout received	SR.SMBTOUT is set. TWCK and TWD are immediately released.
Slave transmitter in SMBus PEC mode has transmitted a PEC byte, that was not identical to the PEC calculated by the master receiver.	Master receiver will transmit a NAK as usual after the last byte of a master receiver transfer. Master receiver will retry the transfer at a later time.
Slave receiver discovers SMBus PEC Error	SR.SMBPECERR is set. NAK returned after the data byte.

23.9 User Interface

Table 23-6. TWIS Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Read/Write	0x00000000
0x04	NBYTES Register	NBYTES	Read/Write	0x00000000
0x08	Timing Register	TR	Read/Write	0x00000000
0x0C	Receive Holding Register	RHR	Read-only	0x00000000
0x10	Transmit Holding Register	THR	Write-only	0x00000000
0x14	Packet Error Check Register	PECR	Read-only	0x00000000
0x18	Status Register	SR	Read-only	0x00000002
0x1c	Interrupt Enable Register	IER	Write-only	0x00000000
0x20	Interrupt Disable Register	IDR	Write-only	0x00000000
0x24	Interrupt Mask Register	IMR	Read-only	0x00000000
0x28	Status Clear Register	SCR	Write-only	0x00000000
0x2C	Parameter Register	PR	Read-only	(1)
0x30	Version Register	VR	Read-only	(1)

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

23.9.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	TENBIT	ADR[9:8]	

23	22	21	20	19	18	17	16
ADR[7:0]							

15	14	13	12	11	10	9	8
-	SOAM	CUP	ACK	PECEN	SMHH	SMDA	SMBALERT

7	6	5	4	3	2	1	0
SWRST	-	-	STREN	GCMATCH	SMATCH	SMEN	SEN

- **TENBIT: Ten Bit Address Match**
  Write this bit to zero to disable Ten Bit Address Match.
  Write this bit to one to enable Ten Bit Address Match.
- **ADR: Slave Address**
  Slave address used in slave address match. Bits 9:0 are used if in 10-bit mode, bits 6:0 otherwise.
- **SOAM: Stretch Clock on Address Match**
  Writing this bit to zero will not strech bus clock after address match.
  Writing this bit to one will strech bus clock after address match.
- **CUP: NBYTES Count Up**
  Writing this bit to zero causes NBYTES to count down (decrement) per byte transferred.
  Writing this bit to one causes NBYTES to count up (increment) per byte transferred.
- **ACK: Slave Receiver Data Phase ACK Value**
  Writing this bit to zero causes a low value to be returned in the ACK cycle of the data phase in slave receiver mode.
  Writing this bit to one causes a high value to be returned in the ACK cycle of the data phase in slave receiver mode.
- **PECEN: Packet Error Checking Enable**
  Writing this bit to zero disables SMBus PEC (CRC) generation and check.
  Writing this bit to one enables SMBus PEC (CRC) generation and check.
- **SMHH: SMBus Host Header**
  Writing this bit to zero causes TWIS not to acknowledge the SMBus Host Header.
  Writing this bit to one causes TWIS to acknowledge the SMBus Host Header.
- **SMDA: SMBus Default Address**
  Writing this bit to zero causes TWIS not to acknowledge the SMBus Default Address.
  Writing this bit to one causes TWIS to acknowledge the SMBus Default Address.
- **SMBALERT: SMBus Alert**
  Writing this bit to zero causes TWIS to release the SMBALERT line and not to acknowledge the SMBus Alert Response Address (ARA).
  Writing this bit to zero causes TWIS to pull down the SMBALERT line and to acknowledge the SMBus Alert Response Address (ARA).

- **SWRST: Software Reset**
  - This bit will always read as 0.
  - Writing a zero to this bit has no effect.
  - Writing a one to this bit resets the TWIS.
- **STREN: Clock Stretch Enable**
  - Writing this bit to zero disables clock stretching if RHR/THR buffer full/empty. May cause over/underrun.
  - Writing this bit to one enables clock stretching if RHR/THR buffer full/empty.
- **GCMATCH: General Call Address Match**
  - Writing this bit to zero causes TWIS not to acknowledge the General Call Address.
  - Writing this bit to one causes TWIS to acknowledge the General Call Address.
- **SMATCH: Slave Address Match**
  - Writing this bit to zero causes TWIS not to acknowledge the Slave Address.
  - Writing this bit to one causes TWIS to acknowledge the Slave Address.
- **SMEN: SMBus Mode Enable**
  - Writing this bit to zero disables SMBus mode.
  - Writing this bit to one enables SMBus mode.
- **SEN: Slave Enable**
  - Writing this bit to zero disables the slave interface.
  - Writing this bit to one enables the slave interface.

23.9.2 NBYTES Register

Name: NBYTES

Access Type: Read/Write

Offset: 0x04

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
NBYTES							

- **NBYTES: Number of Bytes to Transfer**
  Writing to this field updates the NBYTES counter. Can also be read to to learn the progress of the transfer. Can be incremented or decremented automatically by hardware.

23.9.3 Timing Register

Name: TR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
EXP				-			

23	22	21	20	19	18	17	16
SUDAT							

15	14	13	12	11	10	9	8
TTOUT							

7	6	5	4	3	2	1	0
TLOWS							

- **EXP: Clock Prescaler**
  Used to specify how to prescale the SMBus TLOWS counter. The counter is prescaled according to the following formula:

$$f_{prescaled} = \frac{f_{clkpb}}{2^{(EXP+1)}}$$

- **SUDAT: Data Setup Cycles**
  Non-prescaled clock cycles for data setup count. Used to time T_{SU_DAT}. Data is driven SUDAT cycles after TWCK low detected. This timing is used for timing the ACK/NAK bits, and any data bits driven in slave transmitter mode.
- **TTOUT: SMBus Ttimeout Cycles**
  Prescaled clock cycles used to time SMBus $T_{TIMEOUT}$.
- **TLOWS: SMBus Tlow:sext Cycles**
  Prescaled clock cycles used to time SMBus $T_{LOW:SEXT}$.

23.9.4 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x0C

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: Received Data Byte**
  When the RXRDY bit in the Status Register (SR) is set, this field contains a byte received from the TWI bus.

23.9.5 Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x10

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: Data Byte to Transmit**
  Write data to be transferred on the TWI bus here.

23.9.6 Packet Error Check Register

Name: PECR

Access Type: Read-only

Offset: 0x14

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
PEC							

- **PEC: Calculated PEC Value**

  The calculated PEC value. Updated automatically by hardware after each byte has been transferred. Reset by hardware after a STOP condition. Provided if the user manually wishes to control when the PEC byte is transmitted, or wishes to access the PEC value for other reasons. In ordinary operation, the PEC handling is done automatically by hardware.

23.9.7 Status Register

Name: SR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x000000002

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	TRA	-	TCOMP	SEN	TXRDY	RXRDY

- **BTF: Byte Transfer Finished**
  This bit is set when byte transfer has completed.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **REP: Repeated Start Received**
  This bit is set when REPEATED START condition received.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **STO: Stop Received**
  This bit is set when STOP condition received.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBDAM: SMBus Default Address Match**
  This bit is set when received address matched SMBus Default Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBHHM: SMBus Host Header Address Match**
  This bit is set when received address matched SMBus Host Header Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBALERTM: SMBus Alert Response Address Match**
  This bit is set when received address matched SMBus Alert Response Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **GCM: General Call Match**
  This bit is set when received address matched General Call Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SAM: Slave Address Match**
  This bit is set when received address matched Slave Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **BUSERR: Bus Error**
  This bit is set when a misplaced start or stop condition has occurred.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **SMBPECERR: SMBus PEC Error**
  This bit is set when SMBus PEC error has occurred.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBTOUT: SMBus Timeout**
  This bit is set when SMBus timeout has occurred.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **NAK: NAK Received**
  This bit is set when NAK was received from master during slave transmitter operation.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **ORUN: Overrun**
  This bit is set when overrun has occurred in slave receiver mode. Can only occur if CR.STREN=0.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **URUN: Underrun**
  This bit is set when underrun has occurred in slave transmitter mode. Can only occur if CR.STREN=0.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **TRA: Transmitter Mode**
  0: The slave is in slave receiver mode.
  1: The slave is in slave transmitter mode.
- **TCOMP: Transmission Complete**
  This bit is set when transmission is complete. Set after receiving a STOP after being addressed.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SEN: Slave Enabled**
  0: The slave interface is disabled.
  1: The slave interface is enabled.
- **TXRDY: TX Buffer Ready**
  0: The TX buffer is full and should not be written to.
  1: The TX buffer is empty, and can accept new data.
- **RXRDY: RX Buffer Ready**
  0: No RX data ready in RHR.
  1: RX data is ready to be read from RHR.

23.9.8 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x1C

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

23.9.9 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

23.9.10 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x24

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	TXRDY	RXRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
This bit is cleared when the corresponding bit in IDR is written to one.
This bit is set when the corresponding bit in IER is written to one.

23.9.11 Status Clear Register

Name: SCR

Access Type: Read/Write

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

23.9.12 Parameter Register

Name: PR

Access Type: Read-only

Offset: 0x2C

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
-	-	-	-	-	-	-	-

23.9.13 Version Register (VR)

Name: VR

Access Type: Read-only

Offset: 0x30

Reset Value: Device-specific

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION [11:8]			

7	6	5	4	3	2	1	0
VERSION [7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

23.10 Module Configuration

The specific configuration for each TWIS instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 23-7. Module Clock Name

Module name	Clock name
TWIS0	**CLK_TWIS0**
TWIS1	**CLK_TWIS1**

Table 23-8. Register Reset Values

Register	Reset Value
VR	0x00000100
PR	0x00000000

24. Two-Wire Master Interface (TWIM)

Rev 1.0.0.0

24.1 Features

- **Compatible with I²C standard**
  - **Multi-master support**
  - **100 and 400 kbit/s transfer speeds**
  - **7- and 10-bit and General Call addressing**
- **Compatible with SMBus standard**
  - **Hardware Packet Error Checking (CRC) generation and verification with ACK control**
  - **SMBus ALERT interface**
  - **25 ms clock low timeout delay**
  - **10 ms master cumulative clock low extend time**
  - **25 ms slave cumulative clock low extend time**
- **Compatible with PMBus**
- **Compatible with Atmel Two-Wire Interface Serial Memories**
- **DMA interface for reducing CPU load**
- **Arbitrary transfer lengths, including 0 data bytes**
- **Optional clock stretching if transmit or receive buffers not ready for data transfer**

24.2 Overview

The Atmel Two-wire Interface Master (TWIM) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 kbit/s, 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 a real rime clock (RTC), dot matrix/graphic LCD controller and temperature sensor, to name a few. TWIM is always a bus master and can transfer sequential or single bytes. Multiple master capability is supported. Arbitration of the bus is performed internally and relinquishes the bus automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.Table 24-1 on page 457 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I²C compatible device.

Table 24-1. Atmel TWIM Compatibility with I²C Standard

I²C Standard	Atmel TWIM
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)	Supported
Clock Stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

Table 24-2 on page 458 lists the compatibility level of the Atmel Two-wire Master Interface and a full SMBus compatible master.

Table 24-2. Atmel TWIM Compatibility with SMBus Standard

SMBus Standard	Atmel TWIM
Bus Timeouts	Supported
Address Resolution Protocol	Supported
Alert	Supported
Host Functionality	Supported
Packet Error Checking	Supported

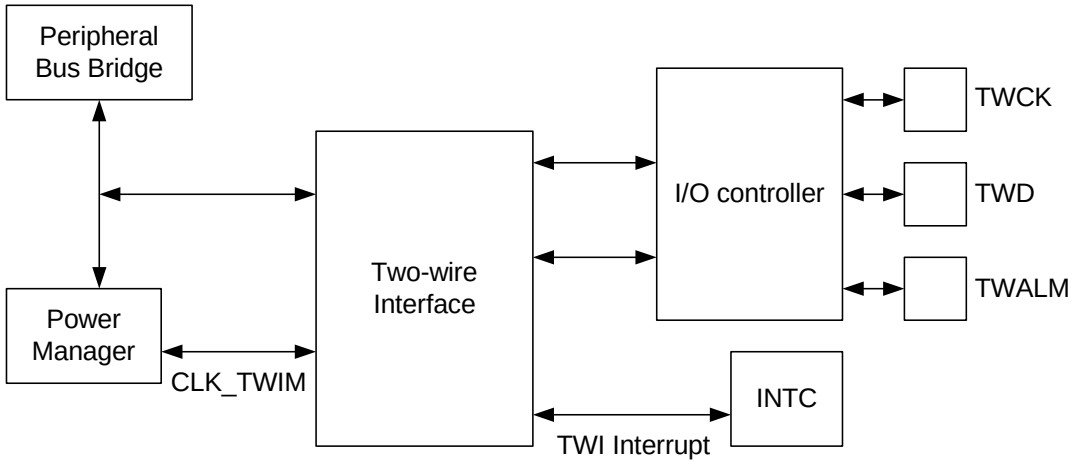
24.3 List of Abbreviations

Table 24-3. 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

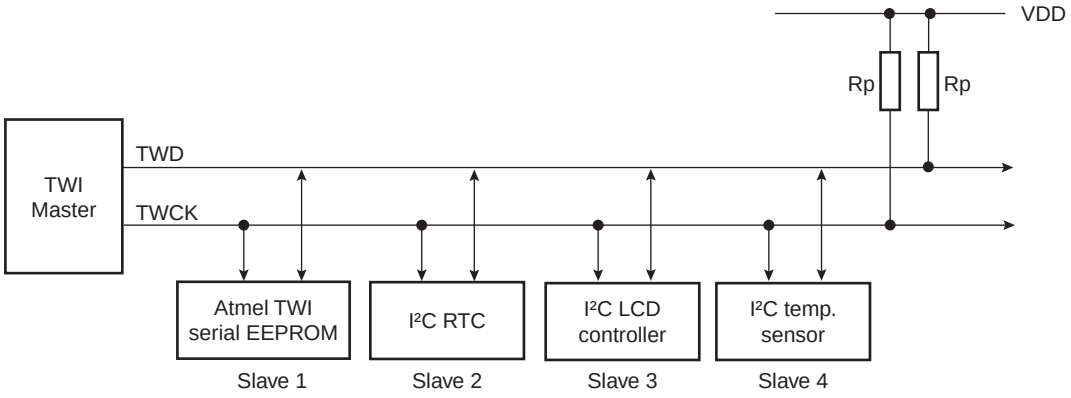
24.4 Block Diagram

Figure 24-1. Block Diagram



24.5 Application Block Diagram

Figure 24-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

24.6 I/O Lines Description

Table 24-4. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output
TWALM	SMBus SMBALERT	Input/Output

24.7 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

24.7.1 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 24-2 on page 459). 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.

TWALM is used to implement the optional SMBus SMBALERT signal.

TWD, TWCK and TWALM pins may be multiplexed with I/O Controller lines. To enable the TWIM, the programmer must perform the following steps:

- Program the I/O Controller to:
  - Dedicate TWD, TWCK and optionally TWALM as peripheral lines.
  - Define TWD, TWCK and optionally TWALM as open-drain.

24.7.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the TWIM, the TWIM will stop functioning and resume operation after the system wakes up from sleep mode.

24.7.3 Clocks

The clock for the TWIM bus interface (CLK_TWIM) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the TWIM before disabling the clock, to avoid freezing the TWIM in an undefined state.

24.7.4 Interrupts

The TWIM interrupt request lines are connected to the interrupt controller. Using the TWIM interrupts requires the interrupt controller to be programmed first.

24.7.5 Debug Operation

When an external debugger forces the CPU into debug mode, the TWIM continues normal operation. If the TWIM is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

24.8 Functional Description

24.8.1 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 24-4 on page 461).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 24-4 on page 461).

- 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 24-3. START and STOP Conditions

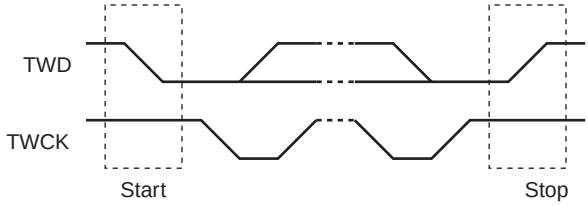
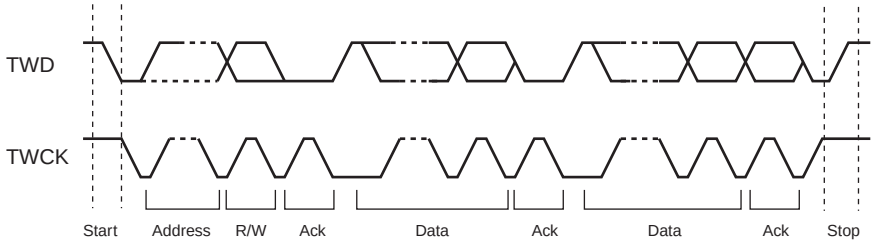


Figure 24-4. Transfer Format



24.8.2 Operation

The TWIM has two modes of operation:

- Master transmitter mode
- Master receiver mode

The master is the device which starts and stops a transfer and generates the TWCK clock. These modes are described in the following chapters.

24.8.2.1 Clock Generation

The Clock Waveform Generator Register (CWGR) is used to control the waveform of the TWCK clock. CWGR must be programmed so that the desired TWI bus timings are generated. CWGR describes bus timings as a function of cycles of a prescaled clock. The clock prescaling can be selected through the EXP field in CWGR.

$$f_{prescaled} = \frac{f_{clkpb}}{2^{(EXP+1))}}$$

CWGR has the following fields:

LOW: Prescaled clock cycles in clock low count. Used to time T_{LOW}. and T_{BUF}.

HIGH: Prescaled clock cycles in clock high count. Used to time T_{HIGH}.

STASTO: Prescaled clock cycles in clock high count. Used to time T_{HD_STA}, T_{SU_STA}, T_{SU_STO}.

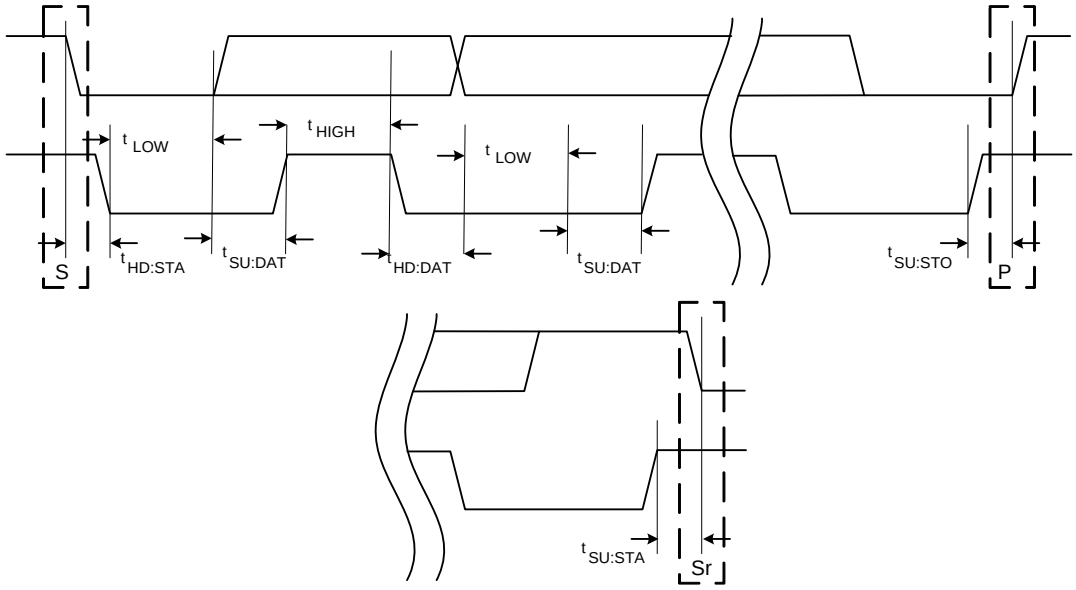
DATA: Prescaled clock cycles for data setup and hold count. Used to time T_{HD_DAT}, T_{SU_DAT}.

EXP: Specifies the clock prescaler setting.

Note that the total clock low time generated is the sum of T_{HD_DAT} + T_{SU_DAT} + T_{LOW}.

Any slave or other bus master taking part in the transfer may extend the TWCK low period at any time.

Figure 24-5. Bus Timing Diagram



24.8.2.2 Setting up and Performing a Transfer

Operation of TWIM is mainly controlled by the Control Register (CR) and the Command Register (CMDR). The following list presents the main steps in a typical communication:

1. Before any transfers can be performed, bus timings must be configured by programming the Clock Waveform Generator Register (CWGR). If operating in SMBus mode, the SMBus Timing Register (SMBTR) register must also be configured.
2. If a DMA controller is to be used for the transfers, it must be set up.
3. CMDR or NCMDR must be programmed with a value describing the transfer to be performed.

The interrupt system can be set up to give interrupt request on specific events or error conditions, for example when the transfer is complete or if arbitration is lost.

The controller will refuse to start a new transfer while ANAK, DNAK or ARBLST is set in the Status Register (SR). This is necessary to avoid a race when the software issues a continuation of the current transfer at the same time as one of these errors happen. Also, if ANAK or DNAK occur, a STOP condition is sent automatically. The programmer will have to restart the transmission by clearing the errors bit in SR after resolving the cause for the NACK.

After a data or address NACK from the slave, a STOP will be transmitted automatically. Note that the VALID bit in CMDR is NOT cleared in this case. If this transfer is to be discarded, the VALID bit can be cleared manually allowing any command in NCMDR to be copied into CMDR.

24.8.3 Master Transmitter Mode

A START condition is transmitted and master transmitter mode is initiated when the bus is free and CMDR has been written with START=1 and READ=0. START and SADR+W will then be transmitted. During the address acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to acknowledge the address. The master polls the data line during this clock pulse and sets the Address Not Acknowledged bit (ANAK**)** in the Status Register if no slave acknowledges the address.

After the address phase, the following is repeated:

while (NBYTES>0)

1. Wait until THR contains a valid data byte, stretching low period of TWCK. SR.TXRDY indicates the state of THR. Software or a DMA controller must write the data byte to THR.
2. Transmit this data byte
3. Decrement NBYTES
4. If (NBYTES==0) and STOP=1, transmit STOP condition

Programming CMDR with START=STOP=1 and NBYTES=0 will generate a transmission with no data bytes, ie START, SADR+W, STOP.

TWI transfers require the slave to acknowledge each received data 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 Data Acknowledge bit (DNACK) in the Status Register if the slave does not acknowledge the data byte. As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable Register (TWIM_IER).

TXRDY is used as Transmit Ready for the Peripheral DMA Controller transmit channel.

The end of a command is marked by setting the SR.CCOMP bit to one. See Figure 24-6 on page 464 and Figure 24-7 on page 464.

Figure 24-6. Master Write with One Data Byte

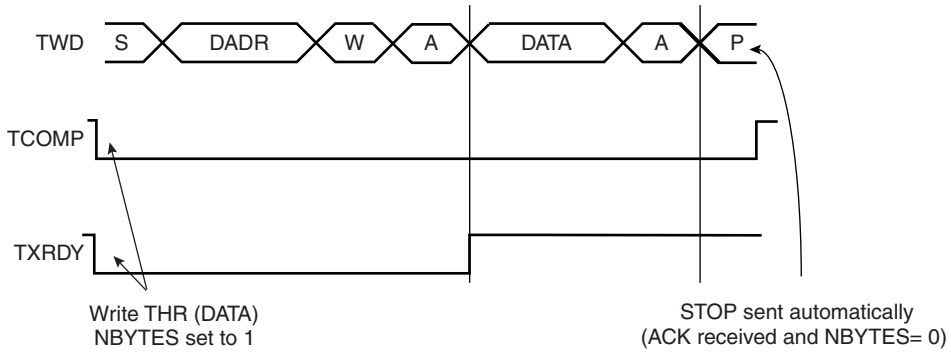
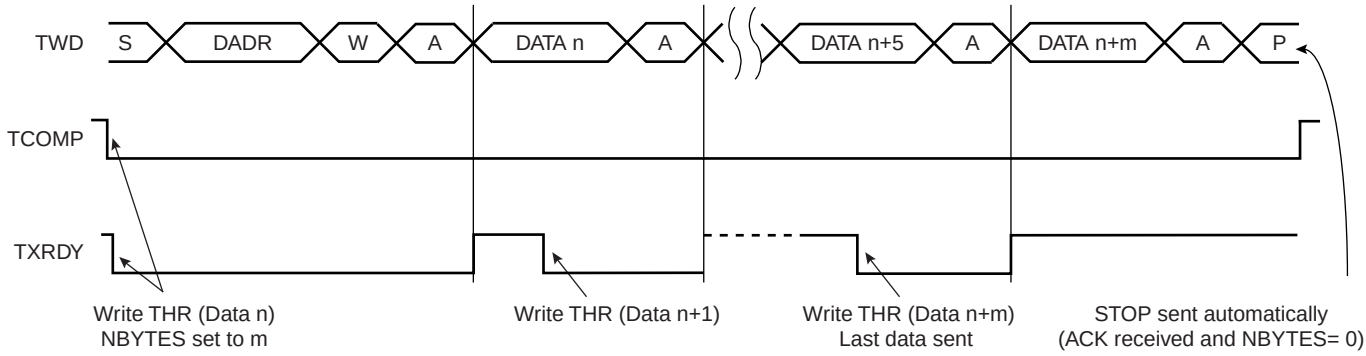


Figure 24-7. Master Write with Multiple Data Bytes



24.8.4 Master Receiver Mode

A START condition is transmitted and master receiver mode is initiated when the bus is free and CMDR has been written with START=1 and READ=1. START and SADR+R will then be transmitted. During the address acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to acknowledge the address. The master polls the data line during this clock pulse and sets the Address Not Acknowledged bit (ANAK) in the Status Register if no slave acknowledges the address.

After the address phase, the following is repeated:

while (NBYTES>0)

1. Wait until RHR is empty, stretching low period of TWCK. SR.RXRDY indicates the state of RHR. Software or a DMA controller must read any data byte present in RHR.
2. Release TWCK generating a clock that the slave uses to transmit a data byte.
3. Place the received data byte in RHR, set RXRDY.
4. If NBYTES=0, generate a NAK after the data byte, otherwise generate an ACK.
5. Decrement NBYTES
6. If (NBYTES==0) and STOP=1, transmit STOP condition.

Programming CMDR with START=STOP=1 and NBYTES=0 will generate a transmission with no data bytes, ie START, DADR+R, STOP

The TWI transfers require the master to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the slave releases the data line (HIGH), enabling the master to pull it down in order to generate the acknowledge. All data bytes except the last are

acknowledged by the master. Not acknowledging the last byte informs the slave that the transfer is finished.

RXRDY is used as Receive Ready for the Peripheral DMA Controller receive channel.

Figure 24-8. Master Read with One Data Byte

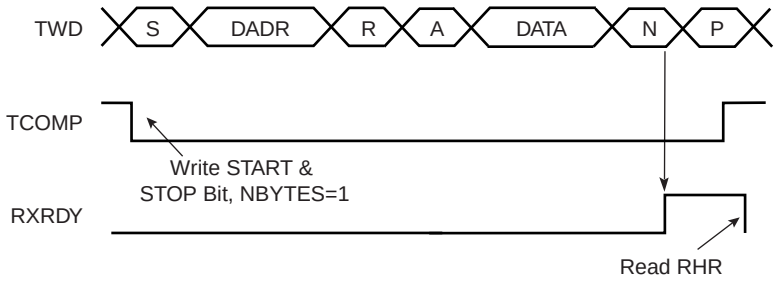
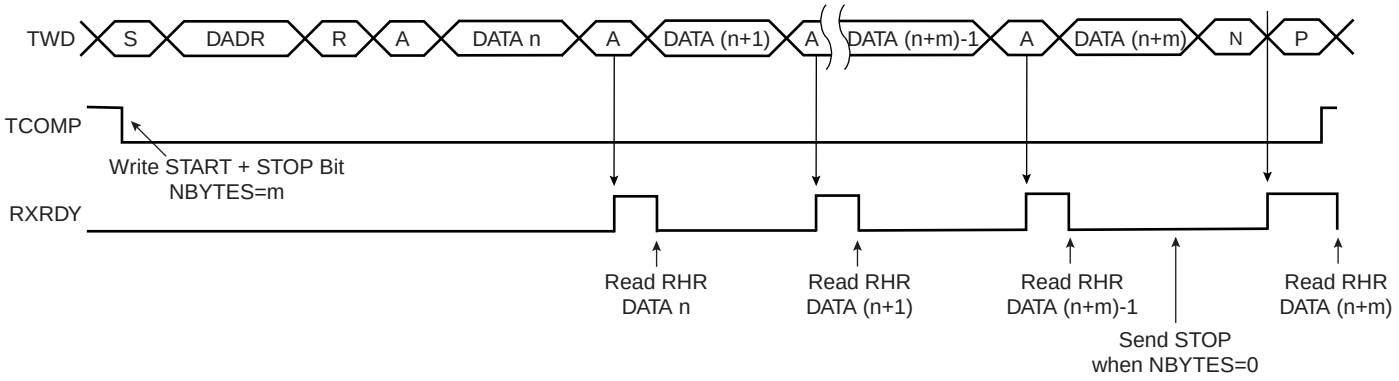


Figure 24-9. Master Read with Multiple Data Bytes



24.8.5 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The programmer can set up ring buffers for the DMA controller, containing data to transmit or free buffer space to place received data.

To assure correct behavior, respect the following programming sequences:

24.8.5.1 Data Transmit with the Peripheral DMA Controller

1. Initialize the transmit Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIM (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller TXTEN bit.
4. Wait for the Peripheral DMA Controller end TX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller TXDIS bit.

24.8.5.2 *Data Receive with the Peripheral DMA Controller*

1. Initialize the receive Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIM (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller RXTEN bit.
4. Wait for the Peripheral DMA Controller end RX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller RXDIS bit.

24.8.6 Multi-master Mode

More than one master may access 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. The SR.ARBLST flag will be set. When the STOP is detected, the master who lost arbitration may reinitiate the data transfer.

Arbitration is illustrated in Figure 24-11 on page 467.

If the user starts a transfer and if the bus is busy, TWIM automatically waits for a STOP condition on the bus before initiating the transfer (see Figure 24-10 on page 466).

Note: The state of the bus (busy or free) is not indicated in the user interface.

Figure 24-10. Programmer Sends Data While the Bus is Busy

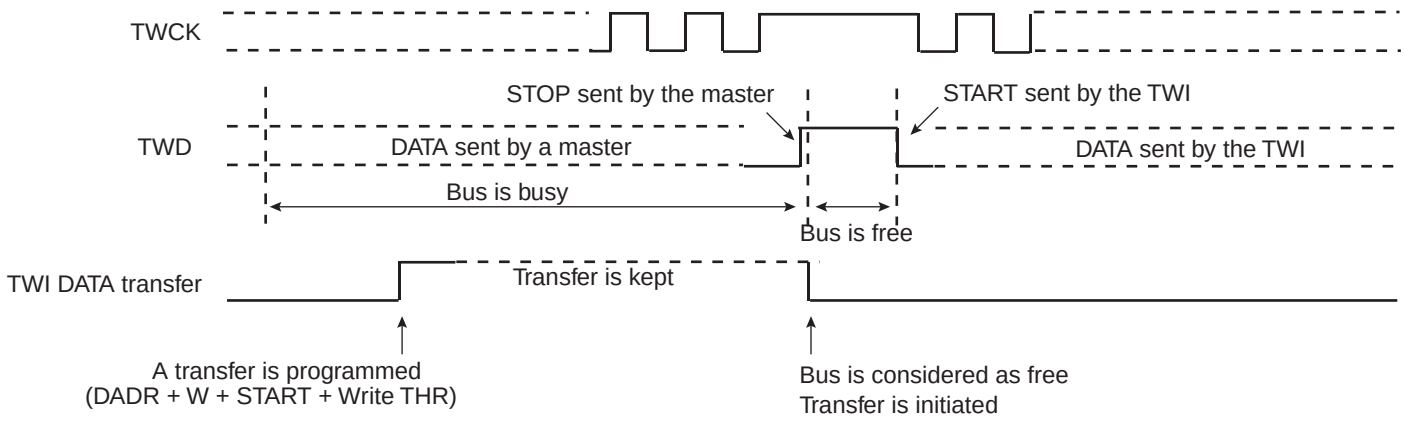
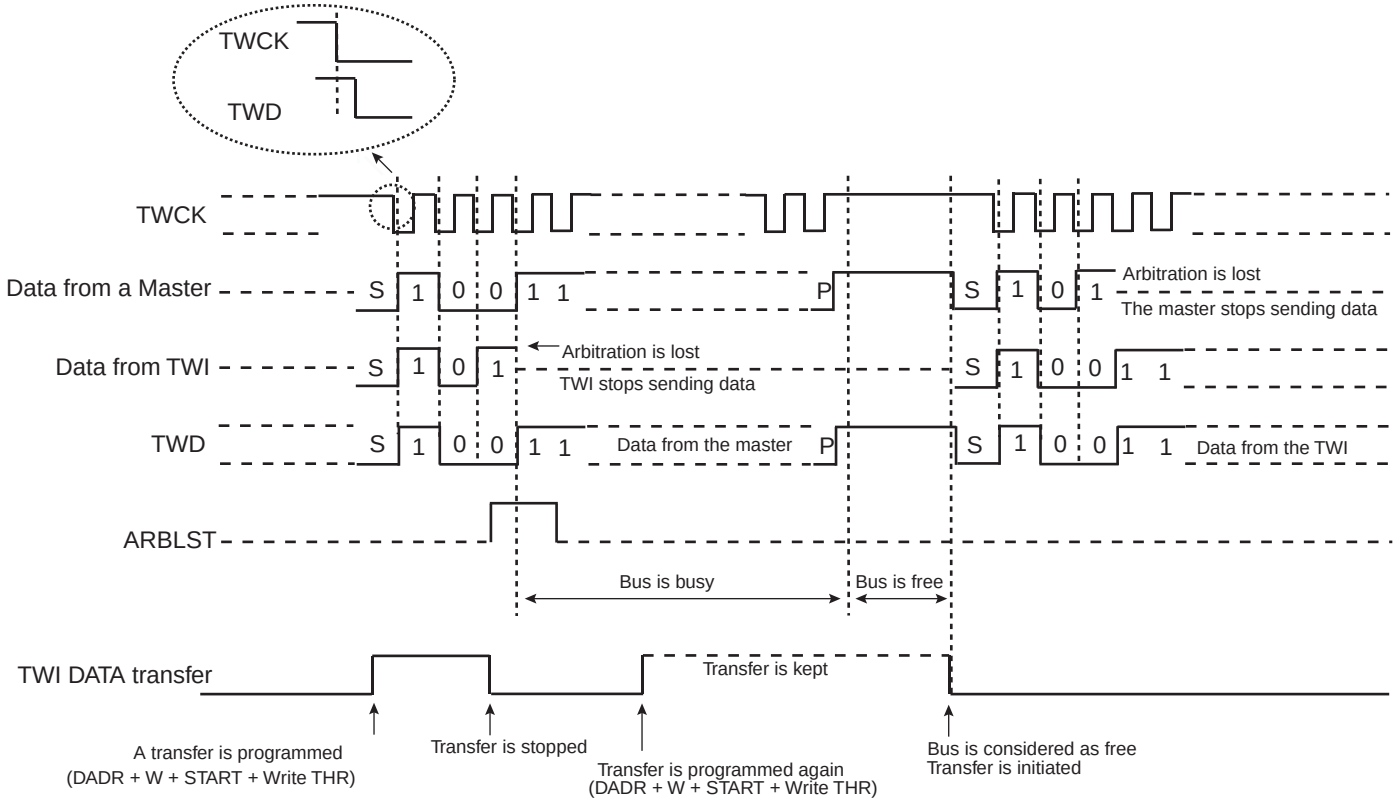


Figure 24-11. Arbitration Cases



24.8.7 Combined Transfers

CMDR and NCMDR may be used to generate longer sequences of connected transfers, since generation of START and/or STOP conditions is programmable on a per-command basis.

Programming NCMDR with START=1 when the previous transfer was programmed with STOP=0 will cause a REPEATED START on the bus. The ability to generate such connected transfers allows arbitrary transfer lengths, since it is legal to program CMDR with both START=0 and STOP=0. If this is done in master receiver mode, the CMDR.ACKLAST bit must also be controlled.

As for single data transfers, the TXRDY and RXRDY bits in the Status Register indicates when data to transmit can be written to the THR, or when received data can be read from RHR. Transfer of data to THR and from RHR can also be done automatically by DMA, see “Using the Peripheral DMA Controller” on page 465

24.8.7.1 Write Followed by Write

Consider the following transfer:

START, DADR+W, DATA+A, DATA+A, REPSTART, DADR+W, DATA+A, DATA+A, STOP.

To generate this transfer:

1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=0.
3. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
4. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.

5. Wait until SR.TXRDY==1, then write third data byte to transfer to THR.
6. Wait until SR.TXRDY==1, then write fourth data byte to transfer to THR.

24.8.7.2 *Read Followed by Read*

Consider the following transfer:

START, DADR+R, DATA+A, DATA+NA, REPSTART, DADR+R, DATA+A, DATA+NA, STOP.

To generate this transfer:

1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=1.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=1.
3. Wait until SR.RXRDY==1, then read first data byte received from RHR.
4. Wait until SR.RXRDY==1, then read second data byte received from RHR.
5. Wait until SR.RXRDY==1, then read third data byte received from RHR.
6. Wait until SR.RXRDY==1, then read fourth data byte received from RHR.

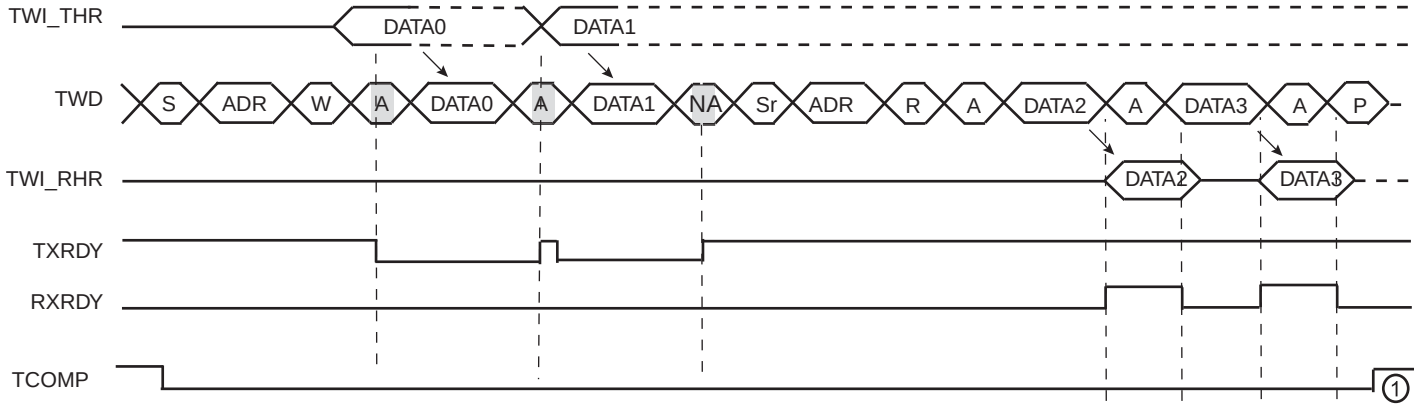
If combining several transfers, without any STOP or REPEATED START between them, remember to set the ACKLAST bit in CMDR to keep from ending each of the partial transfers with a NACK.

24.8.7.3 *Write Followed by Read*

Consider the following transfer:

START, DADR+W, DATA+A, DATA+A, REPSTART, DADR+R, DATA+A, DATA+NA, STOP.

Figure 24-12. Combining a Write and Read Transfer



To generate this transfer:

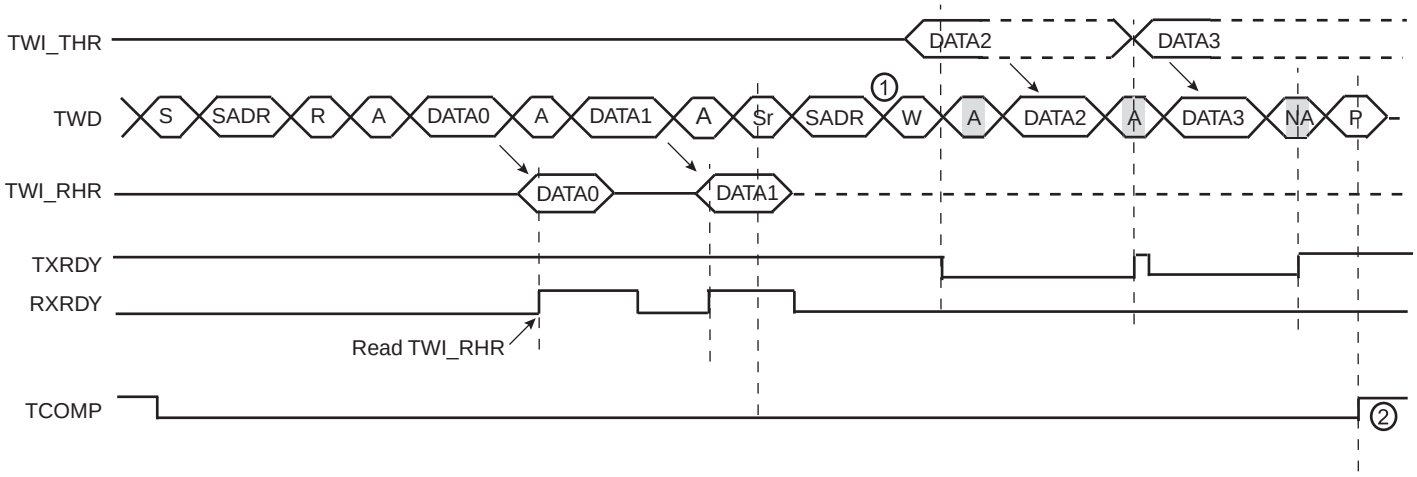
1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=1.
3. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
4. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.
5. Wait until SR.RXRDY==1, then read first data byte received from RHR.
6. Wait until SR.RXRDY==1, then read second data byte received from RHR.

24.8.7.4 *Read Followed by Write*

Consider the following transfer:

START, DADR+R, DATA+A, DATA+NA, REPSTART, DADR+W, DATA+A, DATA+A, STOP.

Figure 24-13. Combining a Read and Write Transfer



To generate this transfer:

1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=1.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=0.
3. Wait until SR.RXRDY==1, then read first data byte received from RHR.
4. Wait until SR.RXRDY==1, then read second data byte received from RHR.
5. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
6. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.

24.8.8 Ten Bit Addressing

Setting CMDR.TENBIT enables 10-bit addressing in hardware. Performing transfers with 10-bit addressing is similar to transfers with 7-bit addresses, except that bits 10:7 of CMDR.ADR must be set appropriately.

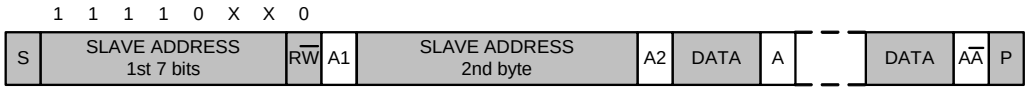
In Figure 24-14 on page 469 and Figure 24-15 on page 470, the grey boxes represent signals driven by the master, the white boxes are driven by the slave.

24.8.8.1 Master Transmitter

To perform a master transmitter transfer,

1. Program CMDR with TENBIT=1, REPSAME=0, READ=0, START=1, STOP=1 and the desired address and NBYTES value.

Figure 24-14. A Write Transfer with 10-bit Addressing



24.8.8.2 Master Receiver

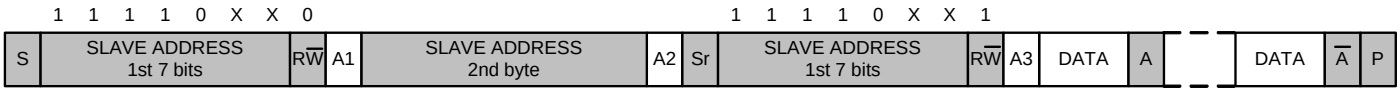
When using master receiver mode with 10-bit addressing, CMDR.REPSAME must also be controlled. CMDR.REPSAME must be written to one when the address phase of the transfer should

consist of only 1 address byte (the 11110xx byte) and not 2 address bytes. The I²C standard specifies that such addressing is required when addressing a slave for reads using 10-bit addressing.

To perform a master receiver transfer,

1. Program CMDR with TENBIT=1, REPSAME=0, READ=0, START=1, STOP=0, NBYTES=0 and the desired address.
2. Program NCMDR with TENBIT=1, REPSAME=1, READ=1, START=1, STOP=1 and the desired address and NBYTES value.

Figure 24-15. A Read Transfer with 10-bit Addressing



24.8.9 SMBus Mode

SMBus mode is enabled and disabled by the SMEN and SMDIS bits in CR. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A dedicated bus line SMBALERT, allows a slave to get a master's attention.
- A set of addresses have been reserved for protocol handling, such as Alert Response Address (ARA) and Host Header (HH) Address.

24.8.9.1 Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing CMDR.PECEN to one enables automatic PEC handling in the current transfer. Transfers with and without PEC can freely be intermixed in the same system, since some slaves may not support PEC. The PEC LFSR is always updated on every bit transmitted or received, so that PEC handling on combined transfers will be correct.

In master transmitter mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave will compare it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave will return an ACK to the master. If the PEC values differ, data was corrupted, and the slave will return a NACK value. The DNAK bit in SR reflects the state of the last received ACK/NACK value. Some slaves may not be able to check the received PEC in time to return a NACK if an error occurred. In this case, the slave should always return an ACK after the PEC byte, and some other mechanism must be implemented to verify that the transmission was received correctly.

In master receiver mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master will compare it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the PECERR bit in SR is set. In master receiver mode, the PEC byte is always followed by a NACK transmitted by the master, since it is the last byte in the transfer.

The PEC byte is automatically inserted in a master transmitter transmission if PEC is enabled when NBYTES reaches zero. The PEC byte is identified in a master receiver transmission if PEC is enabled when NBYTES reaches zero. NBYTES must therefore be set to the total number of data bytes in the transmission, including the PEC byte.

In combined transfers, the PECEN bit should only be set in the last of the combined transfers. Consider the following transfer:

S, ADR+W, COMMAND_BYTE, ACK, SR, ADR+R, DATA_BYTE, ACK, PEC_BYTE, NACK, P

This transfer is generated by writing two commands to the command registers. The first command is a write with NBYTES=1 and PECEN=0, and the second is a read with NBYTES=2 and PECEN=1.

Writing a one to the STOP bit in CR will place a STOP condition on the bus after the current byte. No PEC byte will be sent in this case.

24.8.9.2 Timeouts

The TLOWS and TLOWM fields in SMBTR configure the SMBus timeout values. If a timeout occurs, the master will transmit a STOP condition and leave the bus. The SR.TOUT bit is also set.

24.8.9.3 SMBus ALERT Signal

A slave can get the master's attention by pulling the TWALM line low. SR.SMBAL will then be set. This can be set up to trigger an interrupt, and software can then take the appropriate action, as defined in the SMBus standard.

24.8.10 Identifying Bus Events

This chapter lists the different bus events, and how these affects bits in the TWIM registers. This is intended to help writing drivers for the TWIM.

Table 24-5. Bus Events

Event	Effect
Master transmitter has sent a data byte	SR.THR is cleared.
Master receiver has received a data byte	SR.RHR is set.
Start+Sadr sent, no ack received from slave	SR.ANAK is set. CMDR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Data byte sent to slave, no ack received from slave	SR.DNAK is set. CMDR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Arbitration lost	SR.ARBLST is set. CMDR.CCOMP not set. CMDR.VALID remains set. TWCK and TWD immediately released to a pulled-up state.
SMBus Alert received	SR.SMBAL is set.

Table 24-5. Bus Events

Event	Effect
SMBus timeout received	SR.SMBTOUT is set. CMDR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Master transmitter receives SMBus PEC Error	SR.DNAK is set. CMDR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Master receiver discovers SMBus PEC Error	SR.PECERR is set. CMDR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
CR.STOP is written by user	SR.STOP is set. CMDR.CCOMP set. CMDR.VALID remains set. STOP transmitted on bus after current byte transfer has finished.

24.9 User Interface

Table 24-6. TWIM Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control	CR	Write-only	0x00000000
0x04	Clock Waveform Generator	CWGR	Read/Write	0x00000000
0x08	SMBus Timing	SMBTR	Read/Write	0x00000000
0x0C	Command	CMDR	Read/Write	0x00000000
0x10	Next Command	NCMDR	Read/Write	0x00000000
0x14	Receive Holding	RHR	Read-only	0x00000000
0x18	Transmit Holding	THR	Write-only	0x00000000
0x1C	Status	SR	Read-only	0x00000002
0x20	Interrupt Enable Register	IER	Write-only	0x00000000
0x24	Interrupt Disable Register	IDR	Write-only	0x00000000
0x28	Interrupt Mask Register	IMR	Read-only	0x00000000
0x2C	Status Clear Register	SCR	Write-only	0x00000000
0x30	Parameter Register	PR	Read-only	(1)
0x34	Version Register	VR	Read-only	(1)

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

24.9.1 Control Register (CR)

Name: CR

Access Type: Write-only

Offset: 0x00

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
-	-	-	-	-	-	-	STOP

7	6	5	4	3	2	1	0
SWRST	-	SMDIS	SMEN	-	-	MDIS	MEN

- **STOP: Stop the current transfer**
  Writing a one to this bit terminates the current transfer, sending a STOP condition after the shifter has become idle. If there are additional pending transfers, they will have to be explicitly restarted by software after the STOP condition has been successfully sent.
  Writing a zero to this bit has no effect.
- **SWRST: Software Reset**
  Writing a one to this bit resets the TWIM. All transfers are halted immediately, possibly violating the bus semantics.
  Writing a zero to this bit has no effect.
- **SMDIS: SMBus Disable**
  Writing a one to this bit disables SMBus mode.
  Writing a zero to this bit has no effect.
- **SMEN: SMBus Enable**
  Writing a one to this bit enables SMBus mode.
  Writing a zero to this bit has no effect.
- **MDIS: Master Disable**
  Writing a one to this bit disables the master interface.
  Writing a zero to this bit has no effect.
- **MEN: Master enable**
  Writing a one to this bit enables the master interface.
  Writing a zero to this bit has no effect.

24.9.2 Clock Waveform Generator Register (CWGR)

Name: CWGR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	EXP			DATA			

23	22	21	20	19	18	17	16
STASTO							

15	14	13	12	11	10	9	8
HIGH							

7	6	5	4	3	2	1	0
LOW							

- **EXP: Clock Prescaler**
  Used to specify how to prescale the TWCK clock. Counters are prescaled according to the following formula

$$f_{prescaled} = \frac{f_{clkpb}}{2^{(EXP+1)}}$$

- **DATA: Data Setup and Hold Cycles**
  Clock cycles for data setup and hold count. Prescaled by CWGR.EXP. Used to time T_{HD_DAT}, T_{SU_DAT}.
- **STASTO: START and STOP Cycles**
  Clock cycles in clock high count. Prescaled by CWGR.EXP. Used to time T_{HD_STA}, T_{SU_STA}, T_{SU_STO}
- **HIGH: Clock High Cycles**
  Clock cycles in clock high count. Prescaled by CWGR.EXP. Used to time T_{HIGH}.
- **LOW: Clock Low Cycles**
  Clock cycles in clock low count. Prescaled by CWGR.EXP. Used to time T_{LOW}, T_{BUF}.

24.9.3 SMBus Timing Register (SMBTR)

Name: SMBTR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
EXP				-	-	-	-

23	22	21	20	19	18	17	16
THMAX							

15	14	13	12	11	10	9	8
TLOWM							

7	6	5	4	3	2	1	0
TLOWS							

- **EXP: SMBus Timeout Clock prescaler**
  Used to specify how to prescale the TIM and TLOWM counters in SMBTR. Counters are prescaled according to the following formula

$$f_{prescaled,\,SMBus} = \frac{f_{clkpb}}{2^{(EXP+1)}}$$

- **THMAX: Clock High maximum cycles**
  Clock cycles in clock high maximum count. Prescaled by SMBTR.EXP. Used for bus free detection. Used to time $T_{HIGH:MAX}$.
  NOTE: Uses the prescaler specified by CWGR, NOT the prescaler specified by SMBTR.
- **TLOWM: Master Clock stretch maximum cycles**
  Clock cycles in master maximum clock stretch count. Prescaled by SMBTR.EXP. Used to time $T_{LOW:MEXT}$
- **TLOWS: Slave Clock stretch maximum cycles**
  Clock cycles in slave maximum clock stretch count. Prescaled by SMBTR.EXP. Used to time $T_{LOW:SEXT}$.

24.9.4 Command Register (CMDR)

Name: CMDR

Access Type: Read/Write

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	ACKLAST	PECEN

23	22	21	20	19	18	17	16
NBYTES							

15	14	13	12	11	10	9	8
VALID	STOP	START	REPSAME	TENBIT	SADR[9:7]		

7	6	5	4	3	2	1	0
SADR[6:0]							READ

- **ACKLAST: ACK Last Master RX Byte**
  Writing this bit to zero causes the last byte in master receive mode (when NBYTES has reached 0) to be NACKed. This is the standard way of ending a master receiver transfer.
  Writing this bit to one causes the last byte in master receive mode (when NBYTES has reached 0) to be ACKed. Used for performing linked transfers in master receiver mode with no STOP or REPEATED START between the subtransfers. This is needed when more than 255 bytes are to be received in one single transmission.
- **PECEN: Packet Error Checking Enable**
  Writing this bit to zero causes the transfer not to use PEC byte verification. The PEC LFSR is still updated for every bit transmitted or received. Must be used if SMBus mode is disabled.
  Writing this bit to one causes the transfer to use PEC. PEC byte generation (if master transmitter) or PEC byte verification (if master receiver) will be performed.
- **NBYTES: Number of data bytes in transfer**
  The number of data bytes in the transfer. After the specified number of bytes have been transferred, a STOP condition is transmitted if CMDR.STOP is set. In SMBus mode, if PEC is used, NBYTES includes the PEC byte, ie there are NBYTES-1 data bytes and a PEC byte.
- **VALID: CMDR Valid**
  Writing this to zero indicates that CMDR does not contain a valid command.
  Writing this to one indicates that CMDR contains a valid command. This bit is cleared when the command is finished.
- **STOP: Send STOP condition**
  Write this bit to zero to not transmit a STOP condition after the data bytes have been transmitted.
  Write this bit to one to transmit a STOP condition after the data bytes have been transmitted.
- **START: Send START condition**
  Write this bit to zero if the transfer in CMDR should not commence with a START or REPEATED START condition.
  Write this bit to one if the transfer in CMDR should commence with a START or REPEATED START condition. If the bus is free when the command is executed, a START condition is used, if the bus is busy, a REPEATED START is used.
- **REPSAME: Transfer is to same address as previous address**
  Only used in 10-bit addressing mode, always write to 0 in 7-bit addressing mode.

Write this bit to one if the command in CMDR performs a repeated start to the same slave address as addressed in the previous transfer in order to enter master receiver mode.

Write this bit to zero otherwise.

- **TENBIT: Ten Bit Addressing Mode**

  Write this bit to zero to use 7-bit addressing mode.

  Write this bit to one to use 10-bit addressing mode. Must not be used when TWIM is in SMBus mode.

- **SADR: Slave Address**

  Address of the slave involved in the transfer. Bits 9-7 are don't care if 7-bit addressing is used.

- **READ: Transfer Direction**

  Write this bit to zero to let the master transmit data.

  Write this bit to one to let the master receive data.

24.9.5 Next Command Register (NCMDR)

Name: NCMDR

Access Type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	ACKLAST	PECEN

23	22	21	20	19	18	17	16
NBYTES							

15	14	13	12	11	10	9	8
VALID	STOP	START	REPSAME	TENBIT	SADR[9:7]		

7	6	5	4	3	2	1	0
SADR[6:0]							READ

This register is identical to CMDR. When the VALID bit in CMDR becomes 0, the contents of NCMDR is copied into CMDR, clearing the VALID bit in NCMDR. If the VALID bit in CMDR is cleared when NCMDR is written, the contents are copied immediately.

24.9.6 Receive Holding Register (RHR)

Name: RHR

Access Type: Read-only

Offset: 0x14

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: Received Data**
  When the RXRDY bit in the Status Register (SR) is set, this field contains a byte received from the TWI bus.

24.9.7 Transmit Holding Register (THR)

Name: THR

Access Type: Write-only

Offset: 0x18

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: Data to Transmit**
  Write data to be transferred on the TWI bus here.

24.9.8 Status Register (SR)

Name: SR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000002

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	MENB

15	14	13	12	11	10	9	8
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	-	IDLE	CCOMP	CRDY	TXRDY	RXRDY

- **MENB: Master Interface Enable**
  0: Master interface is disabled.
  1: Master interface is enabled.
- **STOP: Stop Request Accepted**
  This bit is set when STOP request caused by setting CR STOP has been accepted, and transfer has stopped.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **PECERR: PEC Error**
  This bit is set when a SMBus PEC error occurred.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **TOUT: Timeout**
  This bit is set when a SMBus timeout occurred.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **SMBALERT: SMBus Alert**
  This bit is set when a SMBus Alert was received.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **ARBLST: Arbitration Lost**
  This bit is set when the actual state of the SDA line did not correspond to the data driven onto it, indicating a higher-priority transmission in progress by a different master.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **DNAK: NAK in Data Phase Received**
  This bit is set when no ACK was received form slave during data transmission.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **ANAK: NAK in Address Phase Received**
  This bit is set when no ACK was received from slave during address phase
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **IDLE: Master Interface is Idle**
  This bit is set when no command is in progress, and no command waiting to be issued.
  Otherwise, this bit is cleared.

- **CCOMP: Command Complete**
  This bit is set when the current command has completed successfully and STOP has been transmitted.
  Not set if the command failed due to conditions such as a NAK receved from slave.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **CRDY: Ready for More Commands**
  This bit is set when CMDR and/or NCMDR is ready to receive one or more commands.
  This bit is cleared when this is no longer true.
- **TXRDY: THR Data Ready**
  This bit is set when THR is ready for one or more data bytes.
  This bit is cleared when this is no longer true (i.e. THR is full or transmission has stopped).
- **RXRDY: RHR Data Ready**
  This bit is set when RX data are ready to be read from RHR.
  This bit is cleared when this is no longer true.

24.9.9 Interrupt Enable Register (IER)

Name: IER

Access Type: Write-only

Offset: 0x20

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
-	-	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	-	IDLE	CCOMP	CRDY	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR

24.9.10 Interrupt Disable Register (IDR)

Name: IDR

Access Type: Write-only

Offset: 0x24

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
-	-	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	-	IDLE	CCOMP	CRDY	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR

24.9.11 Interrupt Mask Register (IMR)

Name: IMR

Access Type: Read-only

Offset: 0x28

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
-	-	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	-	IDLE	CCOMP	CRDY	TXRDY	RXRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
This bit is cleared when the corresponding bit in IDR is written to one.
This bit is set when the corresponding bit in IER is written to one.

24.9.12 Status Clear Register (SCR)

Name: SCR

Access Type : Write-only

Offset: 0x2C

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
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	-	-	CCOMP	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

24.9.13 Parameter Register (PR)

Name: PR

Access Type: Read-only

Offset: 0x30

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
-	-	-	-	-	-	-	-

24.9.14 Version Register (VR)

Name: VR

Access Type: Read-only

Offset: 0x34

Reset Value: Device-specific

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION [11:8]			

7	6	5	4	3	2	1	0
VERSION [7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

24.10 Module Configuration

The specific configuration for each TWIM instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 24-7. Module Clock Name

Module name	Clock name
TWIM0	**CLK_TWIM0**
TWIM1	**CLK_TWIM1**

Table 24-8. Register Reset Values

Register	Reset Value
VR	0x00000100
PR	0x00000000

25. Synchronous Serial Controller (SSC)

Rev: 3.2.0.2

25.1 Features

- **Provides serial synchronous communication links used in audio and telecom applications**
- **Independent receiver and transmitter, common clock divider**
- **Interfaced with two Peripheral DMA Controller channels to reduce processor overhead**
- **Configurable frame sync and data length**
- **Receiver and transmitter can be configured to start automatically or on detection of different events on the frame sync signal**
- **Receiver and transmitter include a data signal, a clock signa,l and a frame synchronization signal**

25.2 Overview

The Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC consists of a receiver, a transmitter, and a common clock divider. Both the receiver and the transmitter interface with three signals:

- the TX_DATA/RX_DATA signal for data
- the TX_CLOCK/RX_CLOCK signal for the clock
- the TX_FRAME_SYNC/RX_FRAME_SYNC signal for the frame synchronization

The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

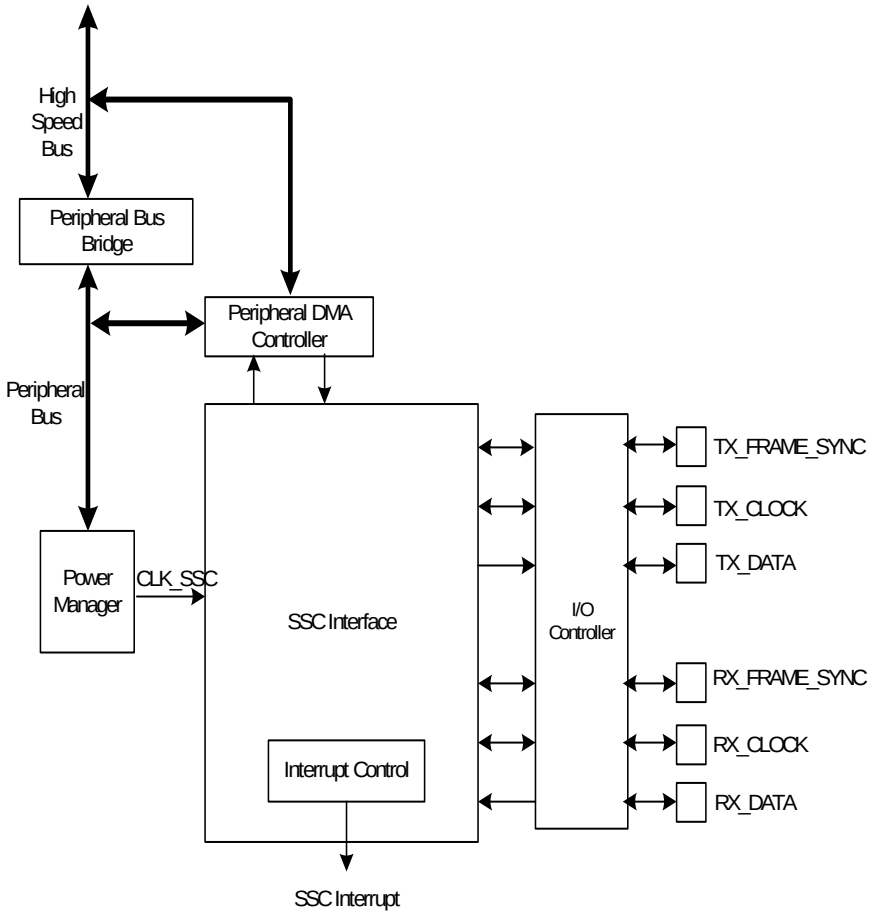
The SSC's high-level of programmability and its two dedicated Peripheral DMA Controller channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two Peripheral DMA Controller 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

25.3 Block Diagram

Figure 25-1. SSC Block Diagram



25.4 Application Block Diagram

Figure 25-2. SSC Application Block Diagram

OS or RTOS Driver	Power Management	Interrupt Management	Test Management
SSC			
Serial AUDIO	Codec	Time Slot Management	Frame Management	Line Interface

25.5 I/O Lines Description

Table 25-1. I/O Lines Description

Pin Name	Pin Description	Type
RX_FRAME_SYNC	Receiver Frame Synchro	Input/Output
RX_CLOCK	Receiver Clock	Input/Output
RX_DATA	Receiver Data	Input
TX_FRAME_SYNC	Transmitter Frame Synchro	Input/Output
TX_CLOCK	Transmitter Clock	Input/Output
TX_DATA	Transmitter Data	Output

25.6 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

25.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O lines.

Before using the SSC receiver, the I/O Controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the I/O Controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

25.6.2 Clocks

The clock for the SSC bus interface (CLK_SSC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SSC before disabling the clock, to avoid freezing the SSC in an undefined state.

25.6.3 Interrupts

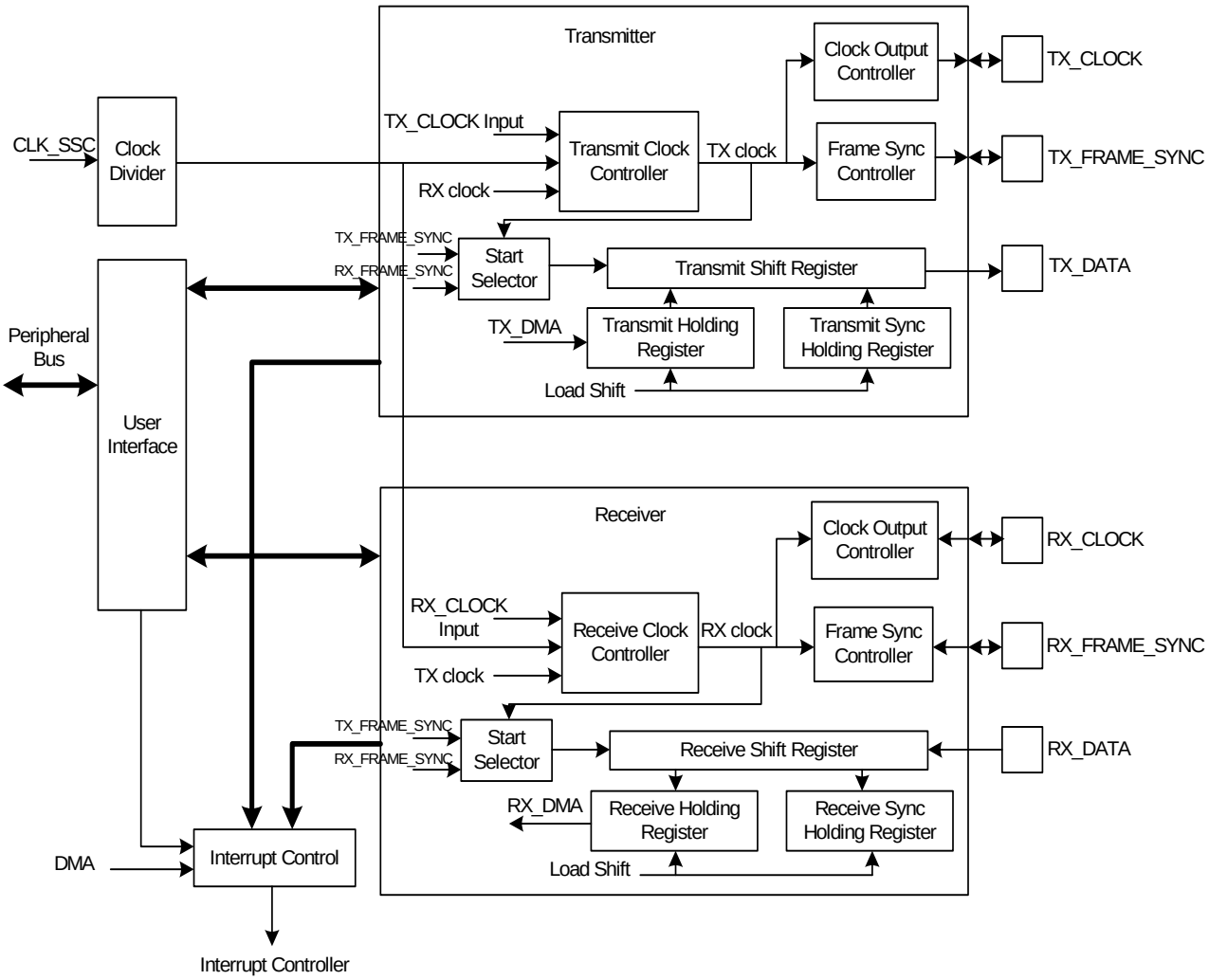
The SSC interrupt request line is connected to the interrupt controller. Using the SSC interrupt requires the interrupt controller to be programmed first.

25.7 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data Framing Format, Start, Transmitter, Receiver, and Frame Sync.

The receiver and the 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 TX_CLOCK or RX_CLOCK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TX_CLOCK and RX_CLOCK pins is CLK_SSC divided by two.

Figure 25-3. SSC Functional Block Diagram



25.7.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TX_CLOCK pin
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

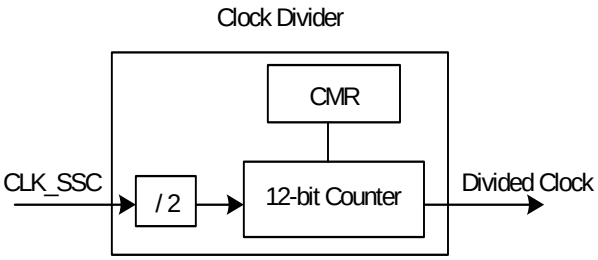
- an external clock received on the RX_CLOCK pin
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TX_CLOCK pin, and the receiver block can generate an external clock on the RX_CLOCK pin.

This allows the SSC to support many Master and Slave Mode data transfers.

25.7.1.1 Clock divider

Figure 25-4. Divided Clock Block Diagram



The peripheral clock divider is determined by the 12-bit Clock Divider field (its maximal value is 4095) in the Clock Mode Register (CMR.DIV), allowing a peripheral clock division by up to 8190. The divided clock is provided to both the receiver and transmitter. When this field is written to zero, the clock divider is not used and remains inactive.

When CMR.DIV is written to a value equal to or greater than one, the divided clock has a frequency of CLK_SSC divided by two times CMR.DIV. Each level of the divided clock has a duration of the peripheral clock multiplied by CMR.DIV. This ensures a 50% duty cycle for the divided clock regardless of whether the CMR.DIV value is even or odd.

Figure 25-5. Divided Clock Generation

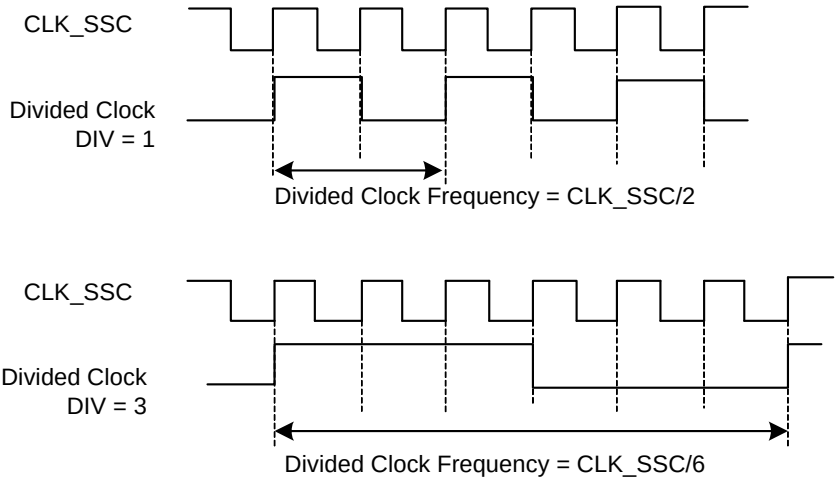


Table 25-2. Range of Clock Divider

Maximum	Minimum
CLK_SSC / 2	CLK_SSC / 8190

25.7.1.2 Transmitter clock management

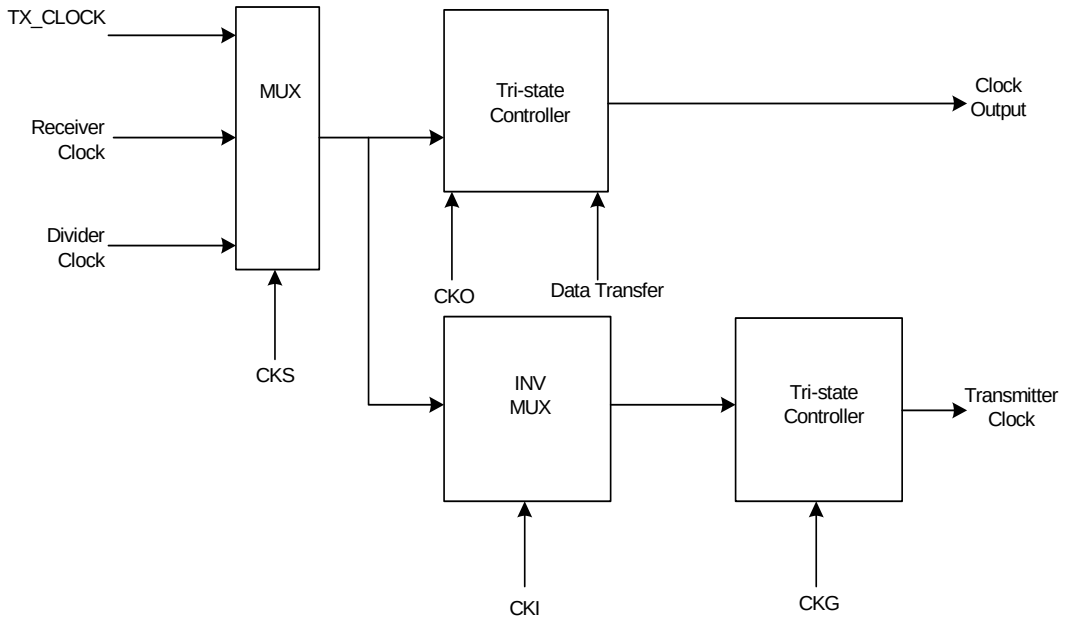
The transmitter clock is generated from the receiver clock, the divider clock, or an external clock scanned on the TX_CLOCK pin. The transmitter clock is selected by writing to the Transmit Clock Selection field in the Transmit Clock Mode Register (TCMR.CKS). The Transmit clock can

be inverted independently by writing a one to the Transmit Clock Inversion bit in TCMR (TCMR.CKI).

The transmitter can also drive the TX_CLOCK pin continuously or be limited to the actual data transfer, depending on the Transmit Clock Output Mode Selection field in the TCMR register (TCMR.CKO). The TCMR.CKI bit has no effect on the clock outputs.

Writing 0b10 to the TCMR.CKS field to select TX_CLOCK pin and 0b001 to the TCMR.CKO field to select Continuous Transmit Clock can lead to unpredictable results.

Figure 25-6. Transmitter Clock Management



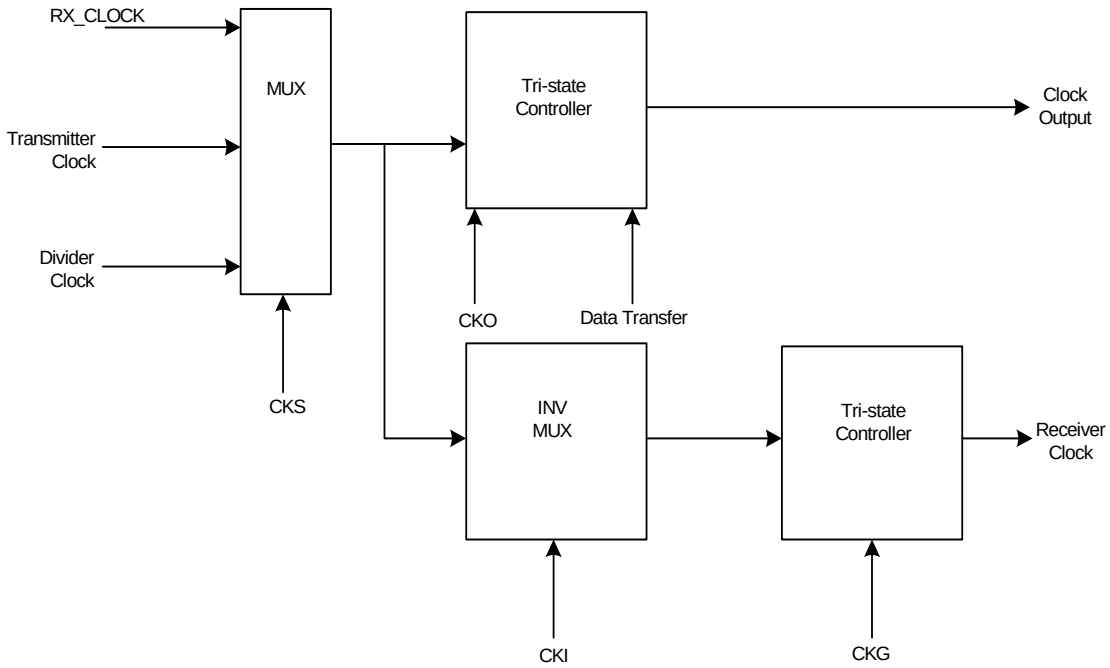
25.7.1.3 *Receiver clock management*

The receiver clock is generated from the transmitter clock, the divider clock, or an external clock scanned on the RX_CLOCK pin. The receive clock is selected by writing to the Receive Clock Selection field in the Receive Clock Mode Register (RCMR.CKS). The receive clock can be inverted independently by writing a one to the Receive Clock Inversion bit in RCMR (RCMR.CKI).

The receiver can also drive the RX_CLOCK pin continuously or be limited to the actual data transfer, depending on the Receive Clock Output Mode Selection field in the RCMR register (RCMR.CKO). The RCMR.CKI bit has no effect on the clock outputs.

Writing 0b10 to the RCMR.CKS field to select RX_CLOCK pin and 0b001 to the RCMR.CKO field to select Continuous Receive Clock can lead to unpredictable results.

Figure 25-7. Receiver Clock Management



25.7.1.4 Serial clock ratio considerations

The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TX_CLOCK or RX_CLOCK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RX_CLOCK pin is:

– CLK_SSC divided by two if RX_FRAME_SYNC is input.

– CLK_SSC divided by three if RX_FRAME_SYNC is output.

In addition, the maximum clock speed allowed on the TX_CLOCK pin is:

– CLK_SSC divided by six if TX_FRAME_SYNC is input.

– CLK_SSC divided by two if TX_FRAME_SYNC is output.

25.7.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

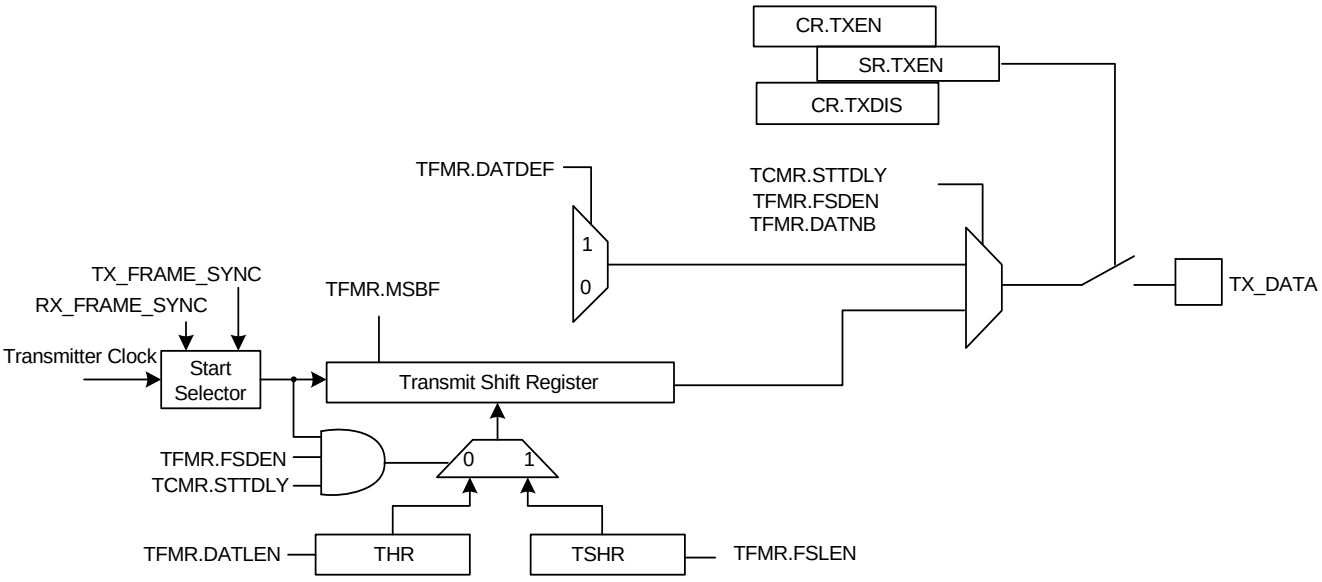
The start event is configured by writing to the TCMR register. See Section 25.7.4.

The frame synchronization is configured by writing to the Transmit Frame Mode Register (TFMR). See Section 25.7.5.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the TCMR register. Data is written by the user to the Transmit Holding Register (THR) then transferred to the shift register according to the data format selected.

When both the THR and the transmit shift registers are empty, the Transmit Empty bit is set in the Status Register (SR.TXEMPTY). When the THR register is transferred in the transmit shift register, the Transmit Ready bit is set in the SR register (SR.TXREADY) and additional data can be loaded in the THR register.

Figure 25-8. Transmitter Block Diagram



25.7.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

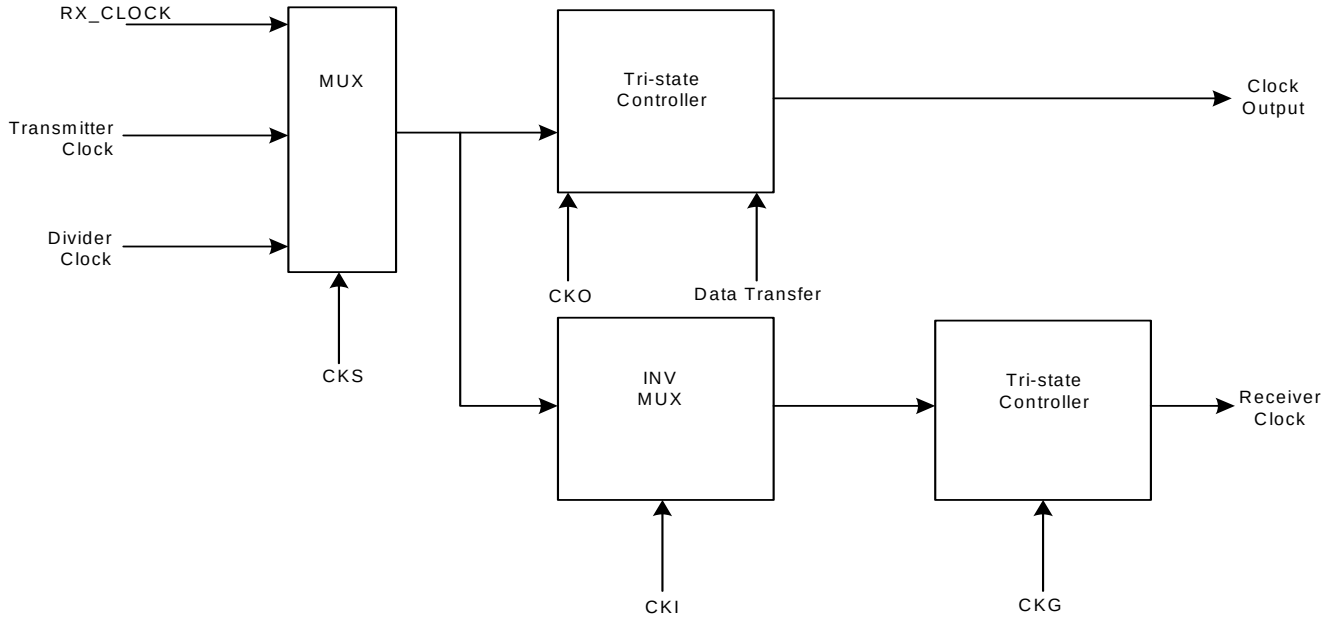
The start event is configured by writing to the RCMR register. See Section 25.7.4.

The frame synchronization is configured by writing to the Receive Frame Mode Register (RFMR). See Section 25.7.5.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the RCMR register. 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 Receive Holding Register (RHR), the Receive Ready bit is set in the SR register (SR.RXREADY) and the data can be read in the RHR register. If another transfer occurs before a read of the RHR register, the Receive Overrun bit is set in the SR register (SR.OVRUN) and the receiver shift register is transferred to the RHR register.

Figure 25-9. Receiver Block Diagram



25.7.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection field of the TCMR register (TCMR.START) and in the Receive Start Selection field of the RCMR register (RCMR.START).

Under the following conditions the start event is independently programmable:

- Continuous: in this case, the transmission starts as soon as a word is written to the THR register and the reception starts as soon as the receiver is enabled
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TX_FRAME_SYNC/RX_FRAME_SYNC
- On detection of a low/high level on TX_FRAME_SYNC/RX_FRAME_SYNC
- On detection of a level change or an edge on TX_FRAME_SYNC/RX_FRAME_SYNC

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Mode Register (TCMR/RCMR). Thus, the start could be on TX_FRAME_SYNC (transmit) or RX_FRAME_SYNC (receive).

Moreover, the receiver can start when data is detected in the bit stream with the compare functions. See Section 25.7.6 for more details on receive compare modes.

Detection on TX_FRAME_SYNC input/output is done by the Transmit Frame Sync Output Selection field in the TFMR register (TFMR.FSOS). Similarly, detection on RX_FRAME_SYNC input/output is done by the Receive Frame Output Sync Selection field in the RFMR register (RFMR.FSOS).

Figure 25-10. Transmit Start Mode

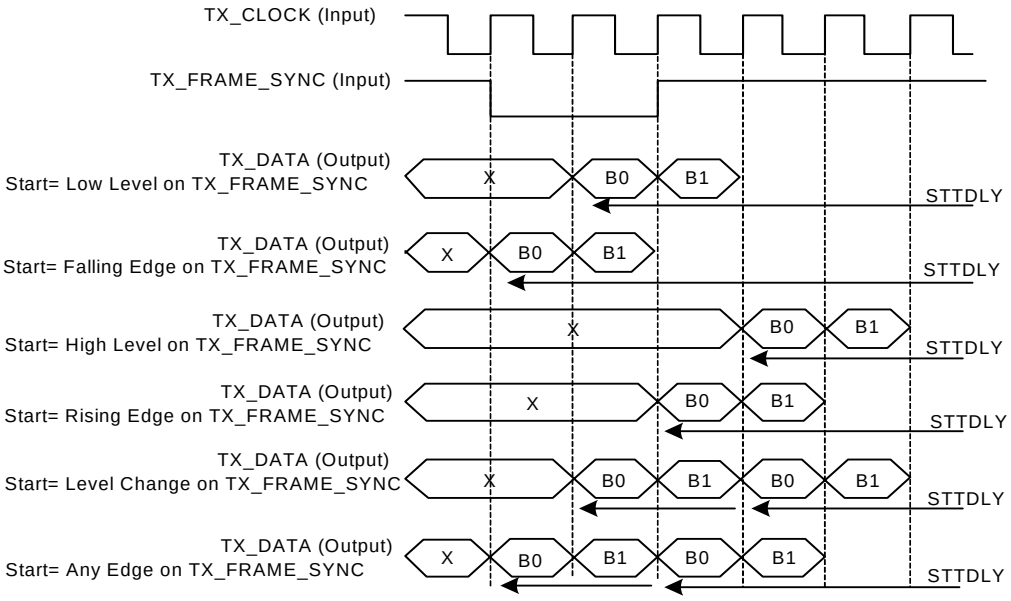
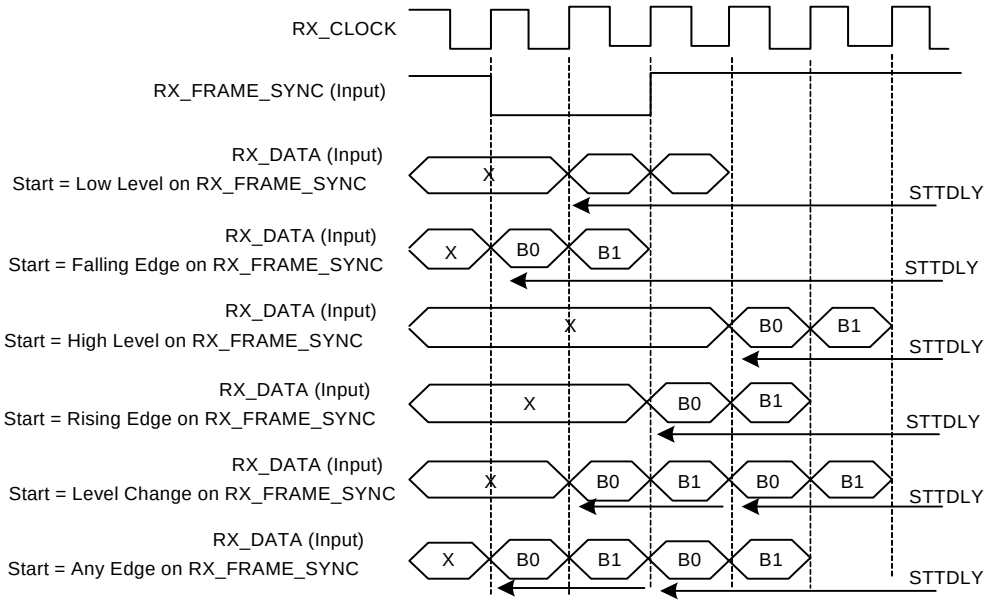


Figure 25-11. Receive Pulse/Edge Start Modes



25.7.5 Frame Sync

The transmitter and receiver frame synchro pins, TX_FRAME_SYNC and RX_FRAME_SYNC, can be programmed to generate different kinds of frame synchronization signals. The RFMR.FSOS and TFMR.FSOS fields 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, in reception, the Receive Frame Sync Length High Part and the Receive Frame Sync Length fields in the RFMR register (RFMR.FSLENHI and RFMR.FSLEN) define the length of the pulse, from 1 bit time up to 256 bit time.

$$\text{Reception Pulse Length} = ((16 \times FSLENHI) + FSLEN + 1) \text{ receive clock periods}$$

Similarly, in transmission, the Transmit Frame Sync Length High Part and the Transmit Frame Sync Length fields in the TFMR register (TFMR.FSLENHI and TFMR.FSLEN) define the length of the pulse, from 1 bit up to 256 bit time.

$$\text{Transmission Pulse Length} = ((16 \times FSLENHI) + FSLEN + 1) \text{ transmit clock periods}$$

The periodicity of the RX_FRAME_SYNC and TX_FRAME_SYNC pulse outputs can be configured respectively through the Receive Period Divider Selection field in the RCMR register (RCMR.PERIOD) and the Transmit Period Divider Selection field in the TCMR register (TCMR.PERIOD).

25.7.5.1 Frame sync data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the receiver can sample the RX_DATA line and store the data in the Receive Sync Holding Register (RSHR) and the transmitter can transfer the Transmit Sync Holding Register (TSHR) in the shifter register.

The data length to be sampled in reception during the Frame Sync signal shall be written to the RFMR.FSLENHI and RFMR.FSLEN fields.

The data length to be shifted out in transmission during the Frame Sync signal shall be written to the TFMR.FSLENHI and TFMR.FSLEN fields.

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 RSHR through the receive shift register.

The Transmit Frame Sync operation is performed by the transmitter only if the Frame Sync Data Enable bit in TFMR register (TFMR.FSDEN) is written to one. 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 TSHR is transferred in the transmit register, then shifted out.

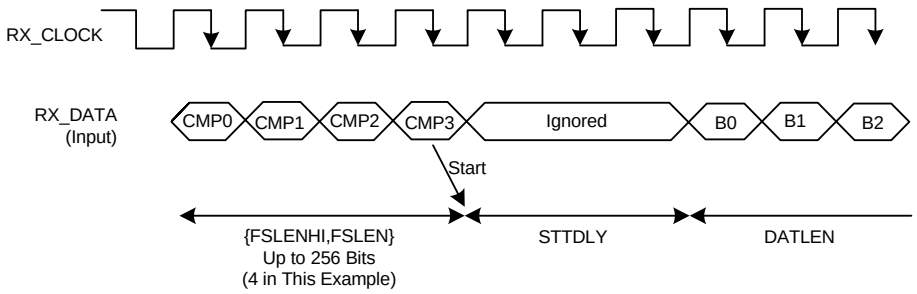
25.7.5.2 Frame sync edge detection

The Frame Sync Edge detection is configured by writing to the Frame Sync Edge Detection bit in the RFMR/TFMR registers (RFMR.FSEDGE and TFMR.FSEDGE). This sets the Receive Sync

and Transmit Sync bits in the SR register (SR.RXSYN and SR.TXSYN) on frame synchro edge detection (signals RX_FRAME_SYNC/TX_FRAME_SYNC).

25.7.6 Receive Compare Modes

Figure 25-12. Receive Compare Modes



25.7.6.1 Compare functions

Compare 0 can be one start event of the receiver. In this case, the receiver compares at each new sample the last {RFMR.FSLENHI, RFMR.FSLEN} bits received to the {RFMR.FSLENHI, RFMR.FSLEN} lower bits of the data contained in the Compare 0 Register (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 Receive Stop Selection bit in the RCMR register (RCMR.STOP).

25.7.7 Data Framing Format

The data framing format of both the transmitter and the receiver are programmable through the TFMR, TCMR, RFMR, and RCMR registers. In either case, the user can independently select:

- the event that starts the data transfer (RCMR.START and TCMR.START)
- the delay in number of bit periods between the start event and the first data bit (RCMR.STTDLY and TCMR.STTDLY)
- the length of the data (RFMR.DATLEN and TFMR.DATLEN)
- the number of data to be transferred for each start event (RFMR.DATNB and TFMR.DATLEN)
- the length of synchronization transferred for each start event (RFMR.FSLENHI, RFMR.FSLEN, TFMR.FSLENHI, and TFMR.FSLEN)
- the bit sense: most or lowest significant bit first (RFMR.MSBF and TFMR.MSBF)

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TX_DATA pin while not in data transfer operation. This is done respectively by writing to the Frame Sync Data Enable and the Data Default Value bits in the TFMR register (TFMR.FSDEN and TFMR.DATDEF).

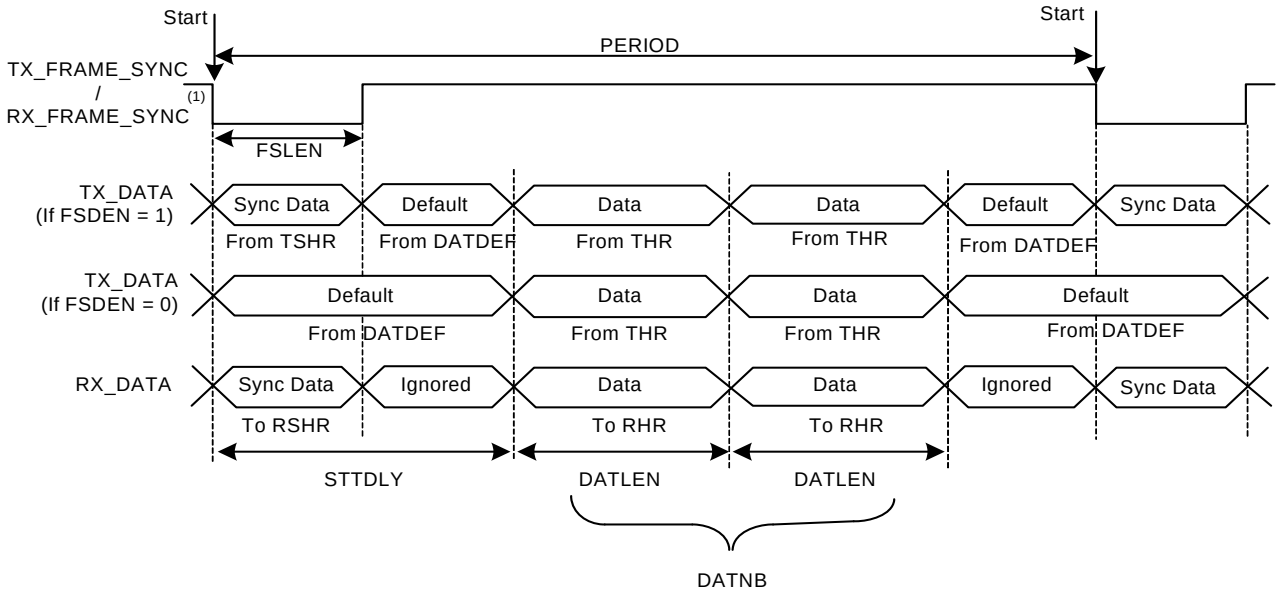
Table 25-3. Data Framing Format Registers

Transmitter	Receiver	Bit/Field	Length	Comment
TCMR	RCMR	PERIOD	Up to 512	Frame size
TCMR	RCMR	START		Start selection
TCMR	RCMR	STTDLY	Up to 255	Size of transmit start delay

Table 25-3. Data Framing Format Registers

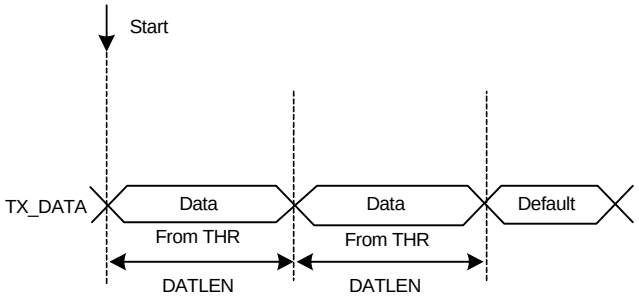
Transmitter	Receiver	Bit/Field	Length	Comment
TFMR	RFMR	DATNB	Up to 16	Number of words transmitted in frame
TFMR	RFMR	DATLEN	Up to 32	Size of word
TFMR	RFMR	{FSLENHI,FSLEN}	Up to 256	Size of Synchro data register
TFMR	RFMR	MSBF		Most significant bit first
TFMR		FSDEN		Enable send TSHR
TFMR		DATDEF		Data default value ended

Figure 25-13. Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: Example of input on falling edge of TX_FRAME_SYNC/RX_FRAME_SYNC.

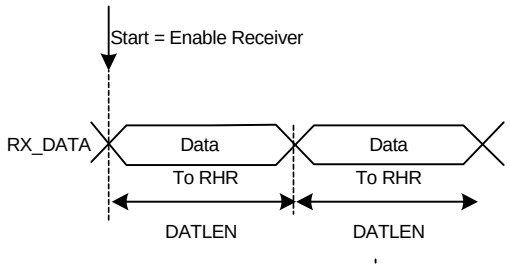
Figure 25-14. Transmit Frame Format in Continuous Mode



Start: 1. TXEMPTY set to one
2. Write into the THR

Note: STTDLY is written to zero. In this example, THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 25-15. Receive Frame Format in Continuous Mode



Note: STTDLY is written to zero.

25.7.8 Loop Mode

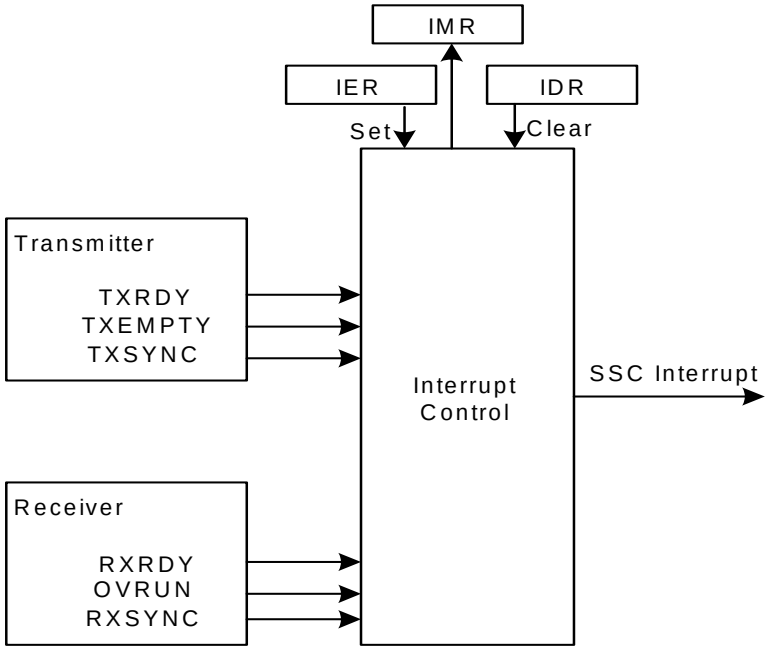
The receiver can be programmed to receive transmissions from the transmitter. This is done by writing a one to the Loop Mode bit in RFMR register (RFMR.LOOP). In this case, RX_DATA is connected to TX_DATA, RX_FRAME_SYNC is connected to TX_FRAME_SYNC and RX_CLOCK is connected to TX_CLOCK.

25.7.9 Interrupt

Most bits in the SR register 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 to the Interrupt Enable Register (IER) and Interrupt Disable Register (IDR). These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in the Interrupt Mask Register (IMR), which controls the generation of interrupts by asserting the SSC interrupt line connected to the interrupt controller.

Figure 25-16. Interrupt Block Diagram



25.8 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 25-17. Audio Application Block Diagram

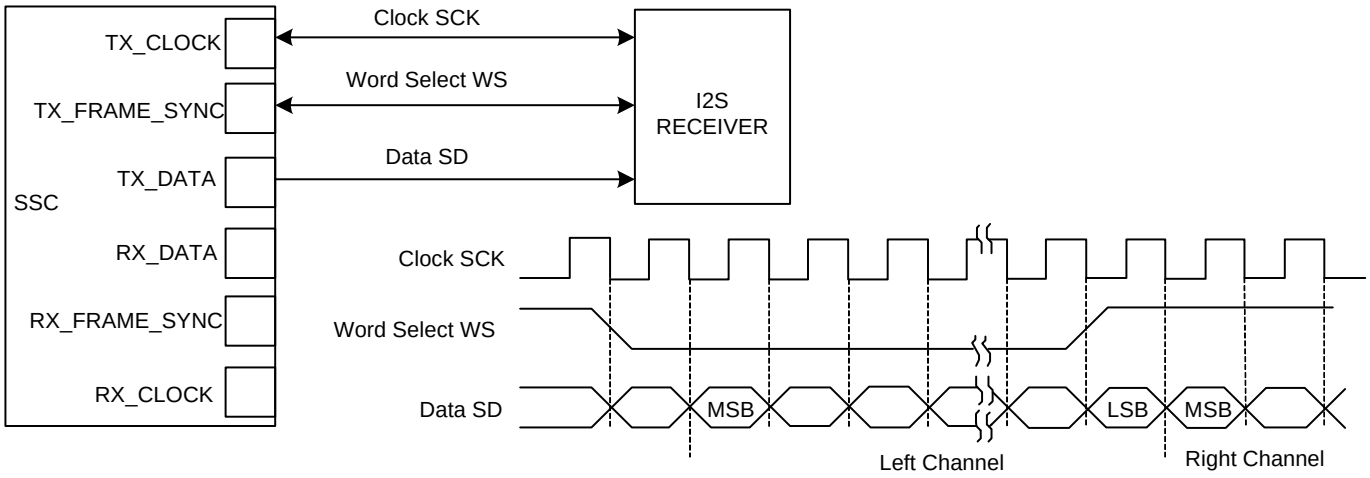


Figure 25-18. Codec Application Block Diagram

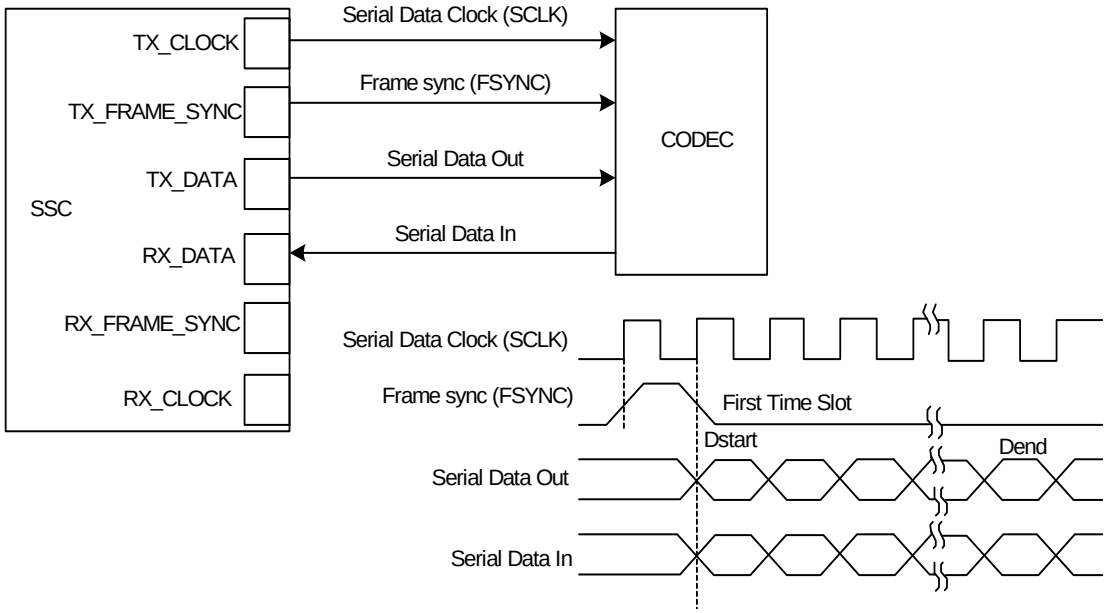
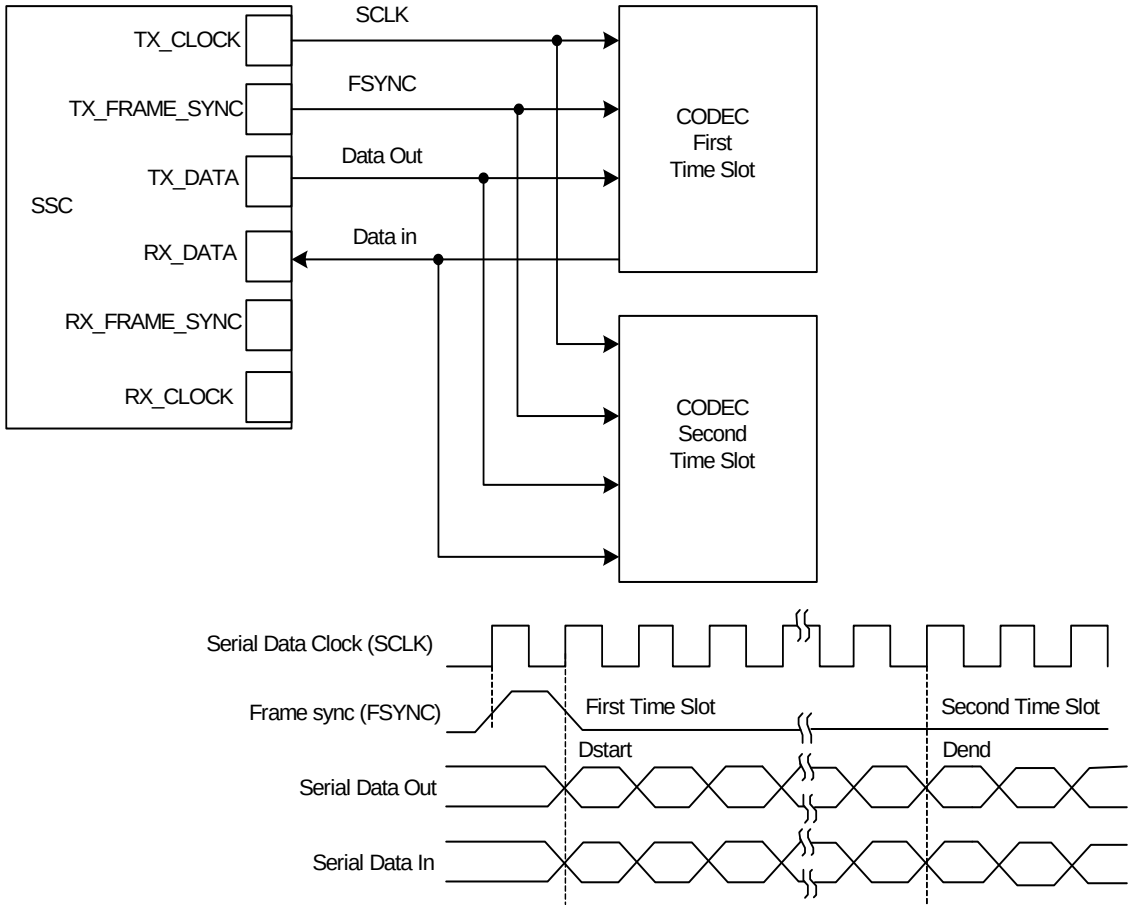


Figure 25-19. Time Slot Application Block Diagram



25.9 User Interface

Table 25-4. SSC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Clock Mode Register	CMR	Read/Write	0x00000000
0x10	Receive Clock Mode Register	RCMR	Read/Write	0x00000000
0x14	Receive Frame Mode Register	RFMR	Read/Write	0x00000000
0x18	Transmit Clock Mode Register	TCMR	Read/Write	0x00000000
0x1C	Transmit Frame Mode Register	TFMR	Read/Write	0x00000000
0x20	Receive Holding Register	RHR	Read-only	0x00000000
0x24	Transmit Holding Register	THR	Write-only	0x00000000
0x30	Receive Synchronization Holding Register	RSHR	Read-only	0x00000000
0x34	Transmit Synchronization Holding Register	TSHR	Read/Write	0x00000000
0x38	Receive Compare 0 Register	RC0R	Read/Write	0x00000000
0x3C	Receive Compare 1 Register	RC1R	Read/Write	0x00000000
0x40	Status Register	SR	Read-only	0x000000CC
0x44	Interrupt Enable Register	IER	Write-only	0x00000000
0x48	Interrupt Disable Register	IDR	Write-only	0x00000000
0x4C	Interrupt Mask Register	IMR	Read-only	0x00000000

25.9.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

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
SWRST	-	-	-	-	-	TXDIS	TXEN

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RXDIS	RXEN

- **SWRST: Software Reset**
  1: Writing a one to this bit will perform a software reset. This software reset has priority on any other bit in CR.
  0: Writing a zero to this bit has no effect.
- **TXDIS: Transmit Disable**
  1: Writing a one to this bit will disable the transmission. If a character is currently being transmitted, the disable occurs at the end of the current character transmission.
  0: Writing a zero to this bit has no effect.
- **TXEN: Transmit Enable**
  1: Writing a one to this bit will enable the transmission if the TXDIS bit is not written to one.
  0: Writing a zero to this bit has no effect.
- **RXDIS: Receive Disable**
  1: Writing a one to this bit will disable the reception. If a character is currently being received, the disable occurs at the end of current character reception.
  0: Writing a zero to this bit has no effect.
- **RXEN: Receive Enable**
  1: Writing a one to this bit will enables the reception if the RXDIS bit is not written to one.
  0: Writing a zero to this bit has no effect.

25.9.2 Clock Mode Register

Name: CMR

Access Type: Read/Write

Offset: 0x04

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
-	-	-	-	DIV[11:8]			

7	6	5	4	3	2	1	0
DIV[7:0]							

- **DIV[11:0]: Clock Divider**

  The divided clock equals the CLK_SSC divided by two times DIV. The maximum bit rate is CLK_SSC/2. The minimum bit rate is CLK_SSC/(2 x 4095) = CLK_SSC/8190.

  The clock divider is not active when DIV equals zero.

$$\text{Divided Clock} = \text{CLK\_SSC} / (DIV \times 2)$$

25.9.3 Receive Clock Mode Register

Name: RCMR

Access Type: Read/Write

Offset: 0x10

Reset value: 0x00000000

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
-	-	-	STOP	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

- **PERIOD: Receive Period Divider Selection**
  This field selects the divider to apply to the selected receive clock in order to generate a periodic Frame Sync Signal.
  If equal to zero, no signal is generated.
  If not equal to zero, a signal is generated each 2 x (PERIOD+1) receive clock periods.
- **STTDLY: Receive Start Delay**
  If STTDLY is not zero, 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 written carefully. If STTDLY must be written, it should be done in relation to Receive Sync Data reception.
- **STOP: Receive Stop Selection**
  1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.
  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.

- **START: Receive Start Selection**

START	Receive Start
0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
1	Transmit start
2	Detection of a low level on RX_FRAME_SYNC signal
3	Detection of a high level on RX_FRAME_SYNC signal
4	Detection of a falling edge on RX_FRAME_SYNC signal
5	Detection of a rising edge on RX_FRAME_SYNC signal
6	Detection of any level change on RX_FRAME_SYNC signal
7	Detection of any edge on RX_FRAME_SYNC signal
8	Compare 0
Others	Reserved

- **CKG: Receive Clock Gating Selection**

CKG	Receive Clock Gating
0	None, continuous clock
1	Receive Clock enabled only if RX_FRAME_SYNC is low
2	Receive Clock enabled only if RX_FRAME_SYNC is high
3	Reserved

- **CKI: Receive Clock Inversion**

  CKI affects only the receive clock and not the output clock signal.

  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.

  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.

- **CKO: Receive Clock Output Mode Selection**

CKO	Receive Clock Output Mode	RX_CLOCK pin
0	None	Input-only
1	Continuous receive clock	Output
2	Receive clock only during data transfers	Output
Others	Reserved	

- **CKS: Receive Clock Selection**

CKS	Selected Receive Clock
0	Divided clock
1	TX_CLOCK clock signal
2	RX_CLOCK pin
3	Reserved

25.9.4 Receive Frame Mode Register

Name: RFMR

Access Type: Read/Write

Offset: 0x14

Reset value: 0x00000000

31	30	29	28	27	26	25	24
FSLENHI				-	-	-	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				

- **FSLENHI: Receive Frame Sync Length High Part**
  The four MSB of the FSLEN field.
- **FSEDGE: Receive Frame Sync Edge Detection**
  Determines which edge on Frame Sync will generate the SR.RXSYN interrupt.

FSEDGE	Frame Sync Edge Detection
0	Positive edge detection
1	Negative edge detection

- **FSOS: Receive Frame Sync Output Selection**

FSOS	Selected Receive Frame Sync Signal	RX_FRAME_SYNC Pin
0	None	Input-only
1	Negative Pulse	Output
2	Positive Pulse	Output
3	Driven Low during data transfer	Output
4	Driven High during data transfer	Output
5	Toggling at each start of data transfer	Output
Others	Reserved	Undefined

- **FSLEN: Receive Frame Sync Length**
  This field defines the length of the Receive Frame Sync signal and the number of bits sampled and stored in the RSHR register. When this mode is selected by the RCMR.START field, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.
  Note: The four most significant bits for this field are located in the FSLENHI field.
  The pulse length is equal to ({FSLENHI,FSLEN} + 1) receive clock periods. Thus, if {FSLENHI,FSLEN} is zero, the Receive Frame Sync signal is generated during one receive clock period.

- **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).
- **MSBF: Most Significant Bit First**
  1: The most significant bit of the data register is sampled first in the bit stream.
  0: The lowest significant bit of the data register is sampled first in the bit stream.
- **LOOP: Loop Mode**
  1: RX_DATA is driven by TX_DATA, RX_FRAME_SYNC is driven by TX_FRAME_SYNC and TX_CLOCK drives RX_CLOCK.
  0: Normal operating mode.
- **DATLEN: Data Length**
  The bit stream contains (DATLEN + 1) data bits.
  This field also defines the transfer size performed by the Peripheral DMA Controller assigned to the receiver.

DATLEN	Transfer Size
0	Forbidden value
1-7	Data transfer are in bytes
8-15	Data transfer are in halfwords
Others	Data transfer are in words

25.9.5 Transmit Clock Mode Register

Name: TCMR

Access Type: Read/Write

Offset: 0x18

Reset value: 0x00000000

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	

- **PERIOD: Transmit Period Divider Selection**
  This field selects the divider to apply to the selected transmit clock in order to generate a periodic Frame Sync Signal.
  If equal to zero, no signal is generated.
  If not equal to zero, a signal is generated each 2 x (PERIOD+1) transmit clock periods.
- **STTDLY: Transmit Start Delay**
  If STTDLY is not zero, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission.
  When the transmitter is programmed to start synchronously with the receiver, the delay is also applied.
  Note: STTDLY must be written carefully, in relation to Transmit Sync Data transmission.
- **START: Transmit Start Selection**

START	Transmit Start
0	Continuous, as soon as a word is written to the THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
1	Receive start
2	Detection of a low level on TX_FRAME_SYNC signal
3	Detection of a high level on TX_FRAME_SYNC signal
4	Detection of a falling edge on TX_FRAME_SYNC signal
5	Detection of a rising edge on TX_FRAME_SYNC signal
6	Detection of any level change on TX_FRAME_SYNC signal
7	Detection of any edge on TX_FRAME_SYNC signal
Others	Reserved

- **CKG: Transmit Clock Gating Selection**

CKG	Transmit Clock Gating
0	None, continuous clock
1	Transmit Clock enabled only if TX_FRAME_SYNC is low
2	Transmit Clock enabled only if TX_FRAME_SYNC is high
3	Reserved

- **CKI: Transmit Clock Inversion**

  CKI affects only the Transmit Clock and not the output clock signal.

  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.

  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.

- **CKO: Transmit Clock Output Mode Selection**

CKO	Transmit Clock Output Mode	TX_CLOCK pin
0	None	Input-only
1	Continuous transmit clock	Output
2	Transmit clock only during data transfers	Output
Others	Reserved	

- **CKS: Transmit Clock Selection**

CKS	Selected Transmit Clock
0	Divided Clock
1	RX_CLOCK clock signal
2	TX_CLOCK Pin
3	Reserved

25.9.6 Transmit Frame Mode Register

Name: TFMR

Access Type: Read/Write

Offset: 0x1C

Reset value: 0x00000000

31	30	29	28	27	26	25	24
FSLENHI				-	-	-	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				

- **FSLENHI: Transmit Frame Sync Length High Part**
  The four MSB of the FSLEN field.
- **FSEDGE: Transmit Frame Sync Edge Detection**
  Determines which edge on Frame Sync will generate the SR.TXSYN interrupt.

FSEDGE	Frame Sync Edge Detection
0	Positive Edge Detection
1	Negative Edge Detection

- **FSDEN: Transmit Frame Sync Data Enable**
  1: TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.
  0: The TX_DATA line is driven with the default value during the Transmit Frame Sync signal.
- **FSOS: Transmit Frame Sync Output Selection**

FSOS	Selected Transmit Frame Sync Signal	TX_FRAME_SYNC Pin
0	None	Input-only
1	Negative Pulse	Output
2	Positive Pulse	Output
3	Driven Low during data transfer	Output
4	Driven High during data transfer	Output
5	Toggling at each start of data transfer	Output
Others	Reserved	Undefined

- **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 TSHR register if TFMR.FSDEN is equal to one.
  Note: The four most significant bits for this field are located in the FSLENHI field.

The pulse length is equal to ({FSLENHI,FSLEN} + 1) transmit clock periods, i.e., the pulse length can range from 1 to 256 transmit clock periods. If {FSLENHI,FSLEN} is zero, the Transmit Frame Sync signal is generated during one transmit clock period.

- **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).
- **MSBF: Most Significant Bit First**
  1: The most significant bit of the data register is shifted out first in the bit stream.
  0: The lowest significant bit of the data register is shifted out first in the bit stream.
- **DATDEF: Data Default Value**
  This bit defines the level driven on the TX_DATA pin while out of transmission.
  Note that if the pin is defined as multi-drive by the I/O Controller, the pin is enabled only if the TX_DATA output is one.
  1: The level driven on the TX_DATA pin while out of transmission is one.
  0: The level driven on the TX_DATA pin while out of transmission is zero.
- **DATLEN: Data Length**
  The bit stream contains (DATLEN + 1) data bits.
  This field also defines the transfer size performed by the Peripheral DMA Controller assigned to the transmitter.

DATLEN	Transfer Size
0	Forbidden value (1-bit data length is not supported)
1-7	Data transfer are in bytes
8-15	Data transfer are in halfwords
Others	Data transfer are in words

25.9.7 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x20

Reset value: 0x00000000

31	30	29	28	27	26	25	24
RDAT[31:24]							

23	22	21	20	19	18	17	16
RDAT[23:16]							

15	14	13	12	11	10	9	8
RDAT[15:8]							

7	6	5	4	3	2	1	0
RDAT[7:0]							

- **RDAT: Receive Data**
  Right aligned regardless of the number of data bits defined by the RFMR.DATLEN field.

25.9.8 Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x24

Reset value: 0x00000000

31	30	29	28	27	26	25	24
TDAT[31:24]							

23	22	21	20	19	18	17	16
TDAT[23:16]							

15	14	13	12	11	10	9	8
TDAT[15:8]							

7	6	5	4	3	2	1	0
TDAT[7:0]							

- **TDAT: Transmit Data**
  Right aligned regardless of the number of data bits defined by the TFMR.DATLEN field.

25.9.9 Receive Synchronization Holding Register

Name: RSHR

Access Type: Read-only

Offset: 0x30

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
RSDAT[15:8]							

7	6	5	4	3	2	1	0
RSDAT[7:0]							

- **RSDAT: Receive Synchronization Data**

25.9.10 Transmit Synchronization Holding Register

Name: TSHR

Access Type: Read/Write

Offset: 0x34

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
TSDAT[15:8]							

7	6	5	4	3	2	1	0
TSDAT[7:0]							

- **TSDAT: Transmit Synchronization Data**

25.9.11 Receive Compare 0 Register

Name: RC0R

Access Type: Read/Write

Offset: 0x38

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
CP0[15:8]							

7	6	5	4	3	2	1	0
CP0[7:0]							

- **CP0: Receive Compare Data 0**

25.9.12 Receive Compare 1 Register

Name: RC1R

Access Type: Read/Write

Offset: 0x3C

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
CP1[[15:8]							

7	6	5	4	3	2	1	0
CP1[7:0]							

- **CP1: Receive Compare Data 1**

25.9.13 Status Register

Name: SR

Access Type: Read-only

Offset: 0x40

Reset value: 0x000000CC

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	RXEN	TXEN
15	14	13	12	11	10	9	8
-	-	-	-	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
-	-	OVRUN	RXRDY	-	-	TXEMPTY	TXRDY

- **RXEN: Receive Enable**
  This bit is set when the CR.RXEN bit is written to one.
  This bit is cleared when no data are being processed and the CR.RXDIS bit has been written to one.
- **TXEN: Transmit Enable**
  This bit is set when the CR.TXEN bit is written to one.
  This bit is cleared when no data are being processed and the CR.TXDIS bit has been written to one.
- **RXSYN: Receive Sync**
  This bit is set when a Receive Sync has occurred.
  This bit is cleared when the SR register is read.
- **TXSYN: Transmit Sync**
  This bit is set when a Transmit Sync has occurred.
  This bit is cleared when the SR register is read.
- **CP1: Compare 1**
  This bit is set when compare 1 has occurred.
  This bit is cleared when the SR register is read.
- **CP0: Compare 0**
  This bit is set when compare 0 has occurred.
  This bit is cleared when the SR register is read.
- **OVRUN: Receive Overrun**
  This bit is set when data has been loaded in the RHR register while previous data has not yet been read.
  This bit is cleared when the SR register is read.
- **RXRDY: Receive Ready**
  This bit is set when data has been received and loaded in the RHR register.
  This bit is cleared when the RHR register is empty.
- **TXEMPTY: Transmit Empty**
  This bit is set when the last data written in the THR register has been loaded in the TSR register and last data loaded in the TSR register has been transmitted.
  This bit is cleared when data remains in the THR register or is currently transmitted from the TSR register.

- **TXRDY: Transmit Ready**

  This bit is set when the THR register is empty.

  This bit is cleared when data has been loaded in the THR register and is waiting to be loaded in the TSR register.

25.9.14 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x44

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
-	-	-	-	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

25.9.15 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x48

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
-	-	-	-	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

25.9.16 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x4C

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
-	-	-	-	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

26. Universal Synchronous Asynchronous Receiver Transmitter (USART)

Rev.4.2.0.3

26.1 Features

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Over-sampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Optional Modem Signal Management DTR-DSR-DCD-RI
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 Kbps
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (CLK) Frequency up to Internal Clock Frequency CLK_USART/4
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 specifications
  - Master or Slave
  - Processing of frames with up to 256 data bytes
  - Response Data length can be configurable or defined automatically by the Identifier
  - Self synchronization in Slave node configuration
  - Automatic processing and verification of the “Synch Break” and the “Synch Field”
  - The “Synch Break” is detected even if it is partially superimposed with a data byte
  - Automatic Identifier parity calculation/sending and verification
  - Parity sending and verification can be disabled
  - Automatic Checksum calculation/sending and verification
  - Checksum sending and verification can be disabled
  - Support both “Classic” and “Enhanced” checksum types
  - Full LIN error checking and reporting
  - Frame Slot Mode: the Master allocates slots to the scheduled frames automatically.
  - Generation of the Wakeup signal
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of Two Peripheral DMA Controller Channels (PDCA)
  - Offers Buffer Transfer without Processor Intervention

26.2 Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programma-

ble (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485, LIN and SPI buses, with ISO7816 T = 0 or T = 1 smart card slots, 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 PDCA provides chained buffer management without any intervention of the processor.

26.3 Block Diagram

Figure 26-1. USART Block Diagram

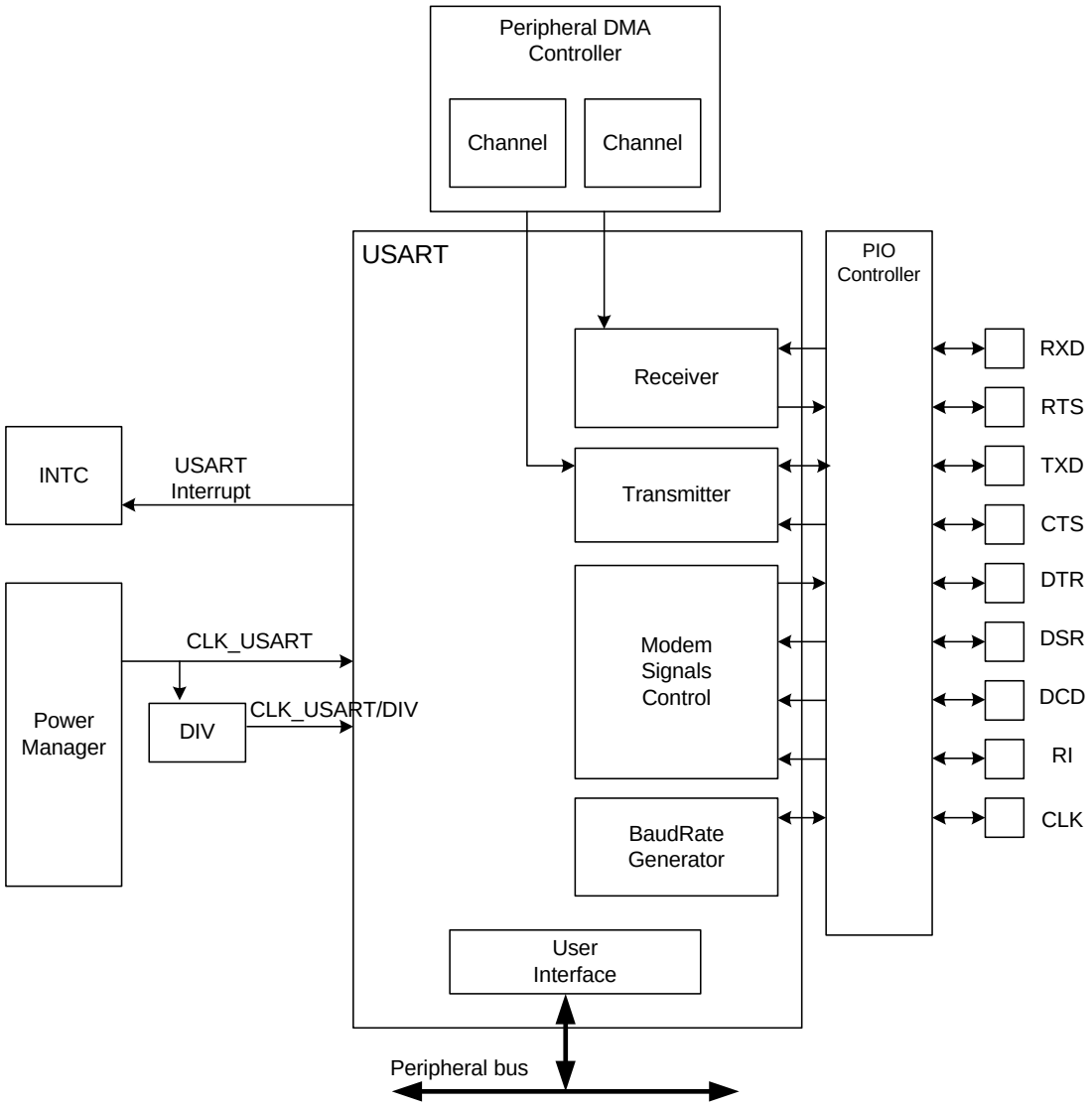
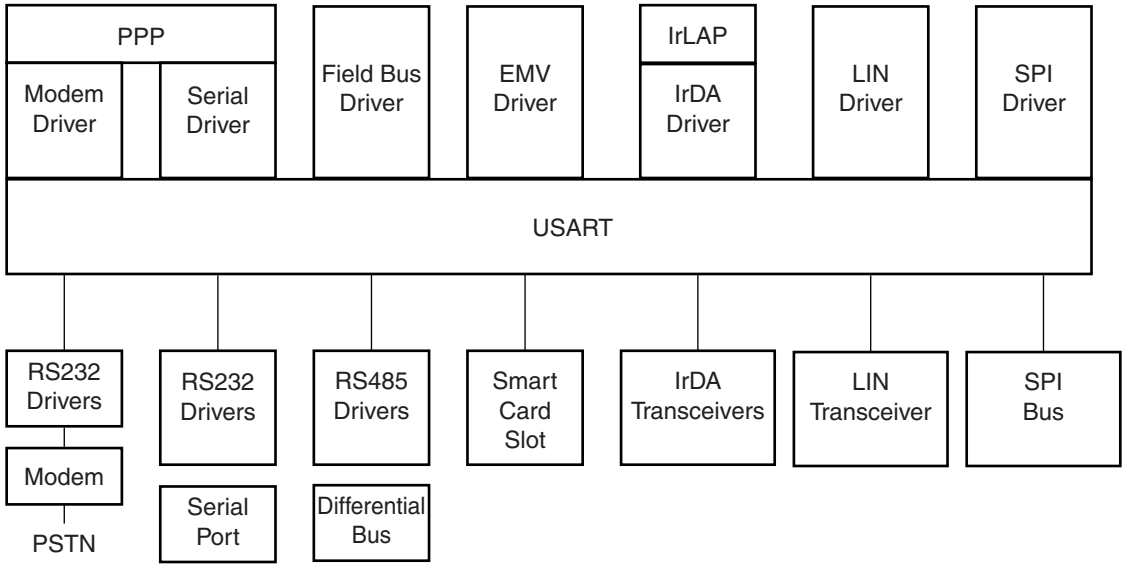


Table 26-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
RXD	RXD	MOSI	MISO
TXD	TXD	MISO	MOSI
RTS	RTS	–	CS
CTS	CTS	CS	–

26.4 Application Block Diagram

Figure 26-2. Application Block Diagram



26.5 I/O Lines Description

Table 26-2. I/O Lines Description

Name	Description	Type	Active Level
CLK	Serial Clock	I/O	
TXD	Transmit Serial Data or Master Out Slave In (MOSI) in SPI Master Mode or Master In Slave Out (MISO) in SPI Slave Mode	Output	
RXD	Receive Serial Data or Master In Slave Out (MISO) in SPI Master Mode or Master Out Slave In (MOSI) in SPI Slave Mode	Input	
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 or Slave Select (NSS) in SPI Slave Mode	Input	Low
RTS	Request to Send or Slave Select (NSS) in SPI Master Mode	Output	Low

26.6 Product Dependencies

26.6.1 I/O Lines

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

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory. If the hardware handshaking feature or Modem mode is used, the internal pull up on TXD must also be enabled.

All the pins of the modems may or may not be implemented on the USART. On USARTs not equipped with the corresponding pins, the associated control bits and statuses have no effect on the behavior of the USART.

26.6.2 Clocks

The clock for the USART bus interface (CLK_USART) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the USART before disabling the clock, to avoid freezing the USART in an undefined state.

26.6.3 Interrupts

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the INTC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

26.7 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional 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
- **SPI Mode**
  - **Master or Slave**
  - **Serial Clock Programmable Phase and Polarity**
  - **SPI Serial Clock (CLK) Frequency up to Internal Clock Frequency CLK_USART/4**
- **LIN Mode**
  - **Compliant with LIN 1.3 and LIN 2.0 specifications**
  - **Master or Slave**
  - **Processing of frames with up to 256 data bytes**
  - **Response Data length can be configurable or defined automatically by the Identifier**
  - **Self synchronization in Slave node configuration**
  - **Automatic processing and verification of the “Synch Break” and the “Synch Field”**
  - **The “Synch Break” is detected even if it is partially superimposed with a data byte**
  - **Automatic Identifier parity calculation/sending and verification**
  - **Parity sending and verification can be disabled**
  - **Automatic Checksum calculation/sending and verification**
  - **Checksum sending and verification can be disabled**
  - **Support both “Classic” and “Enhanced” checksum types**

- **Full LIN error checking and reporting**
- **Frame Slot Mode: the Master allocates slots to the scheduled frames automatically.**
- **Generation of the Wakeup signal**

• Test modes
  – Remote loopback, local loopback, automatic echo

26.7.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

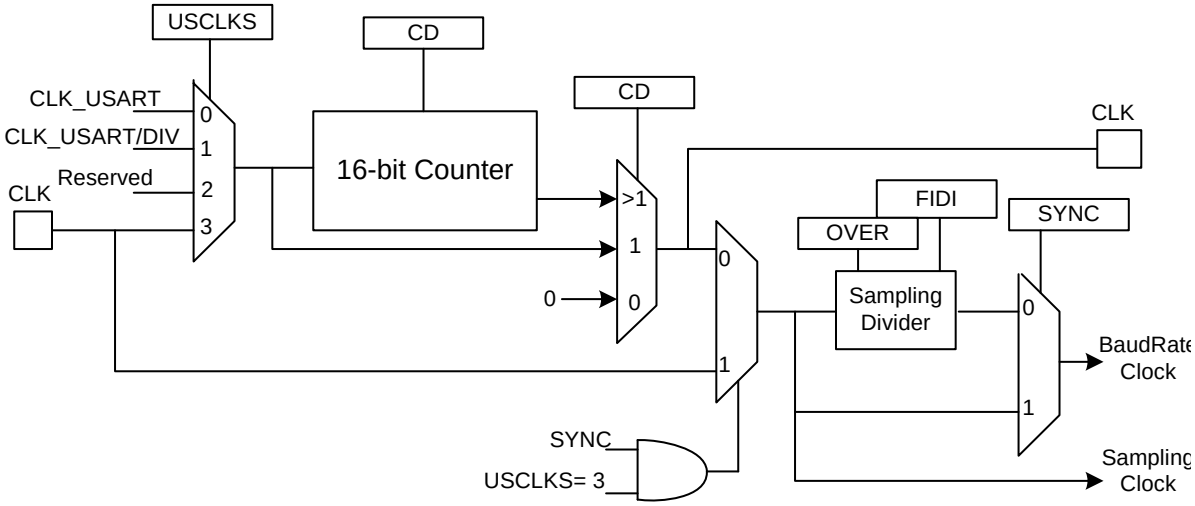
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (MR) between:

- CLK_USART
- a division of CLK_USART, the divider being product dependent, but generally set to 8
- the external clock, available on the CLK 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 (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 CLK clock is selected, the duration of the low and high levels of the signal provided on the CLK pin must be longer than a CLK_USART period. The frequency of the signal provided on CLK must be at least 4.5 times lower than CLK_USART.

Figure 26-3. Baud Rate Generator



26.7.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (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 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 CLK_USART divided by 8, assuming that CLK_USART is the highest possible clock and that OVER is programmed at 1.

26.7.1.2 Baud Rate Calculation Example

Table 26-3 on page 537 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 26-3. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

The baud rate is calculated with the following formula:

$$BaudRate = (CLKUSART)/CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

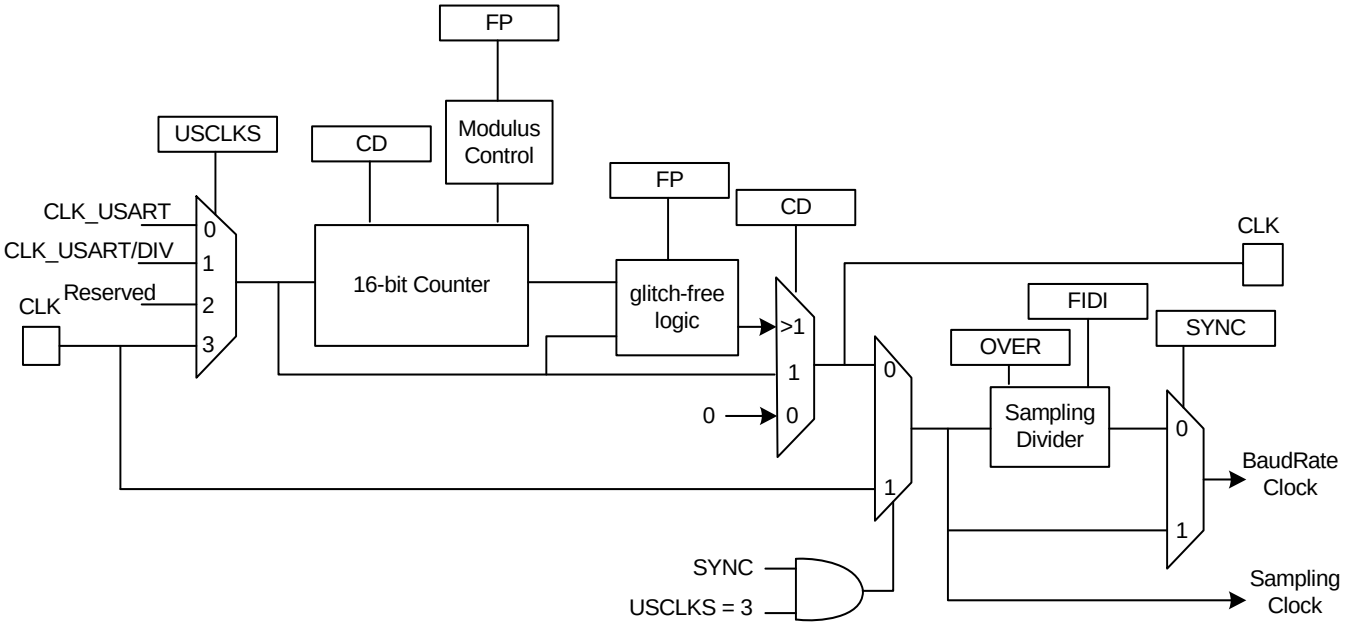
26.7.1.3 Fractional Baud Rate in Asynchronous Mode

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 (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 26-4. Fractional Baud Rate Generator



26.7.1.4 Baud Rate in Synchronous Mode or SPI Mode

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in 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 CLK pin. No division is active. The value written in 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 CLK or the internal clock divided (CLK_USART/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 CLK pin. If the internal clock CLK_USART is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the CLK pin, even if the value programmed in CD is odd.

26.7.1.5 *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 26-4 on page 539.

Table 26-4. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 26-5 on page 539.

Table 26-5. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 26-6 on page 539 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 26-6. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

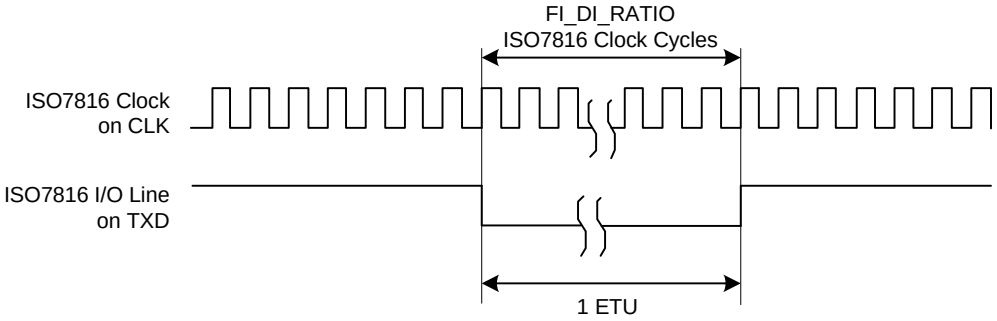
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (BRGR). The resulting clock can be provided to the CLK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (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 26-5 on page 540 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 26-5. Elementary Time Unit (ETU)



26.7.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (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 (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 (CR). The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in 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 (THR). If a time-guard is programmed, it is handled normally.

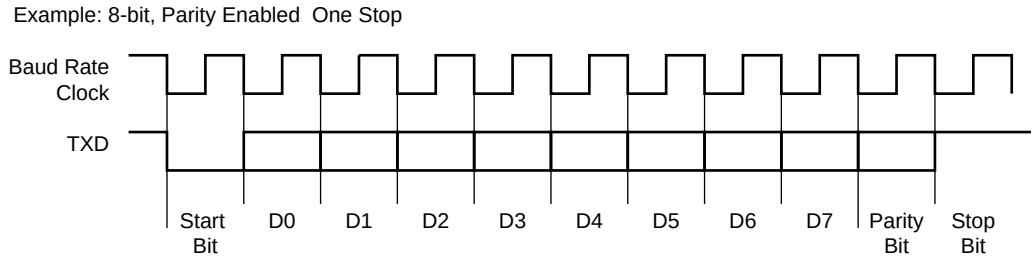
26.7.3 Synchronous and Asynchronous Modes

26.7.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (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 MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in 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 MR. The 1.5 stop bit is supported in asynchronous mode only.

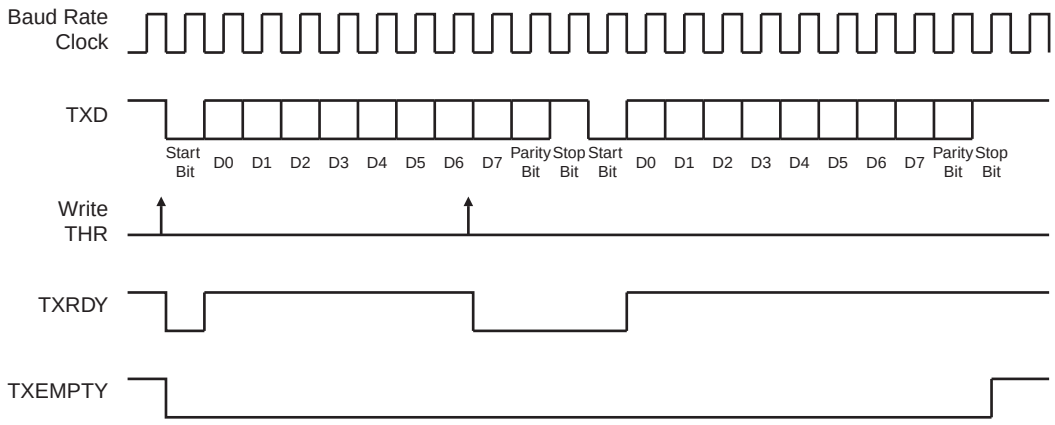
Figure 26-6. Character Transmit



The characters are sent by writing in the Transmit Holding Register (THR). The transmitter reports two status bits in the Channel Status Register (CSR): TXRDY (Transmitter Ready), which indicates that THR is empty and TXEMPTY, which indicates that all the characters written in THR have been processed. When the current character processing is completed, the last character written in THR is transferred into the Shift Register of the transmitter and THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in THR while TXRDY is low has no effect and the written character is lost.

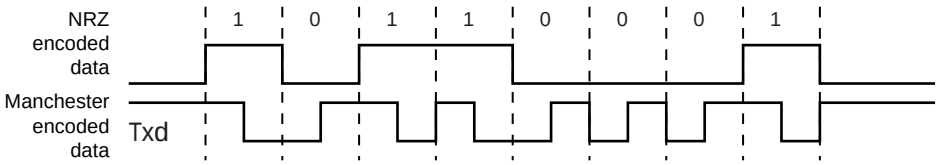
Figure 26-7. Transmitter Status



26.7.3.2 Manchester Encoder

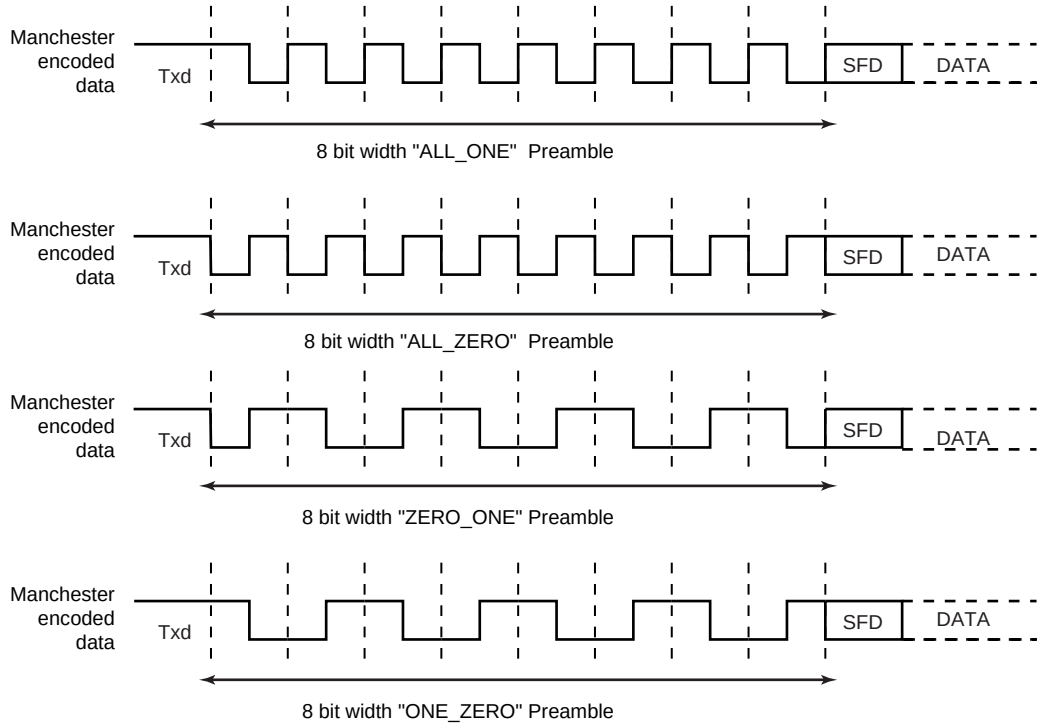
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 26-8 on page 542 illustrates this coding scheme.

Figure 26-8. NRZ to Manchester Encoding



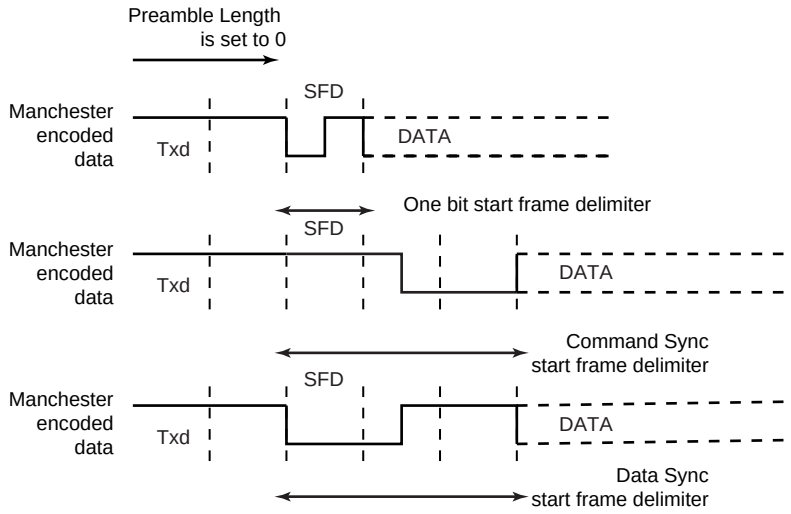
The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the MAN register, the field TX_PL is used to configure the preamble length. Figure 26-9 on page 543 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to one, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

Figure 26-9. Preamble Patterns, Default Polarity Assumed



A start frame delimiter is to be configured using the ONEBIT field in the MR register. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 26-10 on page 544 illustrates these patterns. If the start frame delimiter, also known as start bit, is one bit, (ONEBIT at 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT at 0), a sequence of 3 bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC field in the MR register is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC field in MR register must be set to 1. In this case, the MODSYNC field in MR is bypassed and the sync configuration is held in the TXSYNH in the THR register. The USART character format is modified and includes sync information.

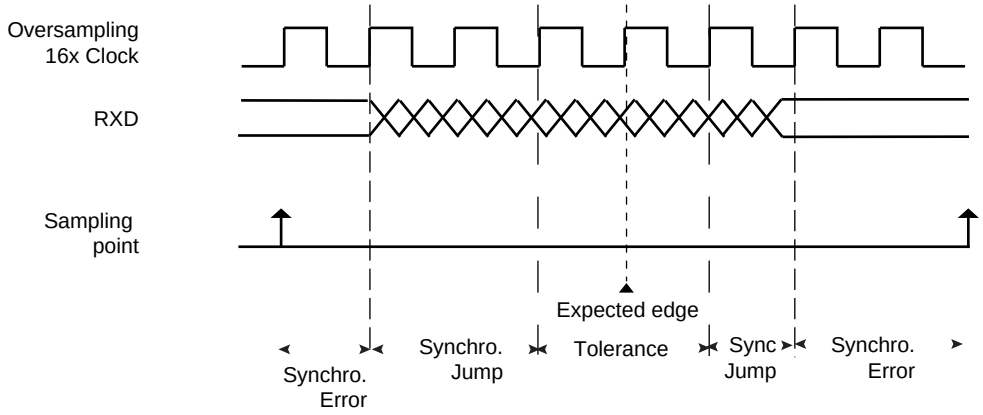
Figure 26-10. Start Frame Delimiter



Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 26-11. Bit Resynchronization



26.7.3.3 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (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 26-12 on page 545 and Figure 26-13 on page 545 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 26-12. Asynchronous Start Detection

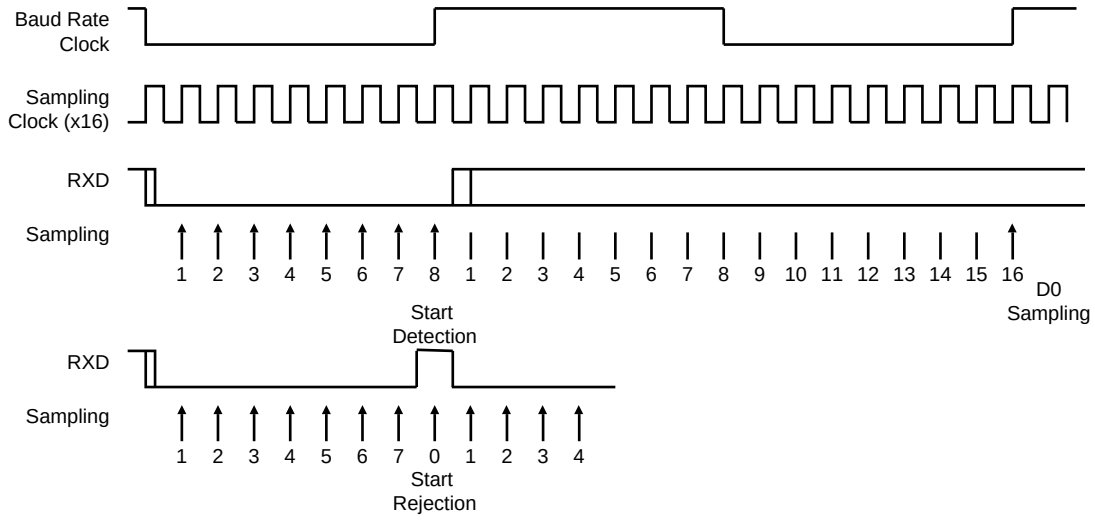
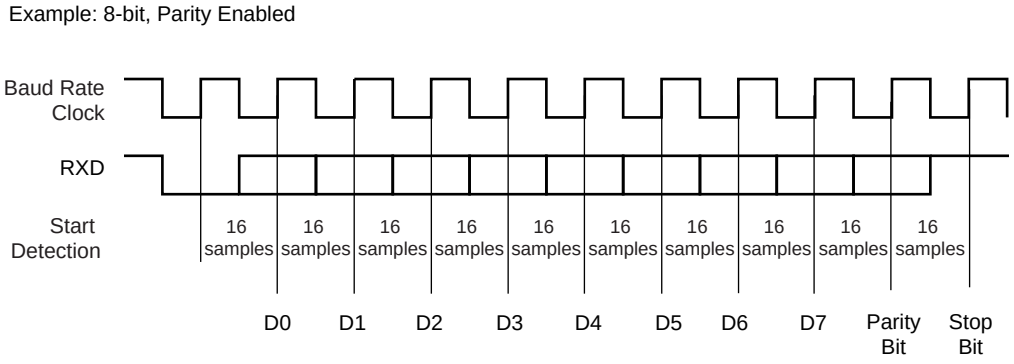


Figure 26-13. Asynchronous Character Reception



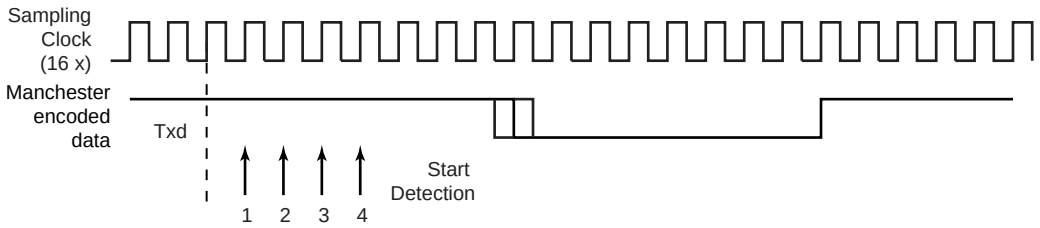
26.7.3.4 Manchester Decoder

When the MAN field in MR register is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL field in MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in MAN. See Figure 26-9 on page 543 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time at zero, a start bit is detected. See Figure 26-14 on page 546. The sample pulse rejection mechanism applies.

Figure 26-14. Asynchronous Start Bit Detection



The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver re-synchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 26-15 on page 547 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in CSR register is raised. It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1. See Figure 26-16 on page 547 for an example of Manchester error detection during data phase.

Figure 26-15. Preamble Pattern Mismatch

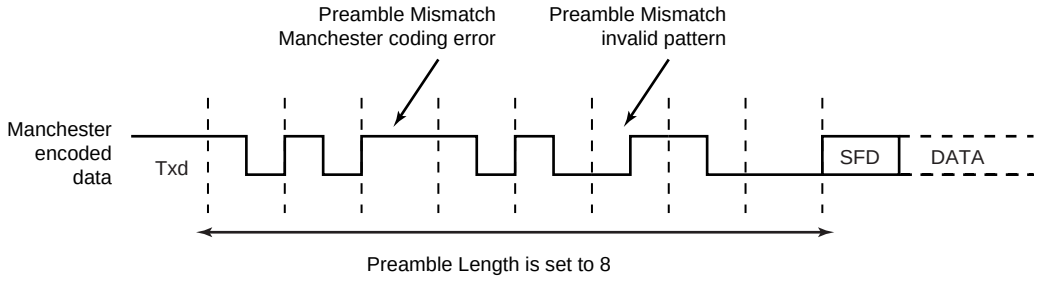
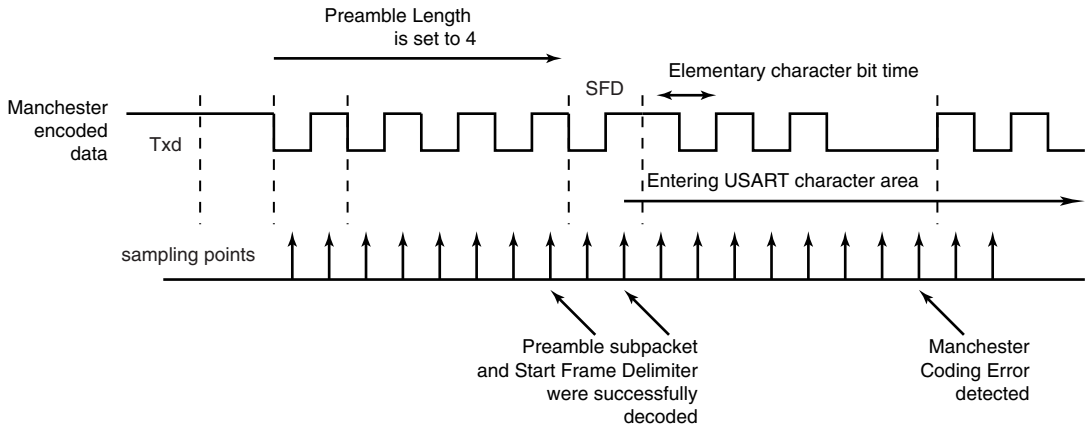


Figure 26-16. Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field at 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the RHR register and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

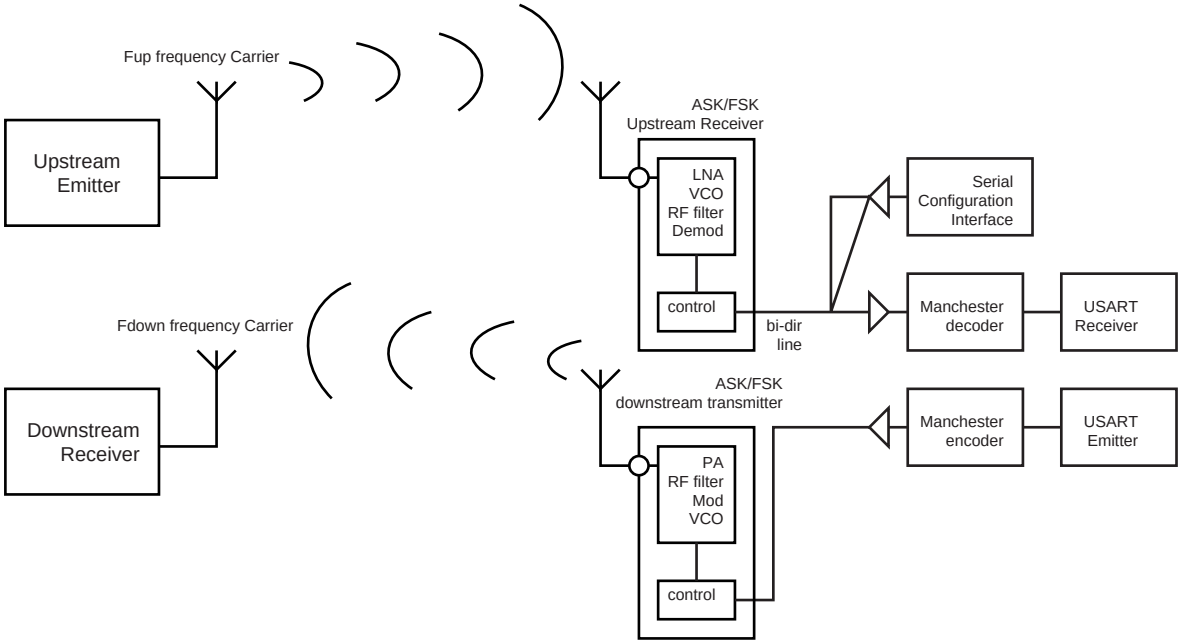
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

26.7.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 26-17 on page 548.

Figure 26-17. Manchester Encoded Characters RF Transmission



The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 26-18 on page 548 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 26-19 on page 549.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 26-18. ASK Modulator Output

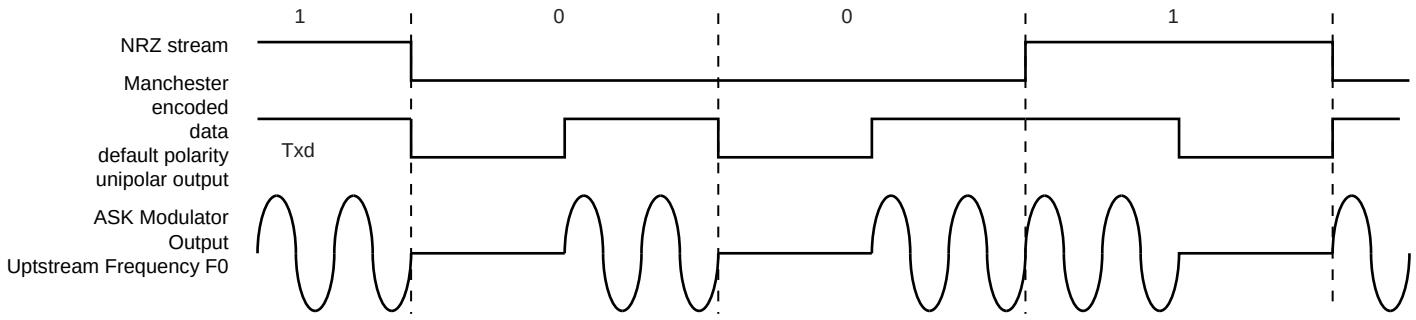
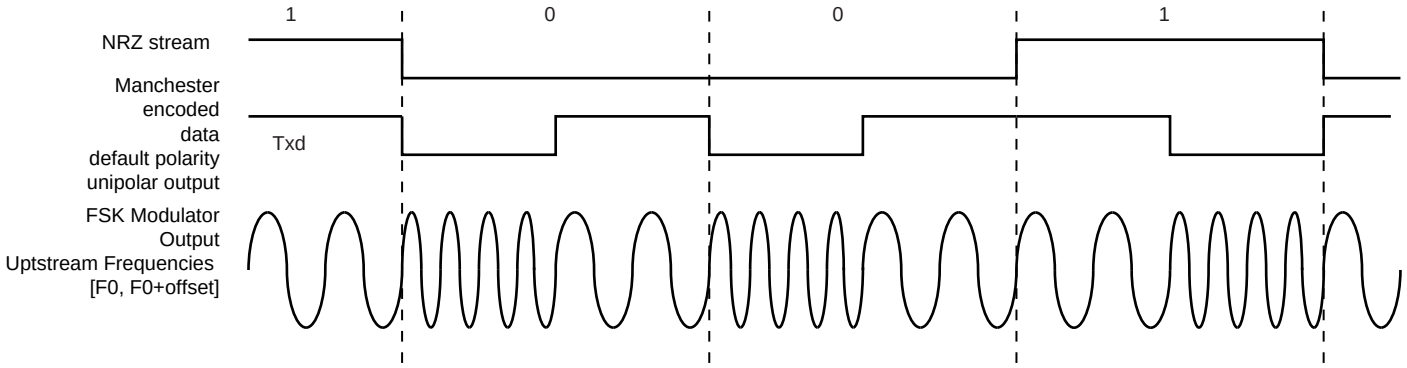


Figure 26-19. FSK Modulator Output



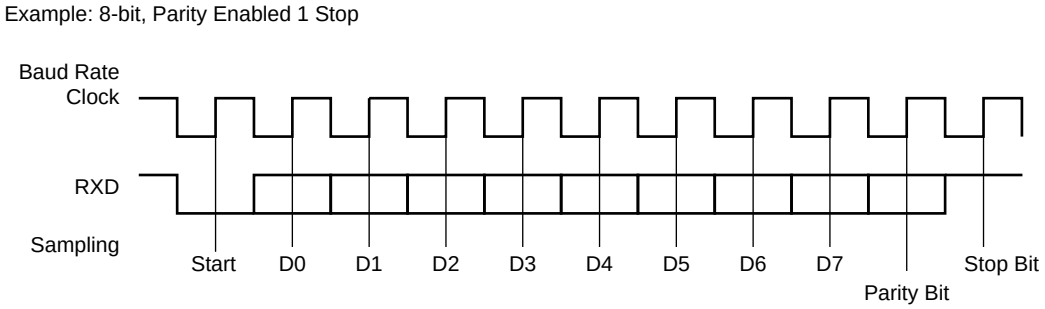
26.7.3.6 *Synchronous Receiver*

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 26-20 on page 549 illustrates a character reception in synchronous mode.

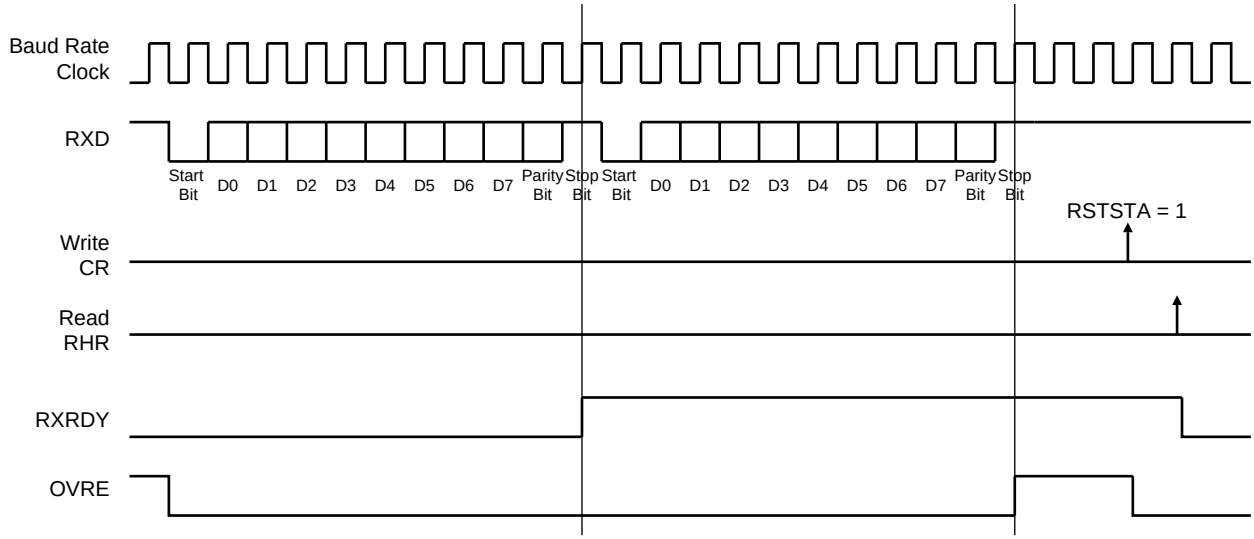
Figure 26-20. Synchronous Mode Character Reception



26.7.3.7 *Receiver Operations*

When a character reception is completed, it is transferred to the Receive Holding Register (RHR) and the RXRDY bit in the Status Register (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 RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

Figure 26-21. Receiver Status



26.7.3.8 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (MR). The PAR field also enables the Multidrop mode, see “Multidrop Mode” on page 552. 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.

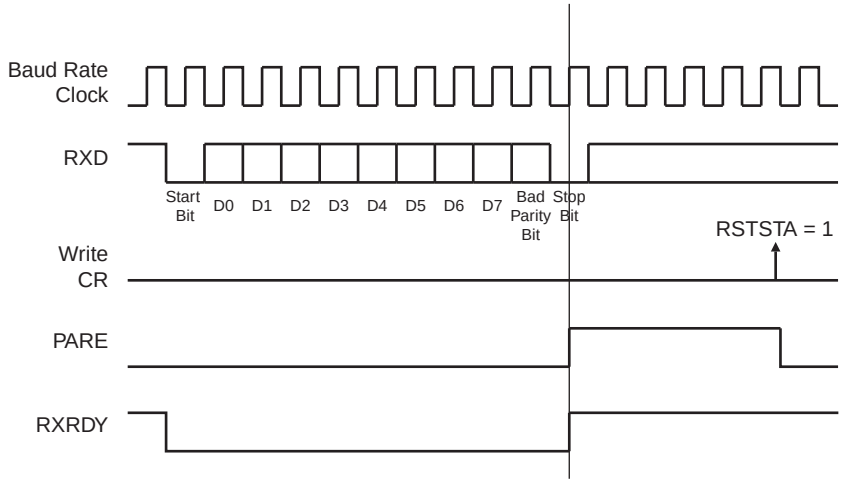
Table 26-7 on page 551 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 26-7. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (CSR). The PARE bit can be cleared by writing the Control Register (CR) with the RSTSTA bit at 1. Figure 26-22 on page 552 illustrates the parity bit status setting and clearing.

Figure 26-22. Parity Error



26.7.3.9 Multidrop Mode

If the PAR field in the Mode Register (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 CR. In this case, the next byte written to THR is transmitted as an address. Any character written in THR without having written the command SENDA is transmitted normally with the parity at 0.

26.7.3.10 Transmitter Timeguard

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (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 26-23 on page 553, 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 THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 26-23. Timeguard Operations

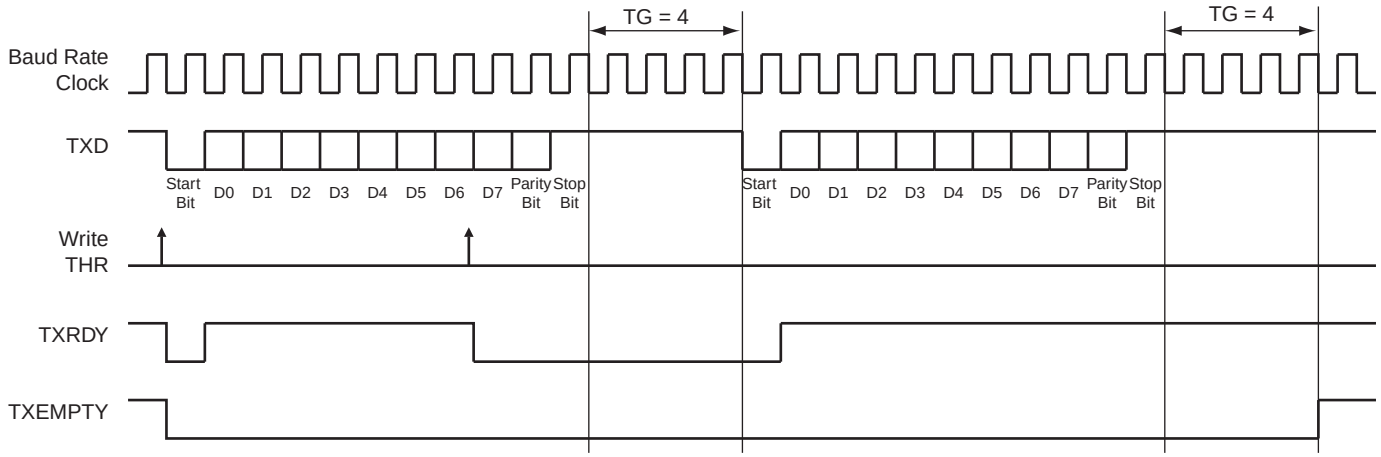


Table 26-8 on page 553 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 26-8. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

26.7.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (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 (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 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 (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 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 26-24 on page 554 shows the block diagram of the Receiver Time-out feature.

Figure 26-24. Receiver Time-out Block Diagram

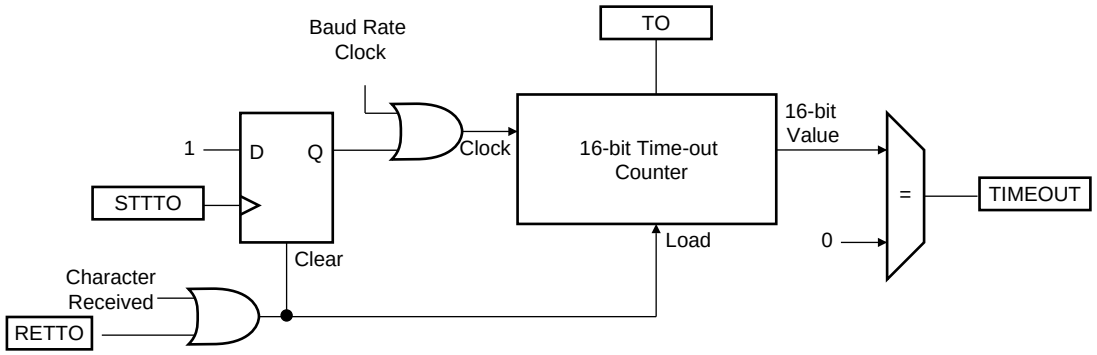


Table 26-9 on page 554 gives the maximum time-out period for some standard baud rates.

Table 26-9. Maximum Time-out Period

Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962

Table 26-9. Maximum Time-out Period (Continued)

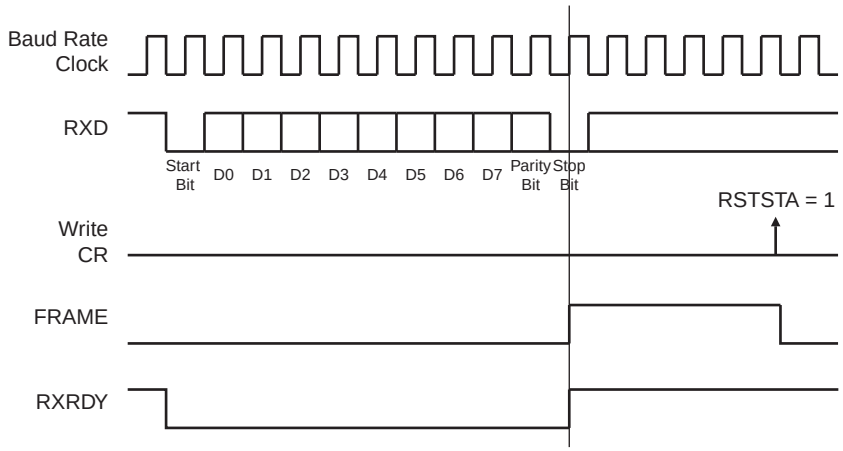
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

26.7.3.12 *Framing Error*

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (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 (CR) with the RSTSTA bit at 1.

Figure 26-25. Framing Error Status



26.7.3.13 *Transmit Break*

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits 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 (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 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 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 CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

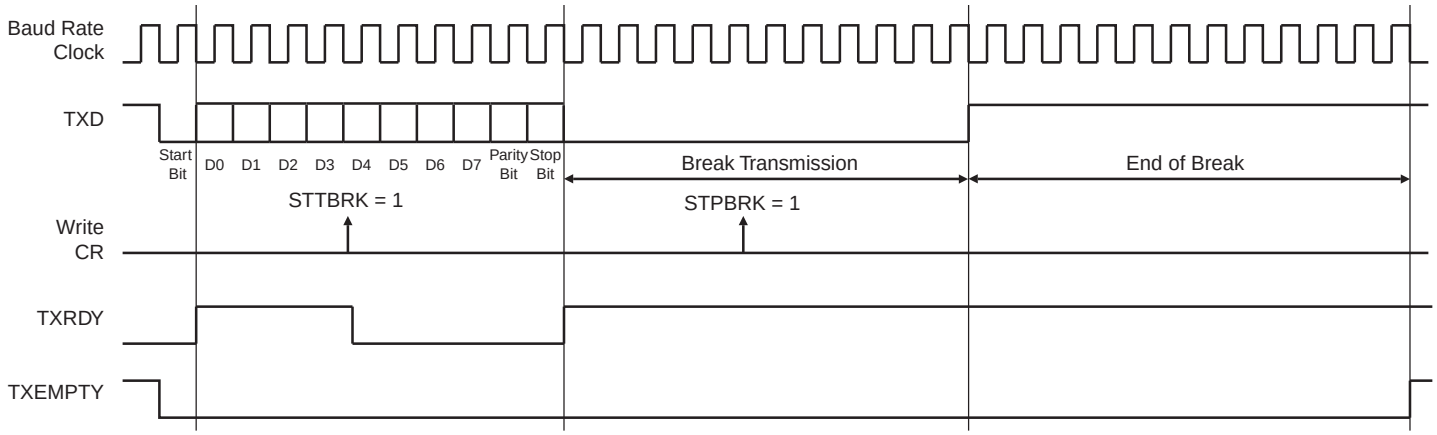
Writing CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 26-26 on page 556 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 26-26. Break Transmission



26.7.3.14 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

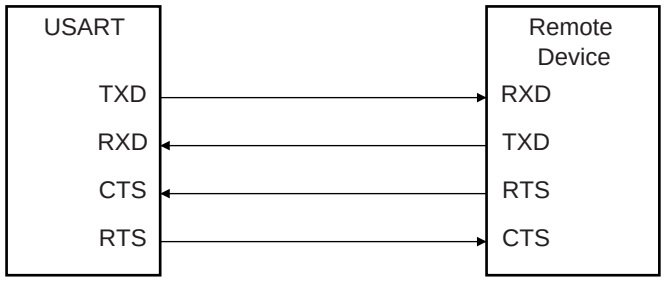
When the low stop bit is detected, the receiver asserts the RXBRK bit in CSR. This bit may be cleared by writing the Control Register (CR) with the bit RSTSTA at 1.

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.

26.7.3.15 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 26-27 on page 557.

Figure 26-27. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the MODE field in the Mode Register (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 PDCA channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 26-28 on page 557 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 PDCA 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 PDCA clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 26-28. Receiver Behavior when Operating with Hardware Handshaking

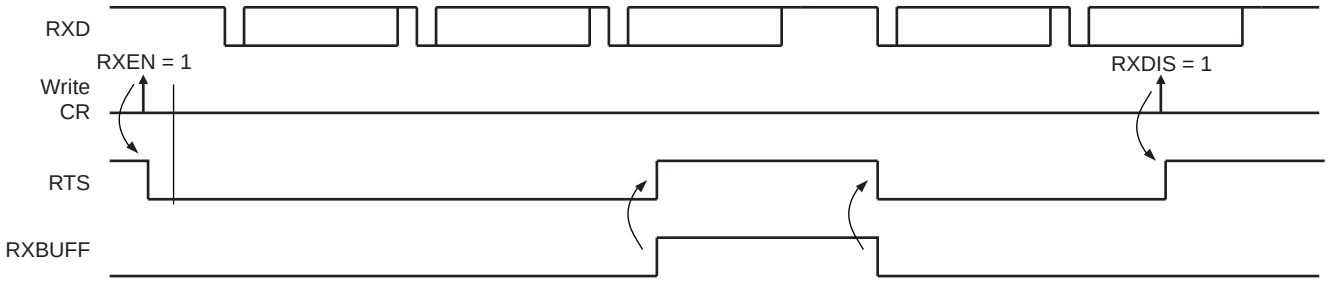
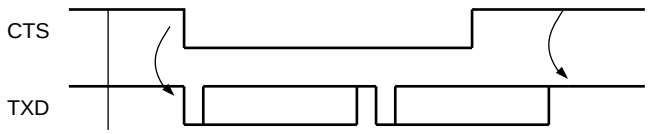


Figure 26-29 on page 557 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 26-29. Transmitter Behavior when Operating with Hardware Handshaking



26.7.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

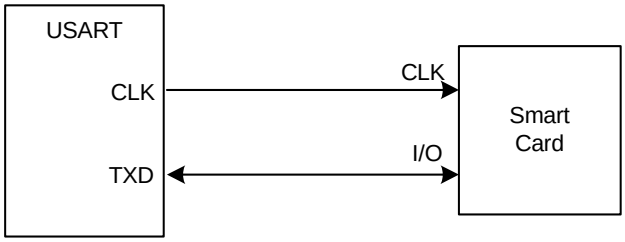
Setting the USART in ISO7816 mode is performed by writing the MODE field in the Mode Register (MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

26.7.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 536).

The USART connects to a smart card as shown in Figure 26-30 on page 558. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the CLK 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 26-30. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to "Mode Register" on page 594 and "PAR: Parity Type" on page 595.

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 (THR) or after reading it in the Receive Holding Register (RHR).

26.7.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 26-31 on page 559.

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 26-32 on page 559. 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 (RHR). It appropriately sets the PARE bit in the Status Register (SR) so that the software can handle the error.

Figure 26-31. T = 0 Protocol without Parity Error

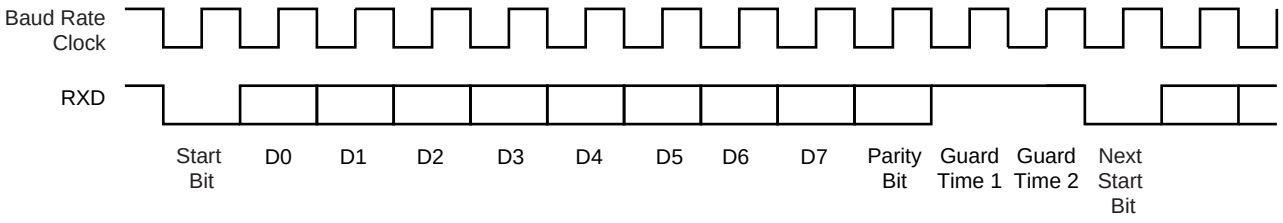
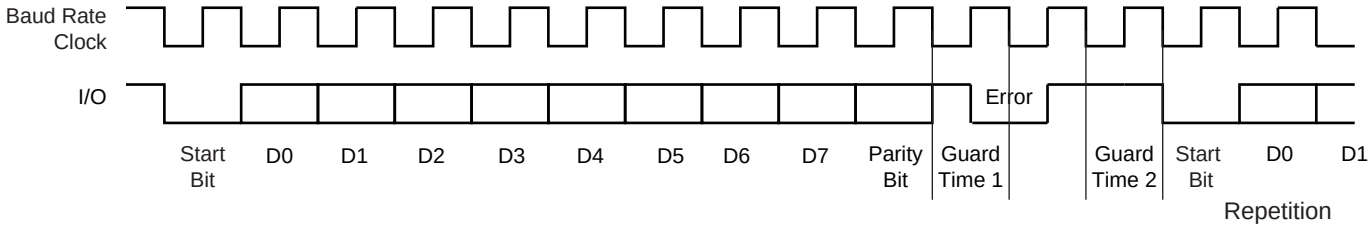


Figure 26-32. T = 0 Protocol with Parity Error



26.7.4.3 Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (NER) register. The NB_ERRORS field can record up to 255 errors. Reading NER automatically clears the NB_ERRORS field.

26.7.4.4 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 (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 (SR). The INACK bit can be cleared by writing the Control Register (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.

26.7.4.5 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 (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 (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 CSR can be cleared by writing the Control Register with the RSIT bit at 1.

26.7.4.6 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 (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.

26.7.4.7 Protocol T = 1

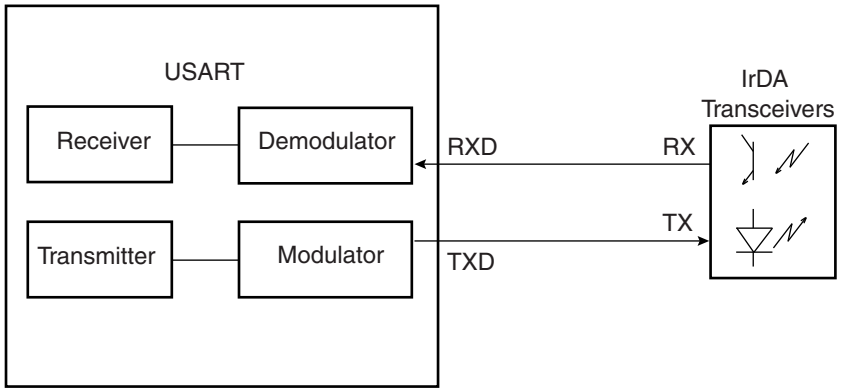
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (CSR).

26.7.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 26-33 on page 560. 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 MODE field in the Mode Register (MR) to the value 0x8. The IrDA Filter Register (IFR) 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 26-33. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

To receive IrDA signals, the following needs to be done:

- Disable TX and Enable RX
- Configure the TXD pin as PIO and set it as an output at 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).

- Receive data

26.7.5.1 IrDA Modulation

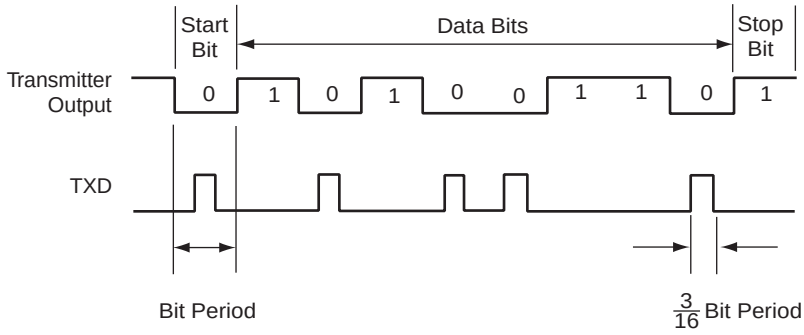
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 26-10 on page 561.

Table 26-10. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 26-34 on page 561 shows an example of character transmission.

Figure 26-34. IrDA Modulation



26.7.5.2 IrDA Baud Rate

Table 26-11 on page 561 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 26-11. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26

Table 26-11. IrDA Baud Rate Error (Continued)

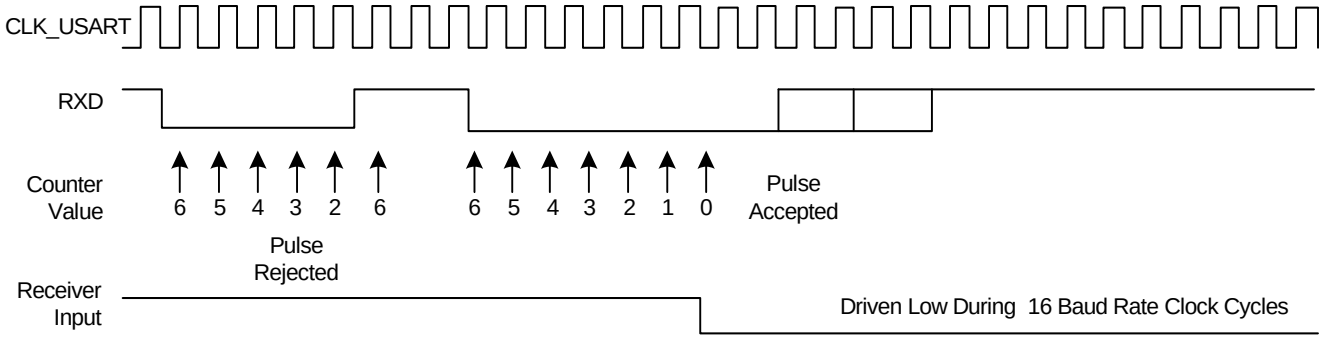
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

26.7.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in IFR. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the CLK_USART speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with IFR. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 26-35 on page 562 illustrates the operations of the IrDA demodulator.

Figure 26-35. IrDA Demodulator Operations

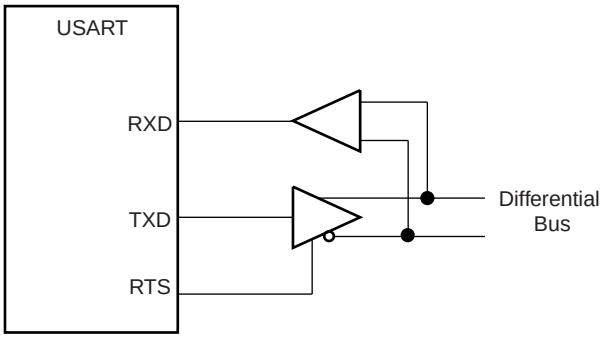


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

26.7.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 26-36 on page 563.

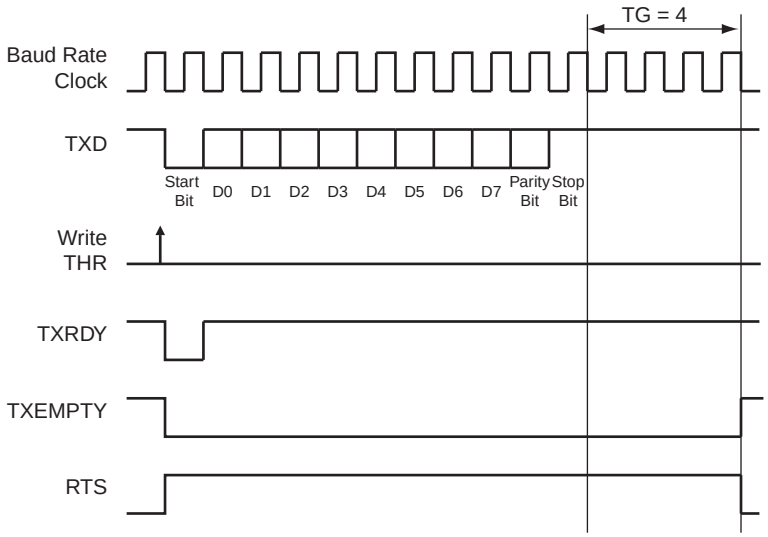
Figure 26-36. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the MODE field in the Mode Register (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 26-37 on page 563 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 26-37. Example of RTS Drive with Timeguard



26.7.7 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 MODE field in the Mode Register (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 26-12 on page 564 gives the correspondence of the USART signals with modem connection standards.

Table 26-12. 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 (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 (CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when 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.

26.7.8 SPI Mode

The Serial Peripheral Interface (SPI) Mode is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master” which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turns being masters and one master may simultaneously shift data into multiple slaves. (Multiple Master Protocol is the opposite of Single Master Protocol, where one CPU is always the master while all of the others are always slaves.) However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when its NSS signal is asserted by the master. The USART in SPI Master mode can address only one SPI Slave because it can generate only one NSS signal.

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input of the slave.
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master.
- Serial Clock (CLK): 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 CLK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows the master to select or deselect the slave.

26.7.8.1 Modes of Operation

The USART can operate in Master Mode or in Slave Mode.

Operation in SPI Master Mode is programmed by writing at 0xE the MODE field in the Mode Register. In this case the SPI lines must be connected as described below:

- the MOSI line is driven by the output pin TXD
- the MISO line drives the input pin RXD
- the CLK line is driven by the output pin CLK
- the NSS line is driven by the output pin RTS

Operation in SPI Slave Mode is programmed by writing at 0xF the MODE field in the Mode Register. In this case the SPI lines must be connected as described below:

- the MOSI line drives the input pin RXD
- the MISO line is driven by the output pin TXD
- the CLK line drives the input pin CLK
- the NSS line drives the input pin CTS

In order to avoid unpredicted behavior, any change of the SPI Mode must be followed by a software reset of the transmitter and of the receiver (except the initial configuration after a hardware reset). (See Section 26.7.9.2 on page 570).

26.7.8.2 Baud Rate

In SPI Mode, the baudrate generator operates in the same way as in USART synchronous mode: See “Baud Rate in Synchronous Mode or SPI Mode” on page 538. However, there are some restrictions:

In SPI Master Mode:

- the external clock CLK must not be selected (USCLKS … 0x3), and the bit CLKO must be set to “1” in the Mode Register (MR), in order to generate correctly the serial clock on the CLK pin.
- to obtain correct behavior of the receiver and the transmitter, the value programmed in CD of must be superior or equal to 4.
- if the internal clock divided (CLK_USART/DIV) is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the CLK pin, this value can be odd if the internal clock is selected (CLK_USART).

In SPI Slave Mode:

- the external clock (CLK) selection is forced regardless of the value of the USCLKS field in the Mode Register (MR). Likewise, the value written in BRGR has no effect, because the clock is provided directly by the signal on the USART CLK pin.
- to obtain correct behavior of the receiver and the transmitter, the external clock (CLK) frequency must be at least 4 times lower than the system clock.

26.7.8.3 Data Transfer

Up to 9 data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (MR). The 9 bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI Mode (Master or Slave).

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Mode Register. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 26-13. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 26-38. SPI Transfer Format (CPHA=1, 8 bits per transfer)

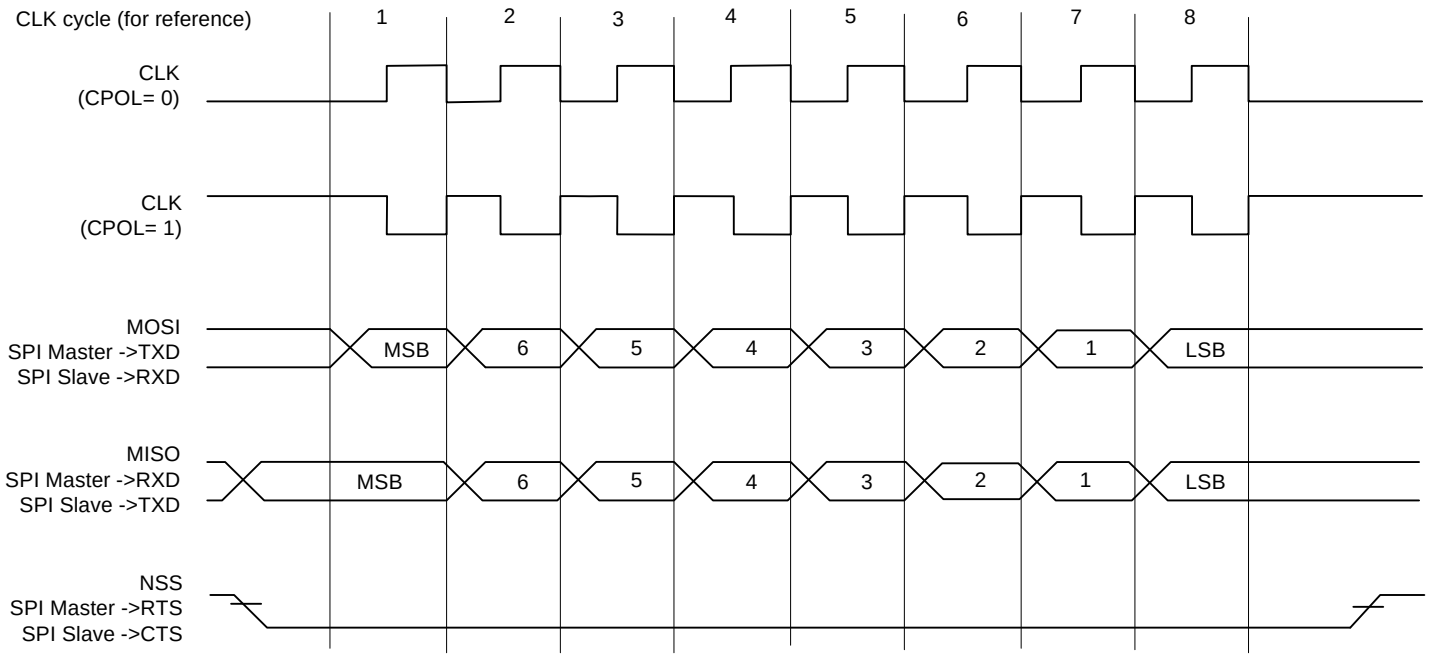
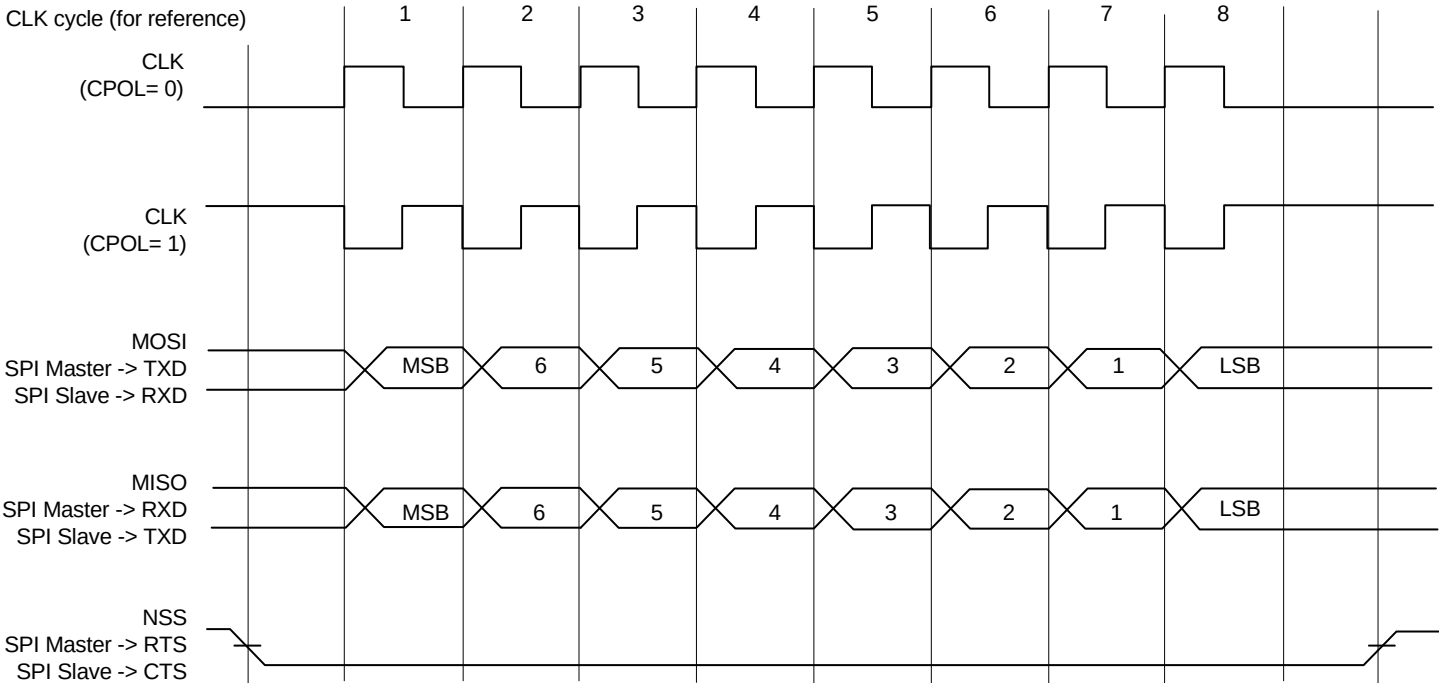


Figure 26-39. SPI Transfer Format (CPHA=0, 8 bits per transfer)



26.7.8.4 Receiver and Transmitter Control

See “Receiver and Transmitter Control” on page 540.

26.7.8.5 Character Transmission

The characters are sent by writing in the Transmit Holding Register (THR). The transmitter reports two status bits in the Channel Status Register (CSR): TXRDY (Transmitter Ready), which indicates that THR is empty and TXEMPTY, which indicates that all the characters written in THR have been processed. When the current character processing is completed, the last character written in THR is transferred into the Shift Register of the transmitter and THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in THR while TXRDY is low has no effect and the written character is lost.

If the USART is in SPI Slave Mode and if a character must be sent while the Transmit Holding Register (THR) is empty, the UNRE (Underrun Error) bit is set. The TXD transmission line stays at high level during all this time. The UNRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

In SPI Master Mode, the slave select line (NSS) is asserted at low level 1 Tbit before the transmission of the MSB bit and released at high level 1 Tbit after the transmission of the LSB bit. So, the slave select line (NSS) is always released between each character transmission and a minimum delay of 3 Tbits always inserted. However, in order to address slave devices supporting the CSAAT mode (Chip Select Active After Transfer), the slave select line (NSS) can be forced at low level by writing the Control Register (CR) with the RTSEN bit at 1. The slave select line (NSS) can be released at high level only by writing the Control Register (CR) with the RTSDIS bit at 1 (for example, when all data have been transferred to the slave device).

In SPI Slave Mode, the transmitter does not require a falling edge of the slave select line (NSS) to initiate a character transmission but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

26.7.8.6 Character Reception

When a character reception is completed, it is transferred to the Receive Holding Register (RHR) and the RXRDY bit in the Status Register (CSR) rises. If a character is completed while RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

To ensure correct behavior of the receiver in SPI Slave Mode, the master device sending the frame must ensure a minimum delay of 1 Tbit between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

26.7.8.7 Receiver Timeout

Because the receiver baudrate clock is active only during data transfers in SPI Mode, a receiver timeout is impossible in this mode, whatever the Time-out value is (field TO) in the Time-out Register (RTOR).

26.7.9 LIN Mode

The LIN Mode provides Master node and Slave node connectivity on a LIN bus.

The LIN (Local Interconnect Network) is a serial communication protocol which efficiently supports the control of mechatronic nodes in distributed automotive applications.

The main properties of the LIN bus are:

- Single Master/Multiple Slaves concept
- Low cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as a pure state machine.
- Self synchronization without quartz or ceramic resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20 kbit/s

LIN provides cost efficient bus communication where the bandwidth and versatility of CAN are not required.

The LIN Mode enables processing LIN frames with a minimum of action from the microprocessor.

26.7.9.1 Modes of operation

The USART can act either as a LIN Master node or as a LIN Slave node.

The node configuration is chosen by setting the MODE field in the USART3 Mode register (MR):

- LIN Master Node (MODE=0xA)
- LIN Slave Node (MODE=0xB)

In order to avoid unpredicted behavior, any change of the LIN node configuration must be followed by a software reset of the transmitter and of the receiver (except the initial node configuration after a hardware reset). (See Section 26.7.9.2 on page 570)

26.7.9.2 Receiver and Transmitter Control

See “Receiver and Transmitter Control” on page 540.

26.7.9.3 Character Transmission

See “Transmitter Operations” on page 541.

26.7.9.4 Character Reception

See “Receiver Operations” on page 549.

26.7.9.5 *Header Transmission (Master Node Configuration)*

All the LIN Frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

So in Master node configuration, the frame handling starts with the sending of the header.

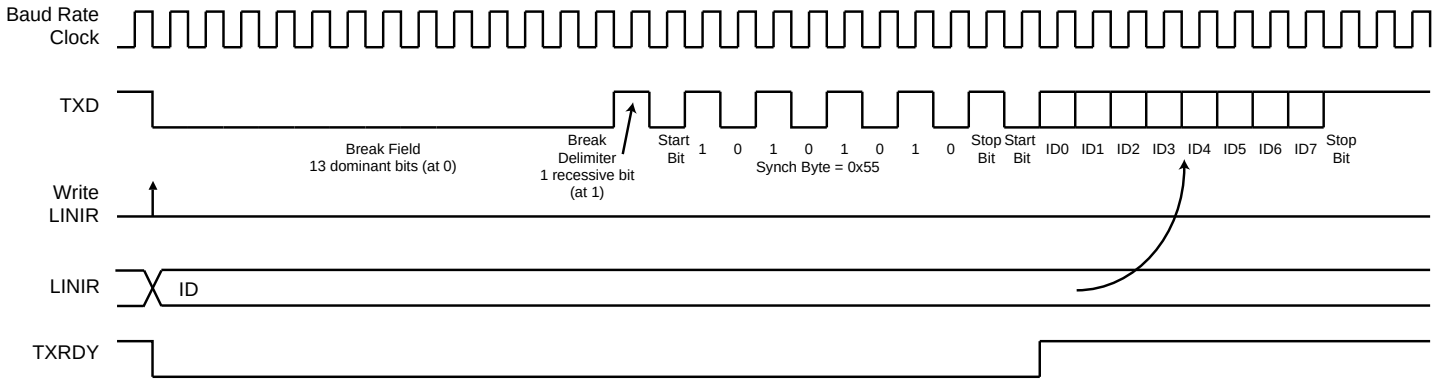
The header is transmitted as soon as the identifier is written in the LIN Identifier register (LINIR). At this moment the flag TXRDY falls.

The Break Field, the Synch Field and the Identifier Field are sent automatically one after the other.

The Break Field consists of 13 dominant bits and 1 recessive bit, the Synch Field is the character 0x55 and the Identifier corresponds to the character written in the LIN Identifier Register (LINIR). The Identifier parity bits can be automatically computed and sent (see Section 26.7.9.8 on page 574).

The flag TXRDY rises when the identifier character is transferred into the Shift Register of the transmitter.

Figure 26-40. Header Transmission



26.7.9.6 Header Reception (Slave Node Configuration)

All the LIN Frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

In Slave node configuration, the frame handling starts with the reception of the header.

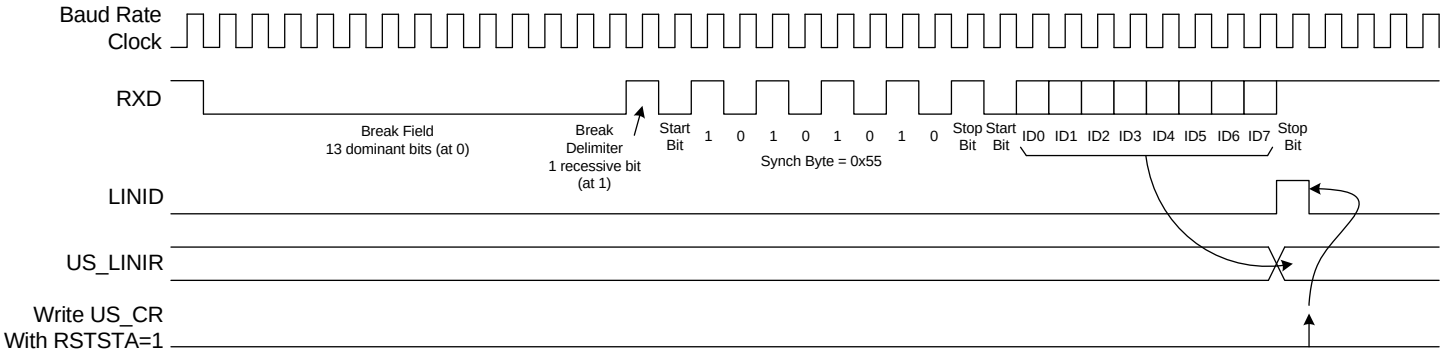
The USART uses a break detection threshold of 11 nominal bit times at the actual baud rate. At any time, if 11 consecutive recessive bits are detected on the bus, the USART detects a Break Field. As long as a Break Field has not been detected, the USART stays idle and the received data are not taken in account.

When a Break Field has been detected, the USART expects the Synch Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized (see Section 26.7.9.7 on page 572). If the received Synch character is not 0x55, an Inconsistent Synch Field error is generated (see Section 26.7.10 on page 579).

After receiving the Synch Field, the USART expects to receive the Identifier Field.

When the Identifier has been received, the flag LINID is set to “1”. At this moment the field IDCHR in the LIN Identifier register (LINIR) is updated with the received character. The Identifier parity bits can be automatically computed and checked (see Section 26.7.9.8 on page 574).

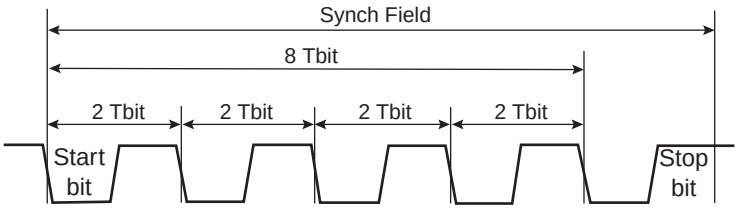
Figure 26-41. Header Reception



26.7.9.7 Slave Node Synchronization

The synchronization is done only in Slave node configuration. The procedure is based on time measurement between falling edges of the Synch Field. The falling edges are available in distances of 2, 4, 6 and 8 bit times.

Figure 26-42. Synch Field

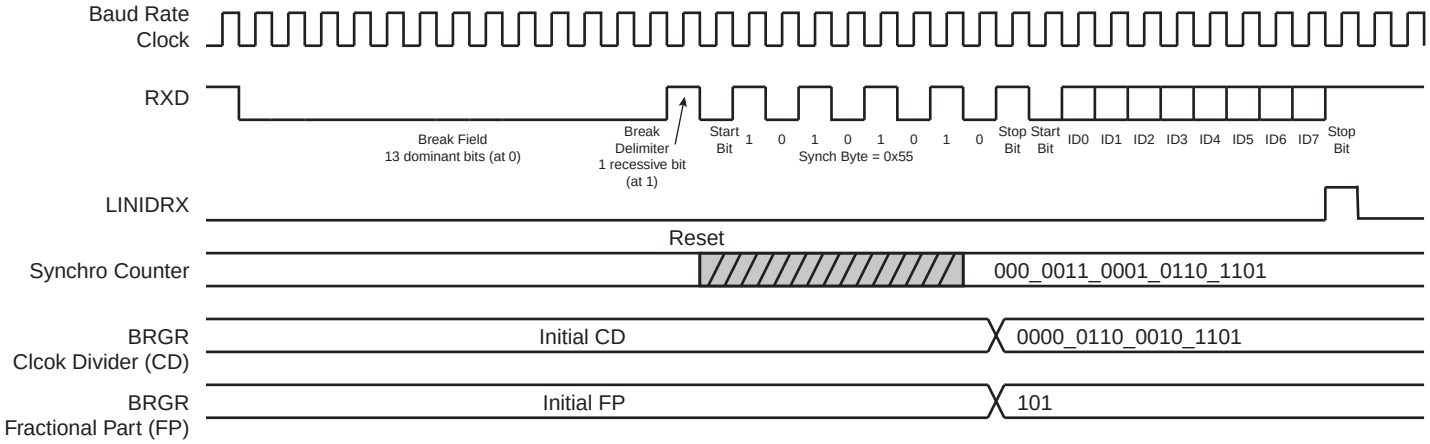


The time measurement is made by a 19-bit counter clocked by the sampling clock (see Section 26.7.1 on page 536).

When the start bit of the Synch Field is detected the counter is reset. Then during the next 8 Tbits of the Synch Field, the counter is incremented. At the end of these 8 Tbits, the counter is stopped. At this moment, the 16 most significant bits of the counter (value divided by 8) gives the new clock divider (CD) and the 3 least significant bits of this value (the remainder) gives the new fractional part (FP).

When the Synch Field has been received, the clock divider (CD) and the fractional part (FP) are updated in the Baud Rate Generator register (BRGR).

Figure 26-43. Slave Node Synchronization



The accuracy of the synchronization depends on several parameters:

- The nominal clock frequency (F_{Nom}) (the theoretical slave node clock frequency)
- The Baudrate
- The oversampling (Over=0 => 16X or Over=0 => 8X)

The following formula is used to compute the deviation of the slave bit rate relative to the master bit rate after synchronization (F_{SLAVE} is the real slave node clock frequency).

$$\text{Baudrate\_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times F_{SLAVE}}\right)\%$$

$$\text{Baudrate\_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times \left(\frac{F_{TOL\_UNSYNCH}}{100}\right) \text{x} F_{Nom}}\right)\%$$

$$-0{,}5 \le \alpha \le +0{,}5 \quad -1 < \beta < +1$$

$F_{TOL\_UNSYNCH}$ is the deviation of the real slave node clock from the nominal clock frequency. The LIN Standard imposes that it must not exceed ±15%. The LIN Standard imposes also that for communication between two nodes, their bit rate must not differ by more than ±2%. This means that the Baudrate_deviation must not exceed ±1%.

It follows from that, a minimum value for the nominal clock frequency:

$$F_{NOM}(min) = \left(100 \times \frac{[0,5 \times 8 \times (2 - Over) + 1] \times Baudrate}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%} \right) Hz$$

Examples:

- Baudrate = 20 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 2.64 MHz
- Baudrate = 20 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 1.47 MHz
- Baudrate = 1 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 132 kHz
- Baudrate = 1 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 74 kHz

If the fractional baud rate is not used, the accuracy of the synchronization becomes much lower. When the counter is stopped, the 16 most significant bits of the counter (value divided by 8) gives the new clock divider (CD). This value is rounded by adding the first insignificant bit. The equation of the Baudrate deviation is the same as given above, but the constants are as follows:

$$-4 \leq \alpha \leq +4 \qquad -1 < \beta < +1$$

It follows from that, a minimum value for the nominal clock frequency:

$$F_{NOM}(min) = \left(100 \times \frac{[4 \times 8 \times (2 - Over) + 1] \times Baudrate}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%} \right) Hz$$

Examples:

- Baudrate = 20 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 19.12 MHz
- Baudrate = 20 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 9.71 MHz
- Baudrate = 1 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 956 kHz
- Baudrate = 1 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 485 kHz

26.7.9.8 Identifier Parity

A protected identifier consists of two sub-fields; the identifier and the identifier parity. Bits 0 to 5 are assigned to the identifier and bits 6 and 7 are assigned to the parity.

The USART interface can generate/check these parity bits, but this feature can also be disabled. The user can choose between two modes by the PARDIS bit of the LIN Mode register (LINMR):

- PARDIS = 0:

During header transmission, the parity bits are computed and sent with the 6 least significant bits of the IDCHR field of the LIN Identifier register (LINIR). The bits 6 and 7 of this register are discarded.

During header reception, the parity bits of the identifier are checked. If the parity bits are wrong, an Identifier Parity error occurs (see Section 26.7.3.8 on page 551). Only the 6 least significant bits of the IDCHR field are updated with the received Identifier. The bits 6 and 7 are stuck at 0.

- PARDIS = 1:

During header transmission, all the bits of the IDCHR field of the LIN Identifier register (LINIR) are sent on the bus.

During header reception, all the bits of the IDCHR field are updated with the received Identifier.

26.7.9.9 Node Action

In function of the identifier, the node is concerned, or not, by the LIN response. Consequently, after sending or receiving the identifier, the USART must be configured. There are three possible configurations:

- PUBLISH: the node sends the response.
- SUBSCRIBE: the node receives the response.
- IGNORE: the node is not concerned by the response, it does not send and does not receive the response.

This configuration is made by the field, Node Action (NACT), in the LINMR register (see Section 26.8.16 on page 613).

Example: a LIN cluster that contains a Master and two Slaves:

- Data transfer from the Master to the Slave 1 and to the Slave 2:
  NACT(Master)=PUBLISH
  NACT(Slave1)=SUBSCRIBE
  NACT(Slave2)=SUBSCRIBE
- Data transfer from the Master to the Slave 1 only:
  NACT(Master)=PUBLISH
  NACT(Slave1)=SUBSCRIBE
  NACT(Slave2)=IGNORE
- Data transfer from the Slave 1 to the Master:
  NACT(Master)=SUBSCRIBE
  NACT(Slave1)=PUBLISH
  NACT(Slave2)=IGNORE
- Data transfer from the Slave1 to the Slave2:
  NACT(Master)=IGNORE
  NACT(Slave1)=PUBLISH
  NACT(Slave2)=SUBSCRIBE
- Data transfer from the Slave2 to the Master and to the Slave1:
  NACT(Master)=SUBSCRIBE
  NACT(Slave1)=SUBSCRIBE
  NACT(Slave2)=PUBLISH

26.7.9.10 Response Data Length

The LIN response data length is the number of data fields (bytes) of the response excluding the checksum.

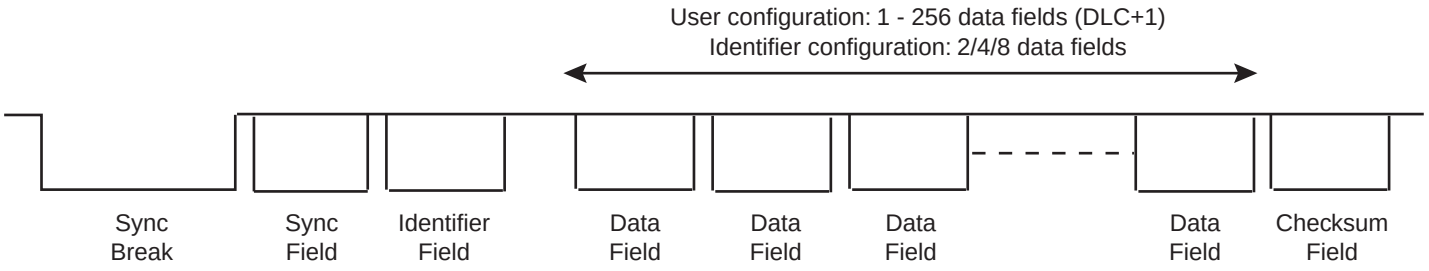
The response data length can either be configured by the user or be defined automatically by bits 4 and 5 of the Identifier (compatibility to LIN Specification 1.1). The user can choose between these two modes by the DLM bit of the LIN Mode register (LINMR):

- DLM = 0: the response data length is configured by the user via the DLC field of the LIN Mode register (LINMR). The response data length is equal to (DLC + 1) bytes. DLC can be programmed from 0 to 255, so the response can contain from 1 data byte up to 256 data bytes.
- DLM = 1: the response data length is defined by the Identifier (IDCHR in LINIR) according to the table below. The DLC field of the LIN Mode register (LINMR) is discarded. The response can contain 2 or 4 or 8 data bytes.

Table 26-14. Response Data Length if DLM = 1

IDCHR[5]	IDCHR[4]	Response Data Length [bytes]
0	0	2
0	1	2
1	0	4
1	1	8

Figure 26-44. Response Data Length



26.7.9.11 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8- bit sum with carry, over all data bytes or all data bytes and the protected identifier. Checksum calculation over the data bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and it is used for communication with LIN 2.0 slaves.

The USART can be configured to:

- Send/Check an Enhanced checksum automatically (CHKDIS = 0 & CHKTYP = 0)
- Send/Check a Classic checksum automatically (CHKDIS = 0 & CHKTYP = 1)
- Not send/check a checksum (CHKDIS = 1)

This configuration is made by the Checksum Type (CHKTYP) and Checksum Disable (CHKDIS) fields of the LIN Mode register (LINMR).

If the checksum feature is disabled, the user can send it manually all the same, by considering the checksum as a normal data byte and by adding 1 to the response data length (see Section 26.7.9.10 on page 576).

26.7.9.12 *Frame Slot Mode*

This mode is useful only for Master nodes. It respects the following rule: each frame slot shall be longer than or equal to TFrame_Maximum.

If the Frame Slot Mode is enabled (FSDIS = 0) and a frame transfer has been completed, the TXRDY flag is set again only after TFrame_Maximum delay, from the start of frame. So the Master node cannot send a new header if the frame slot duration of the previous frame is inferior to TFrame_Maximum.

If the Frame Slot Mode is disabled (FDIS = 1) and a frame transfer has been completed, the TXRDY flag is set again immediately.

The TFrame_Maximum is calculated as below:

If the Checksum is sent (CHKDIS = 0):

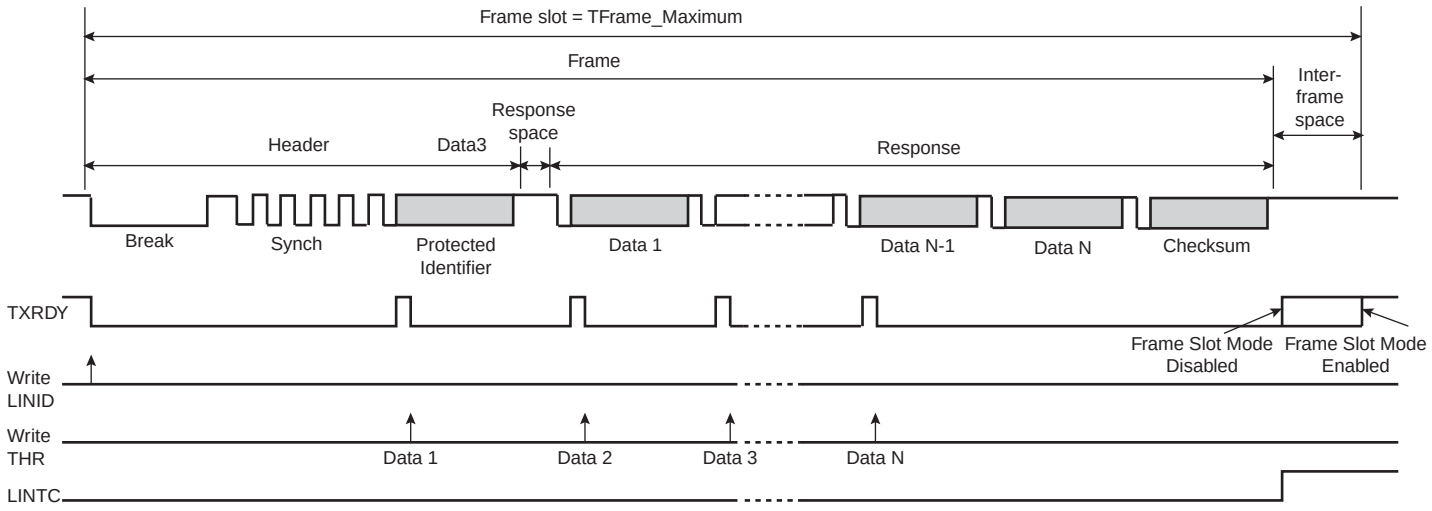
- THeader_Nominal = 34 x TBit
- TResponse_Nominal = 10 x (NData + 1) x TBit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1)(Note:)
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1 + 1) + 1) x TBIT
- TFrame_Maximum = (77 + 14 x DLC) x TBIT

If the Checksum is not sent (CHKDIS = 1):

- THeader_Nominal = 34 x TBit
- TResponse_Nominal = 10 x NData x TBit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1(Note:))
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1) + 1) x TBIT
- TFrame_Maximum = (63 + 14 x DLC) x TBIT

Note: The term “+1” leads to an integer result for TFrame_Max (LIN Specification 1.3)

Figure 26-45. Frame Slot Mode



26.7.10 LIN Errors

26.7.10.1 Bit Error

This error is generated when the USART is transmitting and if the transmitted value on the Tx line is different from the value sampled on the Rx line.

If a bit error is detected, the transmission is aborted at the next byte border.

26.7.10.2 Inconsistent Synch Field Error

This error is generated in Slave node configuration if the Synch Field character received is other than 0x55.

26.7.10.3 Parity Error

This error is generated if the parity of the identifier is wrong. This error can be generated only if the parity feature is enabled (PARDIS = 0).

26.7.10.4 Checksum Error

This error is set if the received checksum is wrong. This error can be generated only if the checksum feature is enabled (CHKDIS = 0).

26.7.10.5 Slave Not Responding Error

This error is set when the USART expects a response from another node (NACT = SUBSCRIBE) but no valid message appears on the bus within the time frame given by the maximum length of the message frame, TFrame_Maximum (see Section 26.7.9.12 on page 578). This error is disabled if the USART does not expect any message (NACT = PUBLISH or NACT = IGNORE).

26.7.11 LIN Frame Handling

26.7.11.1 Master Node Configuration

- Write TXEN and RXEN in CR to enable both the transmitter and the receiver.
- Write MODE in MR to select the LIN mode and the Master Node configuration.
- Write CD and FP in BRGR to configure the baud rate.
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM, FDIS and DLC in LINMR to configure the frame transfer.
- Check that TXRDY in CSR is set to “1”
- Write IDCHR in LINIR to send the header

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in CSR rises
  - Write TCHR in THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in CSR rises
  - Read RCHR in RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in CSR rises
  - Check the LIN errors

Figure 26-46. Master Node Configuration, NACT = PUBLISH

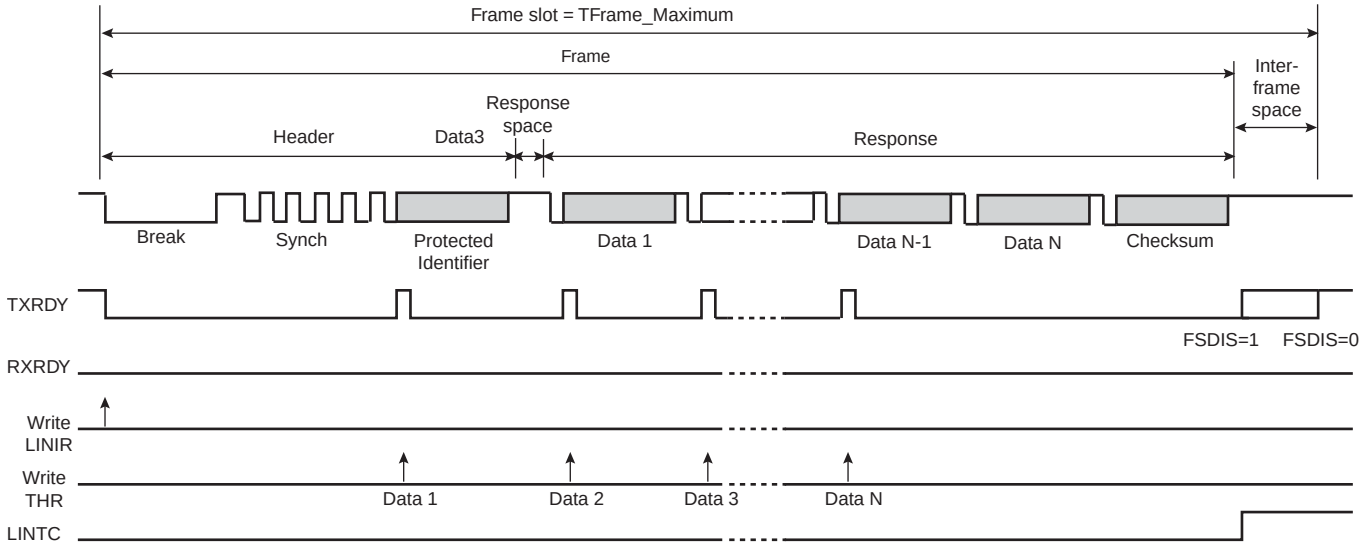


Figure 26-47. Master Node Configuration, NACT=SUBSCRIBE

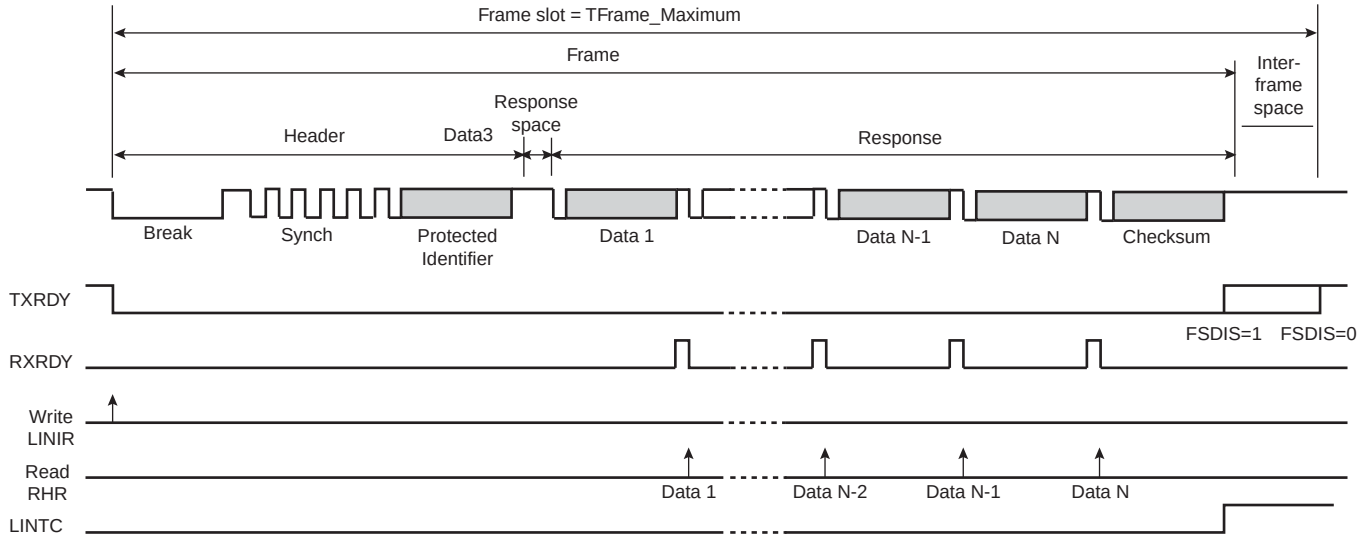
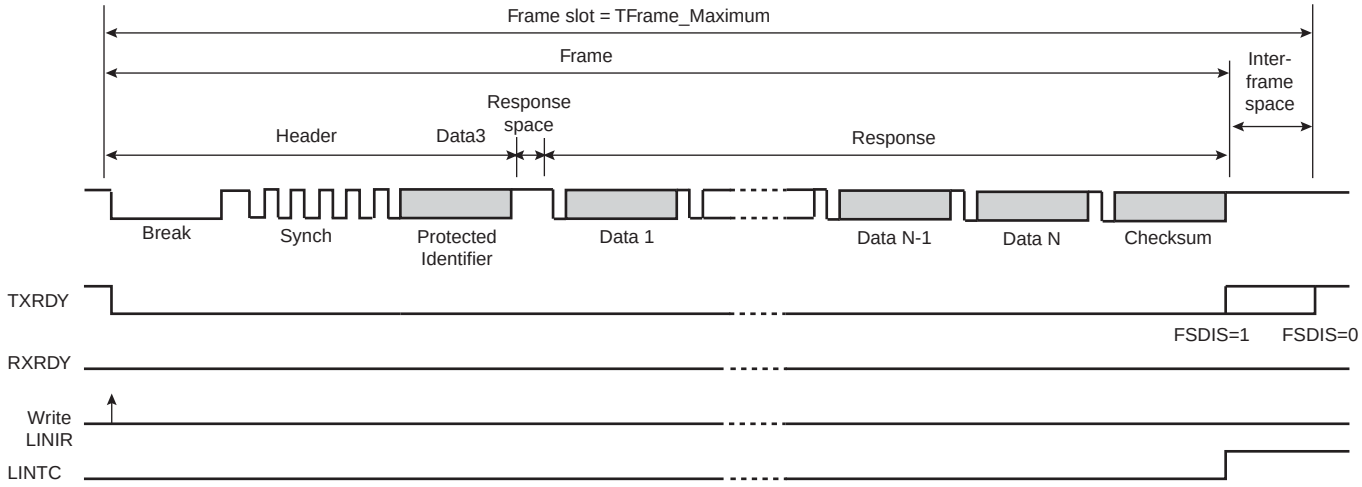


Figure 26-48. Master Node Configuration, NACT=IGNORE



26.7.11.2 Slave Node Configuration

- Write TXEN and RXEN in CR to enable both the transmitter and the receiver.
- Write MODE in MR to select the LIN mode and the Slave Node configuration.
- Write CD and FP in BRGR to configure the baud rate.
- Wait until LINID in CSR rises
- Check LINISFE and LINPE errors
- Read IDCHR in RHR
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM and DLC in LINMR to configure the frame transfer.

IMPORTANT: if the NACT configuration for this frame is PUBLISH, the US_LINMR register, must be write with NACT=PUBLISH even if this field is already correctly configured, that in order to set the TXREADY flag and the corresponding PDC write transfer request.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in CSR rises
  - Write TCHR in THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in CSR rises
  - Read RCHR in RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors

- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in CSR rises
  - Check the LIN errors

Figure 26-49. Slave Node Configuration, NACT = PUBLISH

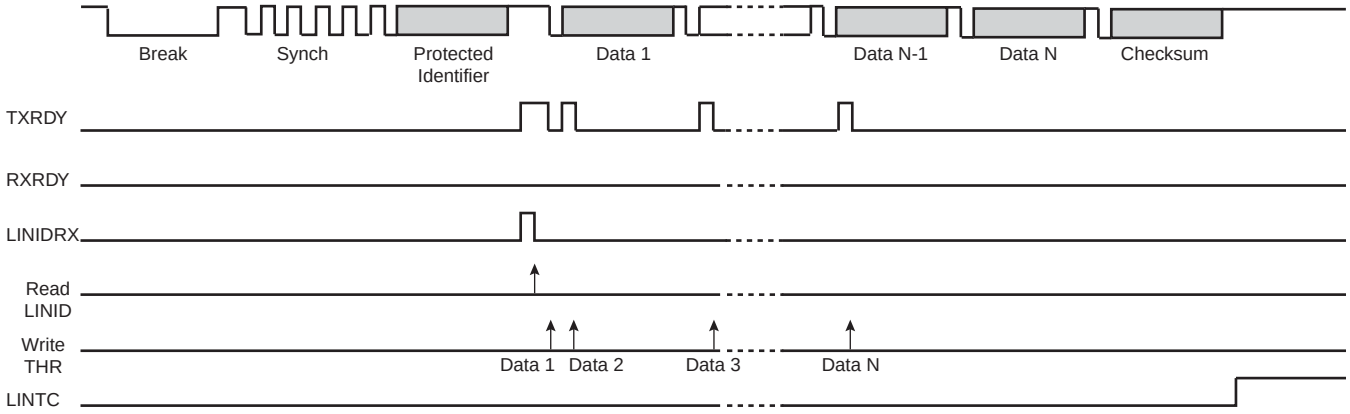


Figure 26-50. Slave Node Configuration, NACT = SUBSCRIBE

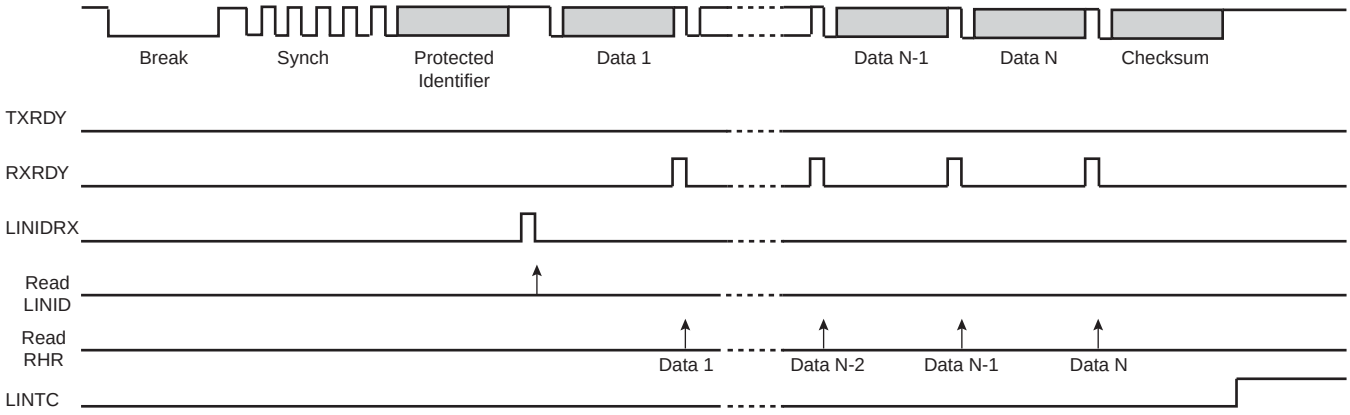
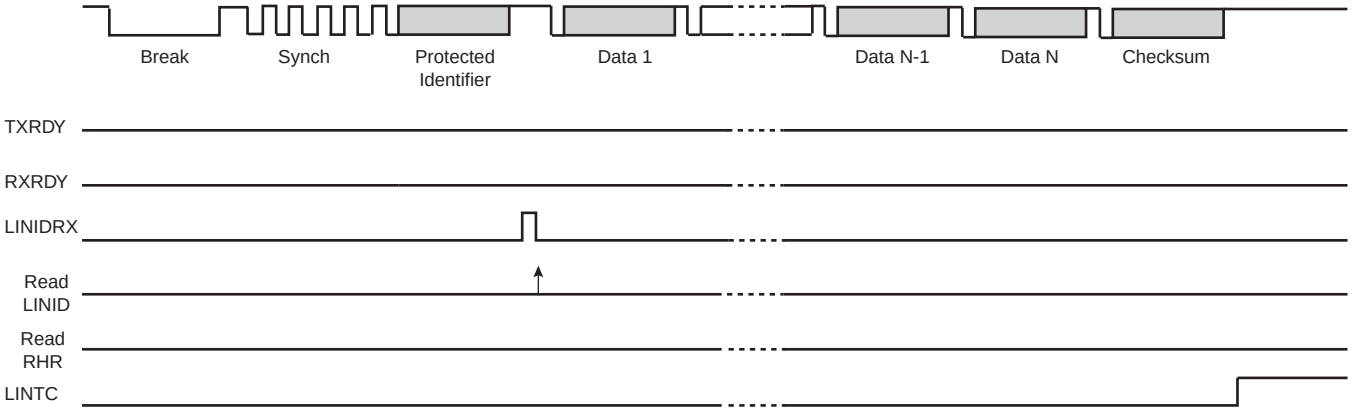


Figure 26-51. Slave Node Configuration, NACT = IGNORE



26.7.12 LIN Frame Handling With The Peripheral DMA Controller

The USART can be used in association with the Peripheral DMA Controller (PDCA) in order to transfer data directly into/from the on- and off-chip memories without any processor intervention.

The PDCA uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The PDCA always writes in the Transmit Holding register (THR) and it always reads in the Receive Holding register (RHR). The size of the data written or read by the PDCA in the USART is always a byte.

26.7.12.1 Master Node Configuration

The user can choose between two PDCA modes by the PDCM bit in the LIN Mode register (LINMR):

- PDCM = 1: the LIN configuration is stored in the WRITE buffer and it is written by the PDCA in the Transmit Holding register THR (instead of the LIN Mode register LINMR). Because the PDCA transfer size is limited to a byte, the transfer is split into two accesses. During the first access the bits, NACT, PARDIS, CHKDIS, CHKTYP, DLM and FDIS are written. During the second access the 8-bit DLC field is written.
- PDCM = 0: the LIN configuration is not stored in the WRITE buffer and it must be written by the user in the LIN Mode register (LINMR).

The WRITE buffer also contains the Identifier and the DATA, if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Figure 26-52. Master Node with PDCA (PDCM=1)

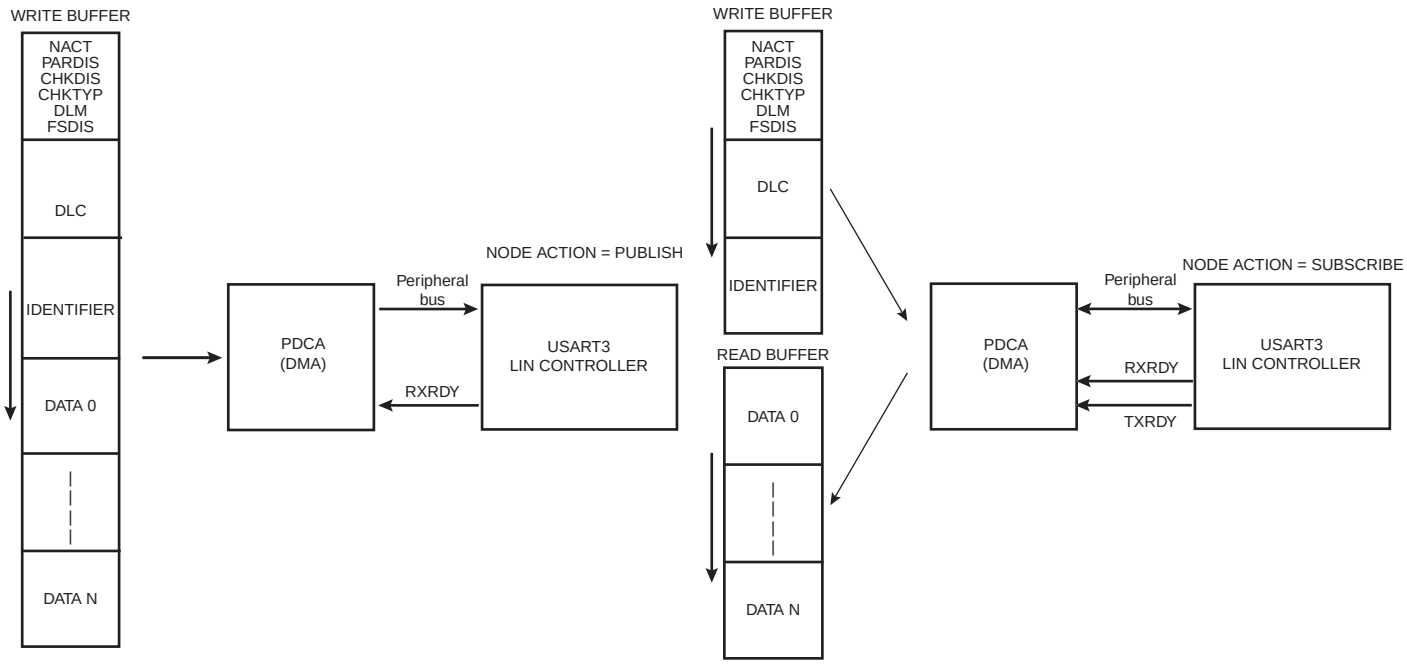
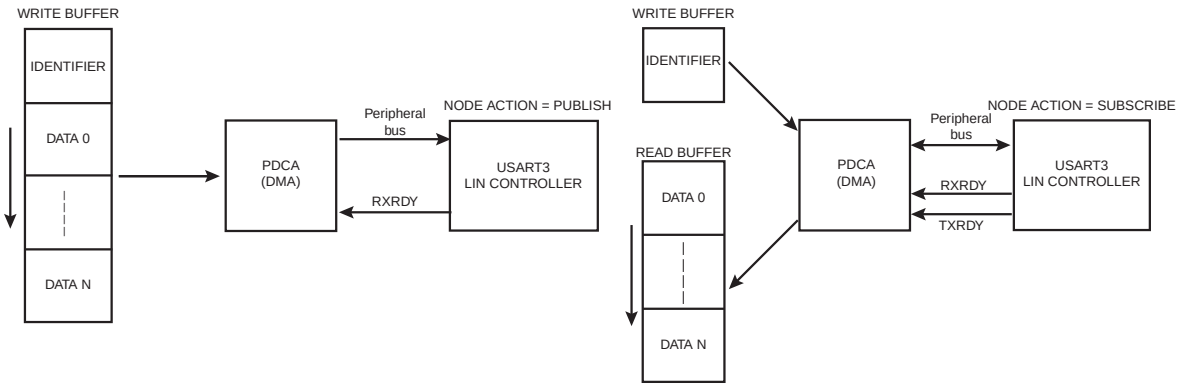


Figure 26-53. Master Node with PDCA (PDCM=0)



26.7.12.2 Slave Node Configuration

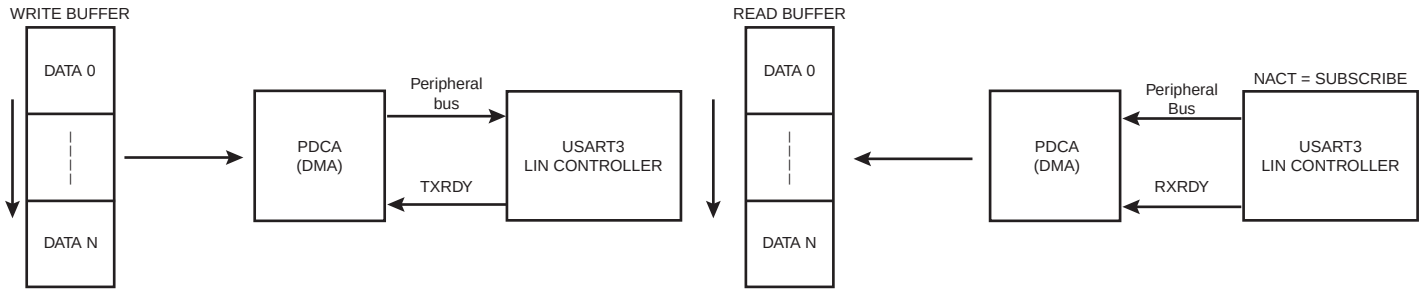
In this configuration, the PDCA transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (LINIR). The LIN mode must be written by the user in the LIN Mode register (LINMR).

The WRITE buffer contains the DATA if the USART sends the response (NACT=PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT=SUBSCRIBE).

IMPORTANT: if the NACT configuration for a frame is PUBLISH, the US_LINMR register, must be write with NACT=PUBLISH even if this field is already correctly configured, that in order to set the TXREADY flag and the corresponding PDC write transfer request.

Figure 26-54. Slave Node with PDCA



26.7.13 Wake-up Request

Any node in a sleeping LIN cluster may request a wake-up.

In the LIN 2.0 specification, the wakeup request is issued by forcing the bus to the dominant state from 250 µs to 5 ms. For this, it is necessary to send the character 0xF0 in order to impose 5 successive dominant bits. Whatever the baud rate is, this character respects the specified timings.

- Baud rate min = 1 kbit/s -> Tbit = 1ms -> 5 Tbits = 5 ms
- Baud rate max = 20 kbit/s -> Tbi t= 50 µs -> 5 Tbits = 250 µs

In the LIN 1.3 specification, the wakeup request should be generated with the character 0x80 in order to impose 8 successive dominant bits.

The user can choose by the WKUPTYP bit in the LIN Mode register (LINMR) either to send a LIN 2.0 wakeup request (WKUPTYP=0) or to send a LIN 1.3 wakeup request (WKUPTYP=1).

A wake-up request is transmitted by writing the Control Register (CR) with the LINWKUP bit at 1. Once the transfer is completed, the LINTC flag is asserted in the Status Register (SR). It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1.

26.7.14 Bus Idle Time-out

If the LIN bus is inactive for a certain duration, the slave nodes shall automatically enter in sleep mode. In the LIN 2.0 specification, this time-out is fixed at 4 seconds. In the LIN 1.3 specification, it is fixed at 25000 Tbits.

In Slave Node configuration, the Receiver Time-out detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (CSR) rises and can generate an interrupt, thus indicating to the driver to go into sleep mode.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (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 CSR remains at 0. Otherwise, the receiver loads a 17-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises.

If STTTO is performed, the counter clock is stopped until a first character is received.

If RETTO is performed, the counter starts counting down immediately from the value TO.

Table 26-15. Receiver Time-out programming

LIN Specification	Baud Rate	Time-out period	TO
2.0	1 000 bit/s	4s	4 000
	2 400 bit/s		9 600
	9 600 bit/s		38 400
	19 200 bit/s		76 800
	20 000 bit/s		80 000
1.3	-	25 000 Tbits	25 000

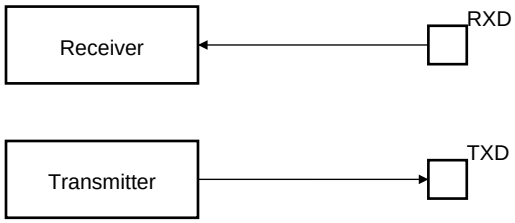
26.7.15 Test Modes

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.

26.7.15.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

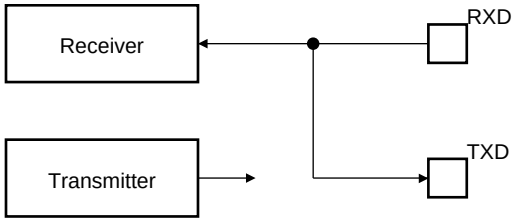
Figure 26-55. Normal Mode Configuration



26.7.15.2 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 26-56 on page 588. 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.

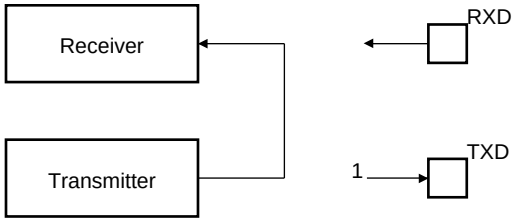
Figure 26-56. Automatic Echo Mode Configuration



26.7.15.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 26-57 on page 588. 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.

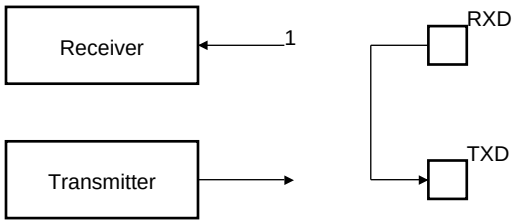
Figure 26-57. Local Loopback Mode Configuration



26.7.15.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 26-58 on page 589. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 26-58. Remote Loopback Mode Configuration



26.7.16 Write Protection Registers

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

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

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

The protected registers are:

- “Mode Register” on page 594
- “Baud Rate Generator Register” on page 605
- “Receiver Time-out Register” on page 606
- “Transmitter Timeguard Register” on page 607
- “FI DI RATIO Register” on page 608
- “IrDA FILTER Register” on page 610
- “Manchester Configuration Register” on page 611

26.8 User Interface

Table 26-16. USART Register Memory Map

Offset	Register	Name	Access	Reset
0x0000	Control Register	CR	Write-only	–
0x0004	Mode Register	MR	Read-write	0x00000000
0x0008	Interrupt Enable Register	IER	Write-only	–
0x000C	Interrupt Disable Register	IDR	Write-only	–
0x0010	Interrupt Mask Register	IMR	Read-only	0x00000000
0x0014	Channel Status Register	CSR	Read-only	0x00000000
0x0018	Receiver Holding Register	RHR	Read-only	0x00000000
0x001C	Transmitter Holding Register	THR	Write-only	–
0x0020	Baud Rate Generator Register	BRGR	Read-write	0x00000000
0x0024	Receiver Time-out Register	RTOR	Read-write	0x00000000
0x0028	Transmitter Timeguard Register	TTGR	Read-write	0x00000000
0x0040	FI DI Ratio Register	FIDI	Read-write	0x00000174
0x0044	Number of Errors Register	NER	Read-only	0x00000000
0x004C	IrDA Filter Register	IFR	Read-write	0x00000000
0x0050	Manchester Encoder Decoder Register	MAN	Read-write	0x30011004
0x0054	LIN Mode Register	LINMR	Read-write	0x00000000
0x0058	LIN Identifier Register	LINIR	Read-write(Note:)	0x00000000
0xE4	Write Protect Mode Register	WPMR	Read-write	0x00000000
0xE8	Write Protect Status Register	WPSR	Read-only	0x00000000
0xFC	Version Register	VERSION	Read-only	0x–(Note:)

Note: Write is possible only in LIN Master node configuration.

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

26.8.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x0

Reset Value: -

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

23	22	21	20	19	18	17	16
–	–	LINWKUP	LINABT	RTSDIS/RCS	RTSEN/FCS	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	–	–

- **LINWKUP: Send LIN Wakeup Signal**
  0: No effect:
  1: Sends a wakeup signal on the LIN bus.
- **LINABT: Abort LIN Transmission**
  0: No effect.
  1: Abort the current LIN transmission.
- **RTSDIS/RCS: Request to Send Disable/Release SPI Chip Select**
  If USART does not operate in SPI Master Mode (MODE … 0xE):
  0: No effect.
  1: Drives the pin RTS to 1.
  If USART operates in SPI Master Mode (MODE = 0xE):
  RCS = 0: No effect.
  RCS = 1: Releases the Slave Select Line NSS (RTS pin).
- **RTSEN/FCS: Request to Send Enable/Force SPI Chip Select**
  If USART does not operate in SPI Master Mode (MODE … 0xE):
  0: No effect.
  1: Drives the pin RTS to 0.
  If USART operates in SPI Master Mode (MODE = 0xE):
  FCS = 0: No effect.
  FCS = 1: Forces the Slave Select Line NSS (RTS pin) to 0, even if USART is no transmitting, in order to address SPI slave devices supporting the CSAAT Mode (Chip Select Active After Transfer).
- **DTRDIS: Data Terminal Ready Disable**
  0: No effect.
  1: Drives the pin DTR to 1.
- **DTREN: Data Terminal Ready Enable**
  0: No effect.
  1: Drives the pin DTR at 0.
- **RETTO: Rearm Time-out**
  0: No effect
  1: Restart Time-out

- **RSTNACK: Reset Non Acknowledge**
  - 0: No effect
  - 1: Resets NACK in CSR.
- **RSTIT: Reset Iterations**
  - 0: No effect.
  - 1: Resets ITERATION in CSR. No effect if the ISO7816 is not enabled.
- **SENDA: Send Address**
  - 0: No effect.
  - 1: In Multidrop Mode only, the next character written to the THR is sent with the address bit set.
- **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 CSR.
- **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.
- **STTBRK: Start Break**
  - 0: No effect.
  - 1: Starts transmission of a break after the characters present in THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.
- **RSTSTA: Reset Status Bits**
  - 0: No effect.
  - 1: Resets the status bits PARE, FRAME, OVRE, MANERR, LINBE, LINSFE, LINIPE, LINCE, LINSNRE and RXBRK in CSR.
- **TXDIS: Transmitter Disable**
  - 0: No effect.
  - 1: Disables the transmitter.
- **TXEN: Transmitter Enable**
  - 0: No effect.
  - 1: Enables the transmitter if TXDIS is 0.
- **RXDIS: Receiver Disable**
  - 0: No effect.
  - 1: Disables the receiver.
- **RXEN: Receiver Enable**
  - 0: No effect.
  - 1: Enables the receiver, if RXDIS is 0.
- **RSTTX: Reset Transmitter**
  - 0: No effect.
  - 1: Resets the transmitter.
- **RSTRX: Reset Receiver**
  - 0: No effect.
  - 1: Resets the receiver.

26.8.2 Mode Register

Name: MR

Access Type: Read-write

Offset: 0x4

Reset Value: -

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC	MAN	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF/CPOL

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC/CPHA

7	6	5	4	3	2	1	0
CHRL		USCLKS		MODE			

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **ONEBIT: Start Frame Delimiter Selector**
  0: Start Frame delimiter is COMMAND or DATA SYNC.
  1: Start Frame delimiter is One Bit.
- **MODSYNC: Manchester Synchronization Mode**
  0:The Manchester Start bit is a 0 to 1 transition
  1: The Manchester Start bit is a 1 to 0 transition.
- **MAN: Manchester Encoder/Decoder Enable**
  0: Manchester Encoder/Decoder are disabled.
  1: Manchester Encoder/Decoder are enabled.
- **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).
- **MAX_ITERATION**
  Defines the maximum number of iterations in mode ISO7816, protocol T= 0.
- **VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter**
  0: User defined configuration of command or data sync field depending on SYNC value.
  1: The sync field is updated when a character is written into THR register.
- **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.
- **INACK: Inhibit Non Acknowledge**
  0: The NACK is generated.
  1: The NACK is not generated.
- **OVER: Oversampling Mode**
  0: 16x Oversampling.
  1: 8x Oversampling.

- **CLKO: Clock Output Select**
  0: The USART does not drive the CLK pin.
  1: The USART drives the CLK pin if USCLKS does not select the external clock CLK.
- **MODE9: 9-bit Character Length**
  0: CHRL defines character length.
  1: 9-bit character length.
- **MSBF/CPOL: Bit Order or SPI Clock Polarity**
  If USART does not operate in SPI Mode (MODE … 0xE and 0xF):
  MSBF = 0: Least Significant Bit is sent/received first.
  MSBF = 1: Most Significant Bit is sent/received first.
  If USART operates in SPI Mode (Slave or Master, MODE = 0xE or 0xF):
  CPOL = 0: The inactive state value of SPCK is logic level zero.
  CPOL = 1: The inactive state value of SPCK is logic level one.
  CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with CPHA to produce the required clock/data relationship between master and slave devices.
- **CHMODE: Channel Mode**

Table 26-17.

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.

- **NBSTOP: Number of Stop Bits**

Table 26-18.

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

- **PAR: Parity Type**

Table 26-19.

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

- **SYNC/CPHA: Synchronous Mode Select or SPI Clock Phase**

  If USART does not operate in SPI Mode (MODE is … 0xE and 0xF):
  SYNC = 0: USART operates in Asynchronous Mode.
  SYNC = 1: USART operates in Synchronous Mode.
  If USART operates in SPI Mode (MODE = 0xE or 0xF):
  CPHA = 0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.
  CPHA = 1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  CPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.
- **CHRL: Character Length.**

Table 26-20.

CHRL		Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **USCLKS: Clock Selection**

Table 26-21.

USCLKS		Selected Clock
0	0	CLK_USART
0	1	CLK_USART/DIV (DIV = xx)
1	0	Reserved
1	1	CLK

- **MODE**

Table 26-22.

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	1	0	IS07816 Protocol: T = 1
1	0	0	0	IrDA
1	0	1	0	LIN Master
1	0	1	1	LIN Slave
1	1	1	0	SPI Master
1	1	1	1	SPI Slave
Others				Reserved

26.8.3 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x8

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANEA
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINiD	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

For backward compatibility the MANE bit has been duplicated to the MANEA bit position. Writing either one or the other has the same effect.

26.8.4 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0xC

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANEA
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

For backward compatibility the MANE bit has been duplicated to the MANEA bit position. Writing either one or the other has the same effect.

26.8.5 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x10

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANEA
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in IDR is written to one.

A bit in this register is cleared when the corresponding bit in IER is written to one.

For backward compatibility the MANE bit has been duplicated to the MANEA bit position. Reading either one or the other has the same effect.

26.8.6 Channel Status Register

Name: CSR

Access Type: Read-only

Offset: 0x14

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANERR
23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

- **LINSNRE: LIN Slave Not Responding Error**
  - 0: No LIN Slave Not Responding Error has been detected since the last RSTSTA.
  - 1: A LIN Slave Not Responding Error has been detected since the last RSTSTA.
- **LINCE: LIN Checksum Error**
  - 0: No LIN Checksum Error has been detected since the last RSTSTA.
  - 1: A LIN Checksum Error has been detected since the last RSTSTA.
- **LINIPE: LIN Identifier Parity Error**
  - 0: No LIN Identifier Parity Error has been detected since the last RSTSTA.
  - 1: A LIN Identifier Parity Error has been detected since the last RSTSTA.
- **LINISFE: LIN Inconsistent Synch Field Error**
  - 0: No LIN Inconsistent Synch Field Error has been detected since the last RSTSTA
  - 1: The USART is configured as a Slave node and a LIN Inconsistent Synch Field Error has been detected since the last RSTSTA.
- **LINBE: LIN Bit Error**
  - 0: No Bit Error has been detected since the last RSTSTA.
  - 1: A Bit Error has been detected since the last RSTSTA.
- **MANERR: Manchester Error**
  - 0: No Manchester error has been detected since the last RSTSTA.
  - 1: At least one Manchester error has been detected since the last RSTSTA.
- **CTS: Image of CTS Input**
  - 0: CTS is at 0.
  - 1: CTS is at 1.
- **DCD: Image of DCD Input**
  - 0: DCD is at 0.
  - 1: DCD is at 1.
- **DSR: Image of DSR Input**
  - 0: DSR is at 0
  - 1: DSR is at 1.

- **RI: Image of RI Input**
  - 0: RI is at 0.
  - 1: RI is at 1.
- **CTSIC: Clear to Send Input Change Flag**
  - 0: No input change has been detected on the CTS pin since the last read of CSR.
  - 1: At least one input change has been detected on the CTS pin since the last read of CSR.
- **DCDIC: Data Carrier Detect Input Change Flag**
  - 0: No input change has been detected on the DCD pin since the last read of CSR.
  - 1: At least one input change has been detected on the DCD pin since the last read of CSR.
- **DSRIC: Data Set Ready Input Change Flag**
  - 0: No input change has been detected on the DSR pin since the last read of CSR.
  - 1: At least one input change has been detected on the DSR pin since the last read of CSR.
- **RIIC: Ring Indicator Input Change Flag**
  - 0: No input change has been detected on the RI pin since the last read of CSR.
  - 1: At least one input change has been detected on the RI pin since the last read of CSR.
- **LINTC: LIN Transfer Completed**
  - 0: The USART is idle or a LIN transfer is ongoing.
  - 1: A LIN transfer has been completed since the last RSTSTA.
- **LINIR: LIN Identifier Received**
  - 0: No LIN Identifier Received
  - 1: The USART is configured as a Slave node and a LIN Identifier has been received since the last RSTSTA.
- **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.
- **RXBUFF: Reception Buffer Full**
  - 0: The signal Buffer Full from the Receive PDCA channel is inactive.
  - 1: The signal Buffer Full from the Receive PDCA channel is active.
- **ITER/UNRE: Max number of Repetitions Reached or SPI Underrun Error**
  - If USART does not operate in SPI Slave Mode (MODE … 0xF):
  - ITER = 0: Maximum number of repetitions has not been reached since the last RSTSTA.
  - ITER = 1: Maximum number of repetitions has been reached since the last RSTSTA.
  - If USART operates in SPI Slave Mode (MODE = 0xF):
  - UNRE = 0: No SPI underrun error has occurred since the last RSTSTA.
  - UNRE = 1: At least one SPI underrun error has occurred since the last RSTSTA.
- **TXEMPTY: Transmitter Empty**
  - 0: There are characters in either THR or the Transmit Shift Register, or the transmitter is disabled.
  - 1: There are no characters in THR, nor in the Transmit Shift Register.
- **TIMEOUT: Receiver Time-out**
  - 0: There has not been a time-out since the last Start Time-out command (STTTO in CR) or the Time-out Register is 0.
  - 1: There has been a time-out since the last Start Time-out command (STTTO in CR).
- **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.
- **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.
- **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.
- **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.

- **TXRDY: Transmitter Ready**
  0: A character is in the 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 THR.
- **RXRDY: Receiver Ready**
  0: No complete character has been received since the last read of 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 RHR has not yet been read.

26.8.7 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x18

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
RXSYNH	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXSYNH: Received Sync**
  0: Last Character received is a Data.
  1: Last Character received is a Command.
- **RXCHR: Received Character**
  Last character received if RXRDY is set.

26.8.8 USART Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x1C

Reset Value: -

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							

- **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.

- **TXCHR: Character to be Transmitted**

  Next character to be transmitted after the current character if TXRDY is not set.

26.8.9 Baud Rate Generator Register

Name: BRGR

Access Type: Read-write

Offset: 0x20

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **FP: Fractional Part**
  0: Fractional divider is disabled.
  1 - 7: Baudrate resolution, defined by FP x 1/8.
- **CD: Clock Divider**

Table 26-23.

	MODE ≠ ISO7816			
	SYNC = 0		**SYNC = 1 or MODE = SPI (Master or Slave)**	
CD	**OVER = 0**	**OVER = 1**		**MODE = ISO7816**
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

26.8.10 Receiver Time-out Register

Name: RTOR

Access Type: Read-write

Offset: 0x24

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **TO: Time-out Value**

  0: The Receiver Time-out is disabled.

  1 - 131071: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

26.8.11 Transmitter Timeguard Register

Name: TTGR

Access Type: Read-write

Offset: 0x28

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
TG							

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **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.

26.8.12 FI DI RATIO Register

Name: FIDI

Access Type: Read-write

Offset: 0x40

Reset Value: 0x00000174

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

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

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

7	6	5	4	3	2	1	0
FI_DI_RATIO							

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **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 CLK divided by FI_DI_RATIO.

26.8.13 Number of Errors Register

Name: NER

Access Type: Read-only

Offset: 0x44

Reset Value: -

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

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.

26.8.14 IrDA FILTER Register

Name: IFR

Access Type: Read-write

Offset: 0x4C

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
IRDA_FILTER							

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **IRDA_FILTER: IrDA Filter**
  Sets the filter of the IrDA demodulator.

26.8.15 Manchester Configuration Register

Name: MAN

Access Type: Read-write

Offset: 0x50

Reset Value: 0x30011004

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

This register can only be written if the WPEN bit is cleared in “Write Protect Mode Register” on page 616.

- **DRIFT: Drift compensation**
  0: The USART can not recover from an important clock drift
  1: The USART can recover from clock drift. The 16X clock mode must be enabled.
- **RX_MPOL: Receiver Manchester Polarity**
  0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.
  1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.
- **RX_PP: Receiver Preamble Pattern detected**

Table 26-24.

RX_PP		Preamble Pattern default polarity assumed (RX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **RX_PL: Receiver Preamble Length**
  0: The receiver preamble pattern detection is disabled
  1 - 15: The detected preamble length is RX_PL x Bit Period
- **TX_MPOL: Transmitter Manchester Polarity**
  0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.
  1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **TX_PP: Transmitter Preamble Pattern**

Table 26-25.

TX_PP		Preamble Pattern default polarity assumed (TX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **TX_PL: Transmitter Preamble Length**

  0: The Transmitter Preamble pattern generation is disabled

  1 - 15: The Preamble Length is TX_PL x Bit Period

26.8.16 LIN Mode Register

Name: LINMR

Access Type: Read-write

Offset: 0x54

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
DLC							

7	6	5	4	3	2	1	0
WKUPTYP	FSDIS	DLM	CHKTYP	CHKDIS	PARDIS	NACT	

- **PDCM: PDCA Mode**
  0: The LIN mode register LINMR is not written by the PDCA.
  1: The LIN mode register LINMR (excepting that flag) is written by the PDCA.
- **DLC: Data Length Control**
  0 - 255: Defines the response data length if DLM=0,in that case the response data length is equal to DLC+1 bytes.
- **WKUPTYP: Wakeup Signal Type**
  0: setting the bit LINWKUP in the control register sends a LIN 2.0 wakeup signal.
  1: setting the bit LINWKUP in the control register sends a LIN 1.3 wakeup signal.
- **FDIS: Frame Slot Mode Disable**
  0: The Frame Slot Mode is enabled.
  1: The Frame Slot Mode is disabled.
- **DLM: Data Length Mode**
  0: The response data length is defined by the field DLC of this register.
  1: The response data length is defined by the bits 5 and 6 of the Identifier (IDCHR in LINIR).
- **CHKTYP: Checksum Type**
  0: LIN 2.0 “Enhanced” Checksum
  1: LIN 1.3 “Classic” Checksum
- **CHKDIS: Checksum Disable**
  0: In Master node configuration, the checksum is computed and sent automatically. In Slave node configuration, the checksum is checked automatically.
  1: Whatever the node configuration is, the checksum is not computed/sent and it is not checked.
- **PARDIS: Parity Disable**
  0: In Master node configuration, the Identifier Parity is computed and sent automatically. In Master node and Slave node configuration, the parity is checked automatically.
  1:Whatever the node configuration is, the Identifier parity is not computed/sent and it is not checked.

- **NACT: LIN Node Action**

Table 1.

NACT		Mode Description
0	0	PUBLISH: The USART transmits the response.
0	1	SUBSCRIBE: The USART receives the response.
1	0	IGNORE: The USART does not transmit and does not receive the response.
1	1	Reserved

26.8.17 LIN Identifier Register

Name: LINIR

Access Type: Read-write or Read-only

Offset: 0x58

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
IDCHR							

- **IDCHR: Identifier Character**

  If MODE=0xA (Master node configuration):
  IDCHR is Read-write and its value is the Identifier character to be transmitted.
  if MODE=0xB (Slave node configuration):
  IDCHR is Read-only and its value is the last Identifier character that has been received.

26.8.18 Write Protect Mode Register

Register Name: WPMR

Access Type: Read-write

Offset: 0xE4

Reset Value: See Table 26-16 on page 591

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

- **WPKEY: Write Protect KEY**

  Should be written at value 0x858365 ("USA" in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

- **WPEN: Write Protect Enable**

  0 = Disables the Write Protect if WPKEY corresponds to 0x858365 ("USA" in ASCII).

  1 = Enables the Write Protect if WPKEY corresponds to 0x858365 ("USA" in ASCII).

Protects the registers:

- "Mode Register" on page 594
- "Baud Rate Generator Register" on page 605
- "Receiver Time-out Register" on page 606
- "Transmitter Timeguard Register" on page 607
- "FI DI RATIO Register" on page 608
- "IrDA FILTER Register" on page 610
- "Manchester Configuration Register" on page 611

26.8.19 Write Protect Status Register

Register Name: WPSR

Access Type: Read-only

Offset: 0xE8

Reset Value: See Table 26-16 on page 591

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

- **WPVSRC: Write Protect Violation Source**

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

- **WPVS: Write Protect Violation Status**

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

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

Note: Reading WPSR automatically clears all fields.

26.8.20 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

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

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

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

7	6	5	4	3	2	1	0
VERSION							

- **VARIANT**

  Reserved. No functionality associated.

- **VERSION**

  Version of the module. No functionality associated.

26.9 Module Configuration

The specific configuration for each USART instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 26-26. Module Configuration

Feature	USART0	USART1	USART2	USART3
SPI Logic	Implemented	Implemented	Implemented	Implemented
LIN Logic	Implemented	Implemented	Implemented	Implemented
Manchester Logic	Not Implemented	Implemented	Not Implemented	Not Implemented
Modem Logic	Not Implemented	Implemented	Not Implemented	Not Implemented
IRDA Logic	Not Implemented	Implemented	Not Implemented	Not Implemented
Fractional Baudrate	Implemented	Implemented	Implemented	Implemented
ISO7816	Not Implemented	Implemented	Not Implemented	Not Implemented
DIV	8	8	8	8

Table 26-27. Module Clock Name

Module name	Clock name
USART0	CLK_USART0
USART1	CLK_USART1
USART2	CLK_USART2
USART3	CLK_USART3

Table 26-28. Register Reset Values

Module name	Reset Value
VERSION	0x00000420

27. Hi-Speed USB On-The-Go Interface (USBB)

Rev: 3.2.0.1

27.1 Features

- **Compatible with the USB 2.0 specification**
- **Supports High (480 Mbps), Full (12 Mbps) and Low (1.5 Mbps) speed communication and On-The-Go**
- nine **pipes/endpoints**
- **2368 of Embedded Dual-Port RAM (DPRAM) for Pipes/Endpoints**
- **Up to 2 memory banks per Pipe/Endpoint (Not for Control Pipe/Endpoint)**
- **Flexible Pipe/Endpoint configuration and management with dedicated DMA channels**
- **On-Chip UTMI transceiver including Pull-Ups/Pull-downs**
- **On-Chip OTG pad including VBUS analog comparator**

27.2 Overview

The Universal Serial Bus (USB) MCU device complies with the Universal Serial Bus (USB) 2.0 specification, in all speeds.

Each pipe/endpoint can be configured in one of several transfer types. It can be associated with one or more banks of a dual-port RAM (DPRAM) used to store the current data payload. If several banks are used (“ping-pong” mode), then one DPRAM bank is read or written by the CPU or the DMA while the other is read or written by the USBB core. This feature is mandatory for isochronous pipes/endpoints.

Table 27-1 on page 620 describes the hardware configuration of the USB MCU device.

Table 27-1. Description of USB Pipes/Endpoints

Pipe/Endpoint	Mnemonic	Max. Size	Max. Nb. Banks	DMA	Type
0	PEP0	64 bytes	1	N	Control
1	PEP1	512 bytes	2	Y	Isochronous/Bulk/Interrupt
2	PEP2	512 bytes	2	Y	Isochronous/Bulk/Interrupt
3	PEP3	512 bytes	2	Y	Isochronous/Bulk/Interrupt
4	PEP4	512 bytes	2	Y	Isochronous/Bulk/Interrupt
5	PEP5	512 bytes	2	Y	Isochronous/Bulk/Interrupt
6	PEP6	512 bytes	2	Y	Isochronous/Bulk/Interrupt
7	PEP7	512 bytes	2	Y	Isochronous/Bulk/Interrupt

The theoretical maximal pipe/endpoint configuration (458752) exceeds the real DPRAM size (2368). The user needs to be aware of this when configuring pipes/endpoints. To fully use the

2368 of DPRAM, the user could for example use the configuration described inTable 27-2 on page 621.

Table 27-2. Example of Configuration of Pipes/Endpoints Using the Whole DPRAM

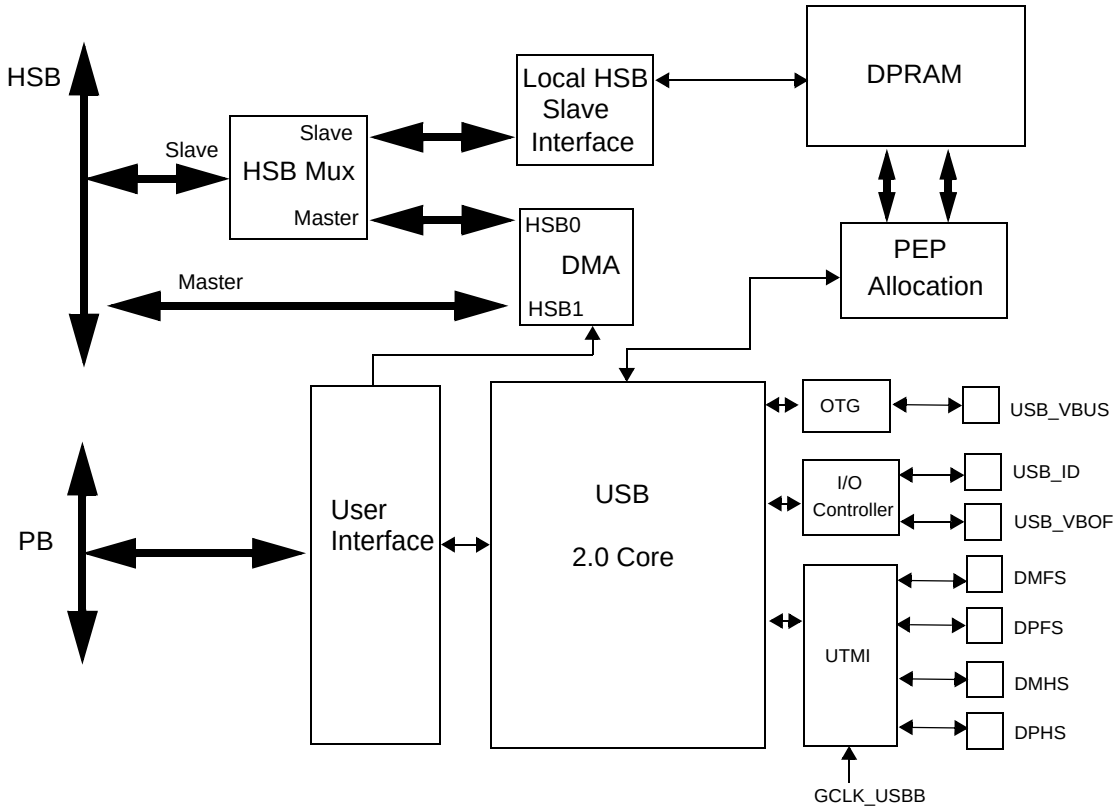
Pipe/Endpoint	Mnemonic	Size	Nb. Banks
0	PEP0	64 bytes	1
1	PEP1	512 bytes	2
2	PEP2	512 bytes	2
3	PEP3	256 bytes	1

27.3 Block Diagram

The USBB provides a hardware device to interface a USB link to a data flow stored in a dual-port RAM (DPRAM).

The UTMI transceiver requires an external 12MHz clock as a reference to its internal 480MHz PLL. The internal 480MHz PLL is used to clock an internal DLL module to recover the USB differential data at 480Mbps.

Figure 27-1. USBB Block Diagram



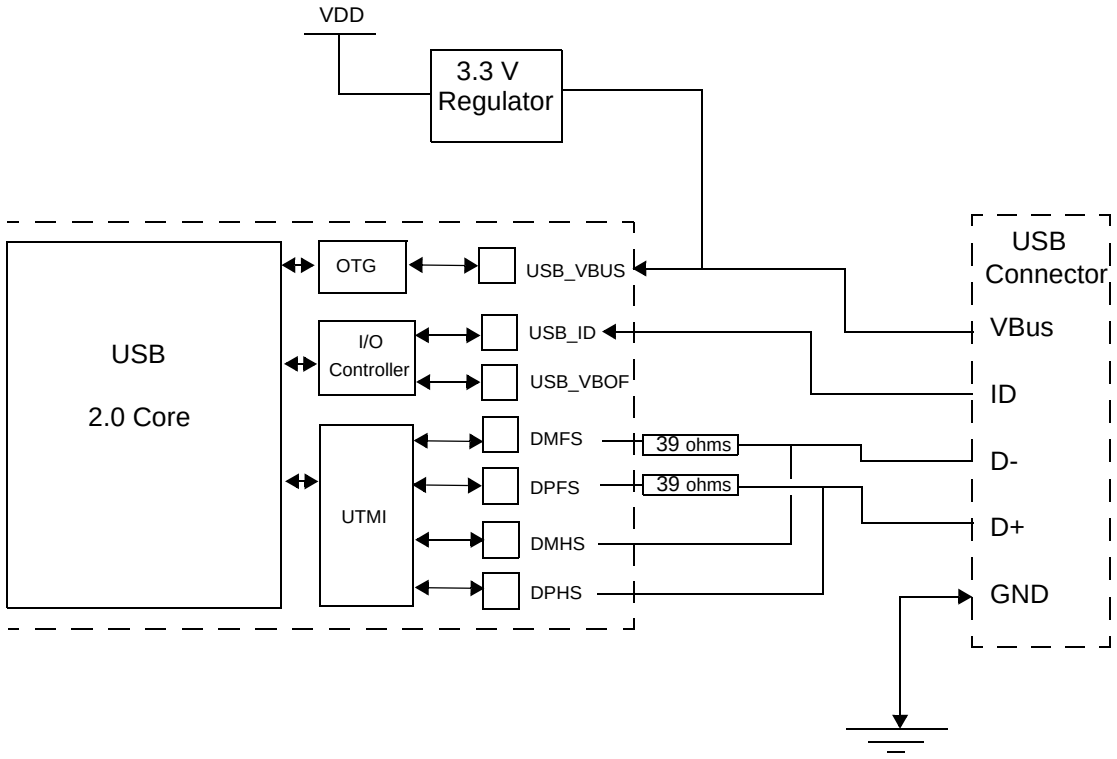
27.4 Application Block Diagram

Depending on the USB operating mode (device-only, reduced-host or OTG mode) and the power source (bus-powered or self-powered), there are different typical hardware implementations.

27.4.1 Device Mode

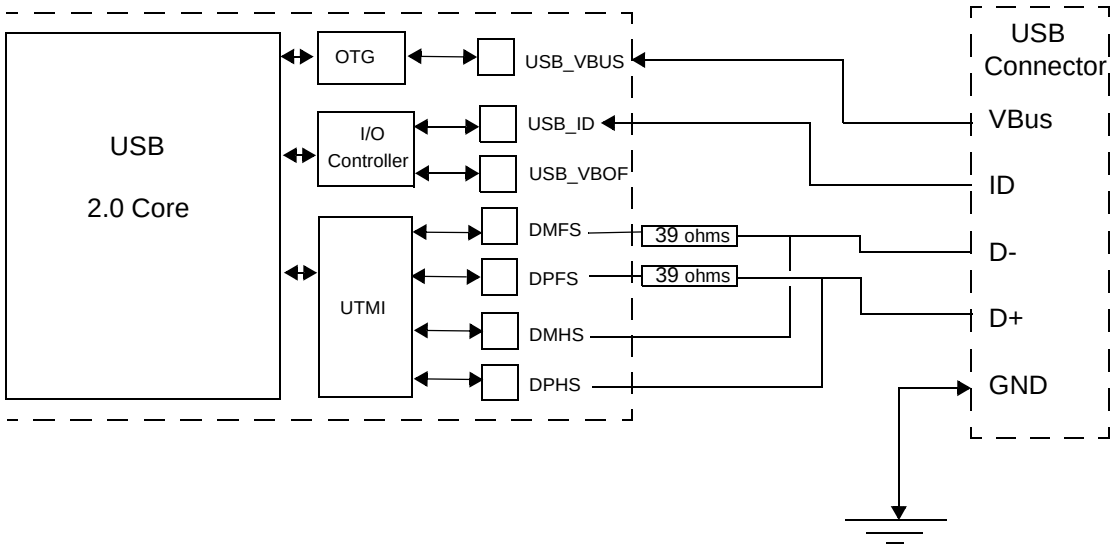
27.4.1.1 Bus-Powered device

Figure 27-2. Bus-Powered Device Application Block Diagram



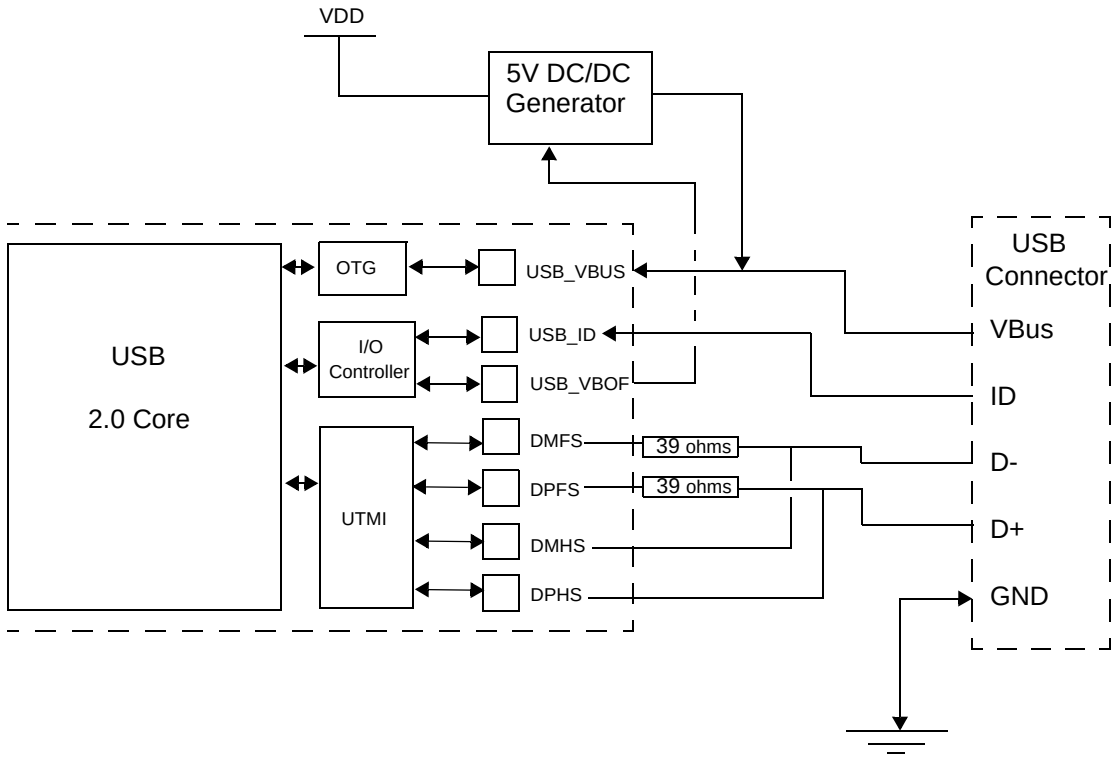
27.4.1.2 Self-Powered device

Figure 27-3. Self-powered Device Application Block Diagram



27.4.2 Host and OTG Modes

Figure 27-4. Host and OTG Application Block Diagram



27.5 I/O Lines Description

Table 27-3. I/O Lines Description

PIn Name	Pin Description	Type	Active Level
USB_VBOF	USB VBus On/Off: Bus Power Control Port	Output	$\overline{\text{VBUSPO}}$
USB_VBUS	VBus: Bus Power Measurement Port	Input	
DMF	FS Data -: Full-Speed Differential Data Line - Port	Input/Output	
DPF	FS Data +: Full-Speed Differential Data Line + Port	Input/Output	
DMH	HS Data -: Hi-Speed Differential Data Line - Port	Input/Output	
DPH	HS Data +: Hi-Speed Differential Data Line + Port	Input/Output	
USB_ID	USB Identification: Mini Connector Identification Port	Input	Low: Mini-A plug High Z: Mini-B plug

27.6 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

27.6.1 I/O Lines

The USB_VBOF and USB_ID pins are multiplexed with I/O Controller lines and may also be multiplexed with lines of other peripherals. In order to use them with the USB, the user must first configure the I/O Controller to assign them to their USB peripheral functions.

If USB_ID is used, the I/O Controller must be configured to enable the internal pull-up resistor of its pin.

If USB_VBOF or USB_ID is not used by the application, the corresponding pin can be used for other purposes by the I/O Controller or by other peripherals.

27.6.2 Clocks

The clock for the USBB bus interface (CLK_USBB) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the USBB before disabling the clock, to avoid freezing the USBB in an undefined state.

The UTMI transceiver needs a 12 MHz clock as a clock reference for its internal 480 MHz PLL. Before using the USB, the user must ensure that this 12 MHz clock is available. The 12 MHz input is connected to a Generic Clock (GCLK_USBB) provided by the Power Manager.

27.6.3 Interrupts

The USBB interrupt request line is connected to the interrupt controller. Using the USBB interrupt requires the interrupt controller to be programmed first.

27.7 Functional Description

27.7.1 USB General Operation

27.7.1.1 Introduction

After a hardware reset, the USBB is disabled. When enabled, the USBB runs either in device mode or in host mode according to the ID detection.

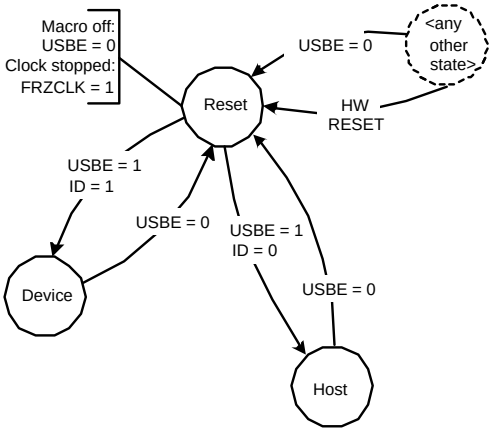
If the USB_ID pin is not connected to ground, the USB_ID Pin State bit in the General Status register (USBSTA.ID) is set (the internal pull-up resistor of the USB_ID pin must be enabled by the I/O Controller) and device mode is engaged.

The USBSTA.ID bit is cleared when a low level has been detected on the USB_ID pin. Host mode is then engaged.

27.7.1.2 Power-On and reset

Figure 27-5 on page 626 describes the USBB main states.

Figure 27-5. General States



After a hardware reset, the USBB is in the Reset state. In this state:

- The macro is disabled. The USBB Enable bit in the General Control register (USBCON.USBE) is zero.
- The macro clock is stopped in order to minimize power consumption. The Freeze USB Clock bit in USBCON (USBON.FRZCLK) is set.
- The UTMI is in suspend mode.
- The internal states and registers of the device and host modes are reset.
- The DPRAM is not cleared and is accessible.
- The USBSTA.ID bit and the VBus Level bit in the UBSTA (UBSTA.VBUS) reflect the states of the USB_ID and USB_VBUS input pins.
- The OTG Pad Enable (OTGPADE) bit, the VBus Polarity (VBUSPO) bit, the FRZCLK bit, the USBE bit, the USB_ID Pin Enable (UIDE) bit, the USBB Mode (UIMOD) bit in USBCON, and the Low-Speed Mode Force bit in the Device General Control (UDCON.LS) register can be written by software, so that the user can program pads and speed before enabling the macro, but their value is only taken into account once the macro is enabled and unfrozen.

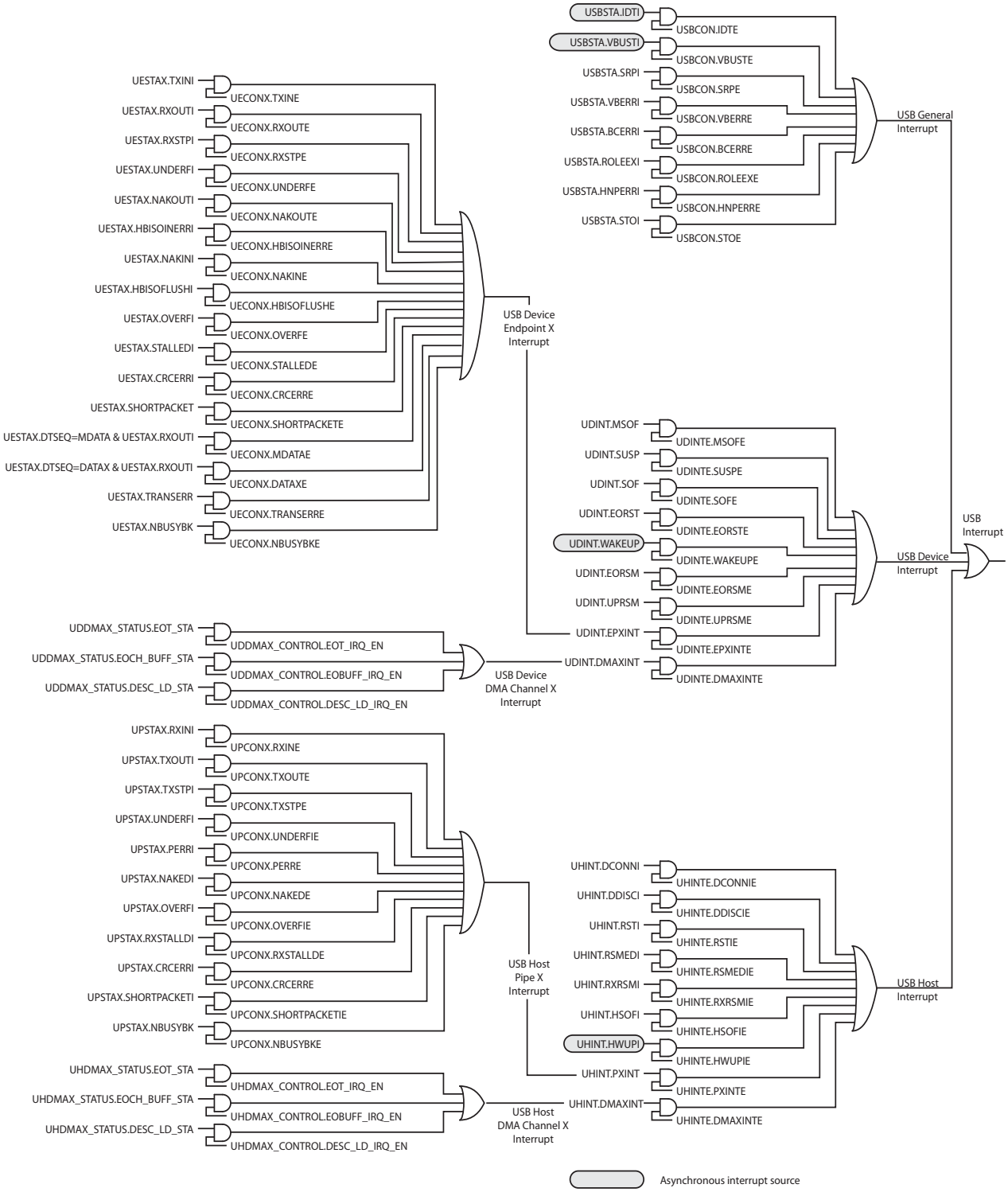
After writing a one to USBCON.USBE, the USBB enters the Device or the Host mode (according to the ID detection) in idle state.

The USBB can be disabled at any time by writing a zero to USBCON.USBE. In fact, writing a zero to USBCON.USBE acts as a hardware reset, except that the OTGPADE, VBUSPO, FRZCLK, UIDE, UIMOD and, LS bits are not reset.

27.7.1.3 Interrupts

One interrupt vector is assigned to the USB interface. Figure 27-6 on page 628 shows the structure of the USB interrupt system.

Figure 27-6. Interrupt System



See Section 27.7.2.19 and Section 27.7.3.13 for further details about device and host interrupts.

There are two kinds of general interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

The processing general interrupts are:

- The ID Transition Interrupt (IDTI)
- The VBus Transition Interrupt (VBUSTI)
- The SRP Interrupt (SRPI)
- The Role Exchange Interrupt (ROLEEXI)

The exception general interrupts are:

- The VBus Error Interrupt (VBERRI)
- The B-Connection Error Interrupt (BCERRI)
- The HNP Error Interrupt (HNPERRI)
- The Suspend Time-Out Interrupt (STOI)

27.7.1.4 MCU Power modes

•Run mode

In this mode, all MCU clocks can run, including the USB clock.

•Idle mode

In this mode, the CPU is halted, i.e. the CPU clock is stopped. The Idle mode is entered whatever the state of the USBB. The MCU wakes up on any USB interrupt.

•Frozen mode

Same as the Idle mode, except that the HSB module is stopped, so the USB DMA, which is an HSB master, can not be used. Moreover, the USB DMA must be stopped before entering this sleep mode in order to avoid erratic behavior. The MCU wakes up on any USB interrupt.

•Standby, Stop, DeepStop and Static modes

Same as the Frozen mode, except that the USB generic clock and other clocks are stopped, so the USBB is frozen. In the current version of the MCU, no USB interrupt can wake up the MCU in these modes, even the asynchronous interrupt sources.

•USB clock frozen

In the run, idle and frozen MCU modes, the USBB can be frozen when the USB line is in the suspend mode, by writing a one to the FRZCLK bit, what reduces power consumption.

In this case, it is still possible to access the following elements, but only in Run mode:

- The OTGPADE, VBUSPO, FRZCLK, USBE, UIDE, UIMOD and LS bits in the USBCON register
- The DPRAM (through the USB Pipe/Endpoint n FIFO Data (USBFIFOnDATA) registers, but not through USB bus transfers which are frozen)

Moreover, when FRZCLK is written to one, only the asynchronous interrupt sources may trigger the USB interrupt:

- The ID Transition Interrupt (IDTI)
- The VBus Transition Interrupt (VBUSTI)

- The Wake-up Interrupt (WAKEUP)
- The Host Wake-up Interrupt (HWUPI)

•USB Suspend mode

In peripheral mode, the Suspend Interrupt bit in the Device Global Interrupt register (UDINT.SUSP)indicates that the USB line is in the suspend mode. In this case, the transceiver is automatically set in suspend mode to reduce the consumption.The 480MHz internal PLL is stopped. The USBSTA.CLKUSABLE bit is cleared.

27.7.1.5 Speed control

•Device mode

When the USB interface is in device mode, the speed selection (full-speed or high-speed) is performed automatically by the USBB during the USB reset according to the host speed capability. At the end of the USB reset, the USBB enables or disables high-speed terminations and pull-up.

It is possible to restraint the USBB to full-speed or low-speed mode by handling the LS and the Speed Configuration (SPDCONF) bits in UDCON.

•Host mode

When the USB interface is in host mode, internal pull-down resistors are connected on both D+ and D- and the interface detects the speed of the connected device, which is reflected by the Speed Status (SPEED) field in USBSTA.

27.7.1.6 DPRAM management

Pipes and endpoints can only be allocated in ascending order (from the pipe/endpoint 0 to the last pipe/endpoint to be allocated). The user shall therefore configure them in the same order.

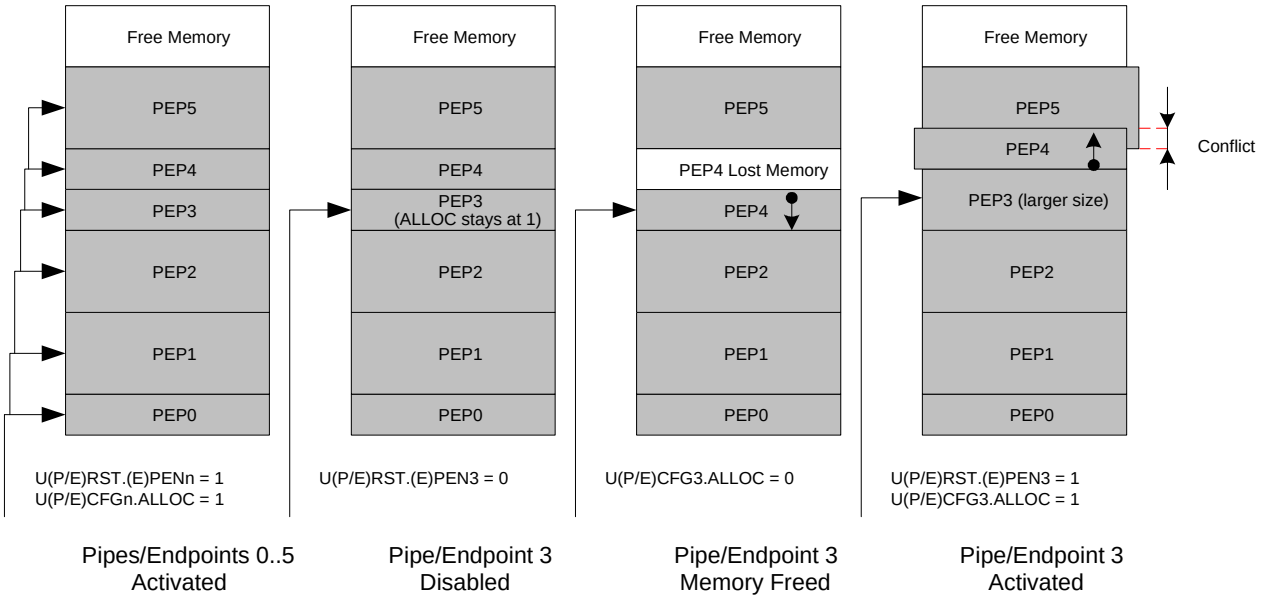
The allocation of a pipe/endpoint n starts when the Endpoint Memory Allocate bit in the Endpoint n Configuration register (UECFGn.ALLOC) is written to one. Then, the hardware allocates a memory area in the DPRAM and inserts it between the n-1 and n+1 pipes/endpoints. The n+1 pipe/endpoint memory window slides up and its data is lost. Note that the following pipe/endpoint memory windows (from n+2) do not slide.

Disabling a pipe, by writing a zero to the Pipe n Enable bit in the Pipe Enable/Reset register (UPRST.PENn), or disabling an endpoint, by writing a zero to the Endpoint n Enable bit in the Endpoint Enable/Reset register (UERST.EPENn), resets neither the UECFGn.ALLOC bit nor its configuration (the Pipe Banks (PBK) field, the Pipe Size (PSIZE) field, the Pipe Token (PTOKEN) field, the Pipe Type (PTYPE) field, the Pipe Endpoint Number (PEPNUM) field, and the Pipe Interrupt Request Frequency (INTFRQ) field in the Pipe n Configuration (UPCFGn) register/the Endpoint Banks (EPBK) field, the Endpoint Size (EPSIZE) field, the Endpoint Direction (EPDIR) field, and the Endpoint Type (EPTYPE) field in UECFGn).

To free its memory, the user shall write a zero to the UECFGn.ALLOC bit. The n+1 pipe/endpoint memory window then slides down and its data is lost. Note that the following pipe/endpoint memory windows (from n+2) does not slide.

Figure 27-7 on page 631 illustrates the allocation and reorganization of the DPRAM in a typical example.

Figure 27-7. Allocation and Reorganization of the DPRAM



1. The pipes/endpoints 0 to 5 are enabled, configured and allocated in ascending order. Each pipe/endpoint then owns a memory area in the DPRAM.
2. The pipe/endpoint 3 is disabled, but its memory is kept allocated by the controller.
3. In order to free its memory, its ALLOC bit is written to zero. The pipe/endpoint 4 memory window slides down, but the pipe/endpoint 5 does not move.
4. If the user chooses to reconfigure the pipe/endpoint 3 with a larger size, the controller allocates a memory area after the pipe/endpoint 2 memory area and automatically slides up the pipe/endpoint 4 memory window. The pipe/endpoint 5 does not move and a memory conflict appears as the memory windows of the pipes/endpoints 4 and 5 overlap. The data of these pipes/endpoints is potentially lost.

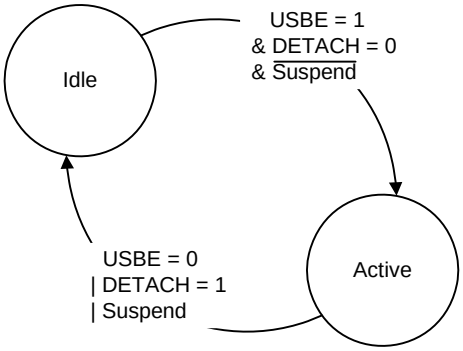
Note that:

- There is no way the data of the pipe/endpoint 0 can be lost (except if it is de-allocated) as memory allocation and de-allocation may affect only higher pipes/endpoints.
- Deactivating then reactivating a same pipe/endpoint with the same configuration only modifies temporarily the controller DPRAM pointer and size for this pipe/endpoint, but nothing changes in the DPRAM, so higher endpoints seem to not have been moved and their data is preserved as far as nothing has been written or received into them while changing the allocation state of the first pipe/endpoint.
- When the user write a one to the ALLOC bit, the Configuration OK Status bit in the Endpoint n Status register (UESTAn.CFGOK) is set only if the configured size and number of banks are correct compared to their maximal allowed values for the endpoint and to the maximal FIFO size (i.e. the DPRAM size), so the value of CFGOK does not consider memory allocation conflicts.

27.7.1.7 Pad Suspend

Figure 27-8 on page 632 shows the pad behavior.

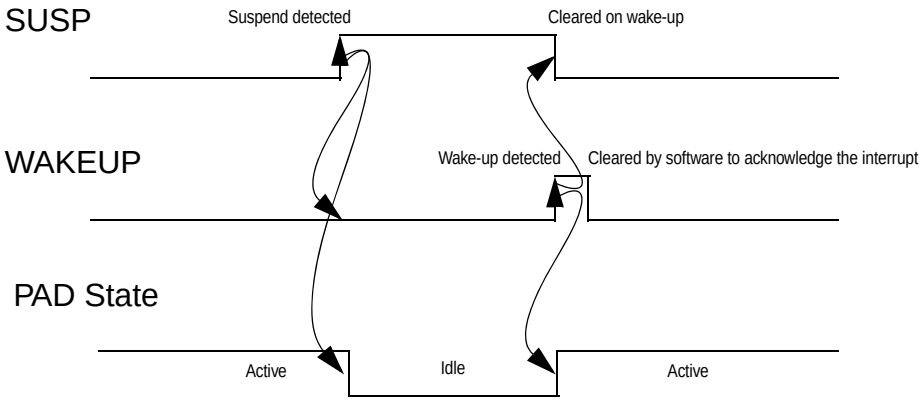
Figure 27-8. Pad Behavior



- In the Idle state, the pad is put in low power consumption mode.
- In the Active state, the pad is working.

Figure 27-9 on page 632 illustrates the pad events leading to a PAD state change.

Figure 27-9. Pad Events



The SUSP bit is set and the Wake-Up Interrupt (WAKEUP) bit in UDINT is cleared when a USB “Suspend” state has been detected on the USB bus. This event automatically puts the USB pad in the Idle state. The detection of a non-idle event sets WAKEUP, clears SUSP and wakes up the USB pad.

Moreover, the pad goes to the Idle state if the macro is disabled or if the DETACH bit is written to one. It returns to the Active state when USBE is written to one and DETACH is written to zero.

27.7.1.8 *Customizing of OTG timers*

It is possible to refine some OTG timers thanks to the Timer Page (TIMPAGE) and Timer Value (TIMVALUE) fields in USBCON, as shown by Table 27-4 on page 633.

Table 27-4. Customizing of OTG Timers

		TIMPAGE			
		0b00 AWaitVrise Time-Out (see OTG Standard[1] Section 6.6.5.1)	**0b01 VbBusPulsing Time-Out (see OTG Standard[1] Section5.3.4)**	**0b10 PdTmOutCnt Time-Out (see OTG Standard[1] Section 5.3.2)**	**0b11 SRPDetTmOut Time-Out (see OTG Standard[1] Section 5.3.3)**
TIMVALUE	**00b**	20 ms	15 ms	93 ms	10 µs
	01b	50 ms	23 ms	105 ms	100 µs
	10b	70 ms	31 ms	118 ms	1 ms
	11b	100 ms	40 ms	131 ms	11 ms

Note: 1. “On-The-Go Supplement to the USB 2.0 Specification Revision 1.0a”.

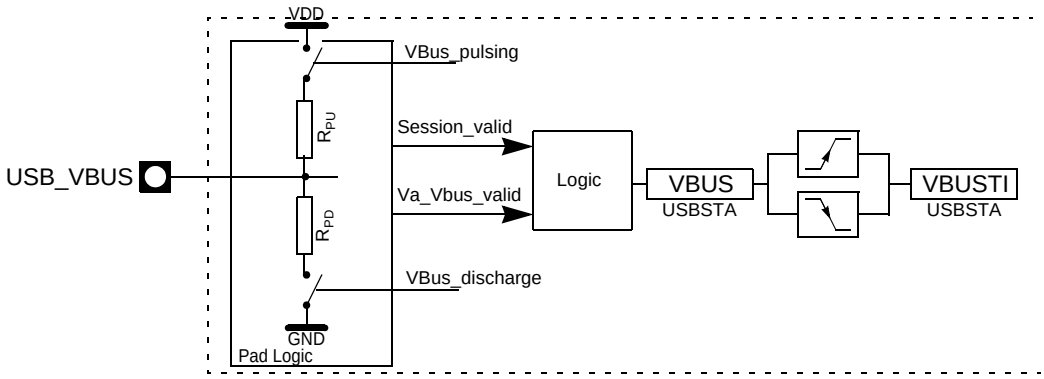
TIMPAGE is used to select the OTG timer to access while TIMVALUE indicates the time-out value of the selected timer.

TIMPAGE and TIMVALUE can be read or written. Before writing them, the user shall unlock write accesses by writing a one to the Timer Access Unlock (UNLOCK) bit in USBCON. This is not required for read accesses, except before accessing TIMPAGE if it has to be written in order to read the TIMVALUE field of another OTG timer.

27.7.1.9 *Plug-In detection*

The USB connection is detected from the USB_VBUS pad. Figure 27-10 on page 633 shows the architecture of the plug-in detector.

Figure 27-10. Plug-In Detection Input Block Diagram



The control logic of the USB_VBUS pad outputs two signals:

- The Session_valid signal is high when the voltage on the USB_VBUS pad is higher than or equal to 1.4V.
- The Va_Vbus_valid signal is high when the voltage on the USB_VBUS pad is higher than or equal to 4.4V.

In device mode, the USBSTA.VBUS bit follows the Session_valid comparator output:

- It is set when the voltage on the USB_VBUS pad is higher than or equal to 1.4V.
- It is cleared when the voltage on the VBUS pad is lower than 1.4V.

In host mode, the USBSTA.VBUS bit follows an hysteresis based on Session_valid and Va_Vbus_valid:

- It is set when the voltage on the USB_VBUS pad is higher than or equal to 4.4V.
- It is cleared when the voltage on the USB_VBUS pad is lower than 1.4V.

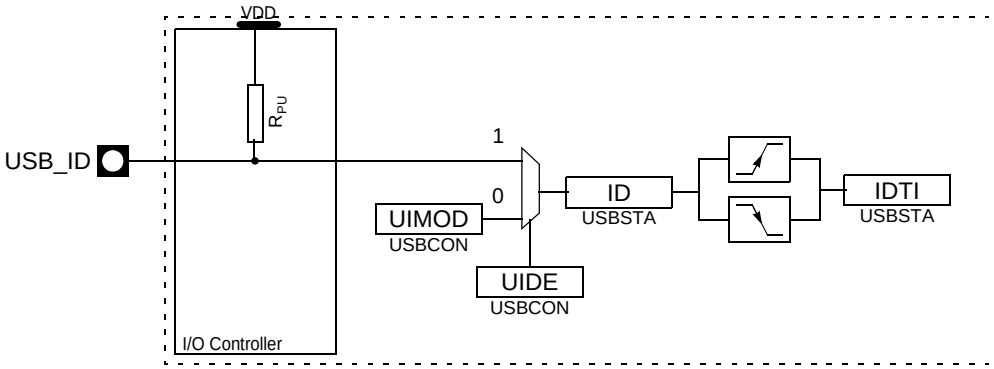
The VBus Transition interrupt (VBUSTI) bit in USBSTA is set on each transition of the USBSTA.VBUS bit.

The USBSTA.VBUS bit is effective whether the USBB is enabled or not.

27.7.1.10 ID detection

Figure 27-11 on page 634 shows how the ID transitions are detected.

Figure 27-11. ID Detection Input Block Diagram



The USB mode (device or host) can be either detected from the USB_ID pin or software selected by writing to the UIMOD bit, according to the UIDE bit. This allows the USB_ID pin to be used as a general purpose I/O pin even when the USB interface is enabled.

By default, the USB_ID pin is selected (UIDE is written to one) and the USBB is in device mode (UBSTA.ID is one), what corresponds to the case where no Mini-A plug is connected, i.e. no plug or a Mini-B plug is connected and the USB_ID pin is kept high by the internal pull-up resistor from the I/O Controller (which must be enabled if USB_ID is used).

The ID Transition Interrupt (IDTI) bit in USBSTA is set on each transition of the ID bit, i.e. when a Mini-A plug (host mode) is connected or disconnected. This does not occur when a Mini-B plug (device mode) is connected or disconnected.

The USBSTA.ID bit is effective whether the USBB is enabled or not.

27.7.2 USB Device Operation

27.7.2.1 Introduction

In device mode, the USBB supports full- and low-speed data transfers.

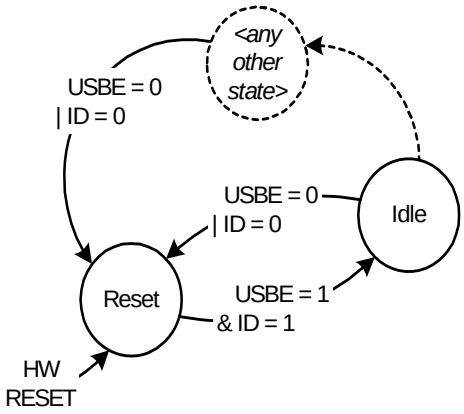
In addition to the default control endpoint, eight endpoints are provided, which can be configured with the types isochronous, bulk or interrupt, as described in .Table 27-1 on page 620.

The device mode starts in the Idle state, so the pad consumption is reduced to the minimum.

27.7.2.2 Power-On and reset

Figure 27-12 on page 635 describes the USBB device mode main states.

Figure 27-12. Device Mode States



After a hardware reset, the USBB device mode is in the Reset state. In this state:

- The macro clock is stopped in order to minimize power consumption (FRZCLK is written to one).
- The internal registers of the device mode are reset.
- The endpoint banks are de-allocated.
- Neither D+ nor D- is pulled up (DETACH is written to one).

D+ or D- will be pulled up according to the selected speed as soon as the DETACH bit is written to zero and VBus is present. See “Device mode” for further details.

When the USBB is enabled (USBE is written to one) in device mode (ID is one), its device mode state goes to the Idle state with minimal power consumption. This does not require the USB clock to be activated.

The USBB device mode can be disabled and reset at any time by disabling the USBB (by writing a zero to USBE) or when host mode is engaged (ID is zero).

27.7.2.3 USB reset

The USB bus reset is managed by hardware. It is initiated by a connected host.

When a USB reset is detected on the USB line, the following operations are performed by the controller:

- All the endpoints are disabled, except the default control endpoint.

- The default control endpoint is reset (see Section 27.7.2.4 for more details).
- The data toggle sequence of the default control endpoint is cleared.
- At the end of the reset process, the End of Reset (EORST) bit in UDINT interrupt is set.
- During a reset, the USBB automatically switches to the Hi-Speed mode if the host is Hi-Speed capable (the reset is called a Hi-Speed reset). The user should observe the USBSTA.SPEED field to know the speed running at the end of the reset (EORST is one).

27.7.2.4 *Endpoint reset*

An endpoint can be reset at any time by writing a one to the Endpoint n Reset (EPRSTn) bit in the UERST register. This is recommended before using an endpoint upon hardware reset or when a USB bus reset has been received. This resets:

- The internal state machine of this endpoint.
- The receive and transmit bank FIFO counters.
- All the registers of this endpoint (UECFGn, UESTAn, the Endpoint n Control (UECONn) register), except its configuration (ALLOC, EPBK, EPSIZE, EPDIR, EPTYPE) and the Data Toggle Sequence (DTSEQ) field of the UESTAn register.

Note that the interrupt sources located in the UESTAn register are not cleared when a USB bus reset has been received.

The endpoint configuration remains active and the endpoint is still enabled.

The endpoint reset may be associated with a clear of the data toggle sequence as an answer to the CLEAR_FEATURE USB request. This can be achieved by writing a one to the Reset Data Toggle Set bit in the Endpoint n Control Set register (UECONnSET.RSTDTS).(This will set the Reset Data Toggle (RSTD) bit in UECONn).

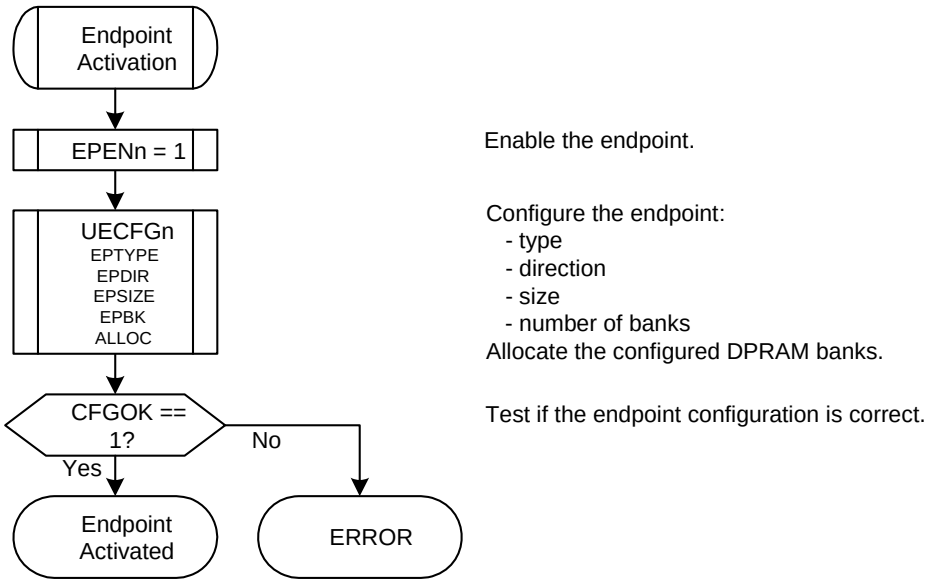
In the end, the user has to write a zero to the EPRSTn bit to complete the reset operation and to start using the FIFO.

27.7.2.5 *Endpoint activation*

The endpoint is maintained inactive and reset (see Section 27.7.2.4 for more details) as long as it is disabled (EPENn is written to zero). DTSEQ is also reset.

The algorithm represented on Figure 27-13 on page 637 must be followed in order to activate an endpoint.

Figure 27-13. Endpoint Activation Algorithm



As long as the endpoint is not correctly configured (CFGOK is zero), the controller does not acknowledge the packets sent by the host to this endpoint.

The CFGOK bit is set only if the configured size and number of banks are correct compared to their maximal allowed values for the endpoint (see Table 27-1 on page 620) and to the maximal FIFO size (i.e. the DPRAM size).

See Section 27.7.1.6 for more details about DPRAM management.

27.7.2.6 Address setup

The USB device address is set up according to the USB protocol.

- After all kinds of resets, the USB device address is 0.
- The host starts a SETUP transaction with a SET_ADDRESS(addr) request.
- The user write this address to the USB Address (UADD) field in UDCON, and write a zero to the Address Enable (ADDEN) bit in UDCON, so the actual address is still 0.
- The user sends a zero-length IN packet from the control endpoint.
- The user enables the recorded USB device address by writing a one to ADDEN.

Once the USB device address is configured, the controller filters the packets to only accept those targeting the address stored in UADD.

UADD and ADDEN shall not be written all at once.

UADD and ADDEN are cleared:

- On a hardware reset.
- When the USBB is disabled (USBE written to zero).
- When a USB reset is detected.

When UADD or ADDEN is cleared, the default device address 0 is used.

27.7.2.7 Suspend and wake-up

When an idle USB bus state has been detected for 3 ms, the controller set the Suspend (SUSP) interrupt bit in UDINT. The user may then write a one to the FRZCLK bit to reduce power consumption. The MCU can also enter the Idle or Frozen sleep mode to lower again power consumption.

To recover from the Suspend mode, the user shall wait for the Wake-Up (WAKEUP) interrupt bit, which is set when a non-idle event is detected, then write a zero to FRZCLK.

As the WAKEUP interrupt bit in UDINT is set when a non-idle event is detected, it can occur whether the controller is in the Suspend mode or not. The SUSP and WAKEUP interrupts are thus independent of each other except that one bit is cleared when the other is set.

27.7.2.8 Detach

The reset value of the DETACH bit is one.

It is possible to initiate a device re-enumeration simply by writing a one then a zero to DETACH.

DETACH acts on the pull-up connections of the D+ and D- pads. See “Device mode” for further details.

27.7.2.9 Remote wake-up

The Remote Wake-Up request (also known as Upstream Resume) is the only one the device may send on its own initiative, but the device should have beforehand been allowed to by a DEVICE_REMOTE_WAKEUP request from the host.

- First, the USBB must have detected a “Suspend” state on the bus, i.e. the Remote Wake-Up request can only be sent after a SUSP interrupt has been set.
- The user may then write a one to the Remote Wake-Up (RMWKUP) bit in UDCON to send an upstream resume to the host for a remote wake-up. This will automatically be done by the controller after 5 ms of inactivity on the USB bus.
- When the controller sends the upstream resume, the Upstream Resume (UPRSM) interrupt is set and SUSP is cleared.
- RMWKUP is cleared at the end of the upstream resume.
- If the controller detects a valid “End of Resume” signal from the host, the End of Resume (EORSM) interrupt is set.

27.7.2.10 STALL request

For each endpoint, the STALL management is performed using:

- The STALL Request (STALLRQ) bit in UECONn to initiate a STALL request.
- The STALLed Interrupt (STALLEDI) bit in UESTAn is set when a STALL handshake has been sent.

To answer the next request with a STALL handshake, STALLRQ has to be set by writing a one to the STALL Request Set (STALLRQS) bit. All following requests will be discarded (RXOUTI, etc. will not be set) and handshaked with a STALL until the STALLRQ bit is cleared, what is done when a new SETUP packet is received (for control endpoints) or when the STALL Request Clear (STALLRQC) bit is written to one.

Each time a STALL handshake is sent, the STALLEDI bit is set by the USBB and the EPnINT interrupt is set.

•*Special considerations for control endpoints*

If a SETUP packet is received into a control endpoint for which a STALL is requested, the Received SETUP Interrupt (RXSTPI) bit in UESTAn is set and STALLRQ and STALLEDI are cleared. The SETUP has to be ACKed.

This management simplifies the enumeration process management. If a command is not supported or contains an error, the user requests a STALL and can return to the main task, waiting for the next SETUP request.

•*STALL handshake and retry mechanism*

The retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the STALLRQ bit is set and if there is no retry required.

27.7.2.11 Management of control endpoints

•*Overview*

A SETUP request is always ACKed. When a new SETUP packet is received, the RXSTPI is set, but not the Received OUT Data Interrupt (RXOUTI) bit.

The FIFO Control (FIFOCON) bit in UECONn and the Read/Write Allowed (RWALL) bit in UESTAn are irrelevant for control endpoints. The user shall therefore never use them on these endpoints. When read, their value are always zero.

Control endpoints are managed using:

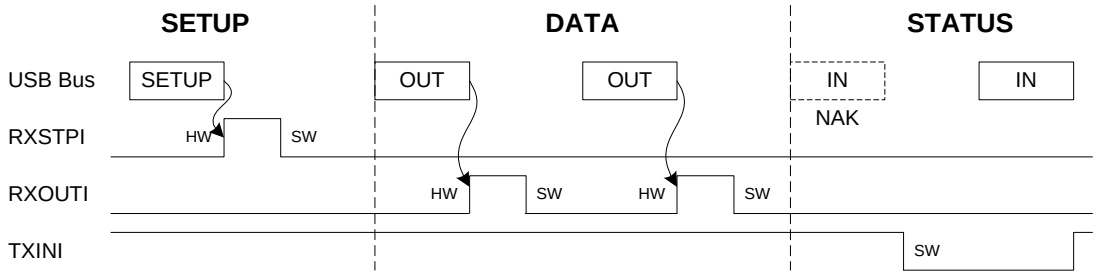
- The RXSTPI bit which is set when a new SETUP packet is received and which shall be cleared by firmware to acknowledge the packet and to free the bank.
- The RXOUTI bit which is set when a new OUT packet is received and which shall be cleared by firmware to acknowledge the packet and to free the bank.
- The Transmitted IN Data Interrupt (TXINI) bit which is set when the current bank is ready to accept a new IN packet and which shall be cleared by firmware to send the packet.

•*Control write*

Figure 27-14 on page 640 shows a control write transaction. During the status stage, the controller will not necessarily send a NAK on the first IN token:

- If the user knows the exact number of descriptor bytes that must be read, it can then anticipate the status stage and send a zero-length packet after the next IN token.
- Or it can read the bytes and wait for the NAKed IN Interrupt (NAKINI) which tells that all the bytes have been sent by the host and that the transaction is now in the status stage.

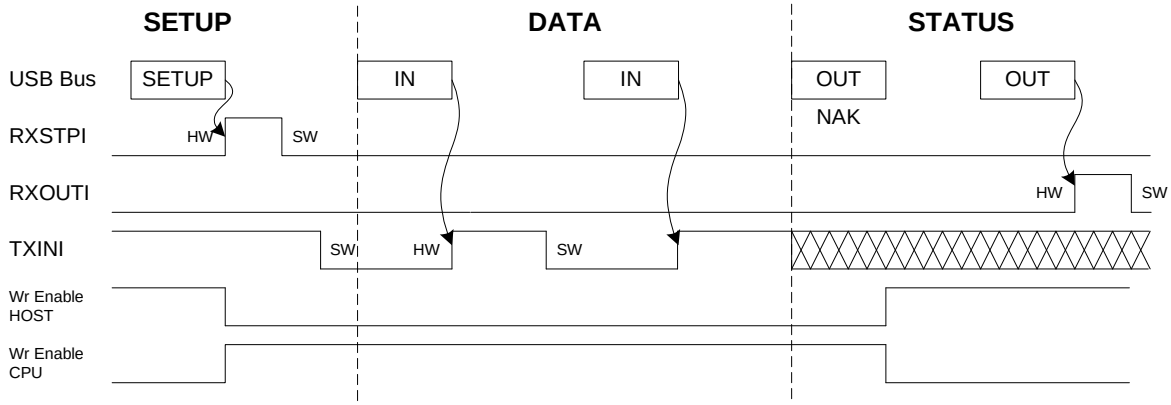
Figure 27-14. Control Write



•*Control read*

Figure 27-15 on page 640 shows a control read transaction. The USBB has to manage the simultaneous write requests from the CPU and the USB host.

Figure 27-15. Control Read



A NAK handshake is always generated on the first status stage command.

When the controller detects the status stage, all the data written by the CPU are lost and clearing TXINI has no effect.

The user checks if the transmission or the reception is complete.

The OUT retry is always ACKed. This reception sets RXOUTI and TXINI. Handle this with the following software algorithm:

```
set TXINI
wait for RXOUTI OR TXINI
if RXOUTI, then clear bit and return
if TXINI, then continue
```

Once the OUT status stage has been received, the USBB waits for a SETUP request. The SETUP request has priority over any other request and has to be ACKed. This means that any other bit should be cleared and the FIFO reset when a SETUP is received.

The user has to take care of the fact that the byte counter is reset when a zero-length OUT packet is received.

27.7.2.12 Management of IN endpoints

•*Overview*

IN packets are sent by the USB device controller upon IN requests from the host. All the data can be written which acknowledges or not the bank when it is full.

The endpoint must be configured first.

The TXINI bit is set at the same time as FIFOCON when the current bank is free. This triggers an EPnINT interrupt if the Transmitted IN Data Interrupt Enable (TXINE) bit in UECONn is one.

TXINI shall be cleared by software (by writing a one to the Transmitted IN Data Interrupt Enable Clear bit in the Endpoint n Control Clear register (UECONnCLR.TXINIC)) to acknowledge the interrupt, what has no effect on the endpoint FIFO.

The user then writes into the FIFO and write a one to the FIFO Control Clear (FIFOCONC) bit in UECONnCLR to clear the FIFOCON bit. This allows the USBB to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The TXINI and FIFOCON bits are updated in accordance with the status of the next bank.

TXINI shall always be cleared before clearing FIFOCON.

The RWALL bit is set when the current bank is not full, i.e. the software can write further data into the FIFO.

Figure 27-16. Example of an IN Endpoint with 1 Data Bank

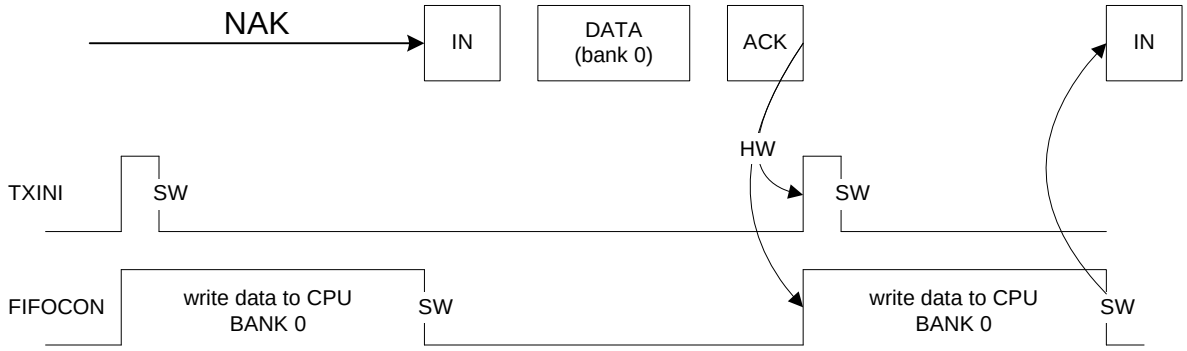
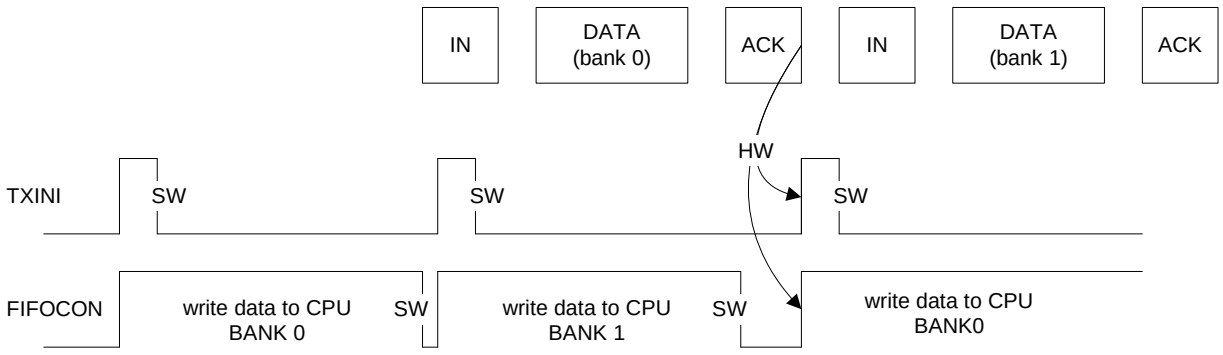


Figure 27-17. Example of an IN Endpoint with 2 Data Banks



•Detailed description

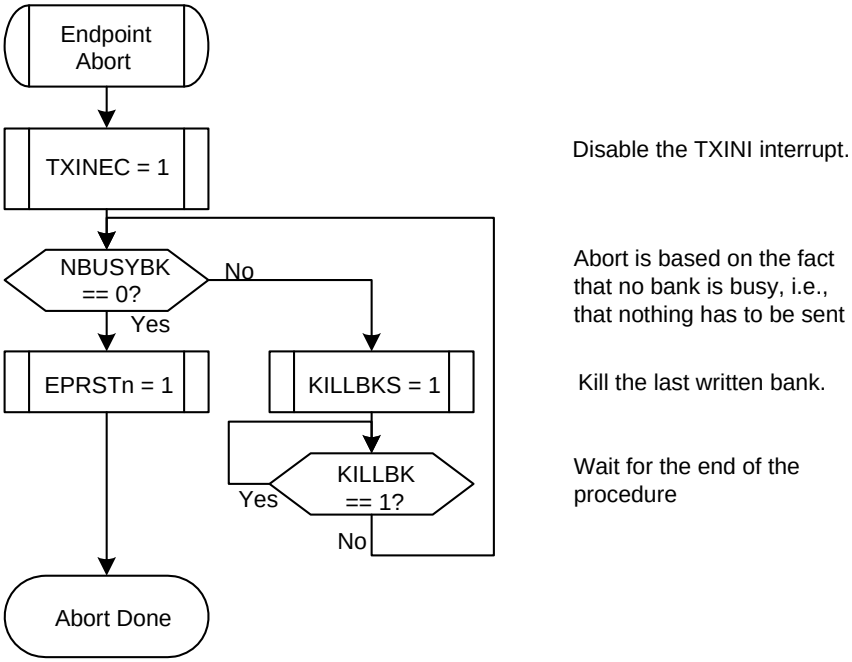
The data is written, following the next flow:

- When the bank is empty, TXINI and FIFOCON are set, what triggers an EPnINT interrupt if TXINE is one.
- The user acknowledges the interrupt by clearing TXINI.
- The user writes the data into the current bank by using the USB Pipe/Endpoint nFIFO Data (USBFIFOnDATA) register, until all the data frame is written or the bank is full (in which case RWALL is cleared and the Byte Count (BYCT) field in UESTAn reaches the endpoint size).
- The user allows the controller to send the bank and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be written while the previous one is being read by the host. Then, when the user clears FIFOCON, the following bank may already be free and TXINI is set immediately.

An “Abort” stage can be produced when a zero-length OUT packet is received during an IN stage of a control or isochronous IN transaction. The Kill IN Bank (KILLBK) bit in UECONn is used to kill the last written bank. The best way to manage this abort is to apply the algorithm represented on Figure 27-18 on page 642.

Figure 27-18. Abort Algorithm



27.7.2.13 *Management of OUT endpoints*

•Overview

OUT packets are sent by the host. All the data can be read which acknowledges or not the bank when it is empty.

The endpoint must be configured first.

The RXOUTI bit is set at the same time as FIFOCON when the current bank is full. This triggers an EPnINT interrupt if the Received OUT Data Interrupt Enable (RXOUTE) bit in UECONn is one.

RXOUTI shall be cleared by software (by writing a one to the Received OUT Data Interrupt Clear (RXOUTIC) bit) to acknowledge the interrupt, what has no effect on the endpoint FIFO.

The user then reads from the FIFO and clears the FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The RXOUTI and FIFOCON bits are updated in accordance with the status of the next bank.

RXOUTI shall always be cleared before clearing FIFOCON.

The RWALL bit is set when the current bank is not empty, i.e. the software can read further data from the FIFO.

Figure 27-19. Example of an OUT Endpoint with one Data Bank

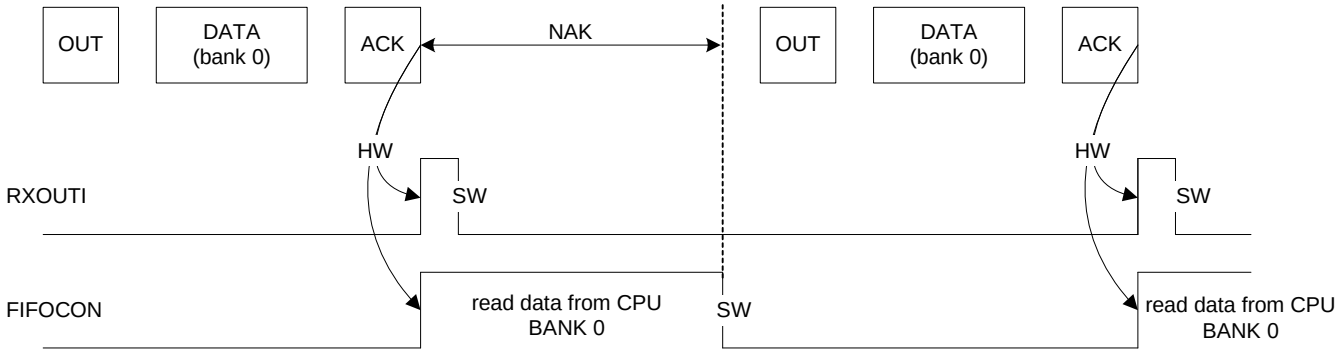
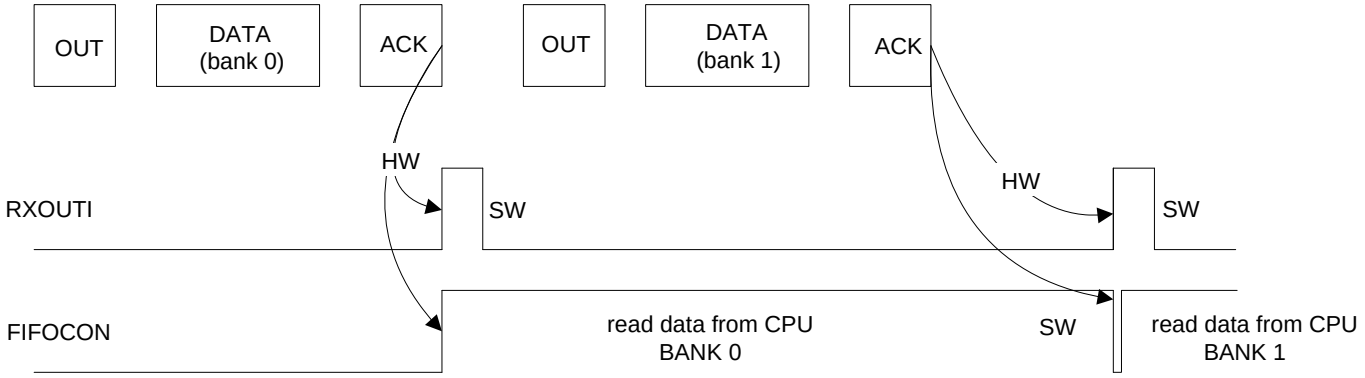


Figure 27-20. Example of an OUT Endpoint with two Data Banks



•Detailed description

The data is read, following the next flow:

- When the bank is full, RXOUTI and FIFOCON are set, what triggers an EPnINT interrupt if RXOUTE is one.
- The user acknowledges the interrupt by writing a one to RXOUTIC in order to clear RXOUTI.

- The user can read the byte count of the current bank from BYCT to know how many bytes to read, rather than polling RWALL.
- The user reads the data from the current bank by using the USBFIFOnDATA register, until all the expected data frame is read or the bank is empty (in which case RWALL is cleared and BYCT reaches zero).
- The user frees the bank and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be read while the following one is being written by the host. Then, when the user clears FIFOCON, the following bank may already be ready and RXOUTI is set immediately.

In Hi-Speed mode, the PING and NYET protocol is handled by the USBB. For single bank, a NYET handshake is always sent to the host (on Bulk-out transaction) to indicate that the current packet is acknowledged but there is no room for the next one. For double bank, the USBB responds to the OUT/DATA transaction with an ACK handshake when the endpoint accepted the data successfully and has room for another data payload (the second bank is free).

27.7.2.14 Underflow

This error exists only for isochronous IN/OUT endpoints. It set the Underflow Interrupt (UNDERFI) bit in UESTAn, what triggers an EPnINT interrupt if the Underflow Interrupt Enable (UNDERFE) bit is one.

An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBB.

An underflow can not occur during OUT stage on a CPU action, since the user may read only if the bank is not empty (RXOUTI is one or RWALL is one).

An underflow can also occur during OUT stage if the host sends a packet while the bank is already full. Typically, the CPU is not fast enough. The packet is lost.

An underflow can not occur during IN stage on a CPU action, since the user may write only if the bank is not full (TXINI is one or RWALL is one).

27.7.2.15 Overflow

This error exists for all endpoint types. It set the Overflow interrupt (OVERFI) bit in UESTAn, what triggers an EPnINT interrupt if the Overflow Interrupt Enable (OVERFE) bit is one.

An overflow can occur during OUT stage if the host attempts to write into a bank that is too small for the packet. The packet is acknowledged and the RXOUTI bit is set as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.

An overflow can not occur during IN stage on a CPU action, since the user may write only if the bank is not full (TXINI is one or RWALL is one).

27.7.2.16 HB IsoIn error

This error exists only for high-bandwidth isochronous IN endpoints.

At the end of the micro-frame, if at least one packet has been sent to the host, if less banks than expected has been validated (by clearing the FIFOCON) for this micro-frame, it set the HBISOINERRORI bit in UESTAn, what triggers an EPnINT interrupt if the High Bandwidth Isochronous IN Error Interrupt Enable (HBISOINERRORE) bit is one.

For instance, if the Number of Transaction per MicroFrame for Isochronous Endpoint (NBTRANS field in UECFGn is three (three transactions per micro-frame), only two banks are

filled by the CPU (three expected) for the current micro-frame. Then, the HBISOINERRI interrupt is generated at the end of the micro-frame. Note that an UNDERFI interrupt is also generated (with an automatic zero-length-packet), except in the case of a missing IN token.

27.7.2.17 HB IsoFlush

This error exists only for high-bandwidth isochronous IN endpoints.

At the end of the micro-frame, if at least one packet has been sent to the host, if there is missing IN token during this micro-frame, the bank(s) destined to this micro-frame is/are flushed out to ensure a good data synchronization between the host and the device.

For instance, if NBTRANS is three (three transactions per micro-frame), if only the first IN token (among 3) is well received by the USBB, then the two last banks will be discarded.

27.7.2.18 CRC error

This error exists only for isochronous OUT endpoints. It set the CRC Error Interrupt (CRCERRI) bit in UESTAn, what triggers an EPnINT interrupt if the CRC Error Interrupt Enable (CRCERRE) bit is one.

A CRC error can occur during OUT stage if the USBB detects a corrupted received packet. The OUT packet is stored in the bank as if no CRC error had occurred (RXOUTI is set).

27.7.2.19 Interrupts

See the structure of the USB device interrupt system on Figure 27-6 on page 628.

There are two kinds of device interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

•Global interrupts

The processing device global interrupts are:

- The Suspend (SUSP) interrupt
- The Start of Frame (SOF) interrupt with no frame number CRC error (the Frame Number CRC Error (FNCERR) bit in the Device Frame Number (UDFNUM) register is zero)
- The Micro Start of Frame (MSOF) interrupt with no CRC error.
- The End of Reset (EORST) interrupt
- The Wake-Up (WAKEUP) interrupt
- The End of Resume (EORSM) interrupt
- The Upstream Resume (UPRSM) interrupt
- The Endpoint n (EPnINT) interrupt
- The DMA Channel n (DMAnINT) interrupt

The exception device global interrupts are:

- The Start of Frame (SOF) interrupt with a frame number CRC error (FNCERR is one)
- The Micro Start of Frame (MSOF) interrupt with a CRC error

•Endpoint interrupts

The processing device endpoint interrupts are:

- The Transmitted IN Data Interrupt (TXINI)

- The Received OUT Data Interrupt (RXOUTI)
- The Received SETUP Interrupt (RXSTPI)
- The Short Packet (SHORTPACKET) interrupt
- The Number of Busy Banks (NBUSYBK) interrupt
- The Received OUT isochronous Multiple Data Interrupt (MDATAI)
- The Received OUT isochronous DataX Interrupt (DATAXI)

The exception device endpoint interrupts are:

- The Underflow Interrupt (UNDERFI)
- The NAKed OUT Interrupt (NAKOUTI)
- The High-bandwidth isochronous IN error Interrupt (HBISOINERRI)
- The NAKed IN Interrupt (NAKINI)
- The High-bandwidth isochronous IN Flush error Interrupt (HBISOFLUSHI)
- The Overflow Interrupt (OVERFI)
- The STALLed Interrupt (STALLEDI)
- The CRC Error Interrupt (CRCERRI)
- The Transaction error (ERRORTRANS) interrupt

•*DMA interrupts*

The processing device DMA interrupts are:

- The End of USB Transfer Status (EOTSTA) interrupt
- The End of Channel Buffer Status (EOCHBUFFSTA) interrupt
- The Descriptor Loaded Status (DESCLDSTA) interrupt

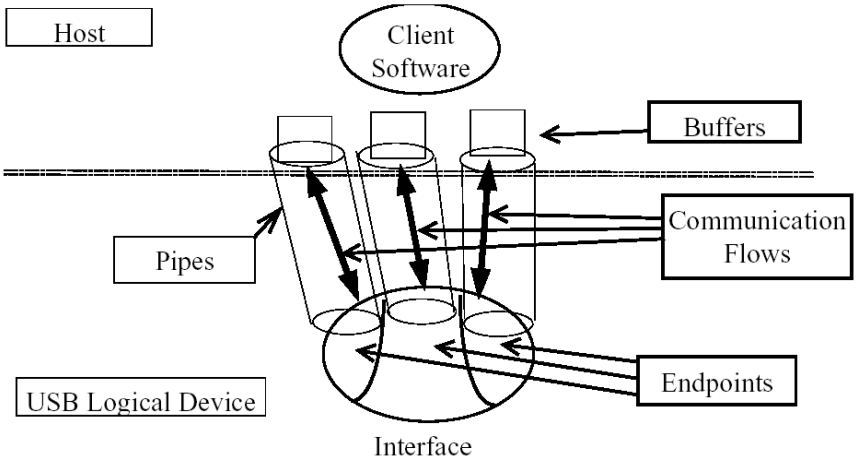
There is no exception device DMA interrupt.

27.7.3 USB Host Operation

27.7.3.1 Description of pipes

For the USBB in host mode, the term “pipe” is used instead of “endpoint” (used in device mode). A host pipe corresponds to a device endpoint, as described by the Figure 27-21 on page 647 from the USB specification.

Figure 27-21. USB Communication Flow

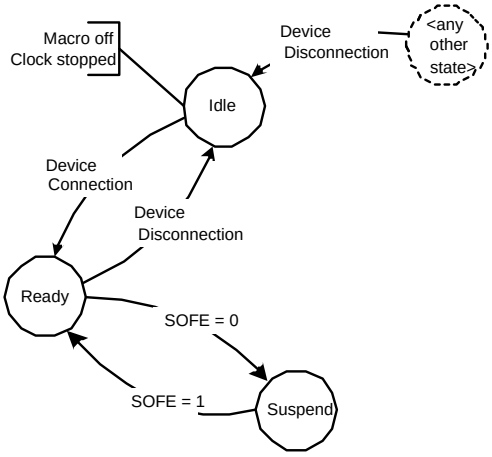


In host mode, the USBB associates a pipe to a device endpoint, considering the device configuration descriptors.

27.7.3.2 Power-On and reset

Figure 27-22 on page 647 describes the USBB host mode main states.

Figure 27-22. Host Mode States



After a hardware reset, the USBB host mode is in the Reset state.

When the USBB is enabled (USBE is one) in host mode (ID is zero), its host mode state goes to the Idle state. In this state, the controller waits for device connection with minimal power con-

sumption. The USB pad should be in the Idle state. Once a device is connected, the macro enters the Ready state, what does not require the USB clock to be activated.

The controller enters the Suspend state when the USB bus is in a “Suspend” state, i.e., when the host mode does not generate the “Start of Frame (SOF)”. In this state, the USB consumption is minimal. The host mode exits the Suspend state when starting to generate the SOF over the USB line.

27.7.3.3 *Device detection*

A device is detected by the USBB host mode when D+ or D- is no longer tied low, i.e., when the device D+ or D- pull-up resistor is connected. To enable this detection, the host controller has to provide the VBus power supply to the device by setting the VBUSRQ bit (by writing a one to the VBUSRQS bit).

The device disconnection is detected by the host controller when both D+ and D- are pulled down.

27.7.3.4 *USB reset*

The USBB sends a USB bus reset when the user write a one to the Send USB Reset bit in the Host General Control register (UHCON.RESET). The USB Reset Sent Interrupt bit in the Host Global Interrupt register (UHINT.RSTI) is set when the USB reset has been sent. In this case, all the pipes are disabled and de-allocated.

If the bus was previously in a “Suspend” state (the Start of Frame Generation Enable (SOFE) bit in UHCON is zero), the USBB automatically switches it to the “Resume” state, the Host Wake-Up Interrupt (HWUPI) bit in UHINT is set and the SOFE bit is set in order to generate SOFs or micro SOFs immediately after the USB reset.

At the end of the reset, the user should check the USBSTA.SPEED field to know the speed running according to the peripheral capability (LS.FS/HS)

27.7.3.5 *Pipe reset*

A pipe can be reset at any time by writing a one to the Pipe n Reset (PRSTn) bit in the UPRST register. This is recommended before using a pipe upon hardware reset or when a USB bus reset has been sent. This resets:

- The internal state machine of this pipe
- The receive and transmit bank FIFO counters
- All the registers of this pipe (UPCFGn, UPSTAn, UPCONn), except its configuration (ALLOC, PBK, PSIZE, PTOKEN, PTYPE, PEPNUM, INTFRQ in UPCFGn) and its Data Toggle Sequence field in the Pipe n Status register (UPSTAn.DTSEQ).

The pipe configuration remains active and the pipe is still enabled.

The pipe reset may be associated with a clear of the data toggle sequence. This can be achieved by setting the Reset Data Toggle bit in the Pipe n Control register (UPCONn.RSTDT) (by writing a one to the Reset Data Toggle Set bit in the Pipe n Control Set register (UPCONnSET.RSTDTS)).

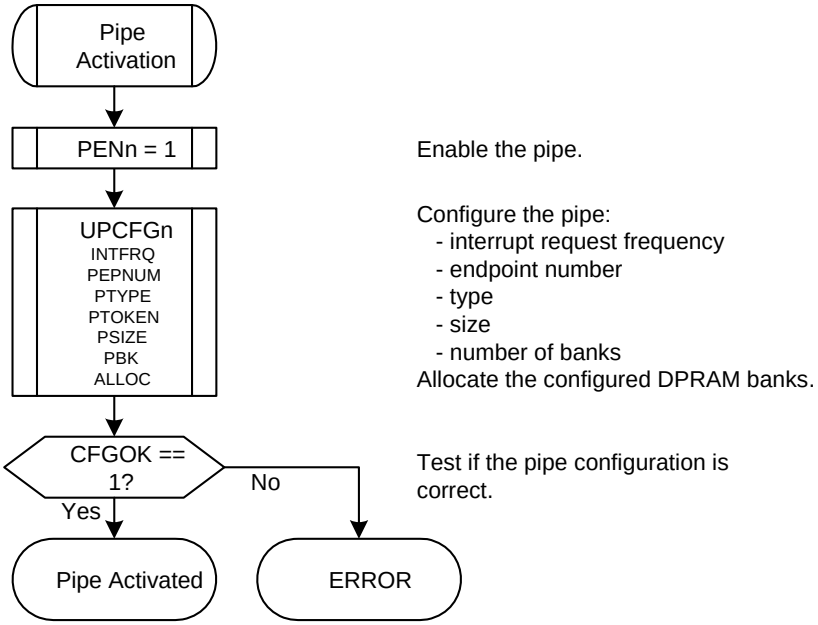
In the end, the user has to write a zero to the PRSTn bit to complete the reset operation and to start using the FIFO.

27.7.3.6 *Pipe activation*

The pipe is maintained inactive and reset (see Section 27.7.3.5 for more details) as long as it is disabled (PENn is zero). The Data Toggle Sequence field (DTSEQ) is also reset.

The algorithm represented on Figure 27-23 on page 649 must be followed in order to activate a pipe.

Figure 27-23. Pipe Activation Algorithm



As long as the pipe is not correctly configured (UPSTAn.CFGOK is zero), the controller can not send packets to the device through this pipe.

The UPSTAn.CFGOK bit is set only if the configured size and number of banks are correct compared to their maximal allowed values for the pipe (see Table 27-1 on page 620) and to the maximal FIFO size (i.e. the DPRAM size).

See Section 27.7.1.6 for more details about DPRAM management.

Once the pipe is correctly configured (UPSTAn.CFGOK is zero), only the PTOKEN and INTFRQ fields can be written by software. INTFRQ is meaningless for non-interrupt pipes.

When starting an enumeration, the user gets the device descriptor by sending a GET_DESCRIPTOR USB request. This descriptor contains the maximal packet size of the device default control endpoint (bMaxPacketSize0) and the user re-configures the size of the default control pipe with this size parameter.

27.7.3.7 *Address setup*

Once the device has answered the first host requests with the default device address 0, the host assigns a new address to the device. The host controller has to send an USB reset to the device and to send a SET_ADDRESS(addr) SETUP request with the new address to be used by the device. Once this SETUP transaction is over, the user writes the new address into the USB Host Address for Pipe n field in the USB Host Device Address register (UHADDR.UHADDRPn). All following requests, on all pipes, will be performed using this new address.

When the host controller sends an USB reset, the UHADDRPn field is reset by hardware and the following host requests will be performed using the default device address 0.

27.7.3.8 *Remote wake-up*

The controller host mode enters the Suspend state when the UHCON.SOFE bit is written to zero. No more “Start of Frame” is sent on the USB bus and the USB device enters the Suspend state 3ms later.

The device awakes the host by sending an Upstream Resume (Remote Wake-Up feature). When the host controller detects a non-idle state on the USB bus, it set the Host Wake-Up interrupt (HWUPI) bit in UHINT. If the non-idle bus state corresponds to an Upstream Resume (K state), the Upstream Resume Received Interrupt (RXRSMI) bit in UHINT is set. The user has to generate a Downstream Resume within 1ms and for at least 20ms by writing a one to the Send USB Resume (RESUME) bit in UHCON. It is mandatory to write a one to UHCON.SOFE before writing a one to UHCON.RESUME to enter the Ready state, else UHCON.RESUME will have no effect.

27.7.3.9 *Management of control pipes*

A control transaction is composed of three stages:

- SETUP
- Data (IN or OUT)
- Status (OUT or IN)

The user has to change the pipe token according to each stage.

For the control pipe, and only for it, each token is assigned a specific initial data toggle sequence:

- SETUP: Data0
- IN: Data1
- OUT: Data1

27.7.3.10 *Management of IN pipes*

IN packets are sent by the USB device controller upon IN requests from the host. All the data can be read which acknowledges or not the bank when it is empty.

The pipe must be configured first.

When the host requires data from the device, the user has to select beforehand the IN request mode with the IN Request Mode bit in the Pipe n IN Request register (UPINRQn.INMODE):

- When INMODE is written to zero, the USBB will perform (INRQ + 1) IN requests before freezing the pipe.
- When INMODE is written to one, the USBB will perform IN requests endlessly when the pipe is not frozen by the user.

The generation of IN requests starts when the pipe is unfrozen (the Pipe Freeze (PFREEZE) field in UPCONn is zero).

The Received IN Data Interrupt (RXINI) bit in UPSTAn is set at the same time as the FIFO Control (FIFOCON) bit in UPCONn when the current bank is full. This triggers a PnINT interrupt if the Received IN Data Interrupt Enable (RXINE) bit in UPCONn is one.

RXINI shall be cleared by software (by writing a one to the Received IN Data Interrupt Clear bit in the Pipe n Control Clear register(UPCONnCLR.RXINIC)) to acknowledge the interrupt, what has no effect on the pipe FIFO.

The user then reads from the FIFO and clears the FIFOCON bit (by writing a one to the FIFO Control Clear (FIFOCONC) bit in UPCONnCLR) to free the bank. If the IN pipe is composed of multiple banks, this also switches to the next bank. The RXINI and FIFOCON bits are updated in accordance with the status of the next bank.

RXINI shall always be cleared before clearing FIFOCON.

The Read/Write Allowed (RWALL) bit in UPSTAn is set when the current bank is not empty, i.e., the software can read further data from the FIFO.

Figure 27-24. Example of an IN Pipe with 1 Data Bank

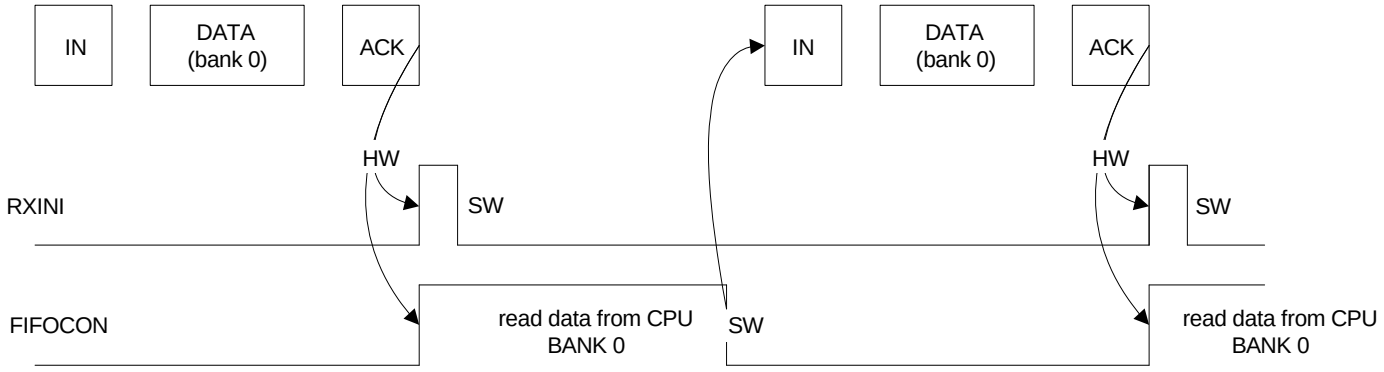
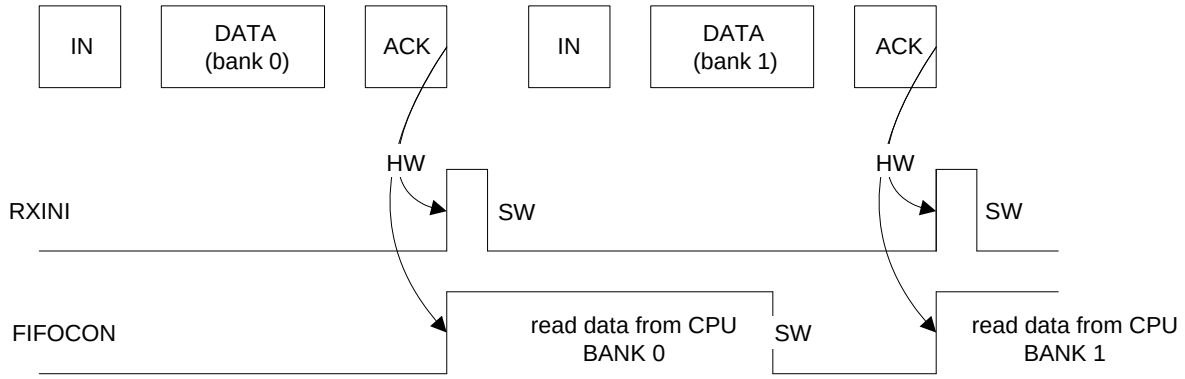


Figure 27-25. Example of an IN Pipe with 2 Data Banks



27.7.3.11 Management of OUT pipes

OUT packets are sent by the host. All the data can be written which acknowledges or not the bank when it is full.

The pipe must be configured and unfrozen first.

The Transmitted OUT Data Interrupt (TXOUTI) bit in UPSTAn is set at the same time as FIFOCON when the current bank is free. This triggers a PnINT interrupt if the Transmitted OUT Data Interrupt Enable (TXOUTE) bit in UPCONn is one.

TXOUTI shall be cleared by software (by writing a one to the Transmitted OUT Data Interrupt Clear (TXOUTIC) bit in UPCONnCLR) to acknowledge the interrupt, what has no effect on the pipe FIFO.

The user then writes into the FIFO and clears the FIFOCON bit to allow the USBB to send the data. If the OUT pipe is composed of multiple banks, this also switches to the next bank. The TXOUTI and FIFOCON bits are updated in accordance with the status of the next bank.

TXOUTI shall always be cleared before clearing FIFOCON.

The UPSTAn.RWALL bit is set when the current bank is not full, i.e., the software can write further data into the FIFO.

Note that if the user decides to switch to the Suspend state (by writing a zero to the UHCON.SOFE bit) while a bank is ready to be sent, the USBB automatically exits this state and the bank is sent.

Note that in High-Speed operating mode, the host controller automatically manages the PING protocol to maximize the USB bandwidth. The user can tune the PING protocol by handling the Ping Enable (PINGEN) bit and the bInterval Parameter for the Bulk-Out/Ping Transaction (BINTERVALL) field in UPCFGn. See the Section 27.8.3.13 for more details.

Figure 27-26. Example of an OUT Pipe with one Data Bank

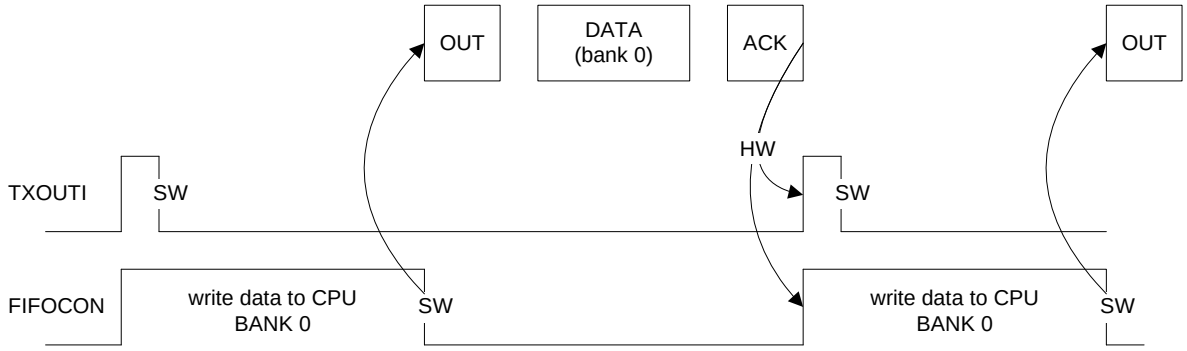


Figure 27-27. Example of an OUT Pipe with two Data Banks and no Bank Switching Delay

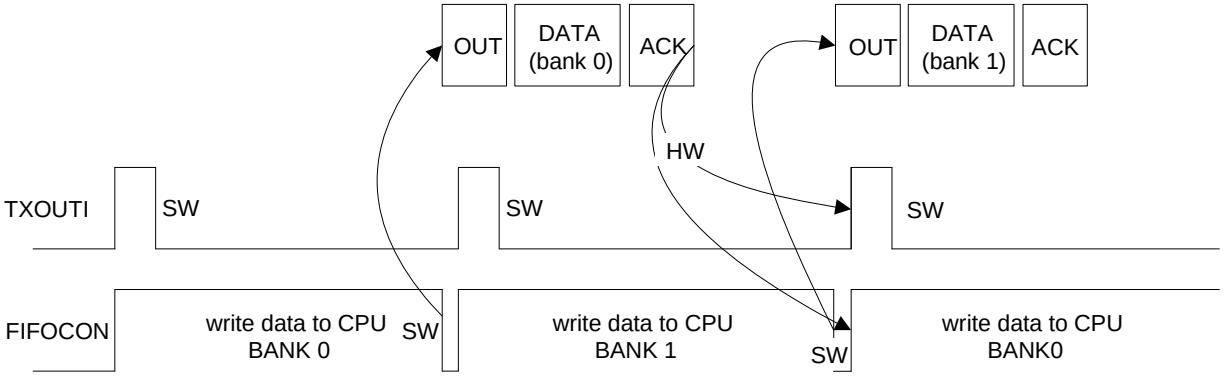
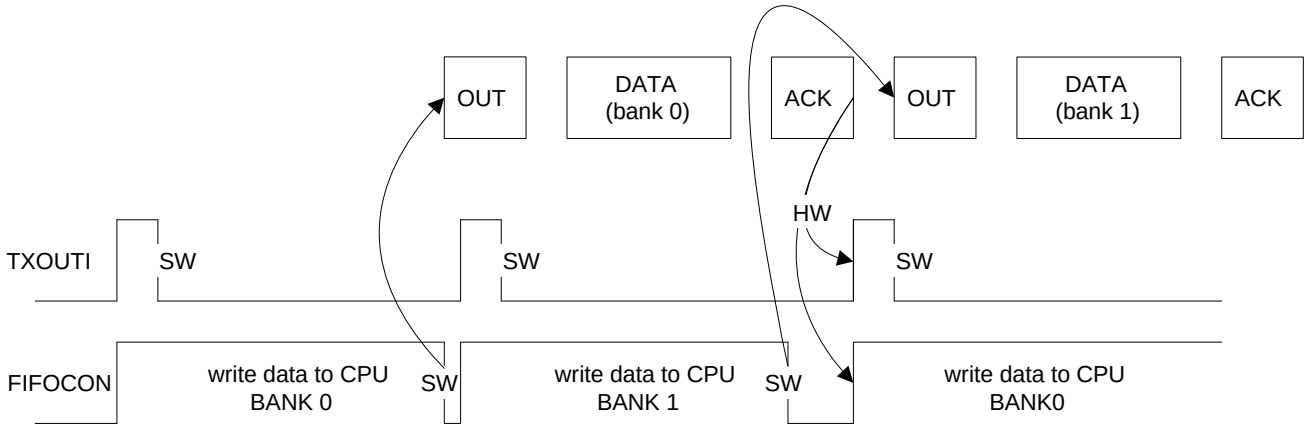


Figure 27-28. Example of an OUT Pipe with two Data Banks and a Bank Switching Delay



27.7.3.12 CRC error

This error exists only for isochronous IN pipes. It set the CRC Error Interrupt (CRCERRI) bit, what triggers a PnINT interrupt if then the CRC Error Interrupt Enable (CRCERRE) bit in UPCONn is one.

A CRC error can occur during IN stage if the USBB detects a corrupted received packet. The IN packet is stored in the bank as if no CRC error had occurred (RXINI is set).

27.7.3.13 Interrupts

See the structure of the USB host interrupt system on Figure 27-6 on page 628.

There are two kinds of host interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

•Global interrupts

The processing host global interrupts are:

- The Device Connection Interrupt (DCONNI)
- The Device Disconnection Interrupt (DDISCI)
- The USB Reset Sent Interrupt (RSTI)
- The Downstream Resume Sent Interrupt (RSMEDI)
- The Upstream Resume Received Interrupt (RXRSMI)
- The Host Start of Frame Interrupt (HSOFI)
- The Host Wake-Up Interrupt (HWUPI)
- The Pipe n Interrupt (PnINT)
- The DMA Channel n Interrupt (DMAnINT)

There is no exception host global interrupt.

•Pipe interrupts

The processing host pipe interrupts are:

- The Received IN Data Interrupt (RXINI)

- The Transmitted OUT Data Interrupt (TXOUTI)
- The Transmitted SETUP Interrupt (TXSTPI)
- The Short Packet Interrupt (SHORTPACKETI)
- The Number of Busy Banks (NBUSYBK) interrupt

The exception host pipe interrupts are:

- The Underflow Interrupt (UNDERFI)
- The Pipe Error Interrupt (PERRI)
- The NAKed Interrupt (NAKEDI)
- The Overflow Interrupt (OVERFI)
- The Received STALLed Interrupt (RXSTALLDI)
- The CRC Error Interrupt (CRCERRI)

•*DMA interrupts*

The processing host DMA interrupts are:

- The End of USB Transfer Status (EOTSTA) interrupt
- The End of Channel Buffer Status (EOCHBUFFSTA) interrupt
- The Descriptor Loaded Status (DESCLDSTA) interrupt

There is no exception host DMA interrupt.

27.7.4 USB DMA Operation

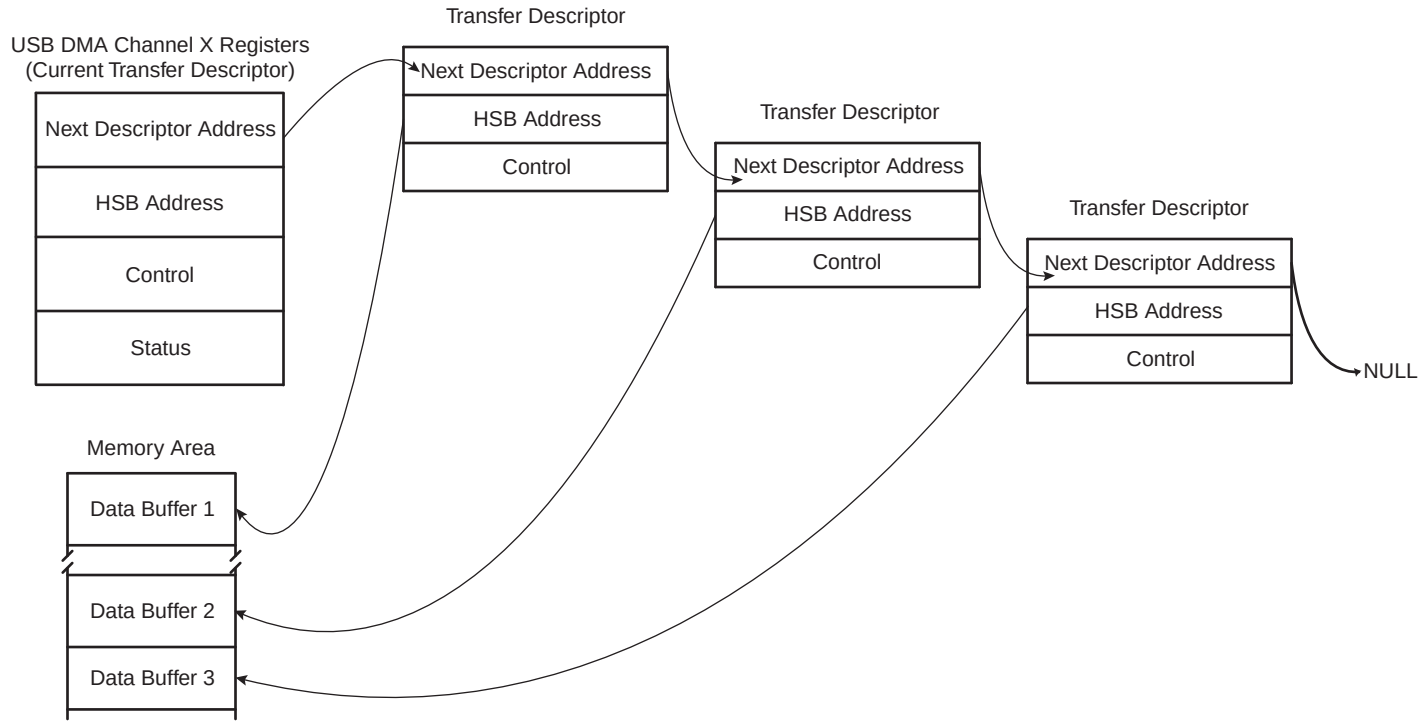
USB packets of any length may be transferred when required by the USBB. These transfers always feature sequential addressing. These two characteristics mean that in case of high USBB throughput, both HSB ports will benefit from “incrementing burst of unspecified length” since the average access latency of HSB slaves can then be reduced.

The DMA uses word “incrementing burst of unspecified length” of up to 256 beats for both data transfers and channel descriptor loading. A burst may last on the HSB busses for the duration of a whole USB packet transfer, unless otherwise broken by the HSB arbitration or the HSB 1 kbyte boundary crossing.

Packet data HSB bursts may be locked on a DMA buffer basis for drastic overall HSB bus bandwidth performance boost with paged memories. This is because these memories row (or bank) changes, which are very clock-cycle consuming, will then likely not occur or occur once instead of dozens of times during a single big USB packet DMA transfer in case other HSB masters address the memory. This means up to 128 words single cycle unbroken HSB bursts for bulk pipes/endpoints and 256 words single cycle unbroken bursts for isochronous pipes/endpoints. This maximal burst length is then controlled by the lowest programmed USB pipe/endpoint size (PSIZE/EPSIZE) and the Channel Byte Length (CHBYTELENGTH) field in the Device DMA Channel n Control (UDDMAnCONTROL) register.

The USBB average throughput may be up to nearly 1.5 Mbps. Its average access latency decreases as burst length increases due to the zero wait-state side effect of unchanged pipe/endpoint. Word access allows reducing the HSB bandwidth required for the USB by four compared to native byte access. If at least 0 wait-state word burst capability is also provided by the other DMA HSB bus slaves, each of both DMA HSB busses need less than 1.1% bandwidth allocation for full USB bandwidth usage at 33 MHz, and less than 0.6% at 66 MHz.

Figure 27-29. Example of DMA Chained List



27.8 User Interface

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Device General Control Register	UDCON	Read/Write	0x00000100
0x0004	Device Global Interrupt Register	UDINT	Read-Only	0x00000000
0x0008	Device Global Interrupt Clear Register	UDINTCLR	Write-Only	0x00000000
0x000C	Device Global Interrupt Set Register	UDINTSET	Write-Only	0x00000000
0x0010	Device Global Interrupt Enable Register	UDINTE	Read-Only	0x00000000
0x0014	Device Global Interrupt Enable Clear Register	UDINTECLR	Write-Only	0x00000000
0x0018	Device Global Interrupt Enable Set Register	UDINTESET	Write-Only	0x00000000
0x001C	Endpoint Enable/Reset Register	UERST	Read/Write	0x00000000
0x0020	Device Frame Number Register	UDFNUM	Read-Only	0x00000000
0x0100	Endpoint 0 Configuration Register	UECFG0	Read/Write	0x00002000
0x0104	Endpoint 1 Configuration Register	UECFG1	Read/Write	0x00002000
0x0108	Endpoint 2 Configuration Register	UECFG2	Read/Write	0x00002000
0x010C	Endpoint 3 Configuration Register	UECFG3	Read/Write	0x00002000
0x0110	Endpoint 4 Configuration Register	UECFG4	Read/Write	0x00002000
0x0114	Endpoint 5 Configuration Register	UECFG5	Read/Write	0x00002000
0x0118	Endpoint 6 Configuration Register	UECFG6	Read/Write	0x00002000
0x011C	Endpoint 7Configuration Register	UECFG7	Read/Write	0x00002000
0x0120	Endpoint 8Configuration Register	UECFG8	Read/Write	0x00002000
0x0130	Endpoint 0 Status Register	UESTA0	Read-Only	0x00000100
0x0134	Endpoint 1 Status Register	UESTA1	Read-Only	0x00000100
0x0138	Endpoint 2 Status Register	UESTA2	Read-Only	0x00000100
0x013C	Endpoint 3 Status Register	UESTA3	Read-Only	0x00000100
0x0140	Endpoint 4 Status Register	UESTA4	Read-Only	0x00000100
0x0144	Endpoint 5 Status Register	UESTA5	Read-Only	0x00000100
0x0148	Endpoint 6 Status Register	UESTA6	Read-Only	0x00000100
0x014C	Endpoint 7Status Register	UESTA7	Read-Only	0x00000100
0x0150	Endpoint 8Status Register	UESTA8	Read-Only	0x00000100
0x0160	Endpoint 0 Status Clear Register	UESTA0CLR	Write-Only	0x00000000
0x0164	Endpoint 1 Status Clear Register	UESTA1CLR	Write-Only	0x00000000
0x0168	Endpoint 2 Status Clear Register	UESTA2CLR	Write-Only	0x00000000
0x016C	Endpoint 3 Status Clear Register	UESTA3CLR	Write-Only	0x00000000
0x0170	Endpoint 4 Status Clear Register	UESTA4CLR	Write-Only	0x00000000
0x0174	Endpoint 5 Status Clear Register	UESTA5CLR	Write-Only	0x00000000
0x0178	Endpoint 6 Status Clear Register	UESTA6CLR	Write-Only	0x00000000
0x017C	Endpoint 7 Status Clear Register	UESTA7CLR	Write-Only	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0180	Endpoint 8 Status Clear Register	UESTA8CLR	Write-Only	0x00000000
0x0190	Endpoint 0 Status Set Register	UESTA0SET	Write-Only	0x00000000
0x0194	Endpoint 1 Status Set Register	UESTA1SET	Write-Only	0x00000000
0x0198	Endpoint 2 Status Set Register	UESTA2SET	Write-Only	0x00000000
0x019C	Endpoint 3 Status Set Register	UESTA3SET	Write-Only	0x00000000
0x01A0	Endpoint 4 Status Set Register	UESTA4SET	Write-Only	0x00000000
0x01A4	Endpoint 5 Status Set Register	UESTA5SET	Write-Only	0x00000000
0x01A8	Endpoint 6 Status Set Register	UESTA6SET	Write-Only	0x00000000
0x01AC	Endpoint 7 Status Set Register	UESTA7SET	Write-Only	0x00000000
0x01B0	Endpoint 8 Status Set Register	UESTA8SET	Write-Only	0x00000000
0x01C0	Endpoint 0 Control Register	UECON0	Read-Only	0x00000000
0x01C4	Endpoint 1 Control Register	UECON1	Read-Only	0x00000000
0x01C8	Endpoint 2 Control Register	UECON2	Read-Only	0x00000000
0x01CC	Endpoint 3 Control Register	UECON3	Read-Only	0x00000000
0x01D0	Endpoint 4 Control Register	UECON4	Read-Only	0x00000000
0x01D4	Endpoint 5 Control Register	UECON5	Read-Only	0x00000000
0x01D8	Endpoint 6 Control Register	UECON6	Read-Only	0x00000000
0x01DC	Endpoint 7 Control Register	UECON7	Read-Only	0x00000000
0x01E0	Endpoint 8 Control Register	UECON8	Read-Only	0x00000000
0x01F0	Endpoint 0 Control Set Register	UECON0SET	Write-Only	0x00000000
0x01F4	Endpoint 1 Control Set Register	UECON1SET	Write-Only	0x00000000
0x01F8	Endpoint 2 Control Set Register	UECON2SET	Write-Only	0x00000000
0x01FC	Endpoint 3 Control Set Register	UECON3SET	Write-Only	0x00000000
0x0200	Endpoint 4 Control Set Register	UECON4SET	Write-Only	0x00000000
0x0204	Endpoint 5 Control Set Register	UECON5SET	Write-Only	0x00000000
0x0208	Endpoint 6 Control Set Register	UECON6SET	Write-Only	0x00000000
0x020C	Endpoint 7 Control Set Register	UECON7SET	Write-Only	0x00000000
0x0210	Endpoint 8 Control Set Register	UECON8SET	Write-Only	0x00000000
0x0220	Endpoint 0 Control Clear Register	UECON0CLR	Write-Only	0x00000000
0x0224	Endpoint 1 Control Clear Register	UECON1CLR	Write-Only	0x00000000
0x0228	Endpoint 2 Control Clear Register	UECON2CLR	Write-Only	0x00000000
0x022C	Endpoint 3 Control Clear Register	UECON3CLR	Write-Only	0x00000000
0x0230	Endpoint 4 Control Clear Register	UECON4CLR	Write-Only	0x00000000
0x0234	Endpoint 5 Control Clear Register	UECON5CLR	Write-Only	0x00000000
0x0238	Endpoint 6 Control Clear Register	UECON6CLR	Write-Only	0x00000000
0x023C	Endpoint 7 Control Clear Register	UECON7CLR	Write-Only	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0240	Endpoint 8 Control Clear Register	UECON8CLR	Write-Only	0x00000000
0x0310	Device DMA Channel 1 Next Descriptor Address Register	UDDMA1 NEXTDESC	Read/Write	0x00000000
0x0314	Device DMA Channel 1 HSB Address Register	UDDMA1 ADDR	Read/Write	0x00000000
0x0318	Device DMA Channel 1 Control Register	UDDMA1 CONTROL	Read/Write	0x00000000
0x031C	Device DMA Channel 1 Status Register	UDDMA1 STATUS	Read/Write	0x00000000
0x0320	Device DMA Channel 2 Next Descriptor Address Register	UDDMA2 NEXTDESC	Read/Write	0x00000000
0x0324	Device DMA Channel 2 HSB Address Register	UDDMA2 ADDR	Read/Write	0x00000000
0x0328	Device DMA Channel 2 Control Register	UDDMA2 CONTROL	Read/Write	0x00000000
0x032C	Device DMA Channel 2 Status Register	UDDMA2 STATUS	Read/Write	0x00000000
0x0330	Device DMA Channel 3 Next Descriptor Address Register	UDDMA3 NEXTDESC	Read/Write	0x00000000
0x0334	Device DMA Channel 3 HSB Address Register	UDDMA3 ADDR	Read/Write	0x00000000
0x0338	Device DMA Channel 3 Control Register	UDDMA3 CONTROL	Read/Write	0x00000000
0x033C	Device DMA Channel 3 Status Register	UDDMA3 STATUS	Read/Write	0x00000000
0x0340	Device DMA Channel 4 Next Descriptor Address Register	UDDMA4 NEXTDESC	Read/Write	0x00000000
0x0344	Device DMA Channel 4 HSB Address Register	UDDMA4 ADDR	Read/Write	0x00000000
0x0348	Device DMA Channel 4 Control Register	UDDMA4 CONTROL	Read/Write	0x00000000
0x034C	Device DMA Channel 4 Status Register	UDDMA4 STATUS	Read/Write	0x00000000
0x0350	Device DMA Channel 5 Next Descriptor Address Register	UDDMA5 NEXTDESC	Read/Write	0x00000000
0x0354	Device DMA Channel 5 HSB Address Register	UDDMA5 ADDR	Read/Write	0x00000000
0x0358	Device DMA Channel 5 Control Register	UDDMA5 CONTROL	Read/Write	0x00000000
0x035C	Device DMA Channel 5 Status Register	UDDMA5 STATUS	Read/Write	0x00000000
0x0360	Device DMA Channel 6 Next Descriptor Address Register	UDDMA6 NEXTDESC	Read/Write	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0364	Device DMA Channel 6 HSB Address Register	UDDMA6 ADDR	Read/Write	0x00000000
0x0368	Device DMA Channel 6 Control Register	UDDMA6 CONTROL	Read/Write	0x00000000
0x036C	Device DMA Channel 6 Status Register	UDDMA6 STATUS	Read/Write	0x00000000
0x0370	Device DMA Channel 7 Next Descriptor Address Register	UDDMA7 NEXTDESC	Read/Write	0x00000000
0x0374	Device DMA Channel 7 HSB Address Register	UDDMA7 ADDR	Read/Write	0x00000000
0x0378	Device DMA Channel 7 Control Register	UDDMA7 CONTROL	Read/Write	0x00000000
0x037C	Device DMA Channel 7Status Register	UDDMA7 STATUS	Read/Write	0x00000000
0x0380	Device DMA Channel 8 Next Descriptor Address Register	UDDMA8 NEXTDESC	Read/Write	0x00000000
0x0384	Device DMA Channel 8 HSB Address Register	UDDMA8 ADDR	Read/Write	0x00000000
0x0388	Device DMA Channel 8 Control Register	UDDMA8 CONTROL	Read/Write	0x00000000
0x038C	Device DMA Channel 8 Status Register	UDDMA8 STATUS	Read/Write	0x00000000
0x0400	Host General Control Register	UHCON	Read/Write	0x00000000
0x0404	Host Global Interrupt Register	UHINT	Read-Only	0x00000000
0x0408	Host Global Interrupt Clear Register	UHINTCLR	Write-Only	0x00000000
0x040C	Host Global Interrupt Set Register	UHINTSET	Write-Only	0x00000000
0x0410	Host Global Interrupt Enable Register	UHINTE	Read-Only	0x00000000
0x0414	Host Global Interrupt Enable Clear Register	UHINTECLR	Write-Only	0x00000000
0x0418	Host Global Interrupt Enable Set Register	UHINTESET	Write-Only	0x00000000
0x0041C	Pipe Enable/Reset Register	UPRST	Read/Write	0x00000000
0x0420	Host Frame Number Register	UHFNUM	Read/Write	0x00000000
0x0424	Host Address 1 Register	UHADDR1	Read/Write	0x00000000
0x0428	Host Address 2 Register	UHADDR2	Read/Write	0x00000000
0x042C	Host Address 3 Register	UHADDR3	Read/Write	0x00000000
0x0500	Pipe 0 Configuration Register	UPCFG0	Read/Write	0x00000000
0x0504	Pipe 1 Configuration Register	UPCFG1	Read/Write	0x00000000
0x0508	Pipe 2 Configuration Register	UPCFG2	Read/Write	0x00000000
0x050C	Pipe 3 Configuration Register	UPCFG3	Read/Write	0x00000000
0x0510	Pipe 4 Configuration Register	UPCFG4	Read/Write	0x00000000
0x0514	Pipe 5 Configuration Register	UPCFG5	Read/Write	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0518	Pipe 6 Configuration Register	UPCFG6	Read/Write	0x00000000
0x051C	Pipe 7 Configuration Register	UPCFG7	Read/Write	0x00000000
0x0520	Pipe 8 Configuration Register	UPCFG8	Read/Write	0x00000000
0x0530	Pipe 0 Status Register	UPSTA0	Read-Only	0x00000000
0x0534	Pipe 1 Status Register	UPSTA1	Read-Only	0x00000000
0x0538	Pipe 2 Status Register	UPSTA2	Read-Only	0x00000000
0x053C	Pipe 3 Status Register	UPSTA3	Read-Only	0x00000000
0x0540	Pipe 4 Status Register	UPSTA4	Read-Only	0x00000000
0x0544	Pipe 5 Status Register	UPSTA5	Read-Only	0x00000000
0x0548	Pipe 6 Status Register	UPSTA6	Read-Only	0x00000000
0x054C	Pipe 7Status Register	UPSTA7	Read-Only	0x00000000
0x0550	Pipe 8 Status Register	UPSTA8	Read-Only	0x00000000
0x0560	Pipe 0 Status Clear Register	UPSTA0CLR	Write-Only	0x00000000
0x0564	Pipe 1 Status Clear Register	UPSTA1CLR	Write-Only	0x00000000
0x0568	Pipe 2 Status Clear Register	UPSTA2CLR	Write-Only	0x00000000
0x056C	Pipe 3 Status Clear Register	UPSTA3CLR	Write-Only	0x00000000
0x0570	Pipe 4 Status Clear Register	UPSTA4CLR	Write-Only	0x00000000
0x0574	Pipe 5 Status Clear Register	UPSTA5CLR	Write-Only	0x00000000
0x0578	Pipe 6 Status Clear Register	UPSTA6CLR	Write-Only	0x00000000
0x057C	Pipe 7 Status Clear Register	UPSTA7CLR	Write-Only	0x00000000
0x0580	Pipe 8 Status Clear Register	UPSTA8CLR	Write-Only	0x00000000
0x0590	Pipe 0 Status Set Register	UPSTA0SET	Write-Only	0x00000000
0x0594	Pipe 1 Status Set Register	UPSTA1SET	Write-Only	0x00000000
0x0598	Pipe 2 Status Set Register	UPSTA2SET	Write-Only	0x00000000
0x059C	Pipe 3 Status Set Register	UPSTA3SET	Write-Only	0x00000000
0x05A0	Pipe 4 Status Set Register	UPSTA4SET	Write-Only	0x00000000
0x05A4	Pipe 5 Status Set Register	UPSTA5SET	Write-Only	0x00000000
0x05A8	Pipe 6 Status Set Register	UPSTA6SET	Write-Only	0x00000000
0x05AC	Pipe 7 Status Set Register	UPSTA7SET	Write-Only	0x00000000
0x05B0	Pipe 8 Status Set Register	UPSTA8SET	Write-Only	0x00000000
0x05C0	Pipe 0 Control Register	UPCON0	Read-Only	0x00000000
0x05C4	Pipe 1 Control Register	UPCON1	Read-Only	0x00000000
0x05C8	Pipe 2 Control Register	UPCON2	Read-Only	0x00000000
0x05CC	Pipe 3 Control Register	UPCON3	Read-Only	0x00000000
0x05D0	Pipe 4 Control Register	UPCON4	Read-Only	0x00000000
0x05D4	Pipe 5 Control Register	UPCON5	Read-Only	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x05D8	Pipe 6 Control Register	UPCON6	Read-Only	0x00000000
0x05DC	Pipe 7 Control Register	UPCON7	Read-Only	0x00000000
0x05E0	Pipe 8 Control Register	UPCON8	Read-Only	0x00000000
0x05F0	Pipe 0 Control Set Register	UPCON0SET	Write-Only	0x00000000
0x05F4	Pipe 1 Control Set Register	UPCON1SET	Write-Only	0x00000000
0x05F8	Pipe 2 Control Set Register	UPCON2SET	Write-Only	0x00000000
0x05FC	Pipe 3 Control Set Register	UPCON3SET	Write-Only	0x00000000
0x0600	Pipe 4 Control Set Register	UPCON4SET	Write-Only	0x00000000
0x0604	Pipe 5 Control Set Register	UPCON5SET	Write-Only	0x00000000
0x0608	Pipe 6 Control Set Register	UPCON6SET	Write-Only	0x00000000
0x060C	Pipe 7 Control Set Register	UPCON7SET	Write-Only	0x00000000
0x0610	Pipe 8 Control Set Register	UPCON8SET	Write-Only	0x00000000
0x0620	Pipe 0 Control Clear Register	UPCON0CLR	Write-Only	0x00000000
0x0624	Pipe 1 Control Clear Register	UPCON1CLR	Write-Only	0x00000000
0x0628	Pipe 2 Control Clear Register	UPCON2CLR	Write-Only	0x00000000
0x062C	Pipe 3 Control Clear Register	UPCON3CLR	Write-Only	0x00000000
0x0630	Pipe 4 Control Clear Register	UPCON4CLR	Write-Only	0x00000000
0x0634	Pipe 5 Control Clear Register	UPCON5CLR	Write-Only	0x00000000
0x0638	Pipe 6 Control Clear Register	UPCON6CLR	Write-Only	0x00000000
0x063C	Pipe 7 Control Clear Register	UPCON7CLR	Write-Only	0x00000000
0x0640	Pipe 8 Control Clear Register	UPCON8CLR	Write-Only	0x00000000
0x0650	Pipe 0 IN Request Register	UPINRQ0	Read/Write	0x00000000
0x0654	Pipe 1 IN Request Register	UPINRQ1	Read/Write	0x00000000
0x0658	Pipe 2 IN Request Register	UPINRQ2	Read/Write	0x00000000
0x065C	Pipe 3 IN Request Register	UPINRQ3	Read/Write	0x00000000
0x0660	Pipe 4 IN Request Register	UPINRQ4	Read/Write	0x00000000
0x0664	Pipe 5 IN Request Register	UPINRQ5	Read/Write	0x00000000
0x0668	Pipe 6 IN Request Register	UPINRQ6	Read/Write	0x00000000
0x066C	Pipe 7 IN Request Register	UPINRQ7	Read/Write	0x00000000
0x0670	Pipe 8 IN Request Register	UPINRQ8	Read/Write	0x00000000
0x0680	Pipe 0 Error Register	UPERR0	Read/Write	0x00000000
0x0684	Pipe 1 Error Register	UPERR1	Read/Write	0x00000000
0x0688	Pipe 2 Error Register	UPERR2	Read/Write	0x00000000
0x068C	Pipe 3 Error Register	UPERR3	Read/Write	0x00000000
0x0690	Pipe 4 Error Register	UPERR4	Read/Write	0x00000000
0x0694	Pipe 5 Error Register	UPERR5	Read/Write	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0698	Pipe 6 Error Register	UPERR6	Read/Write	0x00000000
0x069C	Pipe 7 Error Register	UPERR7	Read/Write	0x00000000
0x0670	Pipe 8 Error Register	UPERR8	Read/Write	0x00000000
0x0710	Host DMA Channel 1 Next Descriptor Address Register	UHDMA1 NEXTDESC	Read/Write	0x00000000
0x0714	Host DMA Channel 1 HSB Address Register	UHDMA1 ADDR	Read/Write	0x00000000
0x0718	Host DMA Channel 1 Control Register	UHDMA1 CONTROL	Read/Write	0x00000000
0x071C	Host DMA Channel 1 Status Register	UHDMA1 STATUS	Read/Write	0x00000000
0x0720	Host DMA Channel 2 Next Descriptor Address Register	UHDMA2 NEXTDESC	Read/Write	0x00000000
0x0724	Host DMA Channel 2 HSB Address Register	UHDMA2 ADDR	Read/Write	0x00000000
0x0728	Host DMA Channel 2 Control Register	UHDMA2 CONTROL	Read/Write	0x00000000
0x072C	Host DMA Channel 2 Status Register	UHDMA2 STATUS	Read/Write	0x00000000
0x0730	Host DMA Channel 3 Next Descriptor Address Register	UHDMA3 NEXTDESC	Read/Write	0x00000000
0x0734	Host DMA Channel 3 HSB Address Register	UHDMA3 ADDR	Read/Write	0x00000000
0x0738	Host DMA Channel 3 Control Register	UHDMA3 CONTROL	Read/Write	0x00000000
0x073C	Host DMA Channel 3Status Register	UHDMA3 STATUS	Read/Write	0x00000000
0x0740	Host DMA Channel 4 Next Descriptor Address Register	UHDMA4 NEXTDESC	Read/Write	0x00000000
0x0744	Host DMA Channel 4 HSB Address Register	UHDMA4 ADDR	Read/Write	0x00000000
0x0748	Host DMA Channel 4 Control Register	UHDMA4 CONTROL	Read/Write	0x00000000
0x074C	Host DMA Channel 4 Status Register	UHDMA4 STATUS	Read/Write	0x00000000
0x0750	Host DMA Channel 5 Next Descriptor Address Register	UHDMA5 NEXTDESC	Read/Write	0x00000000
0x0754	Host DMA Channel 5 HSB Address Register	UHDMA5 ADDR	Read/Write	0x00000000
0x0758	Host DMA Channel 5 Control Register	UHDMA5 CONTROL	Read/Write	0x00000000
0x075C	Host DMA Channel 5 Status Register	UHDMA5 STATUS	Read/Write	0x00000000

Table 27-5. USBB Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0760	Host DMA Channel 6 Next Descriptor Address Register	UHDMA6 NEXTDESC	Read/Write	0x00000000
0x0764	Host DMA Channel 6 HSB Address Register	UHDMA6 ADDR	Read/Write	0x00000000
0x0768	Host DMA Channel 6 Control Register	UHDMA6 CONTROL	Read/Write	0x00000000
0x076C	Host DMA Channel 6 Status Register	UHDMA6 STATUS	Read/Write	0x00000000
0x0770	Host DMA Channel 7 Next Descriptor Address Register	UHDMA7 NEXTDESC	Read/Write	0x00000000
0x0774	Host DMA Channel 7 HSB Address Register	UHDMA7 ADDR	Read/Write	0x00000000
0x0778	Host DMA Channel 7 Control Register	UHDMA7 CONTROL	Read/Write	0x00000000
0x077C	Host DMA Channel 7 Status Register	UHDMA7 STATUS	Read/Write	0x00000000
0x0780	Host DMA Channel 8 Next Descriptor Address Register	UHDMA8 NEXTDESC	Read/Write	0x00000000
0x0784	Host DMA Channel 8 HSB Address Register	UHDMA8 ADDR	Read/Write	0x00000000
0x0788	Host DMA Channel 8 Control Register	UHDMA8 CONTROL	Read/Write	0x00000000
0x078C	Host DMA Channel 8 Status Register	UHDMA8 STATUS	Read/Write	0x00000000
0x0800	General Control Register	USBCON	Read/Write	0x03004000
0x0804	General Status Register	USBSTA	Read-Only	0x00000400
0x0808	General Status Clear Register	USBSTACLR	Write-Only	0x00000000
0x080C	General Status Set Register	USBSTASET	Write-Only	0x00000000
0x0818	IP Version Register	UVERS	Read-Only	-(1)
0x081C	IP Features Register	UFEATURES	Read-Only	-(1)
0x0820	IP PB Address Size Register	UADDRSIZE	Read-Only	-(1)
0x0824	IP Name Register 1	UNAME1	Read-Only	-(1)
0x0828	IP Name Register 2	UNAME2	Read-Only	-(1)
0x082C	USB Finite State Machine Status Register	USBFSM	Read-Only	0x00000009

Table 27-6. USB HSB Memory Map

Offset	Register	Name	Access	Reset Value
0x00000 - 0x0FFFC	Pipe/Endpoint 0 FIFO Data Register	USB FIFO0DATA	Read/Write	Undefined
0x10000 - 0x1FFFC	Pipe/Endpoint 1 FIFO Data Register	USB FIFO1DATA	Read/Write	Undefined
0x20000 - 0x2FFFC	Pipe/Endpoint 2 FIFO Data Register	USB FIFO2DATA	Read/Write	Undefined
0x30000 - 0x3FFFC	Pipe/Endpoint 3 FIFO Data Register	USB FIFO3DATA	Read/Write	Undefined
0x40000 - 0x4FFFC	Pipe/Endpoint 4 FIFO Data Register	USB FIFO4DATA	Read/Write	Undefined
0x50000 - 0x5FFFC	Pipe/Endpoint 5 FIFO Data Register	USB FIFO5DATA	Read/Write	Undefined
0x60000 - 0x6FFFC	Pipe/Endpoint 6 FIFO Data Register	USB FIFO6DATA	Read/Write	Undefined
0x70000 - 0x7FFFC	Pipe/Endpoint 7 FIFO Data Register	USB FIFO7DATA	Read/Write	Undefined
0x80000 - 0x8FFFC	Pipe/Endpoint 8 FIFO Data Register	USB FIFO8DATA	Read/Write	Undefined

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

27.8.1 USB General Registers

27.8.1.1 General Control Register

Name: USBCON

Access Type: Read/Write

Offset: 0x0800

Reset Value: 0x03004000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	UIMOD	UIDE
23	22	21	20	19	18	17	16
-	UNLOCK	TIMPAGE		-	-	TIMVALUE	
15	14	13	12	11	10	9	8
USBE	FRZCLK	VBUSPO	OTGPADE	HNPREQ	SRPREQ	SRPSEL	VBUSHWC
7	6	5	4	3	2	1	0
STOE	HNPERRE	ROLEEXE	BCERRE	VBERRE	SRPE	VBUSTE	IDTE

- **UIMOD: USBB Mode**
  This bit has no effect when UIDE is one (USB_ID input pin activated).
  0: The module is in USB host mode.
  1: The module is in USB device mode.
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBB (by writing a zero to the USBE bit) does not reset this bit.
- **UIDE: USB_ID Pin Enable**
  0: The USB mode (device/host) is selected from the UIMOD bit.
  1: The USB mode (device/host) is selected from the USB_ID input pin.
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBB (by writing a zero to the USBE bit) does not reset this bit.
- **UNLOCK: Timer Access Unlock**
  1: The TIMPAGE and TIMVALUE fields are unlocked.
  0: The TIMPAGE and TIMVALUE fields are locked.
  The TIMPAGE and TIMVALUE fields can always be read, whatever the value of UNLOCK.
- **TIMPAGE: Timer Page**
  This field contains the page value to access a special timer register.
- **TIMVALUE: Timer Value**
  This field selects the timer value that is written to the special time register selected by TIMPAGE. See Section 27.7.1.8 for details.
- **USBE: USBB Enable**
  Writing a zero to this bit will reset the USBB, disable the USB transceiver and, disable the USBB clock inputs. Unless explicitly stated, all registers then will become read-only and will be reset.
  1: The USBB is enabled.
  0: The USBB is disabled.
  This bit can be written even if FRZCLK is one.
- **FRZCLK: Freeze USB Clock**
  1: The clock input are disabled (the resume detection is still active).This reduces power consumption. Unless explicitly stated, all registers then become read-only.

0: The clock inputs are enabled.
This bit can be written even if USBE is zero. Disabling the USBB (by writing a zero to the USBE bit) does not reset this bit, but this freezes the clock inputs whatever its value.

- **VBUSPO: VBus Polarity**
  1: The USB_VBOF output signal is inverted (active low).
  0: The USB_VBOF output signal is in its default mode (active high).
  To be generic. May be useful to control an external VBus power module.
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBB (by writing a zero to the USBE bit) does not reset this bit.
- **OTGPADE: OTG Pad Enable**
  1: The OTG pad is enabled.
  0: The OTG pad is disabled.
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBB (by writing a zero to the USBE bit) does not reset this bit.
- **HNPREQ: HNP Request**
  When the controller is in device mode:
  Writing a one to this bit will initiate a HNP (Host Negociation Protocol).
  Writing a zero to this bit has no effect.
  This bit is cleared when the controller has initiated an HNP.

  When the controller is in host mode:
  Writing a one to this bit will accept a HNP.
  Writing a zero to this bit will reject a HNP.
- **SRPREQ: SRP Request**
  Writing a one to this bit will initiate an SRP when the controller is in device mode.
  Writing a zero to this bit has no effect.
  This bit is cleared when the controller has initiated an SRP.
- **SRPSEL: SRP Selection**
  1: VBus pulsing is selected as SRP method.
  0: Data line pulsing is selected as SRP method.
- **VBUSHWC: VBus Hardware Control**
  1: The hardware control over the USB_VBOF output pin is disabled.
  0: The hardware control over the USB_VBOF output pin is enabled. The USBB resets the USB_VBOF output pin when a VBUS problem occurs.
- **STOE: Suspend Time-Out Interrupt Enable**
  1: The Suspend Time-Out Interrupt (STOI) is enabled.
  0: The Suspend Time-Out Interrupt (STOI) is disabled.
- **HNPERRE: HNP Error Interrupt Enable**
  1: The HNP Error Interrupt (HNPERRI) is enabled.
  0: The HNP Error Interrupt (HNPERRI) is disabled.
- **ROLEEXE: Role Exchange Interrupt Enable**
  1: The Role Exchange Interrupt (ROLEEXI) is enabled.
  0: The Role Exchange Interrupt (ROLEEXI) is disabled.
- **BCERRE: B-Connection Error Interrupt Enable**
  1: The B-Connection Error Interrupt (BCERRI) is enabled.
  0: The B-Connection Error Interrupt (BCERRI) is disabled.
- **VBERRE: VBus Error Interrupt Enable**
  1: The VBus Error Interrupt (VBERRI) is enabled.
  0: The VBus Error Interrupt (VBERRI) is disabled.
- **SRPE: SRP Interrupt Enable**
  1: The SRP Interrupt (SRPI) is enabled.
  0: The SRP Interrupt (SRPI) is disabled.
- **VBUSTE: VBus Transition Interrupt Enable**
  1: The VBus Transition Interrupt (VBUSTI) is enabled.
  0: The VBus Transition Interrupt (VBUSTI) is disabled.
- **IDTE: ID Transition Interrupt Enable**
  1: The ID Transition interrupt (IDTI) is enabled.
  0: The ID Transition interrupt (IDTI) is disabled.

27.8.1.2 General Status Register

Register Name: USBSTA

Access Type: Read-Only

Offset: 0x0804

Reset Value: 0x00000400

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

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

15	14	13	12	11	10	9	8
-	CLKUSABLE	SPEED		VBUS	ID	VBUSRQ	-

7	6	5	4	3	2	1	0
STOI	HNPERRI	ROLEEXI	BCERRI	VBERRI	SRPI	VBUSTI	IDTI

- **CLKUSABLE: UTMI Clock Usable**
  This bit is set when the UTMI 30MHz is usable.
  This bit is cleared when the UTMI 30MHz is not usable.
- **SPEED: Speed Status**
  This field is set according to the controller speed mode. This field shall only be used in device mode.

SPEED		Speed Status
0	0	Full-Speed mode
1	0	Low-Speed mode
0	1	High-Speed mode
1	1	Reserved

- **VBUS: VBus Level**
  This bit is set when the VBus line level is high, even if USBE is zero.
  This bit is cleared when the VBus line level is low, even if USBE is zero.
  This bit can be used in device mode to monitor the USB bus connection state of the application.
- **ID: USB_ID Pin State**
  This bit is cleared when the USB_ID level is low, even if USBE is zero.
  This bit is set when the USB_ID level is high, event if USBE is zero.
- **VBUSRQ: VBus Request**
  This bit is set when the USBSTASET.VBUSRQS bit is written to one.
  This bit is cleared when the USBSTACLR.VBUSRQC bit is written to one or when a VBus error occurs and VBUSHWC is zero.
  1: The USB_VBOF output pin is driven high to enable the VBUS power supply generation.
  0: The USB_VBOF output pin is driven low to disable the VBUS power supply generation.
  This bit shall only be used in host mode.

- **STOI: Suspend Time-Out Interrupt**
  This bit is set when a time-out error (more than 200ms) has been detected after a suspend. This triggers a USB interrupt if STOE is one.
  This bit is cleared when the UBSTACLR.STOIC bit is written to one.
  This bit shall only be used in host mode.
- **HNPERRI: HNP Error Interrupt**
  This bit is set when an error has been detected during a HNP negotiation. This triggers a USB interrupt if HNPERRE is one.
  This bit is cleared when the UBSTACLR.HNPERRIC bit is written to one.
  This bit shall only be used in device mode.
- **ROLEEXI: Role Exchange Interrupt**
  This bit is set when the USBB has successfully switched its mode because of an HNP negotiation (host to device or device to host). This triggers a USB interrupt if ROLEEXE is one.
  This bit is cleared when the UBSTACLR.ROLEEXIC bit is written to one.
- **BCERRI: B-Connection Error Interrupt**
  This bit is set when an error occurs during the B-connection. This triggers a USB interrupt if BCERRE is one.
  This bit is cleared when the UBSTACLR.BCERRIC bit is written to one.
  This bit shall only be used in host mode.
- **VBERRI: VBus Error Interrupt**
  This bit is set when a VBus drop has been detected. This triggers a USB interrupt if VBERRE is one.
  This bit is cleared when the UBSTACLR.VBERRIC bit is written to one.
  This bit shall only be used in host mode.
  If a VBus problem occurs, then the VBERRI interrupt is generated even if the USBB does not go to an error state because of VBUSHWC is one.
- **SRPI: SRP Interrupt**
  This bit is set when an SRP has been detected. This triggers a USB interrupt if SRPE is one.
  This bit is cleared when the UBSTACLR.SRPIC bit is written to one.
  This bit shall only be used in host mode.
- **VBUSTI: VBus Transition Interrupt**
  This bit is set when a transition (high to low, low to high) has been detected on the USB_VBUS pad. This triggers an USB interrupt if VBUSTE is one.
  This bit is cleared when the UBSTACLR.VBUSTIC bit is written to one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **IDTI: ID Transition Interrupt**
  This bit is set when a transition (high to low, low to high) has been detected on the USB_ID input pin. This triggers an USB interrupt if IDTE is one.
  This bit is cleared when the UBSTACLR.IDTIC bit is written to one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.

27.8.1.3 General Status Clear Register

Register Name: USBSTACLR

Access Type: Write-Only

Offset: 0x0808

Read 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
-	-	-	-	-	-	VBUSRQC	-
7	6	5	4	3	2	1	0
STOIC	HNPERRIC	ROLEEXIC	BCERRIC	VBERRIC	SRPIC	VBUSTIC	IDTIC

Writing a one to a bit in this register will clear the corresponding bit in UBSTA.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.1.4 General Status Set Register

Register Name: USBSTASET

Access Type: Write-Only

Offset: 0x080C

Read 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
-	-	-	-	-	-	VBUSRQS	-

7	6	5	4	3	2	1	0
STOIS	HNPERRIS	ROLEEXIS	BCERRIS	VBERRIS	SRPIS	VBUSTIS	IDTIS

Writing a one to a bit in this register will set the corresponding bit in UBSTA, what may be useful for test or debug purposes.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.1.5 Version Register

Register Name: UVERS

Access Type: Read-Only

Offset: 0x0818

Read Value: -

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

27.8.1.6 Features Register

Register Name: UFEATURES

Access Type: Read-Only

Offset: 0x081C

Read Value: -

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

23	22	21	20	19	18	17	16
ENHBISO7	ENHBISO6	ENHBISO5	ENHBISO4	ENHBISO3	ENHBISO2	ENHBISO1	DATABUS

15	14	13	12	11	10	9	8
BYTEWRITE DPRAM	FIFOMAXSIZE			DMAFIFOWORDDEPTH			

7	6	5	4	3	2	1	0
DMABUFFE RSIZE	DMACHANNELNBR			EPTNBRMAX			

- **ENHBISOn: High Bandwidth Isochronous Feature for Endpoint n**
  1: The high bandwidth isochronous is supported.
  1: The high bandwidth isochronous is not supported.
- **DATABUS: Data Bus 16-8**
  1: The UTMI data bus is a 16-bit data path at 30MHz.
  0: The UTMI data bus is a 8-bit data path at 60MHz.
- **BYTEWRITEDPRAM: DPRAM Byte-Write Capability**
  1: The DPRAM is natively byte-write capable.
  0: The DPRAM byte write lanes have shadow logic implemented in the USBB IP interface.
- **FIFOMAXSIZE: Maximal FIFO Size**
  This field indicates the maximal FIFO size, i.e., the DPRAM size:

FIFOMAXSIZE			Maximal FIFO Size
0	0	0	< 256 bytes
0	0	1	< 512 bytes
0	1	0	< 1024 bytes
0	1	1	< 2048 bytes
1	0	0	< 4096 bytes
1	0	1	< 8192 bytes
1	1	0	< 16384 bytes
1	1	1	>= 16384 bytes

- **DMAFIFOWORDDEPTH: DMA FIFO Depth in Words**
  This field indicates the DMA FIFO depth controller in words:

DMAFIFOWORDDEPTH				DMA FIFO Depth in Words
0	0	0	0	16
0	0	0	1	1
0	0	1	0	2
				...
1	1	1	1	15

- **DMABUFFERSIZE: DMA Buffer Size**
  1: The DMA buffer size is 24bits.
  0: The DMA buffer size is 16bits.
- **DMACHANNELNBR: Number of DMA Channels**
  This field indicates the number of hardware-implemented DMA channels:

DMACHANNELNBR			Number of DMA Channels
0	0	0	Reserved
0	0	1	1
0	1	0	2
			...
1	1	1	7

- **EPTNBRMAX: Maximal Number of Pipes/Endpoints**
  This field indicates the number of hardware-implemented pipes/endpoints:

EPTNBRMAX				Maximal Number of Pipes/Endpoints
0	0	0	0	16
0	0	0	1	1
0	0	1	0	2
				...
1	1	1	1	15

27.8.1.7 Address Size Register

Register Name: UADDRSIZE

Access Type: Read-Only

Offset: 0x0820

Read Value: -

31	30	29	28	27	26	25	24
UADDRSIZE[31:24]							

23	22	21	20	19	18	17	16
UADDRSIZE[23:16]							

15	14	13	12	11	10	9	8
UADDRSIZE[15:8]							

7	6	5	4	3	2	1	0
UADDRSIZE[7:0]							

- **UADDRSIZE: IP PB Address Size**
  This field indicates the size of the PB address space reserved for the USBB IP interface.

27.8.1.8 Name Register 1

Register Name: UNAME1

Access Type: Read-Only

Offset: 0x0824

Read Value: -

31	30	29	28	27	26	25	24
UNAME1[31:24]							

23	22	21	20	19	18	17	16
UNAME1[23:16]							

15	14	13	12	11	10	9	8
UNAME1[15:8]							

7	6	5	4	3	2	1	0
UNAME1[7:0]							

- **UNAME1: IP Name Part One**
  This field indicates the first part of the ASCII-encoded name of the USBB IP.

27.8.1.9 Name Register 2

Register Name: UNAME2

Access Type: Read-Only

Offset: 0x0828

Read Value:

31	30	29	28	27	26	25	24
UNAME2[31:24]							

23	22	21	20	19	18	17	16
UNAME2[23:16]							

15	14	13	12	11	10	9	8
UNAME2[15:8]							

7	6	5	4	3	2	1	0
UNAME2[7:0]							

- **UNAME2: IP Name Part Two**
  This field indicates the second part of the ASCII-encoded name of the USBB IP.

27.8.1.10 Finite State Machine Status Register

Register Name: USBFSM

Access Type: Read-Only

Offset: 0x082C

Read Value: 0x00000009

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	DRDSTATE			

- **DRDSTATE**

  This field indicates the state of the USBB.
  Refer to the OTG specification for more details.

DRDSTATE	Description
0	a_idle state: this is the start state for A-devices (when the ID pin is 0)
1	a_wait_vrise: In this state, the A-device waits for the voltage on VBus to rise above the A-device VBus Valid threshold (4.4 V).
2	a_wait_bcon: In this state, the A-device waits for the B-device to signal a connection.
3	a_host: In this state, the A-device that operates in Host mode is operational.
4	a_suspend: The A-device operating as a host is in the suspend mode.
5	a_peripheral: The A-device operates as a peripheral.
6	a_wait_vfall: In this state, the A-device waits for the voltage on VBus to drop below the A-device Session Valid threshold (1.4 V).
7	a_vbus_err: In this state, the A-device waits for recovery of the over-current condition that caused it to enter this state.
8	a_wait_discharge: In this state, the A-device waits for the data usb line to discharge (100 us).
9	b_idle: this is the start state for B-device (when the ID pin is 1).
10	b_peripheral: In this state, the B-device acts as the peripheral.
11	b_wait_begin_hnp: In this state, the B-device is in suspend mode and waits until 3 ms before initiating the HNP protocol if requested.

DRDSTATE	Description
12	b_wait_discharge: In this state, the B-device waits for the data usb line to discharge (100 us) before becoming Host.
13	b_wait_acon: In this state, the B-device waits for the A-device to signal a connect before becoming B-Host.
14	b_host: In this state, the B-device acts as the Host.
15	b_srp_init: In this state, the B-device attempts to start a session using the SRP protocol.

27.8.2 USB Device Registers

27.8.2.1 Device General Control Register

Register Name: UDCON

Access Type: Read/Write

Offset: 0x0000

Reset Value: 0x00000100

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	OPMODE2

15	14	13	12	11	10	9	8
TSTPCKT	TSTK	TSTJ	LS	SPDCONF		RMWKUP	DETACH

7	6	5	4	3	2	1	0
ADDEN	UADD						

- **OPMODE2: Specific Operational mode**
  1: The UTMI transceiver is in the «disable bit stuffing and NRZI encoding» operational mode for test purpose.
  0: The UTMI transceiver is in normal operation mode.
- **TSTPCKT: Test packet mode**
  1: The UTMI transceiver generates test packets for test purpose.
  0: The UTMI transceiver is in normal operation mode.
- **TSTK: Test mode K**
  1: The UTMI transceiver generates high-speed K state for test purpose.
  0: The UTMI transceiver is in normal operation mode.
- **TSTJ: Test mode J**
  1: The UTMI transceiver generates high-speed J state for test purpose.
  0: The UTMI transceiver is in normal operation mode.
- **LS: Low-Speed Mode Force**
  1: The low-speed mode is active.
  0: The full-speed mode is active.
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBB (by writing a zero to the USBE bit) does not reset this bit.

- **SPDCONF: Speed Configuration**
  This field contains the peripheral speed.

SPDCONF		Speed
0	0	Normal mode: the peripheral starts in full-speed mode and performs a high-speed reset to switch to the high-speed mode if the host is high-speed capable.
0	1	reserved, do not use this configuration
1	0	reserved, do not use this configuration
1	1	Full-speed: the peripheral remains in full-speed mode whatever is the host speed capability.

- **RMWKUP: Remote Wake-Up**
  Writing a one to this bit will send an upstream resume to the host for a remote wake-up.
  Writing a zero to this bit has no effect.
  This bit is cleared when the USBB receive a USB reset or once the upstream resume has been sent.
- **DETACH: Detach**
  Writing a one to this bit will physically detach the device (disconnect internal pull-up resistor from D+ and D-).
  Writing a zero to this bit will reconnect the device.
- **ADDEN: Address Enable**
  Writing a one to this bit will activate the UADD field (USB address).
  Writing a zero to this bit has no effect.
  This bit is cleared when a USB reset is received.
- **UADD: USB Address**
  This field contains the device address.
  This field is cleared when a USB reset is received.

27.8.2.2 Device Global Interrupt Register

Register Name: UDINT

Access Type: Read-Only

Offset: 0x0004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INT	DMA6INT	DMA5INT	DMA4INT	DMA3INT	DMA2INT	DMA1INT	-
23	22	21	20	19	18	17	16
-	-	-	EP8INT	EP7INT	EP6INT	EP5INT	EP4INT
15	14	13	12	11	10	9	8
EP3INT	EP2INT	EP1INT	EP0INT	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSM	EORSM	WAKEUP	EORST	SOF	MSOF	SUSP

- **DMAnINT: DMA Channel n Interrupt**
  This bit is set when an interrupt is triggered by the DMA channel n. This triggers a USB interrupt if DMAnINTE is one.
  This bit is cleared when the UDDMAnSTATUS interrupt source is cleared.
- **EPnINT: Endpoint n Interrupt**
  This bit is set when an interrupt is triggered by the endpoint n (UESTAn, UECONn). This triggers a USB interrupt if EPnINTE is one.
  This bit is cleared when the interrupt source is serviced.
- **UPRSM: Upstream Resume Interrupt**
  This bit is set when the USBB sends a resume signal called “Upstream Resume”. This triggers a USB interrupt if UPRSME is one.
  This bit is cleared when the UDINTCLR.UPRSMC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).
- **EORSM: End of Resume Interrupt**
  This bit is set when the USBB detects a valid “End of Resume” signal initiated by the host. This triggers a USB interrupt if EORSME is one.
  This bit is cleared when the UDINTCLR.EORSMC bit is written to one to acknowledge the interrupt.
- **WAKEUP: Wake-Up Interrupt**
  This bit is set when the USBB is reactivated by a filtered non-idle signal from the lines (not by an upstream resume). This triggers an interrupt if WAKEUPE is one.
  This bit is cleared when the UDINTCLR.WAKEUPC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).
  This bit is cleared when the Suspend (SUSP) interrupt bit is set.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **EORST: End of Reset Interrupt**
  This bit is set when a USB “End of Reset” has been detected. This triggers a USB interrupt if EORSTE is one.
  This bit is cleared when the UDINTCLR.EORSTC bit is written to one to acknowledge the interrupt.
- **SOF: Start of Frame Interrupt**
  This bit is set when a USB “Start of Frame” PID (SOF) has been detected (every 1 ms). This triggers a USB interrupt if SOFE is one. The FNUM field is updated. In High-speed mode, the MFNUM field is cleared.
  This bit is cleared when the UDINTCLR.SOFC bit is written to one to acknowledge the interrupt.

- **MSOF: Micro Start of Frame Interrupt**
  This bit is set in High-speed mode when a USB “Micro Start of Frame” PID (SOF) has been detected (every 125 us). This triggers a USB interrupt if MSOFE is one. The MFNUM field is updated. The FNUM field is unchanged.
  This bit is cleared when the UDINTCLR.MSOFC bit is written to one to acknowledge the interrupt.
- **SUSP: Suspend Interrupt**
  This bit is set when a USB “Suspend” idle bus state has been detected for 3 frame periods (J state for 3 ms). This triggers a USB interrupt if SUSPE is one.
  This bit is cleared when the UDINTCLR.SUSPC bit is written to one to acknowledge the interrupt.
  This bit is cleared when the Wake-Up (WAKEUP) interrupt bit is set.

27.8.2.3 Device Global Interrupt Clear Register

Register Name: UDINTCLR

Access Type: Write-Only

Offset: 0x0008

Read 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
-	UPRSMC	EORSMC	WAKEUPC	EORSTC	SOFC	MSOFC	SUSPC

Writing a one to a bit in this register will clear the corresponding bit in UDINT.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.4 Device Global Interrupt Set Register

Register Name: UDINTSET

Access Type: Write-Only

Offset: 0x000C

Read Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTS	DMA6INTS	DMA5INTS	DMA4INTS	DMA3INTS	DMA2INTS	DMA1INTS	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSMS	EORSMS	WAKEUPS	EORSTS	SOFS	MSOFS	SUSPS

Writing a one to a bit in this register will set the corresponding bit in UDINT, what may be useful for test or debug purposes.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.5 Device Global Interrupt Enable Register

Register Name: UDINTE

Access Type: Read-Only

Offset: 0x0010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTE	DMA6INTE	DMA5INTE	DMA4INTE	DMA3INTE	DMA2INTE	DMA1INTE	-
23	22	21	20	19	18	17	16
-	-	-	EP8INTE	EP7INTE	EP6INTE	EP5INTE	EP4INTE
15	14	13	12	11	10	9	8
EP3INTE	EP2INTE	EP1INTE	EP0INTE	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	MSOFE	SUSPE

1: The corresponding interrupt is enabled.
0: The corresponding interrupt is disabled.
A bit in this register is set when the corresponding bit in UDINTESET is written to one.
A bit in this register is cleared when the corresponding bit in UDINTECLR is written to one.

27.8.2.6 Device Global Interrupt Enable Clear Register

Register Name: UDINTECLR

Access Type: Write-Only

Offset: 0x0014

Read Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTEC	DMA6INTEC	DMA5INTEC	DMA4INTEC	DMA3INTEC	DMA2INTEC	DMA1INTEC	-
23	22	21	20	19	18	17	16
-	-	-	EP8INTEC	EP7INTEC	EP6INTEC	EP5INTEC	EP4INTEC
15	14	13	12	11	10	9	8
EP3INTEC	EP2INTEC	EP1INTEC	EP0INTEC	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSMEC	EORSMEC	WAKEUPEC	EORSTEC	SOFEC	MSOFEC	SUSPEC

Writing a one to a bit in this register will clear the corresponding bit in UDINTE.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.7 Device Global Interrupt Enable Set Register

Register Name: UDINTESET

Access Type: Write-Only

Offset: 0x0018

Read Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTES	DMA6INTES	DMA5INTES	DMA4INTES	DMA3INTES	DMA2INTES	DMA1INTES	-
23	22	21	20	19	18	17	16
-	-	-	EP8INTES	EP7INTES	EP6INTES	EP5INTES	EP4INTES
15	14	13	12	11	10	9	8
EP3INTES	EP2INTES	EP1INTES	EP0INTES	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSMES	EORSMES	WAKEUPES	EORSTES	SOFES	MSOFES	SUSPES

Writing a one to a bit in this register will set the corresponding bit in UDINTE.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.8 Endpoint Enable/Reset Register

Register Name: UERST

Access Type: Read/Write

Offset: 0x001C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	EPRST8
23	22	21	20	19	18	17	16
EPRST7	EPRST6	EPRST5	EPRST4	EPRST3	EPRST2	EPRST1	EPRST0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	EPEN8
7	6	5	4	3	2	1	0
EPEN7	EPEN6	EPEN5	EPEN4	EPEN3	EPEN2	EPEN1	EPEN0

- **EPRSTn: Endpoint n Reset**
  Writing a one to this bit will reset the endpoint n FIFO prior to any other operation, upon hardware reset or when a USB bus reset has been received. This resets the endpoint n registers (UECFGn, UESTAn, UECONn) but not the endpoint configuration (ALLOC, EPBK, EPSIZE, EPDIR, EPTYPE).
  All the endpoint mechanism (FIFO counter, reception, transmission, etc.) is reset apart from the Data Toggle Sequence field (DTSEQ) which can be cleared by setting the RSTDT bit (by writing a one to the RSTDTS bit).
  The endpoint configuration remains active and the endpoint is still enabled.
  Writing a zero to this bit will complete the reset operation and start using the FIFO.
  This bit is cleared upon receiving a USB reset.
- **EPENn: Endpoint n Enable**
  1: The endpoint n is enabled.
  0: The endpoint n is disabled, what forces the endpoint n state to inactive (no answer to USB requests) and resets the endpoint n registers (UECFGn, UESTAn, UECONn) but not the endpoint configuration (ALLOC, EPBK, EPSIZE, EPDIR, EPTYPE).

27.8.2.9 Device Frame Number Register

Register Name: UDFNUM

Access Type: Read-Only

Offset: 0x0020

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
FNCERR	-	FNUM[10:5]					

7	6	5	4	3	2	1	0
FNUM[4:0]					MFNUM		

- **FNCERR: Frame Number CRC Error**
  This bit is set when a corrupted frame number (or micro-frame number) is received. This bit and the SOF (or MSOF) interrupt bit are updated at the same time.
  This bit is cleared upon receiving a USB reset.
- **FNUM: Frame Number**
  This field contains the 11-bit frame number information. It is provided in the last received SOF packet.
  This field is cleared upon receiving a USB reset.
  FNUM is updated even if a corrupted SOF is received.
- **MFNUM: Micro Frame Number**
  This field contains the 3-bit micro frame number information. It is provided in the last received MSOF packet.
  This field is cleared at the beginning of each start of frame (SOF interrupt) or upon receiving a USB reset.
  MFNUM is updated even if a corrupted MSOF is received.

27.8.2.10 Endpoint n Configuration Register

Register Name: UECFGn, n in [0..7]

Access Type: Read/Write

Offset: 0x0100 + (n * 0x04)

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
-	NBTRANS		EPTYPE		-	AUTOSW	EPDIR

7	6	5	4	3	2	1	0
-	EPSIZE			EPBK		ALLOC	-

- **NBTRANS: Number of transaction per microframe for isochronous endpoint**
  This field shall be written to the number of transaction per microframe to perform high-bandwidth isochronous transfer
  This field can be written only for endpoint that have this capability (see UFEATURES register, ENHBISOn bit). This field is 0 otherwise.
  This field is irrelevant for non-isochronous endpoint.

NBTRANS		Number of transaction
0	0	reserved to endpoint that does not have the high-bandwidth isochronous capability.
0	1	default value: one transaction per micro-frame.
1	0	2 transactions per micro-frame. This endpoint should be configured as double-bank.
1	1	3 transactions per micro-frame. This endpoint should be configured as triple-bank.

- **EPTYPE: Endpoint Type**
  This field shall be written to select the endpoint type:

EPTYPE		Endpoint Type
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

This field is cleared upon receiving a USB reset.

- **AUTOSW: Automatic Switch**
  This bit is cleared upon receiving a USB reset.
  1: The automatic bank switching is enabled.
  0: The automatic bank switching is disabled.
- **EPDIR: Endpoint Direction**
  This bit is cleared upon receiving a USB reset.
  1: The endpoint direction is IN (nor for control endpoints).
  0: The endpoint direction is OUT.
- **EPSIZE: Endpoint Size**
  This field shall be written to select the size of each endpoint bank:

EPSIZE			Endpoint Size
0	0	0	8 bytes
0	0	1	16 bytes
0	1	0	32 bytes
0	1	1	64 bytes
1	0	0	128 bytes
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

  This field is cleared upon receiving a USB reset (except for the endpoint 0).
- **EPBK: Endpoint Banks**
  This field shall be written to select the number of banks for the endpoint:

EPBK		Endpoint Banks
0	0	1 (single-bank endpoint)
0	1	2 (double-bank endpoint)
1	0	3 (triple-bank endpoint)
1	1	Reserved

  For control endpoints, a single-bank endpoint (0b00) shall be selected.
  This field is cleared upon receiving a USB reset (except for the endpoint 0).
- **ALLOC: Endpoint Memory Allocate**
  Writing a one to this bit will allocate the endpoint memory. The user should check the CFGOK bit to know whether the allocation of this endpoint is correct.
  Writing a zero to this bit will free the endpoint memory.
  This bit is cleared upon receiving a USB reset (except for the endpoint 0).

27.8.2.11 Endpoint n Status Register

Register Name: UESTAn, n in [0..7]

Access Type: Read-Only 0x0100

Offset: 0x0130 + (n * 0x04)

Reset Value: 0x00000100

31	30	29	28	27	26	25	24
-	BYCT						

23	22	21	20	19	18	17	16
BYCT				-	CFGOK	CTRLDIR	RWALL

15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		-	ERRORTRANS	DTSEQ	

7	6	5	4	3	2	1	0
SHORT PACKET	STALLEDI/ CRCERRI	OVERFI	NAKINI/ HBISOFLUSHI	NAKOUTI/ HBISOINERRI	RXSTPI/ UNDERFI	RXOUTI	TXINI

- **BYCT: Byte Count**
  This field is set with the byte count of the FIFO.
  For IN endpoints, incremented after each byte written by the software into the endpoint and decremented after each byte sent to the host.
  For OUT endpoints, incremented after each byte received from the host and decremented after each byte read by the software from the endpoint.
  This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.
- **CFGOK: Configuration OK Status**
  This bit is updated when the ALLOC bit is written to one.
  This bit is set if the endpoint n number of banks (EPBK) and size (EPSIZE) are correct compared to the maximal allowed number of banks and size for this endpoint and to the maximal FIFO size (i.e. the DPRAM size).
  If this bit is cleared, the user shall rewrite correct values to the EPBK and EPSIZE fields in the UECFGn register.
- **CTRLDIR: Control Direction**
  This bit is set after a SETUP packet to indicate that the following packet is an IN packet.
  This bit is cleared after a SETUP packet to indicate that the following packet is an OUT packet.
  Writing a zero or a one to this bit has no effect.
- **RWALL: Read/Write Allowed**
  This bit is set for IN endpoints when the current bank is not full, i.e., the user can write further data into the FIFO.
  This bit is set for OUT endpoints when the current bank is not empty, i.e., the user can read further data from the FIFO.
  This bit is never set if STALLRQ is one or in case of error.
  This bit is cleared otherwise.
  This bit shall not be used for control endpoints.

- **CURRBK: Current Bank**

  This bit is set for non-control endpoints, to indicate the current bank:

CURRBK		Current Bank
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Reserved

  This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

- **NBUSYBK: Number of Busy Banks**

  This field is set to indicate the number of busy banks:

NBUSYBK		Number of Busy Banks
0	0	0 (all banks free)
0	1	1
1	0	2
1	1	3

  For IN endpoints, it indicates the number of banks filled by the user and ready for IN transfer. When all banks are free, this triggers an EPnINT interrupt if NBUSYBKE is one.
  For OUT endpoints, it indicates the number of banks filled by OUT transactions from the host. When all banks are busy, this triggers an EPnINT interrupt if NBUSYBKE is one.
  When the FIFOCON bit is cleared (by writing a one to the FIFOCONC bit) to validate a new bank, this field is updated two or three clock cycles later to calculate the address of the next bank.
  An EPnINT interrupt is triggered if:
  - for IN endpoint, NBUSYBKE is one and all the banks are free.
  - for OUT endpoint, NBUSYBKE is one and all the banks are busy.

- **ERRORTRANS: High-bandwidth isochronous OUT endpoint transaction error Interrupt**

  This bit is set when a transaction error occurs during the current micro-frame (the data toggle sequencing does not respect the usb 2.0 standard). This triggers an EPnINT interrupt if ERRORTRANSE is one.
  This bit is set as long as the current bank (CURRBK) belongs to the bad n-transactions (n=1,2 or 3) transferred during the micro-frame. Shall be cleared by software by clearing (at least once) the FIFOCON bit to switch to the bank that belongs to the next n-transactions (next micro-frame).

- **DTSEQ: Data Toggle Sequence**

  This field is set to indicate the PID of the current bank:

DTSEQ		Data Toggle Sequence
0	0	Data0
0	1	Data1
1	0	Data2 (for high-bandwidth isochronous endpoint)
1	1	MData (for high-bandwidth isochronous endpoint)

  For IN transfers, it indicates the data toggle sequence that will be used for the next packet to be sent. This is not relative to the current bank.
  For OUT transfers, this value indicates the last data toggle sequence received on the current bank.
  By default DTSEQ is 0b01, as if the last data toggle sequence was Data1, so the next sent or expected data toggle sequence should be Data0.
  For High-bandwidth isochronous endpoint, an EPnINT interrupt is triggered if:
  - MDATAE is one and a MData packet has been received (DTSEQ=MData and RXOUTI is one).

- DATAXE is one and a Data0/1/2 packet has been received (DTSEQ=Data0/1/2 and RXOUTI is one)

- **SHORTPACKET: Short Packet Interrupt**
  This bit is set for non-control OUT endpoints, when a short packet has been received.
  This bit is set for non-control IN endpoints, a short packet is transmitted upon ending a DMA transfer, thus signaling an end of isochronous frame or a bulk or interrupt end of transfer, this only if the End of DMA Buffer Output Enable (DMAENDEN) bit and the Automatic Switch (AUTOSW) bit are written to one.
  This triggers an EPnINT interrupt if SHORTPACKETE is one.
  This bit is cleared when the SHORTPACKETC bit is written to one. This will acknowledge the interrupt.
- **STALLEDI: STALLed Interrupt**
  This bit is set to signal that a STALL handshake has been sent. To do that, the software has to set the STALLRQ bit (by writing a one to the STALLRQS bit). This triggers an EPnINT interrupt if STALLEDE is one.
  This bit is cleared when the STALLEDIC bit is written to one. This will acknowledge the interrupt.
- **CRCERRI: CRC Error Interrupt**
  This bit is set to signal that a CRC error has been detected in an isochronous OUT endpoint. The OUT packet is stored in the bank as if no CRC error had occurred. This triggers an EPnINT interrupt if CRCERRE is one.
  This bit is cleared when the CRCERRIC bit is written to one. This will acknowledge the interrupt.
- **OVERFI: Overflow Interrupt**
  This bit is set when an overflow error occurs. This triggers an EPnINT interrupt if OVERFE is one.
  For all endpoint types, an overflow can occur during OUT stage if the host attempts to write into a bank that is too small for the packet. The packet is acknowledged and the RXOUTI bit is set as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.
  This bit is cleared when the OVERFIC bit is written to one. This will acknowledge the interrupt.
- **NAKINI: NAKed IN Interrupt**
  This bit is set when a NAK handshake has been sent in response to an IN request from the host. This triggers an EPnINT interrupt if NAKINE is one.
  This bit is cleared when the NAKINIC bit is written to one. This will acknowledge the interrupt.
- **HBISOFLUSHI: High Bandwidth Isochronous IN Flush Interrupt**
  This bit is set, for High-bandwidth isochronous IN endpoint (with NBTRANS=2 or 3), at the end of the micro-frame, if less than N transaction has been completed by the USBB without underflow error. This may occur in case of a missing IN token. In this case, the bank are flushed out to ensure the data synchronization between the host and the device. This triggers an EPnINT interrupt if HBISOFLUSHE is one.
  This bit is cleared when the HBISOFLUSHIC bit is written to one. This will acknowledge the interrupt.
- **NAKOUTI: NAKed OUT Interrupt**
  This bit is set when a NAK handshake has been sent in response to an OUT request from the host. This triggers an EPnINT interrupt if NAKOUTE is one.
  This bit is cleared when the NAKOUTIC bit is written to one. This will acknowledge the interrupt.
- **HBISOINERRI: High bandwidth isochronous IN Underflow Error Interrupt**
  This bit is set, for High-bandwidth isochronous IN endpoint (with NBTRANS=2 or 3), at the end of the microframe, if less than N bank was written by the cpu within this micro-frame. This triggers an EPnINT interrupt if HBISOINERRE is one.
  This bit is cleared when the HBISOINERRIC bit is written to one. This will acknowledge the interrupt.
- **UNDERFI: Underflow Interrupt**
  This bit is set, for isochronous IN/OUT endpoints, when an underflow error occurs. This triggers an EPnINT interrupt if UNDERFE is one.
  An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBB.
  An underflow can also occur during OUT stage if the host sends a packet while the bank is already full. Typically, the CPU is not fast enough. The packet is lost.
  Shall be cleared by writing a one to the UNDERFIC bit. This will acknowledge the interrupt.
  This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means RXSTPI for control endpoints.

- **RXSTPI: Received SETUP Interrupt**

  This bit is set, for control endpoints, to signal that the current bank contains a new valid SETUP packet. This triggers an EPnINT interrupt if RXSTPE is one.

  Shall be cleared by writing a one to the RXSTPIC bit. This will acknowledge the interrupt and free the bank.

  This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means UNDERFI for isochronous IN/OUT endpoints.

- **RXOUTI: Received OUT Data Interrupt**

  This bit is set, for control endpoints, when the current bank contains a bulk OUT packet (data or status stage). This triggers an EPnINT interrupt if RXOUTE is one.

  Shall be cleared for control end points, by writing a one to the RXOUTIC bit. This will acknowledge the interrupt and free the bank.

  This bit is set for isochronous, bulk and, interrupt OUT endpoints, at the same time as FIFOCON when the current bank is full. This triggers an EPnINT interrupt if RXOUTE is one.

  Shall be cleared for isochronous, bulk and, interrupt OUT endpoints, by writing a one to the RXOUTIC bit. This will acknowledge the interrupt, what has no effect on the endpoint FIFO.

  The user then reads from the FIFO and clears the FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The RXOUTI and FIFOCON bits are set/cleared in accordance with the status of the next bank.

  RXOUTI shall always be cleared before clearing FIFOCON.

  This bit is inactive (cleared) for isochronous, bulk and interrupt IN endpoints.

- **TXINI: Transmitted IN Data Interrupt**

  This bit is set for control endpoints, when the current bank is ready to accept a new IN packet. This triggers an EPnINT interrupt if TXINE is one.

  This bit is cleared when the TXINIC bit is written to one. This will acknowledge the interrupt and send the packet.

  This bit is set for isochronous, bulk and interrupt IN endpoints, at the same time as FIFOCON when the current bank is free. This triggers an EPnINT interrupt if TXINE is one.

  This bit is cleared when the TXINIC bit is written to one. This will acknowledge the interrupt, what has no effect on the endpoint FIFO.

  The user then writes into the FIFO and clears the FIFOCON bit to allow the USBB to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The TXINI and FIFOCON bits are set/cleared in accordance with the status of the next bank.

  TXINI shall always be cleared before clearing FIFOCON.

  This bit is inactive (cleared) for isochronous, bulk and interrupt OUT endpoints.

27.8.2.12 *Endpoint n Status Clear Register*

Register Name: UESTAnCLR, n in [0..7]

Access Type: Write-Only

Offset: 0x0160 + (n * 0x04)

Read 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
SHORT PACKETC	STALLEDIC/ CRCERRIC	OVERFIC	NAKINIC/ HBISOFLUSHIC	NAKOUTIC/ HBISOINERRIC	RXSTPIC/ UNDERFIC	RXOUTIC	TXINIC

Writing a one to a bit in this register will clear the corresponding bit in UESTA.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.13 Endpoint n Status Set Register

Register Name: UESTAnSET, n in [0..7]

Access Type: Write-Only

Offset: 0x0190 + (n * 0x04)

Read 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
-	-	-	NBUSYBKS	-	-		-

7	6	5	4	3	2	1	0
SHORT PACKETS	STALLEDIS/ CRCERRIS	OVERFIS	NAKINIS/ HBISOFLUSHIS	NAKOUTIS/ HBISOINERRIS	RXSTPIS/ UNDERFIS	RXOUTIS	TXINIS

Writing a one to a bit in this register will set the corresponding bit in UESTA, what may be useful for test or debug purposes.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.14 Endpoint n Control Register

Register Name: UECONn, n in [0..7]

Access Type: Read-Only

Offset: 0x01C0 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQ	RSTDT	NYETDIS	EPDISHDMA
15	14	13	12	11	10	9	8
-	FIFOCON	KILLBK	NBUSYBKE	-	ERRORTRANSE	DATAXE	MDATAE
7	6	5	4	3	2	1	0
SHORT PACKETE	STALLEDE/ CRCERRE	OVERFE	NAKINE/ HBISOFLUSHE	NAKOUTE/ HBISOINERRE	RXSTPE/ UNDERFE	RXOUTE	TXINE

- **STALLRQ: STALL Request**
  This bit is set when the STALLRQS bit is written to one. This will request to send a STALL handshake to the host.
  This bit is cleared when a new SETUP packet is received or when the STALLRQC bit is written to zero.
- **RSTDT: Reset Data Toggle**
  This bit is set when the RSTDTS bit is written to one. This will clear the data toggle sequence, i.e., set to Data0 the data toggle sequence of the next sent (IN endpoints) or received (OUT endpoints) packet.
  This bit is cleared instantaneously.
  The user does not have to wait for this bit to be cleared.
- **NYETDIS: NYET token disable**
  This bit is set when the NYETDISS bit is written to one. This will send a ACK handshake instead of a NYET handshake in high-speed mode.
  This bit is cleared when the NYETDISC bit is written to one.This will let the USBB handling the high-speed handshake following the usb 2.0 standard.
- **EPDISHDMA: Endpoint Interrupts Disable HDMA Request Enable**
  This bit is set when the EPDISHDMAS is written to one. This will pause the on-going DMA channel n transfer on any Endpoint n interrupt (EPnINT), whatever the state of the Endpoint n Interrupt Enable bit (EPnINTE).
  The user then has to acknowledge or to disable the interrupt source (e.g. RXOUTI) or to clear the EPDISHDMA bit (by writing a one to the EPDISHDMAC bit) in order to complete the DMA transfer.
  In ping-pong mode, if the interrupt is associated to a new system-bank packet (e.g. Bank1) and the current DMA transfer is running on the previous packet (Bank0), then the previous-packet DMA transfer completes normally, but the new-packet DMA transfer will not start (not requested).
  If the interrupt is not associated to a new system-bank packet (NAKINI, NAKOUTI, etc.), then the request cancellation may occur at any time and may immediately pause the current DMA transfer.
  This may be used for example to identify erroneous packets, to prevent them from being transferred into a buffer, to complete a DMA transfer by software after reception of a short packet, etc.
- **FIFOCON: FIFO Control**
  For control endpoints:
  The FIFOCON and RWALL bits are irrelevant. The software shall therefore never use them on these endpoints. When read, their value is always 0.
  For IN endpoints:

This bit is set when the current bank is free, at the same time as TXINI.
This bit is cleared (by writing a one to the FIFOCONC bit) to send the FIFO data and to switch to the next bank.
For OUT endpoints:
This bit is set when the current bank is full, at the same time as RXOUTI.
This bit is cleared (by writing a one to the FIFOCONC bit) to free the current bank and to switch to the next bank.

- **KILLBK: Kill IN Bank**
  This bit is set when the KILLBKS bit is written to one. This will kill the last written bank.
  This bit is cleared when the bank is killed.
  Caution: The bank is really cleared when the “kill packet” procedure is accepted by the USBB core. This bit is automatically cleared after the end of the procedure:
  The bank is really killed: NBUSYBK is decremented.
  The bank is not cleared but sent (IN transfer): NBUSYBK is decremented.
  The bank is not cleared because it was empty.
  The user shall wait for this bit to be cleared before trying to kill another packet.
  This kill request is refused if at the same time an IN token is coming and the last bank is the current one being sent on the USB line. If at least 2 banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. Indeed, in this case, the current bank is sent (IN transfer) while the last bank is killed.
- **NBUSYBKE: Number of Busy Banks Interrupt Enable**
  This bit is set when the NBUSYBKES bit is written to one. This will enable the Number of Busy Banks interrupt (NBUSYBK).
  This bit is cleared when the NBUSYBKEC bit is written to zero. This will disable the Number of Busy Banks interrupt (NBUSYBK).
- **ERRORTRANSE: Transaction Error Interrupt Enable**
  This bit is set when the ERRORTRANSES bit is written to one. This will enable the transaction error interrupt (ERRORTRANS).
  This bit is cleared when the ERRORTRANSEC bit is written to one. This will disable the transaction error interrupt (ERRORTRANS).
- **DATAXE: DataX Interrupt Enable**
  This bit is set when the DATAXES bit is written to one. This will enable the DATAX interrupt. (see DTSEQ bits)
  This bit is cleared when the DATAXEC bit is written to one. This will disable the DATAX interrupt.
- **MDATAE: MData Interrupt Enable**
  This bit is set when the MDATAES bit is written to one. This will enable the Multiple DATA interrupt. (see DTSEQ bits)
  This bit is cleared when the MDATAEC bit is written to one. This will disable the Multiple DATA interrupt.
- **SHORTPACKETE: Short Packet Interrupt Enable**
  This bit is set when the SHORTPACKETES bit is written to one. This will enable the Short Packet interrupt (SHORTPACKET).
  This bit is cleared when the SHORTPACKETEC bit is written to one. This will disable the Short Packet interrupt (SHORTPACKET).
- **STALLEDE: STALLed Interrupt Enable**
  This bit is set when the STALLEDES bit is written to one. This will enable the STALLed interrupt (STALLEDI).
  This bit is cleared when the STALLEDEC bit is written to one. This will disable the STALLed interrupt (STALLEDI).
- **CRCERRE: CRC Error Interrupt Enable**
  This bit is set when the CRCERRES bit is written to one. This will enable the CRC Error interrupt (CRCERRI).
  This bit is cleared when the CRCERREC bit is written to one. This will disable the CRC Error interrupt (CRCERRI).
- **OVERFE: Overflow Interrupt Enable**
  This bit is set when the OVERFES bit is written to one. This will enable the Overflow interrupt (OVERFI).
  This bit is cleared when the OVERFEC bit is written to one. This will disable the Overflow interrupt (OVERFI).
- **NAKINE: NAKed IN Interrupt Enable**
  This bit is set when the NAKINES bit is written to one. This will enable the NAKed IN interrupt (NAKINI).
  This bit is cleared when the NAKINEC bit is written to one. This will disable the NAKed IN interrupt (NAKINI).
- **HBISOFLUSHE: High Bandwidth Isochronous IN Flush Interrupt Enable**
  This bit is set when the HBISOFLUSHES bit is written to one. This will enable the HBISOFLUSHI interrupt.
  This bit is cleared when the HBISOFLUSHEC bit disable the HBISOFLUSHI interrupt.
- **NAKOUTE: NAKed OUT Interrupt Enable**
  This bit is set when the NAKOUTES bit is written to one. This will enable the NAKed OUT interrupt (NAKOUTI).
  This bit is cleared when the NAKOUTEC bit is written to one. This will disable the NAKed OUT interrupt (NAKOUTI).
- **HBISOINERRE: High Bandwidth Isochronous IN Error Interrupt Enable**
  This bit is set when the HBISOINERRES bit is written to one. This will enable the HBISOINERRI interrupt.
  This bit is cleared when the HBISOINERREC bit disable the HBISOINERRI interrupt.

- **RXSTPE: Received SETUP Interrupt Enable**
  This bit is set when the RXSTPES bit is written to one. This will enable the Received SETUP interrupt (RXSTPI).
  This bit is cleared when the RXSTPEC bit is written to one. This will disable the Received SETUP interrupt (RXSTPI).
- **UNDERFE: Underflow Interrupt Enable**
  This bit is set when the UNDERFES bit is written to one. This will enable the Underflow interrupt (UNDERFI).
  This bit is cleared when the UNDERFEC bit is written to one. This will disable the Underflow interrupt (UNDERFI).
- **RXOUTE: Received OUT Data Interrupt Enable**
  This bit is set when the RXOUTES bit is written to one. This will enable the Received OUT Data interrupt (RXOUT).
  This bit is cleared when the RXOUTEC bit is written to one. This will disable the Received OUT Data interrupt (RXOUT).
- **TXINE: Transmitted IN Data Interrupt Enable**
  This bit is set when the TXINES bit is written to one. This will enable the Transmitted IN Data interrupt (TXINI).
  This bit is cleared when the TXINEC bit is written to one. This will disable the Transmitted IN Data interrupt (TXINI).

27.8.2.15 Endpoint n Control Clear Register

Register Name: UECONnCLR, n in [0..7]

Access Type: Write-Only

Offset: 0x0220 + (n * 0x04)

Read Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	STALLRQC	-	NYETDISC	EPDISHDMAC

15	14	13	12	11	10	9	8
-	FIFOCONC	-	NBUSYBKEC	-	ERRORTRANSEC	DATAXEC	MDATEC

7	6	5	4	3	2	1	0
SHORT PACKETEC	STALLEDEC /CRCERREC	OVERFEC	NAKINEC/ HBISOFLUSHEC	NAKOUTEC/ HBISOINERREC	RXSTPEC/ UNDERFEC	RXOUTEC	TXINEC

Writing a one to a bit in this register will clear the corresponding bit in UECONn.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.16 Endpoint n Control Set Register

Register Name: UECONnSET, n in [0..7]

Access Type: Write-Only

Offset: 0x01F0 + (n * 0x04)

Read Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	STALLRQS	RSTDTS	NYETDISS	EPDISHDMAS

15	14	13	12	11	10	9	8
-	-	KILLBKS	NBUSYBKES	-	ERRORTRANSES	DATAXES	MDATES

7	6	5	4	3	2	1	0
SHORT PACKETES	STALLEDES /CRCERRES	OVERFES	NAKINES/ HBISOFLUSHES	NAKOUTES/ HBISOINERRES	RXSTPES/ UNDERFES	RXOUTES	TXINES

Writing a one to a bit in this register will set the corresponding bit in UECONn.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.2.17 Device DMA Channel n Next Descriptor Address Register

Register Name: *UDDMAnNEXTDESC*, n in [1..7]

Access Type: Read/Write

Offset: 0x0310 + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
NXTDESCADDR[31:24]							

23	22	21	20	19	18	17	16
NXTDESCADDR[23:16]							

15	14	13	12	11	10	9	8
NXTDESCADDR[15:8]							

7	6	5	4	3	2	1	0
NXTDESCADDR[7:4]				-	-	-	-

- **NXTDESCADDR: Next Descriptor Address**
  This field contains the bits 31:4 of the 16-byte aligned address of the next channel descriptor to be processed.
  This field is written either or by descriptor loading.

27.8.2.18 Device DMA Channel n HSB Address Register

Register Name: *UDDMAnADDR*, n in [1..7]

Access Type: Read/Write

Offset: 0x0314 + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
HSBADDR[31:24]							

23	22	21	20	19	18	17	16
HSBADDR[23:16]							

15	14	13	12	11	10	9	8
HSBADDR[15:8]							

7	6	5	4	3	2	1	0
HSBADDR[7:0]							

- **HSBADDR: HSB Address**

  This field determines the HSB bus current address of a channel transfer.

  The address written to the HSB address bus is HSBADDR rounded down to the nearest word-aligned address, i.e., HSBADDR[1:0] is considered as 0b00 since only word accesses are performed.

  Channel HSB start and end addresses may be aligned on any byte boundary.

  The user may write this field only when the Channel Enabled bit (CHEN) of the UDDMAnSTATUS register is cleared.

  This field is updated at the end of the address phase of the current access to the HSB bus. It is incremented of the HSB access byte-width.

  The HSB access width is 4 bytes, or less at packet start or end if the start or end address is not aligned on a word boundary.

  The packet start address is either the channel start address or the next channel address to be accessed in the channel buffer.

  The packet end address is either the channel end address or the latest channel address accessed in the channel buffer.

  The channel start address is written or loaded from the descriptor, whereas the channel end address is either determined by the end of buffer or the end of USB transfer if the Buffer Close Input Enable bit (BUFFCLOSEINEN) is set.

27.8.2.19 Device DMA Channel n Control Register

Register Name: *UDDMAnCONTROL*, n in [1..7]

Access Type: Read/Write

Offset: 0x0318 + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHBYTELENGTH[15:8]							
23	22	21	20	19	18	17	16
CHBYTELENGTH[7:0]							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
BURSTLOCKEN	DESCLDIRQEN	EOBUFFIRQEN	EOTIRQEN	DMAENDEN	BUFFCLOSEINEN	LDNXTCHDESCEN	CHEN

- **CHBYTELENGTH: Channel Byte Length**
  This field determines the total number of bytes to be transferred for this buffer.
  The maximum channel transfer size 64kB is reached when this field is zero (default value).
  If the transfer size is unknown, the transfer end is controlled by the peripheral and this field should be written to zero.
  This field can be written or descriptor loading only after the UDDMAnSTATUS.CHEN bit has been cleared, otherwise this field is ignored.
- **BURSTLOCKEN: Burst Lock Enable**
  1: The USB data burst is locked for maximum optimization of HSB busses bandwidth usage and maximization of fly-by duration.
  0: The DMA never locks the HSB access.
- **DESCLDIRQEN: Descriptor Loaded Interrupt Enable**
  1: The Descriptor Loaded interrupt is enabled.This interrupt is generated when a Descriptor has been loaded from the system bus.
  0: The Descriptor Loaded interrupt is disabled.
- **EOBUFFIRQEN: End of Buffer Interrupt Enable**
  1: The end of buffer interrupt is enabled.This interrupt is generated when the channel byte count reaches zero.
  0: The end of buffer interrupt is disabled.
- **EOTIRQEN: End of USB Transfer Interrupt Enable**
  1: The end of usb OUT data transfer interrupt is enabled. This interrupt is generated only if the BUFFCLOSEINEN bit is set.
  0: The end of usb OUT data transfer interrupt is disabled.
- **DMAENDEN: End of DMA Buffer Output Enable**
  Writing a one to this bit will properly complete the usb transfer at the end of the dma transfer.
  For IN endpoint, it means that a short packet (or a Zero Length Packet) will be sent to the USB line to properly closed the usb transfer at the end of the dma transfer.
  For OUT endpoint, it means that all the banks will be properly released. (NBUSYBK=0) at the end of the dma transfer.

- **BUFFCLOSEINEN: Buffer Close Input Enable**
  For Bulk and Interrupt endpoint, writing a one to this bit will automatically close the current DMA transfer at the end of the USB OUT data transfer (received short packet).
  For Full-speed Isochronous, it does not make sense, so BUFFCLOSEINEN should be left to zero.
  For high-speed OUT isochronous, it may make sense. In that case, if BUFFCLOSEINEN is written to one, the current DMA transfer is closed when the received PID packet is not MDATA.
  Writing a zero to this bit to disable this feature.
- **LDNXTCHDESCEN: Load Next Channel Descriptor Enable**
  1: the channel controller loads the next descriptor after the end of the current transfer, i.e. when the UDDMAnSTATUS.CHEN bit is reset.
  0: no channel register is loaded after the end of the channel transfer.
  If the CHEN bit is written to zero, the next descriptor is immediately loaded upon transfer request (endpoint is free for IN endpoint, or endpoint is full for OUT endpoint).

LDNXTCHDESCEN	CHEN	Current Bank
0	0	stop now
0	1	Run and stop at end of buffer
1	0	Load next descriptor now
1	1	Run and link at end of buffer

- **CHEN: Channel Enable**
  Writing this bit to zero will disabled the DMA channel and no transfer will occur upon request. If the LDNXTCHDESCEN bit is written to zero, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both UDDMAnSTATUS.CHEN and CHACTIVE bits are zero.
  Writing this bit to one will set the UDDMAnSTATUS.CHEN bit and enable DMA channel data transfer. Then any pending request will start the transfer. This may be used to start or resume any requested transfer.
  This bit is cleared when the channel source bus is disabled at end of buffer. If the LDNXTCHDESCEN bit has been cleared by descriptor loading, the user will have to write to one the corresponding CHEN bit to start the described transfer, if needed.
  If a channel request is currently serviced when this bit is zero, the DMA FIFO buffer is drained until it is empty, then the UDDMAnSTATUS.CHEN bit is cleared.
  If the LDNXTCHDESCEN bit is set or after this bit clearing, then the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded.

27.8.2.20 Device DMA Channel n Status Register

Register Name: *UDDMAnSTATUS*, n in [1..7]

Access Type: Read/Write

Offset: 0x031C + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHBYTECNT[15:8]							

23	22	21	20	19	18	17	16
CHBYTECNT[7:0]							

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	DESCLDSTA	EOCHBUFFSTA	EOTSTA	-	-	CHACTIVE	CHEN

- **CHBYTECNT: Channel Byte Count**
  This field contains the current number of bytes still to be transferred for this buffer.
  This field is decremented at each dma access.
  This field is reliable (stable) only if the CHEN bit is zero.
- **DESCLDSTA: Descriptor Loaded Status**
  This bit is set when a Descriptor has been loaded from the HSB bus.
  This bit is cleared when read by the user.
- **EOCHBUFFSTA: End of Channel Buffer Status**
  This bit is set when the Channel Byte Count counts down to zero.
  This bit is automatically cleared when read by software.
- **EOTSTA: End of USB Transfer Status**
  This bit is set when the completion of the usb data transfer has closed the dma transfer. It is valid only if UDDMAnCONTROL.BUFFCLOSEINEN is one.
  This bit is automatically cleared when read by software.
- **CHACTIVE: Channel Active**
  0: the DMA channel is no longer trying to source the packet data.
  1: the DMA channel is currently trying to source packet data, i.e. selected as the highest-priority requesting channel. When a packet transfer cannot be completed due to an EOCHBUFFSTA, this bit stays set during the next channel descriptor load (if any) and potentially until USB packet transfer completion, if allowed by the new descriptor.
  When programming a DMA by descriptor (Load next descriptor now), the CHACTIVE bit is set only once the DMA is running (the endpoint is free for IN transaction, the endpoint is full for OUT transaction).
- **CHEN: Channel Enabled**
  This bit is set (after one cycle latency) when the L.CHEN is written to one or when the descriptor is loaded.
  This bit is cleared when any transfer is ended either due to an elapsed byte count or a USB device initiated transfer end.
  0: the DMA channel no longer transfers data, and may load the next descriptor if the UDDMAnCONTROL.LDNXTCHDESCEN bit is zero.
  1: the DMA channel is currently enabled and transfers data upon request.
  If a channel request is currently serviced when the UDDMAnCONTROL.CHEN bit is written to zero, the DMA FIFO buffer is drained until it is empty, then this status bit is cleared.

27.8.3 USB Host Registers

27.8.3.1 Host General Control Register

Register Name: UHCON

Access Type: Read/Write

Offset: 0x0400

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
-	-	SPDCONF		-	RESUME	RESET	SOFE

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

- **SPDCONF: Speed Configuration**
  This field contains the host speed capability.

SPDCONF		Speed
0	0	Normal mode: the host start in full-speed mode and perform a high-speed reset to switch to the high-speed mode if the downstream peripheral is high-speed capable.
0	1	reserved, do not use this configuration
1	0	reserved, do not use this configuration
1	1	Full-speed: the host remains to full-speed mode whatever is the peripheral speed capability.

- **RESUME: Send USB Resume**
  Writing a one to this bit will generate a USB Resume on the USB bus.
  This bit is cleared when the USB Resume has been sent or when a USB reset is requested.
  Writing a zero to this bit has no effect.
  This bit should be written to one only when the start of frame generation is enable. (SOFE bit is one).
- **RESET: Send USB Reset**
  Writing a one to this bit will generate a USB Reset on the USB bus.
  This bit is cleared when the USB Reset has been sent.
  It may be useful to write a zero to this bit when a device disconnection is detected (UHINT.DDISCI is one) whereas a USB Reset is being sent.
- **SOFE: Start of Frame Generation Enable**
  Writing a one to this bit will generate SOF on the USB bus in full speed mode and keep alive in low speed mode.
  Writing a zero to this bit will disable the SOF generation and to leave the USB bus in idle state.
  This bit is set when a USB reset is requested or an upstream resume interrupt is detected (UHINT.TXRSMI).

27.8.3.2 Host Global Interrupt Register

Register Name: UHINT

Access Type: Read-Only

Offset: 0x0404

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INT	DMA6INT	DMA5INT	DMA4INT	DMA3INT	DMA2INT	DMA1INT	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INT
15	14	13	12	11	10	9	8
P7INT	P6INT	P5INT	P4INT	P3INT	P2INT	P1INT	P0INT
7	6	5	4	3	2	1	0
-	HWUPI	HSOFI	RXRSMI	RSMEDI	RSTI	DDISCI	DCONNI

- **DMAnINT: DMA Channel n Interrupt**
  This bit is set when an interrupt is triggered by the DMA channel n. This triggers a USB interrupt if the corresponding DMAnINTE is one (UHINTE register).
  This bit is cleared when the UHDMAnSTATUS interrupt source is cleared.
- **PnINT: Pipe n Interrupt**
  This bit is set when an interrupt is triggered by the endpoint n (UPSTAn). This triggers a USB interrupt if the corresponding pipe interrupt enable bit is one (UHINTE register).
  This bit is cleared when the interrupt source is served.
- **HWUPI: Host Wake-Up Interrupt**
  This bit is set when the host controller is in the suspend mode (SOFE is zero) and an upstream resume from the peripheral is detected.
  This bit is set when the host controller is in the suspend mode (SOFE is zero) and a peripheral disconnection is detected.
  This bit is set when the host controller is in the Idle state (USBSTA.VBUSRQ is zero, no VBus is generated), and an OTG SRP event initiated by the peripheral is detected (USBSTA.SRPI is one).
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **HSOFI: Host Start of Frame Interrupt**
  This bit is set when a SOF is issued by the Host controller. This triggers a USB interrupt when HSOFE is one. When using the host controller in low speed mode, this bit is also set when a keep-alive is sent.
  This bit is cleared when the HSOFIC bit is written to one.
- **RXRSMI: Upstream Resume Received Interrupt**
  This bit is set when an Upstream Resume has been received from the Device.
  This bit is cleared when the RXRSMIC is written to one.
- **RSMEDI: Downstream Resume Sent Interrupt**
  This bit set when a Downstream Resume has been sent to the Device.
  This bit is cleared when the RSMEDIC bit is written to one.
- **RSTI: USB Reset Sent Interrupt**
  This bit is set when a USB Reset has been sent to the device.
  This bit is cleared when the RSTIC bit is written to one.
- **DDISCI: Device Disconnection Interrupt**
  This bit is set when the device has been removed from the USB bus.
  This bit is cleared when the DDISCIC bit is written to one.

- **DCONNI: Device Connection Interrupt**
  This bit is set when a new device has been connected to the USB bus.
  This bit is cleared when the DCONNIC bit is written to one.

27.8.3.3 Host Global Interrupt Clear Register

Register Name: UHINTCLR

Access Type: Write-Only

Offset: 0x0408

Read 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
-	HWUPIC	HSOFIC	RXRSMIC	RSMEDIC	RSTIC	DDISCIC	DCONNIC

Writing a one to a bit in this register will clear the corresponding bit in UHINT.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.4 Host Global Interrupt Set Register

Register Name: UHINTSET

Access Type: Write-Only

Offset: 0x040C

Read Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTS	DMA6INTS	DMA5INTS	DMA4INTS	DMA3INTS	DMA2INTS	DMA1INTS	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	HWUPIS	HSOFIS	RXRSMIS	RSMEDIS	RSTIS	DDISCIS	DCONNIS

Writing a one to a bit in this register will set the corresponding bit in UHINT, what may be useful for test or debug purposes.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.5 Host Global Interrupt Enable Register

Register Name: UHINTE

Access Type: Read-Only

Offset: 0x0410

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTE	DMA6INTE	DMA5INTE	DMA4INTE	DMA3INTE	DMA2INTE	DMA1INTE	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INTE
15	14	13	12	11	10	9	8
P7INTE	P6INTE	P5INTE	P4INTE	P3INTE	P2INTE	P1INTE	P0INTE
7	6	5	4	3	2	1	0
-	HWUPIE	HSOFIE	RXRSMIE	RSMEDIE	RSTIE	DDISCIE	DCONNIE

- **DMAnINTE: DMA Channel n Interrupt Enable**
  This bit is set when the DMAnINTES bit is written to one. This will enable the DMA Channel n Interrupt (DMAnINT).
  This bit is cleared when the DMAnINTEC bit is written to one. This will disable the DMA Channel n Interrupt (DMAnINT).
- **PnINTE: Pipe n Interrupt Enable**
  This bit is set when the PnINTES bit is written to one. This will enable the Pipe n Interrupt (PnINT).
  This bit is cleared when the PnINTEC bit is written to one. This will disable the Pipe n Interrupt (PnINT).
- **HWUPIE: Host Wake-Up Interrupt Enable**
  This bit is set when the HWUPIES bit is written to one. This will enable the Host Wake-up Interrupt (HWUPI).
  This bit is cleared when the HWUPIEC bit is written to one. This will disable the Host Wake-up Interrupt (HWUPI).
- **HSOFIE: Host Start of Frame Interrupt Enable**
  This bit is set when the HSOFIES bit is written to one. This will enable the Host Start of Frame interrupt (HSOFI).
  This bit is cleared when the HSOFIEC bit is written to one. This will disable the Host Start of Frame interrupt (HSOFI).
- **RXRSMIE: Upstream Resume Received Interrupt Enable**
  This bit is set when the RXRSMIES bit is written to one. This will enable the Upstream Resume Received interrupt (RXRSMI).
  This bit is cleared when the RXRSMIEC bit is written to one. This will disable the Downstream Resume interrupt (RXRSMI).
- **RSMEDIE: Downstream Resume Sent Interrupt Enable**
  This bit is set when the RSMEDIES bit is written to one. This will enable the Downstream Resume interrupt (RSMEDI).
  This bit is cleared when the RSMEDIEC bit is written to one. This will disable the Downstream Resume interrupt (RSMEDI).
- **RSTIE: USB Reset Sent Interrupt Enable**
  This bit is set when the RSTIES bit is written to one. This will enable the USB Reset Sent interrupt (RSTI).
  This bit is cleared when the RSTIEC bit is written to one. This will disable the USB Reset Sent interrupt (RSTI).
- **DDISCIE: Device Disconnection Interrupt Enable**
  This bit is set when the DDISCIES bit is written to one. This will enable the Device Disconnection interrupt (DDISCI).
  This bit is cleared when the DDISCIEC bit is written to one. This will disable the Device Disconnection interrupt (DDISCI).
- **DCONNIE: Device Connection Interrupt Enable**
  This bit is set when the DCONNIES bit is written to one. This will enable the Device Connection interrupt (DCONNI).
  This bit is cleared when the DCONNIEC bit is written to one. This will disable the Device Connection interrupt (DCONNI).

27.8.3.6 Host Global Interrupt Enable Clear Register

Register Name: UHINTECLR

Access Type: Write-Only

Offset: 0x0414

Read Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTEC	DMA6INTEC	DMA5INTEC	DMA4INTEC	DMA3INTEC	DMA2INTEC	DMA1INTEC	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INTEC
15	14	13	12	11	10	9	8
P7INTEC	P6INTEC	P5INTEC	P4INTEC	P3INTEC	P2INTEC	P1INTEC	P0INTEC
7	6	5	4	3	2	1	0
-	HWUPIEC	HSOFIEC	RXRSMIEC	RSMEDIEC	RSTIEC	DDISCIEC	DCONNIEC

Writing a one to a bit in this register will clear the corresponding bit in UHINTE.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.7 Host Global Interrupt Enable Set Register

Register Name: UHINTESET

Access Type: Write-Only

Offset: 0x0418

Read Value: 0x00000000

31	30	29	28	27	26	25	24
DMA7INTES	DMA6INTES	DMA5INTES	DMA4INTES	DMA3INTES	DMA2INTES	DMA1INTES	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INTES
15	14	13	12	11	10	9	8
P7INTES	P6INTES	P5INTES	P4INTES	P3INTES	P2INTES	P1INTES	P0INTES
7	6	5	4	3	2	1	0
-	HWUPIES	HSOFIES	RXRSMIES	RSMEDIES	RSTIES	DDISCIES	DCONNIES

Writing a one to a bit in this register will set the corresponding bit in UHINT.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.8 Host Frame Number Register

Register Name: UHFNUM

Access Type: Read/Write

Offset: 0x0420

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
FLENHIGH							

15	14	13	12	11	10	9	8
-	-	FNUM[10:5]					

7	6	5	4	3	2	1	0
FNUM[4:0]					MFNUM		

- **FLENHIGH: Frame Length**
  In Full speed mode, this field contains the 8 high-order bits of the 16-bit internal frame counter (at 30 MHz, counter length is 30000 to ensure a SOF generation every 1 ms).
  In High speed mode, this field contains the 8 high-order bits of the 16-bit internal frame counter (at 30 MHz, counter length is 3750 to ensure a SOF generation every 125 us).
- **FNUM: Frame Number**
  This field contains the current SOF number.
  This field can be written. In this case, the MFNUM field is reset to zero.
- **MFNUM: Micro Frame Number**
  This field contains the current Micro Frame number (can vary from 0 to 7) updated every 125 us.
  When operating in full-speed mode, this field is tied to zero.

27.8.3.9 USB Host Frame Number Register (UHADDR1)

Register Name: UHADDR1

Access Type: Read/Write

Offset: 0x0424

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	UHADDRP3						

23	22	21	20	19	18	17	16
-	UHADDRP2						

15	14	13	12	11	10	9	8
-	UHADDRP1						

7	6	5	4	3	2	1	0
-	UHADDRP0						

- **UHADDRP3: USB Host Address**
  This field contains the address of the Pipe3 of the USB Device.
  This field is cleared when a USB reset is requested.
- **UHADDRP2: USB Host Address**
  This field contains the address of the Pipe2 of the USB Device.
  This field is cleared when a USB reset is requested.
- **UHADDRP1: USB Host Address**
  This field contains the address of the Pipe1 of the USB Device.
  This field is cleared when a USB reset is requested.
- **UHADDRP0: USB Host Address**
  This field contains the address of the Pipe0 of the USB Device.
  This field is cleared when a USB reset is requested.

27.8.3.10 Host Frame Number Register

Register Name: UHADDR2

Access Type: Read/Write

Offset: 0x0428

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	UHADDRP7						

23	22	21	20	19	18	17	16
-	UHADDRP6						

15	14	13	12	11	10	9	8
-	UHADDRP5						

7	6	5	4	3	2	1	0
-	UHADDRP4						

- **UHADDRP7: USB Host Address**
  This field contains the address of the Pipe7 of the USB Device.
  This field is cleared when a USB reset is requested.
- **UHADDRP6: USB Host Address**
  This field contains the address of the Pipe6 of the USB Device.
  This field is cleared when a USB reset is requested.
- **UHADDRP5: USB Host Address**
  This field contains the address of the Pipe5 of the USB Device.
  This field is cleared when a USB reset is requested.
- **UHADDRP4: USB Host Address**
  This field contains the address of the Pipe4 of the USB Device.
  This field is cleared when a USB reset is requested.

27.8.3.11 Host Frame Number Register

Register Name: UHADDR3

Access Type: Read/Write

Offset: 0x042C

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
-	UHADDRP8						

- **UHADDRP8: USB Host Address**
  This field contains the address of the Pipe8 of the USB Device.
  This field is cleared when a USB reset is requested.

27.8.3.12 Pipe Enable/Reset Register

Register Name: UPRST

Access Type: Read/Write

Offset: 0x0041C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	PRST8
23	22	21	20	19	18	17	16
PRST7	PRST6	PRST5	PRST4	PRST3	PRST2	PRST1	PRST0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	PEN8
7	6	5	4	3	2	1	0
PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0

- **PRSTn: Pipe n Reset**
  Writing a one to this bit will reset the Pipe n FIFO.
  This resets the endpoint n registers (UPCFGn, UPSTAn, UPCONn) but not the endpoint configuration (ALLOC, PBK, PSIZE, PTOKEN, PTYPE, PEPNUM, INTFRQ).
  All the endpoint mechanism (FIFO counter, reception, transmission, etc.) is reset apart from the Data Toggle management.
  The endpoint configuration remains active and the endpoint is still enabled.
  Writing a zero to this bit will complete the reset operation and allow to start using the FIFO.
- **PENn: Pipe n Enable**
  Writing a one to this bit will enable the Pipe n.
  Writing a zero to this bit will disable the Pipe n, what forces the Pipe n state to inactive and resets the pipe n registers (UPCFGn, UPSTAn, UPCONn) but not the pipe configuration (ALLOC, PBK, PSIZE).

27.8.3.13 Pipe n Configuration Register

Register Name: UPCFGn, n in [0..7]

Access Type: Read/Write

Offset: 0x0500 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
INTFRQ/BINTERVAL							

23	22	21	20	19	18	17	16
-	-	-	PINGEN	PEPNUM			

15	14	13	12	11	10	9	8
-	-	PTYPE		-	AUTOSW	PTOKEN	

7	6	5	4	3	2	1	0
-	PSIZE			PBK		ALLOC	-

- **INTFRQ: Pipe Interrupt Request Frequency**
  This field contains the maximum value in millisecond of the polling period for an Interrupt Pipe.
  This value has no effect for a non-Interrupt Pipe.
  This field is cleared upon sending a USB reset.
- **BINTERVAL: bInterval parameter for the Bulk-Out/Ping transaction**
  This field contains the Ping/Bulk-out period.
  If BINTERVAL>0 and PINGEN=1, one PING token is sent every BINTERVAL micro-frame until it is ACKed by the peripheral.
  If BINTERVAL=0 and PINGEN=1, multiple consecutive PING token is sent in the same micro-frame until it is ACKed.
  If BINTERVAL>0 and PINGEN=0, one OUT token is sent every BINTERVAL micro-frame until it is ACKed by the peripheral.
  If BINTERVAL=0 and PINGEN=0, multiple consecutive OUT token is sent in the same micro-frame until it is ACKed.
  This value must be in the range from 0 to 255.
- **PINGEN: Ping Enable**
  This bit is relevant for High-speed Bulk-out transaction only (including the control data stage and the control status stage).
  Writing a zero to this bit will disable the ping protocol.
  Writing a one to this bit will enable the ping mechanism according to the usb 2.0 standard.
  This bit is cleared upon sending a USB reset.
- **PEPNUM: Pipe Endpoint Number**
  This field contains the number of the endpoint targeted by the pipe. This value is from 0 to 15.
  This field is cleared upon sending a USB reset.
- **PTYPE: Pipe Type**
  This field contains the pipe type.

PTYPE		Pipe Type
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

This field is cleared upon sending a USB reset.

- **AUTOSW: Automatic Switch**
  This bit is cleared upon sending a USB reset.
  1: The automatic bank switching is enabled.
  0: The automatic bank switching is disabled.
- **PTOKEN: Pipe Token**
  This field contains the endpoint token.

PTOKEN	Endpoint Direction
00	SETUP
01	IN
10	OUT
11	reserved

- **PSIZE: Pipe Size**
  This field contains the size of each pipe bank.

PSIZE			Endpoint Size
0	0	0	8 bytes
0	0	1	16 bytes
0	1	0	32 bytes
0	1	1	64 bytes
1	0	0	128 bytes
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

This field is cleared upon sending a USB reset.

- **PBK: Pipe Banks**
  This field contains the number of banks for the pipe.

PBK		Endpoint Banks
0	0	1 (single-bank pipe)
0	1	2 (double-bank pipe)
1	0	3 (triple-bank pipe)
1	1	Reserved

For control endpoints, a single-bank pipe (0b00) should be selected.
This field is cleared upon sending a USB reset.

- **ALLOC: Pipe Memory Allocate**
  Writing a one to this bit will allocate the pipe memory.
  Writing a zero to this bit will free the pipe memory.
  This bit is cleared when a USB Reset is requested.
  Refer to the DPRAM Management chapter for more details.

27.8.3.14 Pipe n Status Register

Register Name: UPSTAn, n in [0..7]

Access Type: Read-Only

Offset: 0x0530 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	PBYCT[10:4]						

23	22	21	20	19	18	17	16
PBYCT[3:0]				-	CFGOK	-	RWALL

15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		-	-	DTSEQ	

7	6	5	4	3	2	1	0
SHORT PACKETI	RXSTALLDI/ CRCERRI	OVERFI	NAKEDI	PERRI	TXSTPI/ UNDERFI	TXOUTI	RXINI

- **PBYCT: Pipe Byte Count**
  This field contains the byte count of the FIFO.
  For OUT pipe, incremented after each byte written by the user into the pipe and decremented after each byte sent to the peripheral.
  For IN pipe, incremented after each byte received from the peripheral and decremented after each byte read by the user from the pipe.
  This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.
- **CFGOK: Configuration OK Status**
  This bit is set/cleared when the UPCFGn.ALLOC bit is set.
  This bit is set if the pipe n number of banks (UPCFGn.PBK) and size (UPCFGn.PSIZE) are correct compared to the maximal allowed number of banks and size for this pipe and to the maximal FIFO size (i.e., the DPRAM size).
  If this bit is cleared, the user should rewrite correct values ot the PBK and PSIZE field in the UPCFGn register.
- **RWALL: Read/Write Allowed**
  For OUT pipe, this bit is set when the current bank is not full, i.e., the software can write further data into the FIFO.
  For IN pipe, this bit is set when the current bank is not empty, i.e., the software can read further data from the FIFO.
  This bit is cleared otherwise.
  This bit is also cleared when the RXSTALL or the PERR bit is one.
- **CURRBK: Current Bank**
  For non-control pipe, this field indicates the number of the current bank.

CURRBK		Current Bank
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Reserved

This field may be updated 1 clock cycle after the RWALL bit changes, so the user shall not poll this field as an interrupt bit.

- **NBUSYBK: Number of Busy Banks**
  This field indicates the number of busy bank.
  For OUT pipe, this field indicates the number of busy bank(s), filled by the user, ready for OUT transfer. When all banks are busy, this triggers an PnINT interrupt if UPCONn.NBUSYBKE is one.
  For IN pipe, this field indicates the number of busy bank(s) filled by IN transaction from the Device. When all banks are free, this triggers an PnINT interrupt if UPCONn.NBUSYBKE is one.

NBUSYBK		Number of busy bank
0	0	All banks are free.
0	1	1 busy bank
1	0	2 busy banks
1	1	reserved

- **DTSEQ: Data Toggle Sequence**
  This field indicates the data PID of the current bank.

DTSEQ		Data toggle sequence
0	0	Data0
0	1	Data1
1	0	reserved
1	1	reserved

  For OUT pipe, this field indicates the data toggle of the next packet that will be sent.
  For IN pipe, this field indicates the data toggle of the received packet stored in the current bank.
- **SHORTPACKETI: Short Packet Interrupt**
  This bit is set when a short packet is received by the host controller (packet length inferior to the PSIZE programmed field).
  This bit is cleared when the SHORTPACKETIC bit is written to one.
- **RXSTALLDI: Received STALLed Interrupt**
  This bit is set, for all endpoints but isochronous, when a STALL handshake has been received on the current bank of the pipe. The Pipe is automatically frozen. This triggers an interrupt if the RXSTALLE bit is one.
  This bit is cleared when the RXSTALLDIC bit is written to one.
- **CRCERRI: CRC Error Interrupt**
  This bit is set, for isochronous endpoint, when a CRC error occurs on the current bank of the Pipe. This triggers an interrupt if the TXSTPE bit is one.
  This bit is cleared when the CRCERRIC bit is written to one.
- **OVERFI: Overflow Interrupt**
  This bit is set when the current pipe has received more data than the maximum length of the current pipe. An interrupt is triggered if the OVERFIE bit is one.
  This bit is cleared when the OVERFIC bit is written to one.
- **NAKEDI: NAKed Interrupt**
  This bit is set when a NAK has been received on the current bank of the pipe. This triggers an interrupt if the NAKEDE bit is one.
  This bit is cleared when the NAKEDIC bit written to one.
- **PERRI: Pipe Error Interrupt**
  This bit is set when an error occurs on the current bank of the pipe. This triggers an interrupt if the PERRE bit is set. Refers to the UPERRn register to determine the source of the error.
  This bit is cleared when the error source bit is cleared.
- **TXSTPI: Transmitted SETUP Interrupt**
  This bit is set, for Control endpoints, when the current SETUP bank is free and can be filled. This triggers an interrupt if the TXSTPE bit is one.
  This bit is cleared when the TXSTPIC bit is written to one.

- **UNDERFI: Underflow Interrupt**
  This bit is set, for isochronous and Interrupt IN/OUT pipe, when an error flow occurs. This triggers an interrupt if the UNDERFIE bit is one.
  This bit is set, for Isochronous or interrupt OUT pipe, when a transaction underflow occurs in the current pipe. (the pipe can't send the OUT data packet in time because the current bank is not ready). A zero-length-packet (ZLP) will be sent instead of.
  This bit is set, for Isochronous or interrupt IN pipe, when a transaction flow error occurs in the current pipe. i.e, the current bank of the pipe is not free whereas a new IN USB packet is received. This packet is not stored in the bank. For Interrupt pipe, the overflowed packet is ACKed to respect the USB standard.
  This bit is cleared when the UNDERFIEC bit is written to one.
- **TXOUTI: Transmitted OUT Data Interrupt**
  This bit is set when the current OUT bank is free and can be filled. This triggers an interrupt if the TXOUTE bit is one.
  This bit is cleared when the TXOUTIC bit is written to one.
- **RXINI: Received IN Data Interrupt**
  This bit is set when a new USB message is stored in the current bank of the pipe. This triggers an interrupt if the RXINE bit is one.
  This bit is cleared when the RXINIC bit is written to one.

27.8.3.15 Pipe n Status Clear Register

Register Name: UPSTAnCLR, n in [0..7]

Access Type: Write-Only

Offset: 0x0560 + (n * 0x04)

Read 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
SHORT PACKETIC	RXSTALLDIC/ CRCERRIC	OVERFIC	NAKEDIC	-	TXSTPIC/ UNDERFIC	TXOUTIC	RXINIC

Writing a one to a bit in this register will clear the corresponding bit in UPSTAn.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.16 Pipe n Status Set Register

Register Name: UPSTAnSET, n in [0..7]

Access Type: Write-Only

Offset: 0x0590 + (n * 0x04)

Read 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
-	-	-	NBUSYBKS	-	-	-	-

7	6	5	4	3	2	1	0
SHORT PACKETIS	RXSTALLDIS/ CRCERRIS	OVERFIS	NAKEDIS	PERRIS	TXSTPIS/ UNDERFIS	TXOUTIS	RXINIS

Writing a one to a bit in this register will set the corresponding bit in UPSTAn, what may be useful for test or debug purposes.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.17 Pipe n Control Register

Register Name: UPCONn, n in [0..7]

Access Type: Read-Only

Offset: 0x05C0 + (n * 0x04)

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	RSTDT	PFREEZE	PDISHDMA

15	14	13	12	11	10	9	8
-	FIFOCON	-	NBUSYBKE	-	-	-	-

7	6	5	4	3	2	1	0
SHORT PACKETIE	RXSTALLDE /CRCERRE	OVERFIE	NAKEDE	PERRE	TXSTPE/ UNDERFIE	TXOUTE	RXINE

- **RSTDT: Reset Data Toggle**
  This bit is set when the RSTDTS bit is written to one. This will reset the Data Toggle to its initial value for the current Pipe.
  This bit is cleared when proceed.
- **PFREEZE: Pipe Freeze**
  This bit is set when the PFREEZES bit is written to one or when the pipe is not configured or when a STALL handshake has been received on this Pipe or when an error occurs on the Pipe (PERR is one) or when (INRQ+1) In requests have been processed or when after a Pipe reset (UPRST.PRSTn rising) or a Pipe Enable (UPRST.PEN rising). This will Freeze the Pipe requests generation.
  This bit is cleared when the PFREEZEC bit is written to one. This will enable the Pipe request generation.
- **PDISHDMA: Pipe Interrupts Disable HDMA Request Enable**
  See the UECONn.EPDISHDMA bit description.
- **FIFOCON: FIFO Control**
  For OUT and SETUP Pipe:
  This bit is set when the current bank is free, at the same time than TXOUTI or TXSTPI.
  This bit is cleared when the FIFOCONC bit is written to one. This will send the FIFO data and switch the bank.
  For IN Pipe:
  This bit is set when a new IN message is stored in the current bank, at the same time than RXINI.
  This bit is cleared when the FIFOCONC bit is written to one. This will free the current bank and switch to the next bank.
- **NBUSYBKE: Number of Busy Banks Interrupt Enable**
  This bit is set when the NBUSYBKES bit is written to one.This will enable the Transmitted IN Data interrupt (NBUSYBKE).
  This bit is cleared when the NBUSYBKEC bit is written to one. This will disable the Transmitted IN Data interrupt (NBUSYBKE).
- **SHORTPACKETIE: Short Packet Interrupt Enable**
  This bit is set when the SHORTPACKETES bit is written to one. This will enable the Transmitted IN Data IT (SHORTPACKETIE).
  This bit is cleared when the SHORTPACKETEC bit is written to one. This will disable the Transmitted IN Data IT (SHORTPACKETE).

- **RXSTALLDE: Received STALLed Interrupt Enable**
  This bit is set when the RXSTALLDES bit is written to one. This will enable the Transmitted IN Data interrupt (RXSTALLDE).
  This bit is cleared when the RXSTALLDEC bit is written to one. This will disable the Transmitted IN Data interrupt (RXSTALLDE).
- **CRCERRE: CRC Error Interrupt Enable**
  This bit is set when the CRCERRES bit is written to one. This will enable the Transmitted IN Data interrupt (CRCERRE).
  This bit is cleared when the CRCERREC bit is written to one. This will disable the Transmitted IN Data interrupt (CRCERRE).
- **OVERFIE: Overflow Interrupt Enable**
  This bit is set when the OVERFIES bit is written to one. This will enable the Transmitted IN Data interrupt (OVERFIE).
  This bit is cleared when the OVERFIEC bit is written to one. This will disable the Transmitted IN Data interrupt (OVERFIE).
- **NAKEDE: NAKed Interrupt Enable**
  This bit is set when the NAKEDES bit is written to one. This will enable the Transmitted IN Data interrupt (NAKEDE).
  This bit is cleared when the NAKEDEC bit is written to one. This will disable the Transmitted IN Data interrupt (NAKEDE).
- **PERRE: Pipe Error Interrupt Enable**
  This bit is set when the PERRES bit is written to one. This will enable the Transmitted IN Data interrupt (PERRE).
  This bit is cleared when the PERREC bit is written to one. This will disable the Transmitted IN Data interrupt (PERRE).
- **TXSTPE: Transmitted SETUP Interrupt Enable**
  This bit is set when the TXSTPES bit is written to one. This will enable the Transmitted IN Data interrupt (TXSTPE).
  This bit is cleared when the TXSTPEC bit is written to one. This will disable the Transmitted IN Data interrupt (TXSTPE).
- **UNDERFIE: Underflow Interrupt Enable**
  This bit is set when the UNDERFIES bit is written to one. This will enable the Transmitted IN Data interrupt (UNDERFIE).
  This bit is cleared when the UNDERFIEC bit is written to one. This will disable the Transmitted IN Data interrupt (UNDERFIE).
- **TXOUTE: Transmitted OUT Data Interrupt Enable**
  This bit is set when the TXOUTES bit is written to one. This will enable the Transmitted IN Data interrupt (TXOUTE).
  This bit is cleared when the TXOUTEC bit is written to one. This will disable the Transmitted IN Data interrupt (TXOUTE).
- **RXINE: Received IN Data Interrupt Enable**
  This bit is set when the RXINES bit is written to one. This will enable the Transmitted IN Data interrupt (RXINE).
  This bit is cleared when the RXINEC bit is written to one. This will disable the Transmitted IN Data interrupt (RXINE).

27.8.3.18 Pipe n Control Clear Register

Register Name: UPCONnCLR, n in [0..7]

Access Type: Write-Only

Offset: 0x0620 + (n * 0x04)

Read Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	PFREEZEC	PDISHDMAC

15	14	13	12	11	10	9	8
-	FIFOCONC	-	NBUSYBKEC	-	-	-	-

7	6	5	4	3	2	1	0
SHORT PACKETIEC	RXSTALLDEC /CRCERREC	OVERFIEC	NAKEDEC	PERREC	TXSTPEC/ UNDERFIEC	TXOUTEC	RXINEC

Writing a one to a bit in this register will clear the corresponding bit in UPCONn.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.19 Pipe n Control Set Register

Register Name: UPCONnSET, n in [0..7]

Access Type: Write-Only

Offset: 0x05F0 + (n * 0x04)

Read Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	RSTDTS	PFREEZES	PDISHDMAS

15	14	13	12	11	10	9	8
-	-	-	NBUSYBKES	-	-	-	-

7	6	5	4	3	2	1	0
SHORT PACKETIES	RXSTALLDES /CRCERRES	OVERFIES	NAKEDES	PERRES	TXSTPES/ UNDERFIES	TXOUTES	RXINES

Writing a one to a bit in this register will set the corresponding bit in UPCONn.
Writing a zero to a bit in this register has no effect.
This bit always reads as zero.

27.8.3.20 Pipe n IN Request Register

Register Name: UPINRQn, n in [0..7]

Access Type: Read/Write

Offset: 0x0650 + (n * 0x04)

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
-	-	-	-	-	-	-	INMODE

7	6	5	4	3	2	1	0
INRQ							

- **INMODE: IN Request Mode**
  Writing a one to this bit will allow the USBB to perform infinite IN requests when the Pipe is not frozen.
  Writing a zero to this bit will perform a pre-defined number of IN requests. This number is the INRQ field.
- **INRQ: IN Request Number before Freeze**
  This field contains the number of IN transactions before the USBB freezes the pipe. The USBB will perform (INRQ+1) IN requests before to freeze the Pipe. This counter is automatically decreased by 1 each time a IN request has been successfully performed.
  This register has no effect when the INMODE bit is one (infinite IN requests generation till the pipe is not frozen).

27.8.3.21 Pipe n Error Register

Register Name: UPERRn, n in [0..7]

Access Type: Read/Write

Offset: 0x0680 + (n * 0x04)

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
-	COUNTER		CRC16	TIMEOUT	PID	DATAPID	DATATGL

- **COUNTER: Error Counter**
  This field is incremented each time an error occurs (CRC16, TIMEOUT, PID, DATAPID or DATATGL).
  This field is cleared when receiving a good usb packet without any error.
  When this field reaches 3 (i.e., 3 consecutive errors), this pipe is automatically frozen (UPCONn.PFREEZE is set).
  Writing 0b00 to this field will clear the counter.
- **CRC16: CRC16 Error**
  This bit is set when a CRC16 error has been detected.
  Writing a zero to this bit will clear the bit.
  Writing a one to this bit has no effect.
- **TIMEOUT: Time-Out Error**
  This bit is set when a Time-Out error has been detected.
  Writing a zero to this bit will clear the bit.
  Writing a one to this bit has no effect.
- **PID: PID Error**
  This bit is set when a PID error has been detected.
  Writing a zero to this bit will clear the bit.
  Writing a one to this bit has no effect.
- **DATAPID: Data PID Error**
  This bit is set when a Data PID error has been detected.
  Writing a zero to this bit will clear the bit.
  Writing a one to this bit has no effect.
- **DATATGL: Data Toggle Error**
  This bit is set when a Data Toggle error has been detected.
  Writing a zero to this bit will clear the bit.
  Writing a one to this bit has no effect.

27.8.3.22 Host DMA Channel n Next Descriptor Address Register

Register Name: *UHDMAnNEXTDESC*, n in [1..7]

Access Type: Read/Write

Offset: 0x0710 + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
NXTDESCADDR[31:24]							

23	22	21	20	19	18	17	16
NXTDESCADDR[23:16]							

15	14	13	12	11	10	9	8
NXTDESCADDR[15:8]							

7	6	5	4	3	2	1	0
NXTDESCADDR[7:4]				-	-	-	-

Same as Section 27.8.2.17.

27.8.3.23 Host DMA Channel n HSB Address Register

Register Name: *UHDMAnADDR*, n in [1..7]

Access Type: Read/Write

Offset: 0x0714 + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
HSBADDR[31:24]							

23	22	21	20	19	18	17	16
HSBADDR[23:16]							

15	14	13	12	11	10	9	8
HSBADDR[15:8]							

7	6	5	4	3	2	1	0
HSBADDR[7:0]							

Same as Section 27.8.2.18.

27.8.3.24 USB Host DMA Channel n Control Register

Register Name: *UHDMAnCONTROL*, n in [1..7]

Access Type: Read/Write

Offset: 0x0718 + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHBYTELENGTH[15:8]							

23	22	21	20	19	18	17	16
CHBYTELENGTH[7:0]							

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
BURSTLOCKEN	DESCLDIRQEN	EOBUFFIRQEN	EOTIRQEN	DMAENDEN	BUFFCLOSEINEN	LDNXTCHDESCEN	CHEN

Same as Section 27.8.2.19.

(just replace the IN endpoint term by OUT endpoint, and vice-versa)

27.8.3.25 USB Host DMA Channel n Status Register

Register Name: *UHDMAnSTATUS*, n in [1..7]

Access Type: Read/Write

Offset: 0x071C + (n - 1) * 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHBYTECNT[15:8]							

23	22	21	20	19	18	17	16
CHBYTECNT[7:0]							

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	DESCLDSTA	EOCHBUFFSTA	EOTSTA	-	-	CHACTIVE	CHEN

Same as Section 27.8.2.20.

27.8.4 USB Pipe/Endpoint n FIFO Data Register (USBFIFOnDATA)

The application has access to the physical DPRAM reserved for the Endpoint/Pipe through a 64KB logical address space. The application can access a 64KB buffer linearly or fixedly as the DPRAM address increment is fully handled by hardware. Byte, half-word and word access are supported. Data should be access in a big-endian way.

Disabling the USBB (by writing a zero to the USBE bit) does not reset the DPRAM.

27.9 Module Configuration

The specific configuration for the USBB instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 27-7. Module Clock Name

Module name	Clock name	Clock name
USBB	CLK_USBB_HSB	CLK_USBB_PB

Table 27-8. Register Reset Values

Register	Reset Value
UVERS	0x00000320
UFEATURES	to be defined
UADDRSIZE	to be defined
UNAME1	to be defined
UNAME2	to be defined

28. Timer/Counter (TC)

Rev: 2.2.3.1

28.1 Features

- **Three 16-bit Timer Counter channels**
- **A wide range of functions including:**
  - **Frequency measurement**
  - **Event counting**
  - **Interval measurement**
  - **Pulse generation**
  - **Delay timing**
  - **Pulse width modulation**
  - **Up/down capabilities**
- **Each channel is user-configurable and contains:**
  - **Three external clock inputs**
  - **Five internal clock inputs**
  - **Two multi-purpose input/output signals**
- **Internal interrupt signal**
- **Two global registers that act on all three TC channels**

28.2 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.

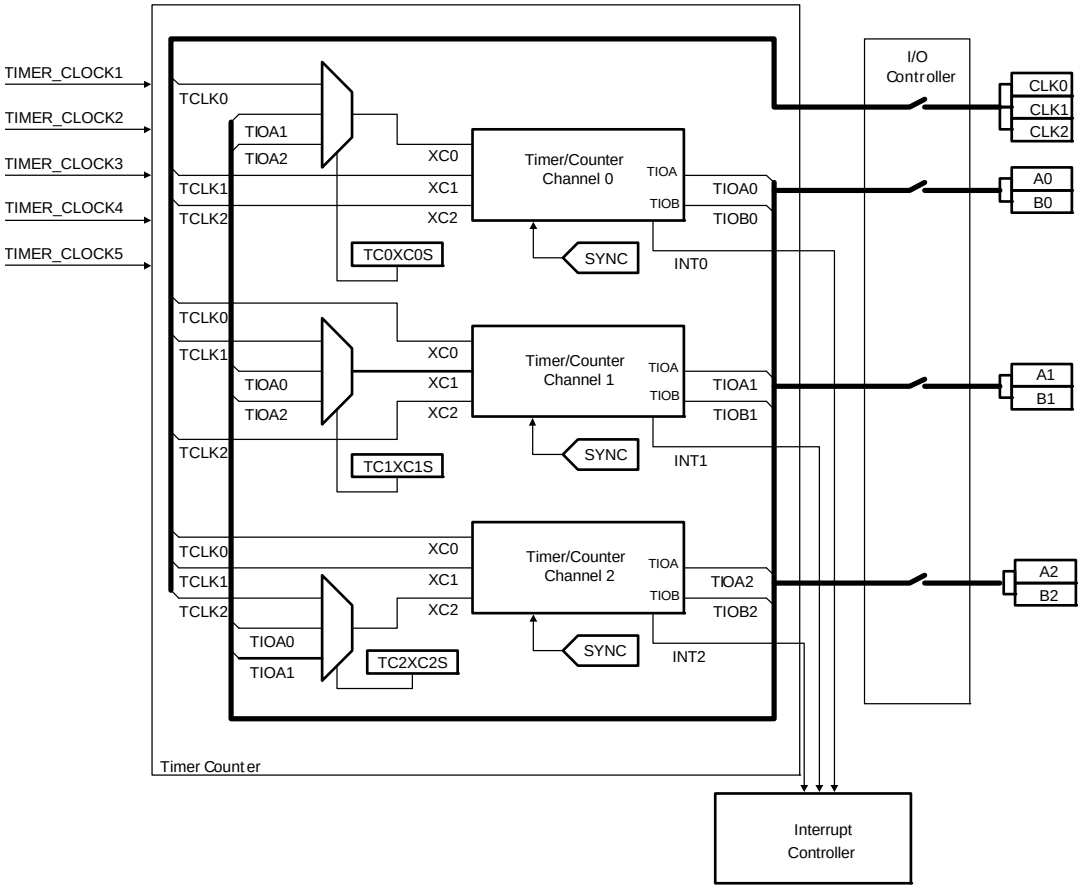
The TC block has two global registers which act upon all three TC channels.

The Block Control Register (BCR) allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register (BMR) defines the external clock inputs for each channel, allowing them to be chained.

28.3 Block Diagram

Figure 28-1. TC Block Diagram



28.4 I/O Lines Description

Table 28-1. I/O Lines Description

Pin Name	Description	Type
CLK0-CLK2	External Clock Input	Input
A0-A2	I/O Line A	Input/Output
B0-B2	I/O Line B	Input/Output

28.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

28.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O lines. The user must first program the I/O Controller to assign the TC pins to their peripheral functions.

28.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the TC, the TC will stop functioning and resume operation after the system wakes up from sleep mode.

28.5.3 Clocks

The clock for the TC bus interface (CLK_TC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the TC before disabling the clock, to avoid freezing the TC in an undefined state.

28.5.4 Interrupts

The TC interrupt request line is connected to the interrupt controller. Using the TC interrupt requires the interrupt controller to be programmed first.

28.5.5 Debug Operation

The Timer Counter clocks are frozen during debug operation, unless the OCD system keeps peripherals running in debug operation.

28.6 Functional Description

28.6.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Figure 28-3 on page 757.

28.6.1.1 Channel I/O Signals

As described in Figure 28-1 on page 741, each Channel has the following I/O signals.

Table 28-2. Channel I/O Signals 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

28.6.1.2 16-bit counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the Counter Overflow Status bit in the Channel n Status Register (SRn.COVFS) is set.

The current value of the counter is accessible in real time by reading the Channel n Counter Value Register (CVn). 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.

28.6.1.3 Clock selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the configurable I/O signals A0, A1 or A2 for chaining by writing to the BMR register. See Figure 28-2 on page 743.

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. See the Module Configuration Chapter for details about the connection of these clock sources.
- External clock signals: XC0, XC1 or XC2. See the Module Configuration Chapter for details about the connection of these clock sources.

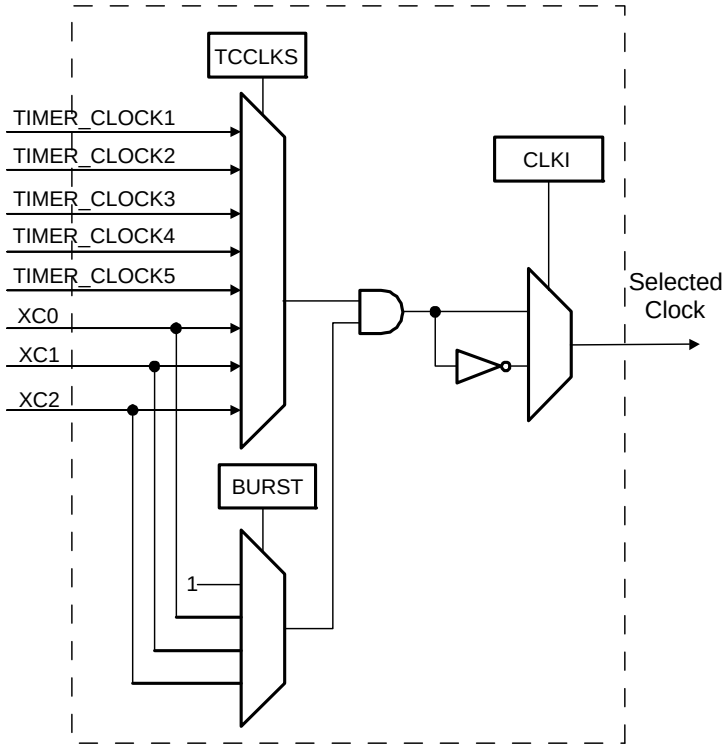
This selection is made by the Clock Selection field in the Channel n Mode Register (CMRn.TCCLKS).

The selected clock can be inverted with the Clock Invert bit in CMRn (CMRn.CLKI). 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 Signal Selection field in the CMRn register (CMRn.BURST) defines this signal.

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the CLK_TC period. The external clock frequency must be at least 2.5 times lower than the CLK_TC.

Figure 28-2. Clock Selection

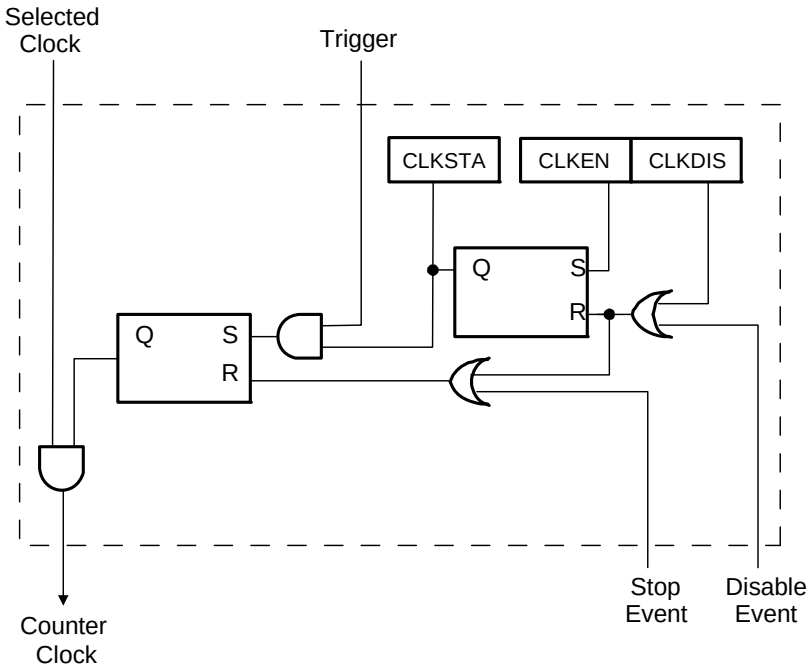


28.6.1.4 Clock control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 28-3 on page 744.

- The clock can be enabled or disabled by the user by writing to the Counter Clock Enable/Disable Command bits in the Channel n Clock Control Register (CCRn.CLKEN and CCRn.CLKDIS). In Capture mode it can be disabled by an RB load event if the Counter Clock Disable with RB Loading bit in CMRn is written to one (CMRn.LDBDIS). In Waveform mode, it can be disabled by an RC Compare event if the Counter Clock Disable with RC Compare bit in CMRn is written to one (CMRn.CPCDIS). When disabled, the start or the stop actions have no effect: only a CLKEN command in CCRn can re-enable the clock. When the clock is enabled, the Clock Enabling Status bit is set in SRn (SRn.CLKSTA).
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. In Capture mode the clock can be stopped by an RB load event if the Counter Clock Stopped with RB Loading bit in CMRn is written to one (CMRn.LDBSTOP). In Waveform mode it can be stopped by an RC compare event if the Counter Clock Stopped with RC Compare bit in CMRn is written to one (CMRn.CPCSTOP). The start and the stop commands have effect only if the clock is enabled.

Figure 28-3. Clock Control



28.6.1.5 TC operating modes

Each channel can independently operate in two different modes:

- Capture mode provides measurement on signals.
- Waveform mode provides wave generation.

The TC operating mode selection is done by writing to the Wave bit in the CCRn register (CCRn.WAVE).

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.

28.6.1.6 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: each channel has a software trigger, available by writing a one to the Software Trigger Command bit in CCRn (CCRn.SWTRG).
- 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 a one to the Synchro Command bit in the BCR register (BCR.SYNC).
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if the RC Compare Trigger Enable bit in CMRn (CMRn.CPCTRG) is written to one.

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 to be one of the following signals: TIOB, XC0, XC1, or XC2. This external event can then be programmed to perform a trigger by writing a one to the External Event Trigger Enable bit in CMRn (CMRn.ENETRG).

If an external trigger is used, the duration of the pulses must be longer than the CLK_TC 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.

28.6.2 Capture Operating Mode

This mode is entered by writing a zero to the CMRn.WAVE bit.

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 28-4 on page 747 shows the configuration of the TC channel when programmed in Capture mode.

28.6.2.1 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 RA Loading Selection field in CMRn (CMRn.LDRA) defines the TIOA edge for the loading of the RA register, and the RB Loading Selection field in CMRn (CMRn.LDRB) defines the TIOA edge for the loading of the RB register.

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.

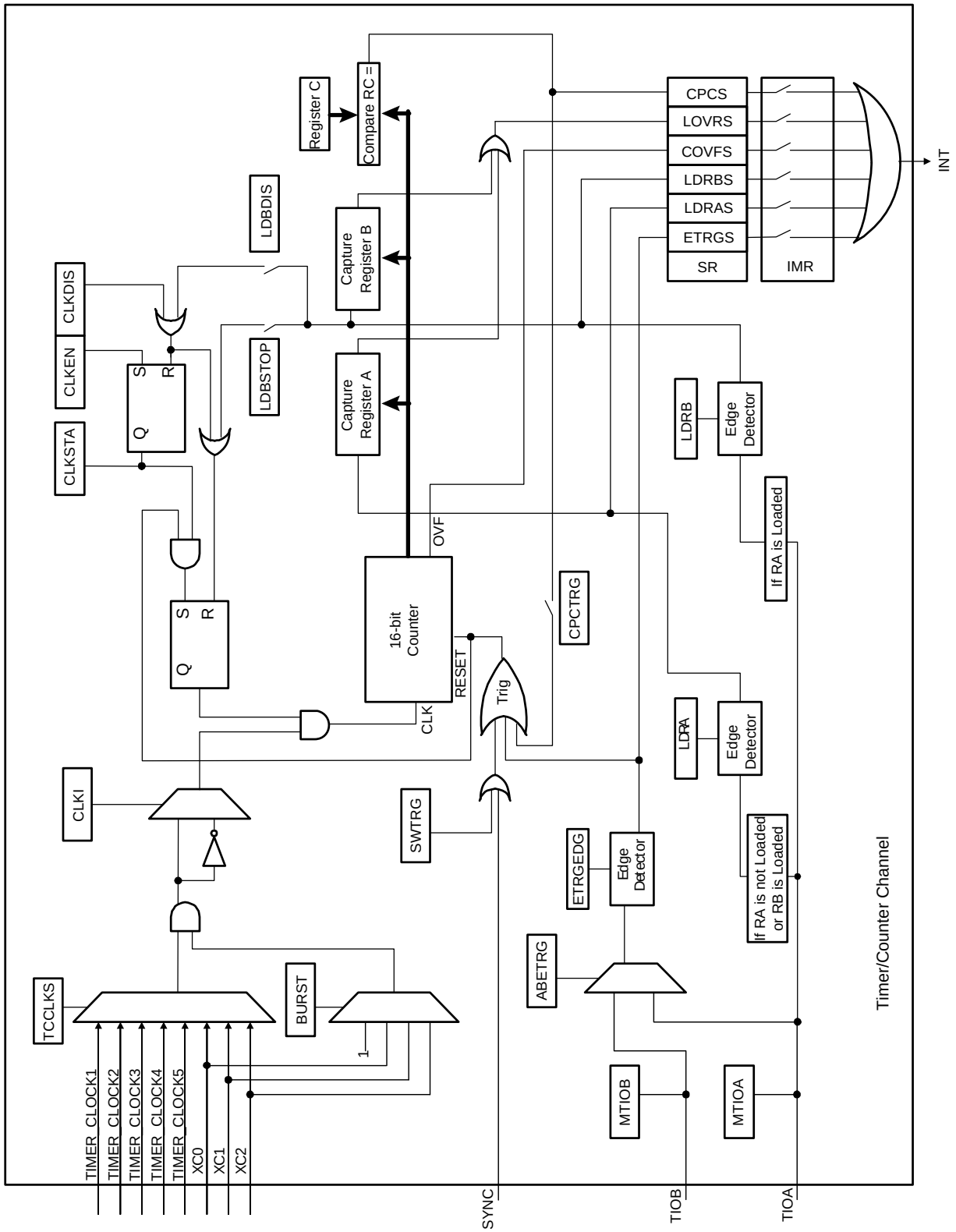
Loading RA or RB before the read of the last value loaded sets the Load Overrun Status bit in SRn (SRn.LOVRS). In this case, the old value is overwritten.

28.6.2.2 Trigger conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The TIOA or TIOB External Trigger Selection bit in CMRn (CMRn.ABETRG) selects TIOA or TIOB input signal as an external trigger. The External Trigger Edge Selection bit in CMRn (CMRn.ETREDG) defines the edge (rising, falling or both) detected to generate an external trigger. If CMRn.ETRGEDG is zero (none), the external trigger is disabled.

Figure 28-4. Capture Mode



28.6.3 Waveform Operating Mode

Waveform operating mode is entered by writing a one to the CMRn.WAVE bit.

In Waveform operating mode the TC channel generates one or two 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.

Figure 28-5 on page 749 shows the configuration of the TC channel when programmed in Waveform operating mode.

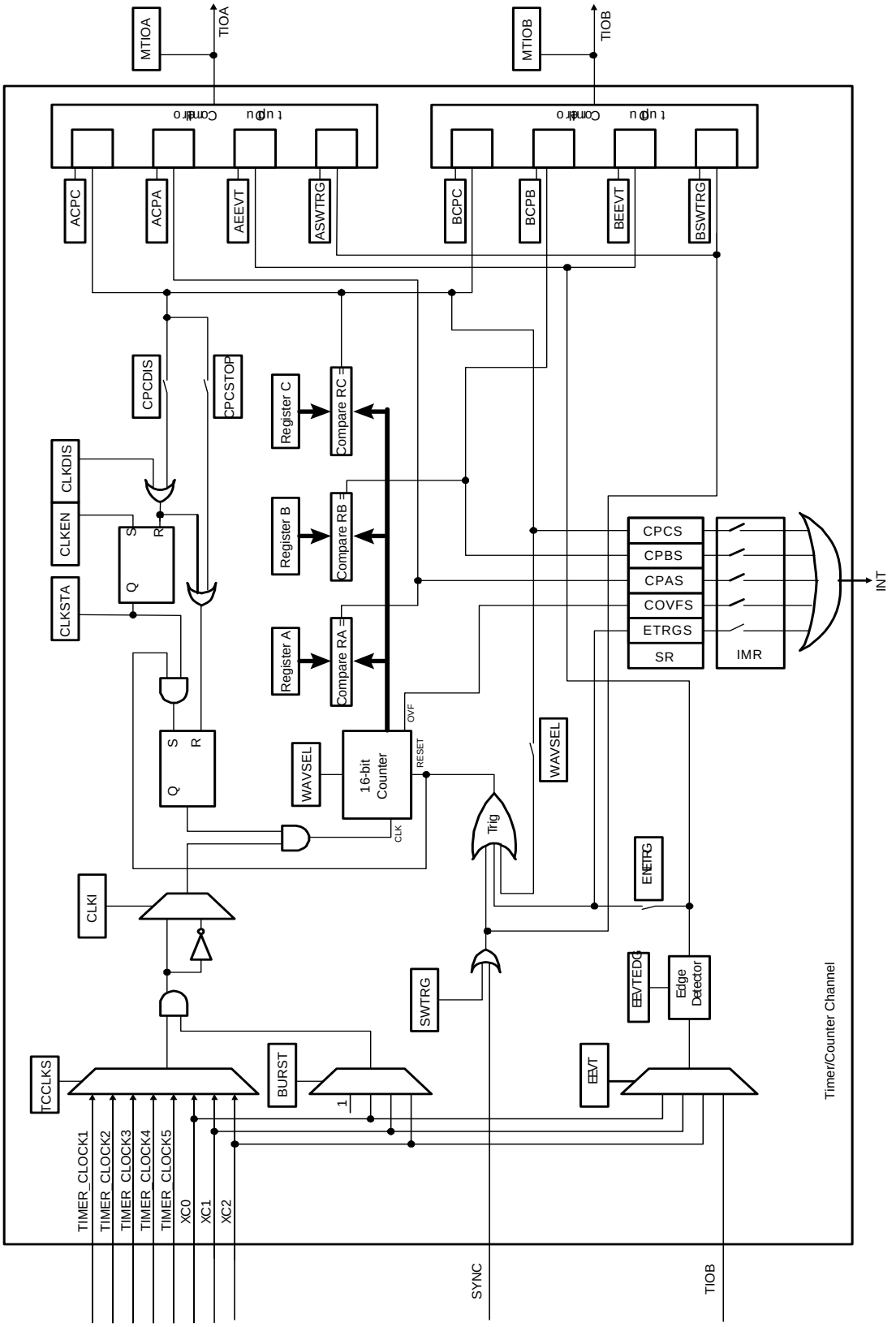
28.6.3.1 Waveform selection

Depending on the Waveform Selection field in CMRn (CMRn.WAVSEL), the behavior of CVn 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 28-5. Waveform Mode



28.6.3.2 WAVSEL = 0

When CMRn.WAVSEL is zero, the value of CVn is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of CVn is reset. Incrementation of CVn starts again and the cycle continues. See Figure 28-6 on page 750.

An external event trigger or a software trigger can reset the value of CVn. It is important to note that the trigger may occur at any time. See Figure 28-7 on page 751.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 28-6. WAVSEL= 0 Without Trigger

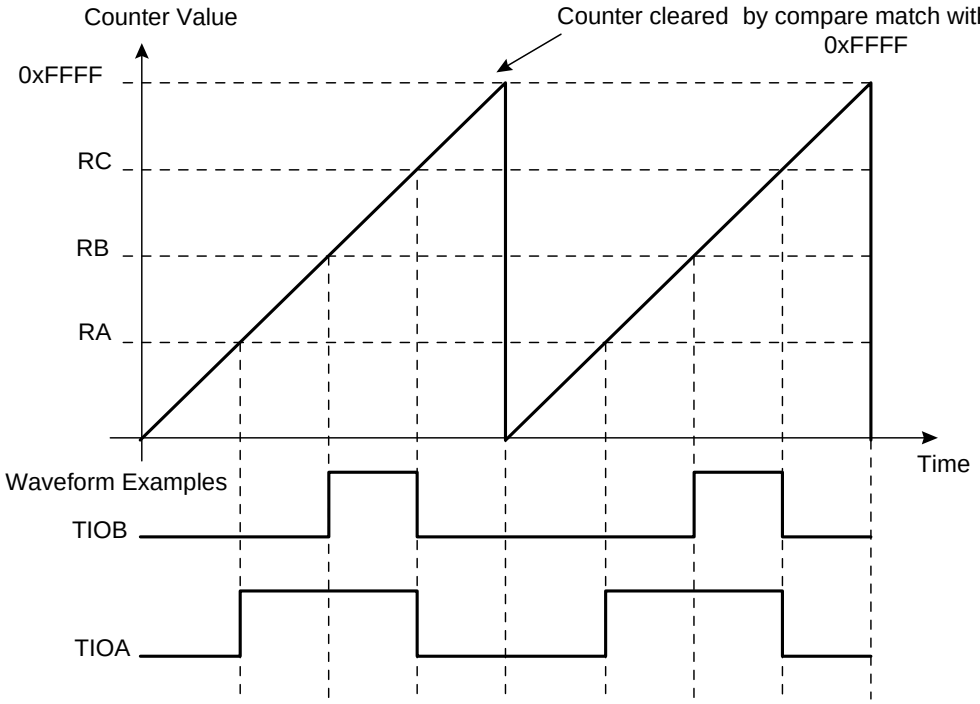
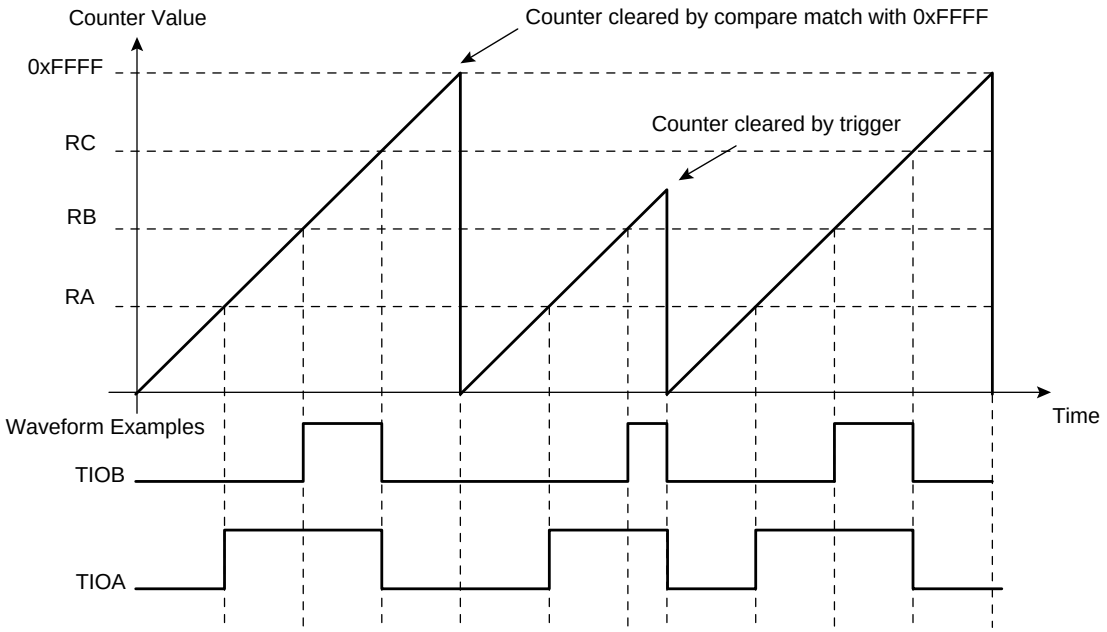


Figure 28-7. WAVSEL= 0 With Trigger



28.6.3.3 *WAVSEL = 2*

When CMRn.WAVSEL is two, the value of CVn is incremented from zero to the value of RC, then automatically reset on a RC Compare. Once the value of CVn has been reset, it is then incremented and so on. See Figure 28-8 on page 752.

It is important to note that CVn can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 28-9 on page 752.

In addition, RC Compare can stop the counter clock (CMRn.CPCSTOP) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 28-8. WAVSEL = 2 Without Trigger

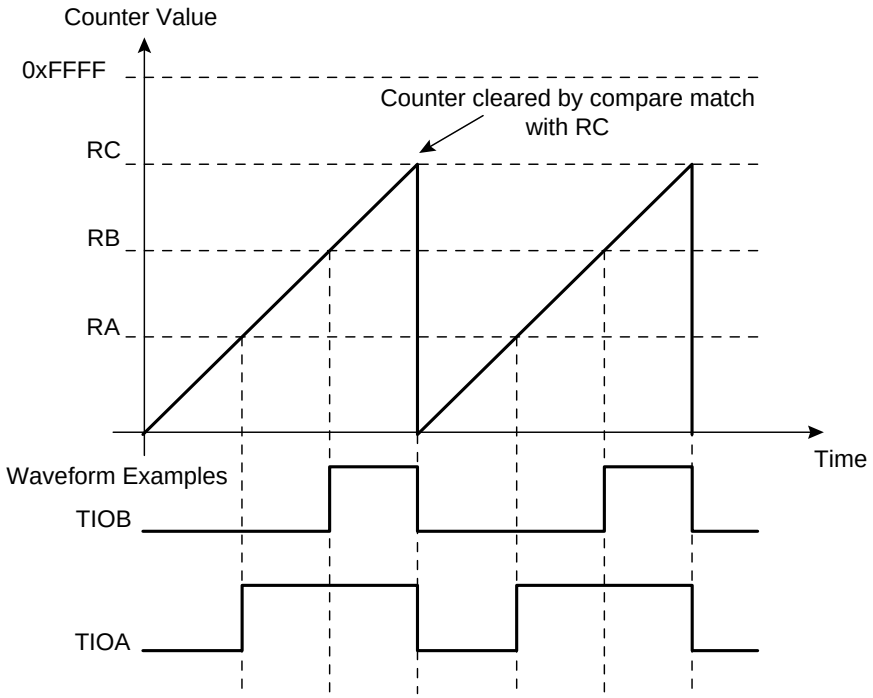
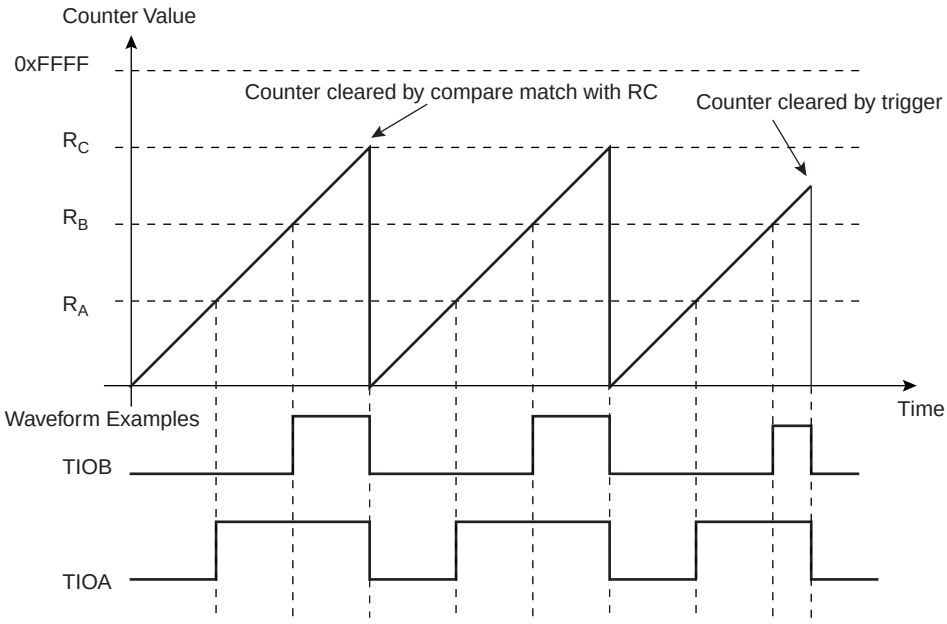


Figure 28-9. WAVSEL = 2 With Trigger



28.6.3.4 WAVSEL = 1

When CMRn.WAVSEL is one, the value of CVn is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of CVn is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 28-10 on page 753.

A trigger such as an external event or a software trigger can modify CVn at any time. If a trigger occurs while CVn is incrementing, CVn then decrements. If a trigger is received while CVn is decrementing, CVn then increments. See Figure 28-11 on page 753.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 28-10. WAVSEL = 1 Without Trigger

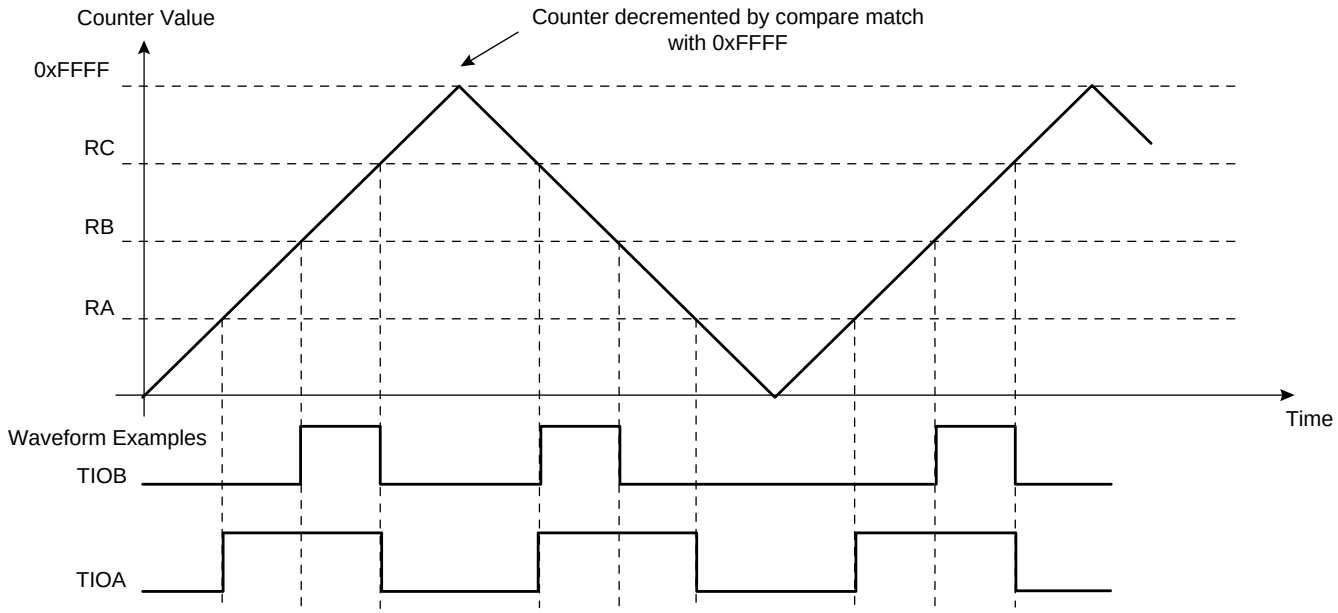
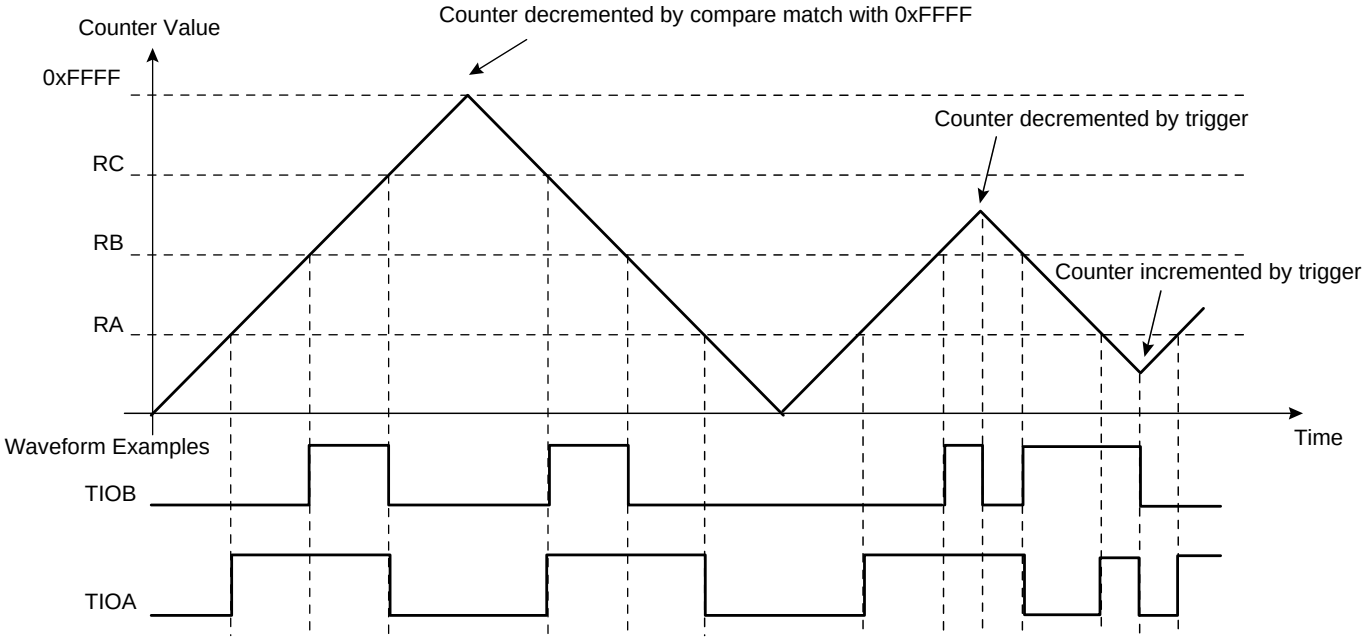


Figure 28-11. WAVSEL = 1 With Trigger



28.6.3.5 *WAVSEL = 3*

When CMRn.WAVSEL is three, the value of CVn is incremented from zero to RC. Once RC is reached, the value of CVn is decremented to zero, then re-incremented to RC and so on. See Figure 28-12 on page 754.

A trigger such as an external event or a software trigger can modify CVn at any time. If a trigger occurs while CVn is incrementing, CVn then decrements. If a trigger is received while CVn is decrementing, CVn then increments. See Figure 28-13 on page 755.

RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 28-12. WAVSEL = 3 Without Trigger

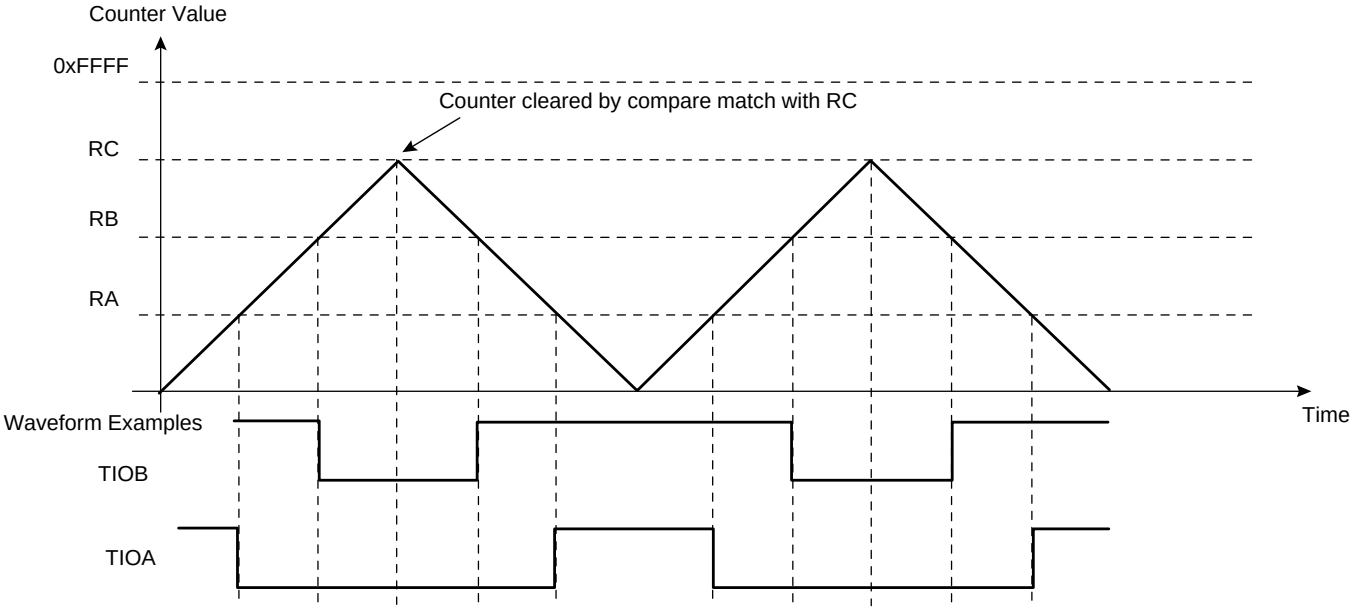
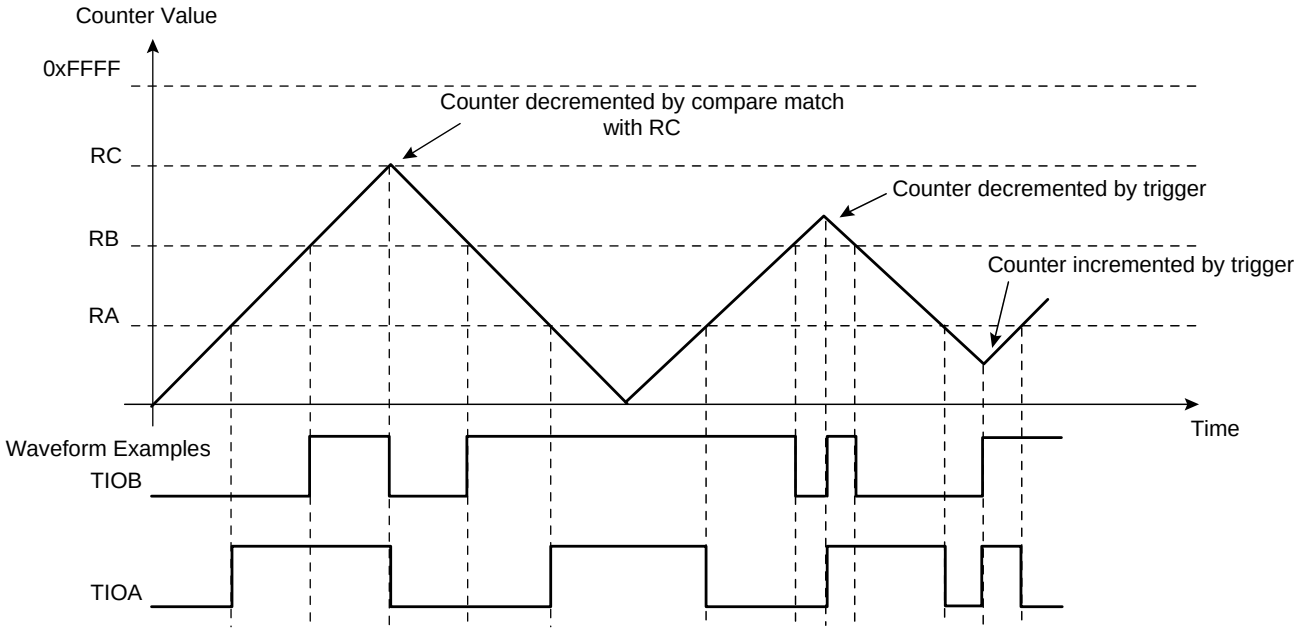


Figure 28-13. WAVSEL = 3 With Trigger



28.6.3.6 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 External Event Selection field in CMRn (CMRn.EEVT) selects the external trigger. The External Event Edge Selection field in CMRn (CMRn.EEVTEDG) defines the trigger edge for each of the possible external triggers (rising, falling or both). If CMRn.EEVTEDG is written to zero, no external event is defined.

If TIOB is defined as an external event signal (CMRn.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 writing a one to the CMRn.ENETRG bit.

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 CMRn.WAVSEL field.

28.6.3.7 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
- 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 following fields in CMRn:

- RC Compare Effect on TIOB (CMRn.BCPC)

- RB Compare Effect on TIOB (CMRn.BCPB)
- RC Compare Effect on TIOA (CMRn.ACPC)
- RA Compare Effect on TIOA (CMRn.ACPA)

28.7 User Interface

Table 28-3. TC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Channel 0 Control Register	CCR0	Write-only	0x00000000
0x04	Channel 0 Mode Register	CMR0	Read/Write	0x00000000
0x10	Channel 0 Counter Value	CV0	Read-only	0x00000000
0x14	Channel 0 Register A	RA0	Read/Write(1)	0x00000000
0x18	Channel 0 Register B	RB0	Read/Write(1)	0x00000000
0x1C	Channel 0 Register C	RC0	Read/Write	0x00000000
0x20	Channel 0 Status Register	SR0	Read-only	00x00000000
0x24	Interrupt Enable Register	IER0	Write-only	0x00000000
0x28	Channel 0 Interrupt Disable Register	IDR0	Write-only	0x00000000
0x2C	Channel 0 Interrupt Mask Register	IMR0	Read-only	0x00000000
0x40	Channel 1 Control Register	CCR1	Write-only	0x00000000
0x44	Channel 1 Mode Register	CMR1	Read/Write	0x00000000
0x50	Channel 1 Counter Value	CV1	Read-only	0x00000000
0x54	Channel 1 Register A	RA1	Read/Write(1)	0x00000000
0x58	Channel 1 Register B	RB1	Read/Write(1)	0x00000000
0x5C	Channel 1 Register C	RC1	Read/Write	0x00000000
0x60	Channel 1 Status Register	SR1	Read-only	0x00000000
0x64	Channel 1 Interrupt Enable Register	IER1	Write-only	0x00000000
0x68	Channel 1 Interrupt Disable Register	IDR1	Write-only	0x00000000
0x6C	Channel 1 Interrupt Mask Register	IMR1	Read-only	0x00000000
0x80	Channel 2 Control Register	CCR2	Write-only	0x00000000
0x84	Channel 2 Mode Register	CMR2	Read/Write	0x00000000
0x90	Channel 2 Counter Value	CV2	Read-only	0x00000000
0x94	Channel 2 Register A	RA2	Read/Write(1)	0x00000000
0x98	Channel 2 Register B	RB2	Read/Write(1)	0x00000000
0x9C	Channel 2 Register C	RC2	Read/Write	0x00000000
0xA0	Channel 2 Status Register	SR2	Read-only	0x00000000
0xA4	Channel 2 Interrupt Enable Register	IER2	Write-only	0x00000000
0xA8	Channel 2 Interrupt Disable Register	IDR2	Write-only	0x00000000
0xAC	Channel 2 Interrupt Mask Register	IMR2	Read-only	0x00000000
0xC0	Block Control Register	BCR	Write-only	0x00000000
0xC4	Block Mode Register	BMR	Read/Write	0x00000000

Notes: 1. Read-only if CMRn.WAVE is zero

28.7.1 Channel Control Register

Name: CCR

Access Type: Write-only

Offset: 0x00 + n * 0x40

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
-	-	-	-	-	SWTRG	CLKDIS	CLKEN

- **SWTRG: Software Trigger Command**
  1: Writing a one to this bit will perform a software trigger: the counter is reset and the clock is started.
  0: Writing a zero to this bit has no effect.
- **CLKDIS: Counter Clock Disable Command**
  1: Writing a one to this bit will disable the clock.
  0: Writing a zero to this bit has no effect.
- **CLKEN: Counter Clock Enable Command**
  1: Writing a one to this bit will enable the clock if CLKDIS is not one.
  0: Writing a zero to this bit has no effect.

28.7.2 Channel Mode Register: Capture Mode

Name: CMR

Access Type: Read/Write

Offset: 0x04 + n * 0x40

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	LDRB		LDRA	

15	14	13	12	11	10	9	8
WAVE	CPCTRG	-	-	-	ABETRG	ETRGEDG	

7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **LDRB: RB Loading Selection**

LDRB	Edge
0	none
1	rising edge of TIOA
2	falling edge of TIOA
3	each edge of TIOA

- **LDRA: RA Loading Selection**

LDRA	Edge
0	none
1	rising edge of TIOA
2	falling edge of TIOA
3	each edge of TIOA

- **WAVE**
  1: Capture mode is disabled (Waveform mode is enabled).
  0: Capture mode is enabled.
- **CPCTRG: RC Compare Trigger Enable**
  1: RC Compare resets the counter and starts the counter clock.
  0: RC Compare has no effect on the counter and its clock.
- **ABETRG: TIOA or TIOB External Trigger Selection**
  1: TIOA is used as an external trigger.

0: TIOB is used as an external trigger.

- **ETRGEDG: External Trigger Edge Selection**

ETRGEDG	Edge
0	none
1	rising edge
2	falling edge
3	each edge

- **LDBDIS: Counter Clock Disable with RB Loading**
  1: Counter clock is disabled when RB loading occurs.
  0: Counter clock is not disabled when RB loading occurs.
- **LDBSTOP: Counter Clock Stopped with RB Loading**
  1: Counter clock is stopped when RB loading occurs.
  0: Counter clock is not stopped when RB loading occurs.
- **BURST: Burst Signal Selection**

BURST	Burst Signal Selection
0	The clock is not gated by an external signal
1	XC0 is ANDed with the selected clock
2	XC1 is ANDed with the selected clock
3	XC2 is ANDed with the selected clock

- **CLKI: Clock Invert**
  1: The counter is incremented on falling edge of the clock.
  0: The counter is incremented on rising edge of the clock.
- **TCCLKS: Clock Selection**

TCCLKS	Clock Selected
0	TIMER_CLOCK1
1	TIMER_CLOCK2
2	TIMER_CLOCK3
3	TIMER_CLOCK4
4	TIMER_CLOCK5
5	XC0
6	XC1
7	XC2

28.7.3 Channel Mode Register: Waveform Mode

Name: CMR

Access Type: Read/Write

Offset: 0x04 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	

23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	

15	14	13	12	11	10	9	8
WAVE	WAVSEL		ENETRG	EEVT		EEVTEDG	

7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **BSWTRG: Software Trigger Effect on TIOB**

BSWTRG	Effect
0	none
1	set
2	clear
3	toggle

- **BEEVT: External Event Effect on TIOB**

BEEVT	Effect
0	none
1	set
2	clear
3	toggle

- **BCPC: RC Compare Effect on TIOB**

BCPC	Effect
0	none
1	set
2	clear
3	toggle

- **BCPB: RB Compare Effect on TIOB**

BCPB	Effect
0	none
1	set
2	clear
3	toggle

- **ASWTRG: Software Trigger Effect on TIOA**

ASWTRG	Effect
0	none
1	set
2	clear
3	toggle

- **AEEVT: External Event Effect on TIOA**

AEEVT	Effect
0	none
1	set
2	clear
3	toggle

- **ACPC: RC Compare Effect on TIOA**

ACPC	Effect
0	none
1	set
2	clear
3	toggle

- **ACPA: RA Compare Effect on TIOA**

ACPA	Effect
0	none
1	set
2	clear
3	toggle

- **WAVE**
  1: Waveform mode is enabled.
  0: Waveform mode is disabled (Capture mode is enabled).
- **WAVSEL: Waveform Selection**

WAVSEL	Effect
0	UP mode without automatic trigger on RC Compare
1	UPDOWN mode without automatic trigger on RC Compare
2	UP mode with automatic trigger on RC Compare
3	UPDOWN mode with automatic trigger on RC Compare

- **ENETRG: External Event Trigger Enable**
  1: The external event resets the counter and starts the counter clock.
  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.
- **EEVT: External Event Selection**

EEVT	Signal selected as external event	TIOB Direction
0	TIOB	input(1)
1	XC0	output
2	XC1	output
3	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.*

- **EEVTEDG: External Event Edge Selection**

EEVTEDG	Edge
0	none
1	rising edge
2	falling edge
3	each edge

- **CPCDIS: Counter Clock Disable with RC Compare**
  1: Counter clock is disabled when counter reaches RC.
  0: Counter clock is not disabled when counter reaches RC.
- **CPCSTOP: Counter Clock Stopped with RC Compare**
  1: Counter clock is stopped when counter reaches RC.

0: Counter clock is not stopped when counter reaches RC.

- **BURST: Burst Signal Selection**

BURST	Burst Signal Selection
0	The clock is not gated by an external signal.
1	XC0 is ANDed with the selected clock.
2	XC1 is ANDed with the selected clock.
3	XC2 is ANDed with the selected clock.

- **CLKI: Clock Invert**

1: Counter is incremented on falling edge of the clock.

0: Counter is incremented on rising edge of the clock.

- **TCCLKS: Clock Selection**

TCCLKS	Clock Selected
0	TIMER_CLOCK1
1	TIMER_CLOCK2
2	TIMER_CLOCK3
3	TIMER_CLOCK4
4	TIMER_CLOCK5
5	XC0
6	XC1
7	XC2

28.7.4 Channel Counter Value Register

Name: CV

Access Type: Read-only

Offset: 0x10 + n * 0x40

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
CV[15:8]							

7	6	5	4	3	2	1	0
CV[7:0]							

- **CV: Counter Value**
  CV contains the counter value in real time.

28.7.5 Channel Register A

Name: RA

Access Type: Read-only if CMRn.WAVE = 0, Read/Write if CMRn.WAVE = 1

Offset: 0x14 + n * 0X40

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
RA[15:8]							

7	6	5	4	3	2	1	0
RA[7:0]							

- **RA: Register A**
  RA contains the Register A value in real time.

28.7.6 Channel Register B

Name: RB

Access Type: Read-only if CMRn.WAVE = 0, Read/Write if CMRn.WAVE = 1

Offset: 0x18 + n * 0x40

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
RB[15:8]							

7	6	5	4	3	2	1	0
RB[7:0]							

- **RB: Register B**
  RB contains the Register B value in real time.

28.7.7 Channel Register C

Name: RC

Access Type: Read/Write

Offset: 0x1C + n * 0x40

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
RC[15:8]							

7	6	5	4	3	2	1	0
RC[7:0]							

- **RC: Register C**
  RC contains the Register C value in real time.

28.7.8 Channel Status Register

Name: SR

Access Type: Read-only

Offset: 0x20 + n * 0x40

Reset Value: 0x00000000

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

Note: Reading the Status Register will also clear the interrupt bit for the corresponding interrupts.

- **MTIOB: TIOB Mirror**
  1: TIOB is high. If CMRn.WAVE is zero, this means that TIOB pin is high. If CMRn.WAVE is one, this means that TIOB is driven high.
  0: TIOB is low. If CMRn.WAVE is zero, this means that TIOB pin is low. If CMRn.WAVE is one, this means that TIOB is driven low.
- **MTIOA: TIOA Mirror**
  1: TIOA is high. If CMRn.WAVE is zero, this means that TIOA pin is high. If CMRn.WAVE is one, this means that TIOA is driven high.
  0: TIOA is low. If CMRn.WAVE is zero, this means that TIOA pin is low. If CMRn.WAVE is one, this means that TIOA is driven low.
- **CLKSTA: Clock Enabling Status**
  1: This bit is set when the clock is enabled.
  0: This bit is cleared when the clock is disabled.
- **ETRGS: External Trigger Status**
  1: This bit is set when an external trigger has occurred.
  0: This bit is cleared when the SR register is read.
- **LDRBS: RB Loading Status**
  1: This bit is set when an RB Load has occurred and CMRn.WAVE is zero.
  0: This bit is cleared when the SR register is read.
- **LDRAS: RA Loading Status**
  1: This bit is set when an RA Load has occurred and CMRn.WAVE is zero.
  0: This bit is cleared when the SR register is read.
- **CPCS: RC Compare Status**
  1: This bit is set when an RC Compare has occurred.
  0: This bit is cleared when the SR register is read.

- **CPBS: RB Compare Status**
  1: This bit is set when an RB Compare has occurred and CMRn.WAVE is one.
  0: This bit is cleared when the SR register is read.
- **CPAS: RA Compare Status**
  1: This bit is set when an RA Compare has occurred and CMRn.WAVE is one.
  0: This bit is cleared when the SR register is read.
- **LOVRS: Load Overrun Status**
  1: This bit is set when RA or RB have been loaded at least twice without any read of the corresponding register and CMRn.WAVE is zero.
  0: This bit is cleared when the SR register is read.
- **COVFS: Counter Overflow Status**
  1: This bit is set when a counter overflow has occurred.
  0: This bit is cleared when the SR register is read.

28.7.9 Channel Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x24 + n * 0x40

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
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

28.7.10 Channel Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x28 + n * 0x40

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
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

28.7.11 Channel Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x2C + n * 0x40

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
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in IDR is written to one.

A bit in this register is set when the corresponding bit in IER is written to one.

28.7.12 Block Control Register

Name: BCR

Access Type: Write-only

Offset: 0xC0

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
-	-	-	-	-	-	-	SYNC

- **SYNC: Synchro Command**
  1: Writing a one to this bit asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.
  0: Writing a zero to this bit has no effect.

28.7.13 Block Mode Register

Name: BMR

Access Type: Read/Write

Offset: 0xC4

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
-	-	TC2XC2S		TC1XC1S		TC0XC0S	

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S	Signal Connected to XC2
0	TCLK2
1	none
2	TIOA0
3	TIOA1

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S	Signal Connected to XC1
0	TCLK1
1	none
2	TIOA0
3	TIOA2

- **TC0XC0S: External Clock Signal 0 Selection**

TC0XC0S	Signal Connected to XC0
0	TCLK0

1	none
2	TIOA1
3	TIOA2

28.8 Module Configuration

The specific configuration for each TC instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 28-4. Module Clock Name

Module name	Clock name
TC0	CLK_TC0
TC1	CLK_TC1

28.8.1 Clock Connections

Each Timer/Counter channel can independently select an internal or external clock source for its counter:

Table 28-5. Timer/Counter Internal Clock Connections

Name	Connection
TIMER_CLOCK1	32 KHz clock
TIMER_CLOCK2	PBA Clock / 2
TIMER_CLOCK3	PBA Clock / 8
TIMER_CLOCK4	PBA Clock / 32
TIMER_CLOCK5	PBA Clock / 128

29. Analog-to-Digital Converter (ADC)

Rev: 2.0.0.1

29.1 Features

- **Integrated multiplexer offering up to eight independent analog inputs**
- **Individual enable and disable of each channel**
- **Hardware or software trigger**
  - **External trigger pin**
  - **Timer counter outputs (corresponding TIOA trigger)**
- **PDC support**
- **Possibility of ADC timings configuration**
- **Sleep mode and conversion sequencer**
  - **Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels**

29.2 Overview

The Analog-to-Digital Converter (ADC) is based on a Successive Approximation Register (SAR) 10-bit ADC. It also integrates an ADC_NB_CHANNELS-to-1 analog multiplexer, making possible the analog-to-digital conversions of ADC_NB_CHANNELS analog lines. The conversions extend from 0V to VDDANA.

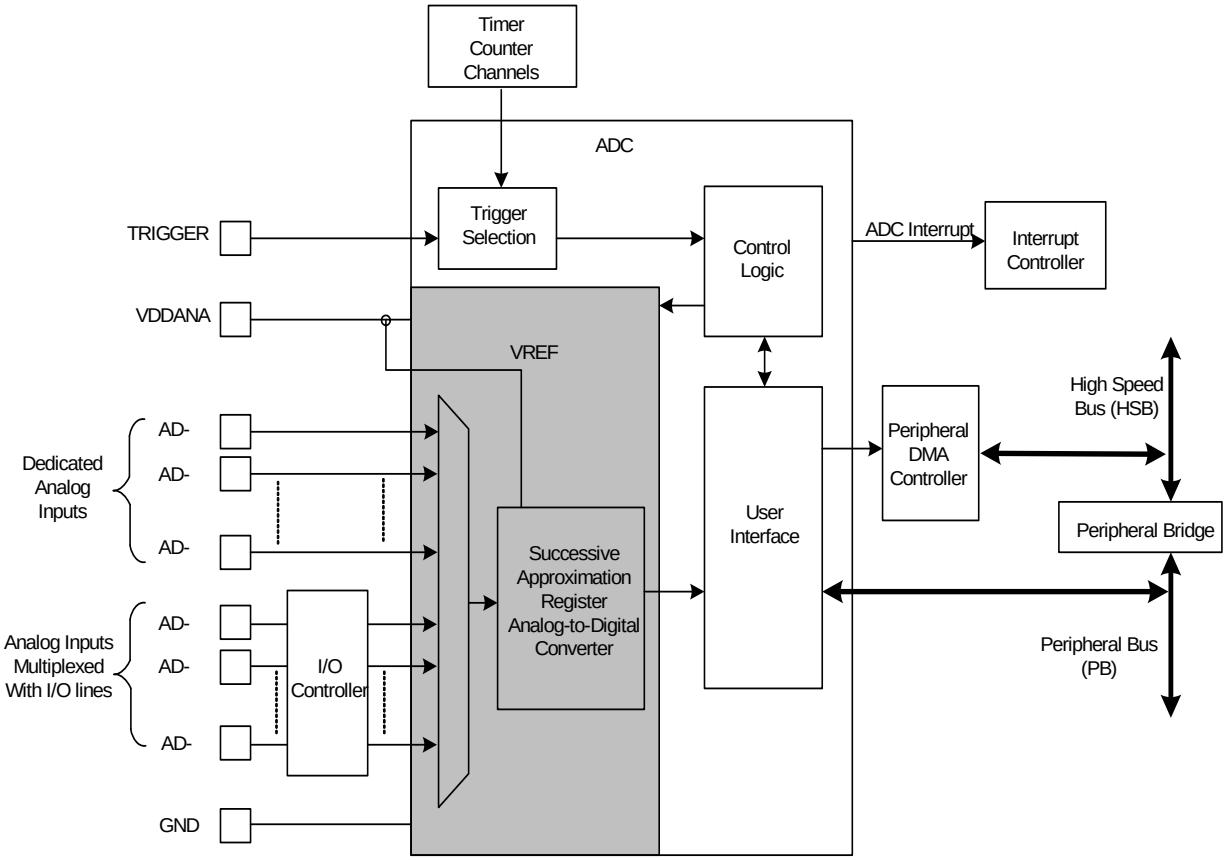
The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the TRIGGER pin, or internal triggers from timer counter output(s) are configurable.

The ADC also integrates a sleep mode and a conversion sequencer and connects with a Peripheral DMA Controller 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.

29.3 Block Diagram

Figure 29-1. ADC Block Diagram



29.4 I/O Lines Description

Table 29-1. ADC Pins Description

Pin Name	Description
VDDANA	Analog power supply
AD[0] - AD[ADC_NB_CHANNELS-1]	Analog input channels
TRIGGER	External trigger

29.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

29.5.1 I/O Lines

The TRIGGER pin may be shared with other peripheral functions through the I/O Controller. Refer to the Peripheral Event System chapter for details

29.5.2 Power Management

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 Manager has no effect on the ADC behavior.

29.5.3 Clocks

The clock for the ADC bus interface (CLK_ADC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the ADC before disabling the clock, to avoid freezing the ADC in an undefined state.

29.5.4 Interrupts

The ADC interrupt request line is connected to the interrupt controller. Using the ADC interrupt requires the interrupt controller to be programmed first.

29.5.5 Analog Inputs

The analog input pins can be multiplexed with I/O lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing a one to the corresponding bit in the Channel Enable Register (CHER). By default, after reset, the I/O line is configured as input with its pull-up enabled and the ADC input is connected to the ground.

29.5.6 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.

29.6 Functional Description

29.6.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires sample and hold clock cycles as defined in the Sample and Hold Time field of the Mode Register (MR.SHTIM) and 10 ADC Clock cycles. The ADC Clock frequency is selected in the Prescaler Rate Selection field of the MR register (MR.PRESCAL).

The ADC Clock range is between CLK_ADC/2, if the PRESCAL field is 0, and CLK_ADC/128, if the PRESCAL field is 63 (0x3F). The PRESCAL field must be written in order to provide an ADC Clock frequency according to the parameters given in the Electrical Characteristics chapter.

29.6.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage connected to VDDANA. Analog input values between these voltages are converted to digital values based on a linear conversion.

29.6.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by writing a one to the Resolution bit in the MR register (MR.LOWRES). By default, after a reset, the resolution is the highest and the Converted Data field in the Channel Data Registers (CDRn.DATA) is fully used. By writing a one to the LOWRES bit, the ADC switches in the lowest resolution and the conversion results can be read in the eight lowest significant bits of the Channel Data Registers (CDRn). The two highest bits of the DATA field in the corresponding CDRn register will be read as zero. The two highest bits of the Last Data Converted field in the Last Converted Data Register (LCDR.LDATA) will be read as zero too.

Moreover, when a Peripheral DMA channel is connected to the ADC, a 10-bit resolution sets the transfer request size to 16-bit. Writing a one to the LOWRES bit automatically switches to 8-bit data transfers. In this case, the destination buffers are optimized.

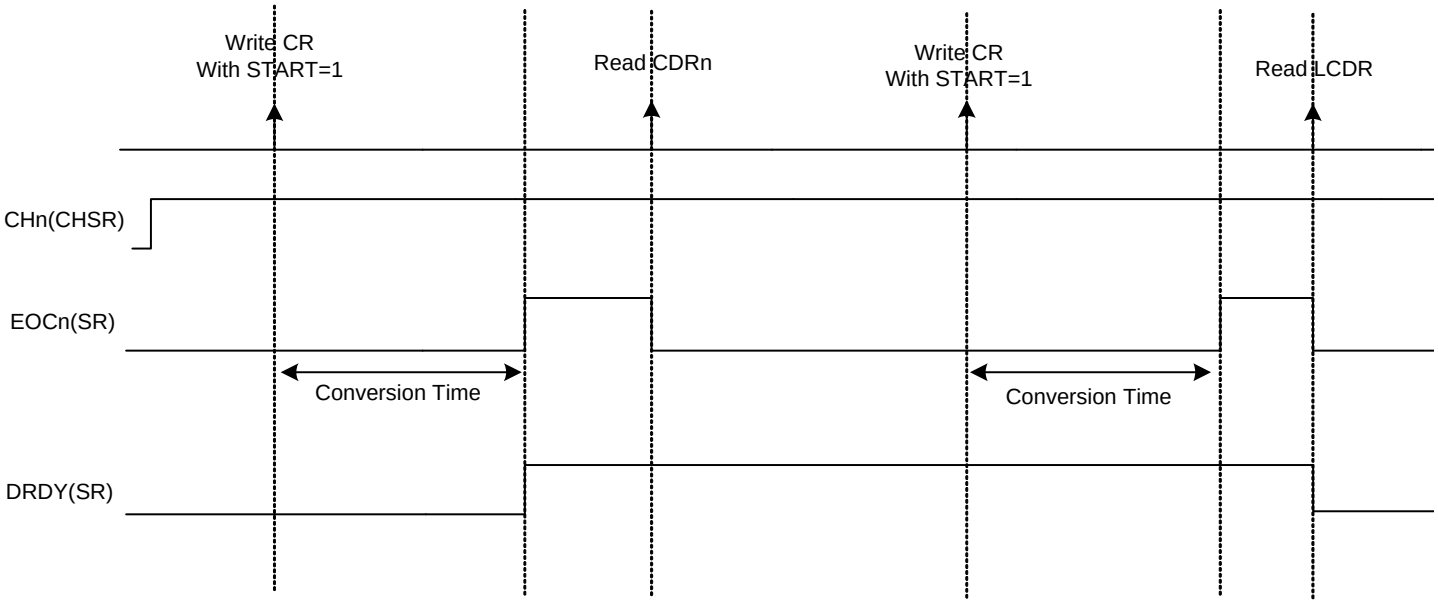
29.6.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the CDR register of the current channel and in the LCDR register. Channels are enabled by writing a one to the Channel n Enable bit (CHn) in the CHER register.

The corresponding channel End of Conversion bit in the Status Register (SR.EOCn) and the Data Ready bit in the SR register (SR.DRDY) are set. In the case of a connected Peripheral DMA channel, DRDY rising triggers a data transfer request. In any case, either EOC or DRDY can trigger an interrupt.

Reading one of the CDRn registers clears the corresponding EOC bit. Reading LCDR clears the DRDY bit and the EOC bit corresponding to the last converted channel.

Figure 29-2. EOCn and DRDY Flag Behavior

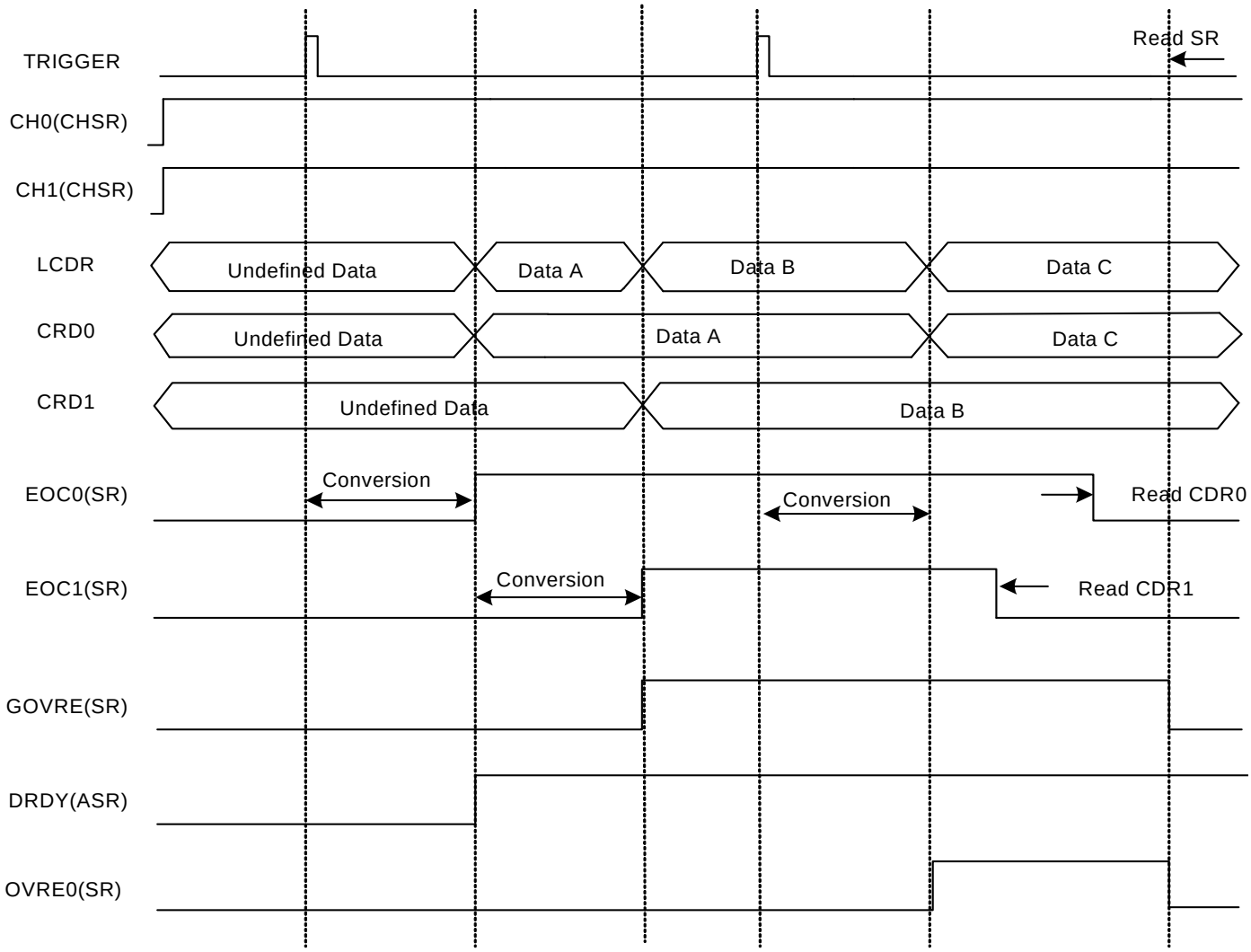


If the CDR register is not read before further incoming data is converted, the corresponding Overrun Error bit in the SR register (SR.OVREn) is set.

In the same way, new data converted when DRDY is high sets the General Overrun Error bit in the SR register (SR.GOVRE).

The OVREn and GOVRE bits are automatically cleared when the SR register is read.

Figure 29-3. GOVRE and OVREn 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 SR are unpredictable.

29.6.5 Conversion Triggers

Conversions of the active analog channels are started with a software or a hardware trigger. The software trigger is provided by writing a one to the START bit in the Control Register (CR.START).

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, or the external trigger input of the ADC (TRIGGER). The hardware trigger is selected with the Trigger Selection field in the Mode Register (MR.TRIGSEL). The selected hardware trigger is enabled by writing a one to the Trigger Enable bit in the Mode Register (MR.TRGEN).

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 (CHER) and Channel Disable (CHDR) Registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a Peripheral DMA Controller, 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.

29.6.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by writing a one to the Sleep Mode bit in the Mode Register (MR.SLEEP).

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 Peripheral DMA Controller.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

29.6.7 ADC Timings

Each ADC has its own minimal startup time that is defined through the Start Up Time field in the Mode Register (MR.STARTUP). This startup time is given in the Electrical Characteristics chapter.

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 defined through the Sample and Hold Time field in the Mode Register (MR.SHTIM). This time depends on the input impedance of the analog input, but also on the output impedance of the driver providing the signal to the analog input, as there is no input buffer amplifier.

29.6.8 Conversion Performances

For performance and electrical characteristics of the ADC, see the Electrical Characteristics chapter.

29.7 User Interface

Table 29-2. ADC Register Memory Map

Offset	Register	Name	Access	Reset State
0x00	Control Register	CR	Write-only	0x00000000
0x04	Mode Register	MR	Read/Write	0x00000000
0x10	Channel Enable Register	CHER	Write-only	0x00000000
0x14	Channel Disable Register	CHDR	Write-only	0x00000000
0x18	Channel Status Register	CHSR	Read-only	0x00000000
0x1C	Status Register	SR	Read-only	0x000C0000
0x20	Last Converted Data Register	LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	IER	Write-only	0x00000000
0x28	Interrupt Disable Register	IDR	Write-only	0x00000000
0x2C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x30	Channel Data Register 0	CDR0	Read-only	0x00000000
...	...(if implemented)	...	...	...
0x4C	Channel Data Register 7(if implemented)	CDR7	Read-only	0x00000000
0xFC	Version Register	VERSION	Read-only	- (1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

29.7.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

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
–	–	–	–	–	–	START	SWRST

- **START: Start Conversion**
  Writing a one to this bit will begin an analog-to-digital conversion.
  Writing a zero to this bit has no effect.
  This bit always reads zero.
- **SWRST: Software Reset**
  Writing a one to this bit will reset the ADC.
  Writing a zero to this bit has no effect.
  This bit always reads zero.

29.7.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

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

- **SHTIM: Sample & Hold Time**
  Sample & Hold Time = (SHTIM+1) / ADCClock
- **STARTUP: Start Up Time**
  Startup Time = (STARTUP+1) * 8 / ADCClock
- **PRESCAL: Prescaler Rate Selection**
  ADCClock = CLK_ADC / ((PRESCAL+1) * 2)
- **SLEEP: Sleep Mode**
  1: Sleep Mode is selected.
  0: Normal Mode is selected.
- **LOWRES: Resolution**
  1: 8-bit resolution is selected.
  0: 10-bit resolution is selected.
- **TRGSEL: Trigger Selection**

TRGSEL			Selected TRGSEL
0	0	0	Internal Trigger 0, depending of chip integration
0	0	1	Internal Trigger 1, depending of chip integration
0	1	0	Internal Trigger 2, depending of chip integration
0	1	1	Internal Trigger 3, depending of chip integration
1	0	0	Internal Trigger 4, depending of chip integration
1	0	1	Internal Trigger 5, depending of chip integration
1	1	0	External trigger

- **TRGEN: Trigger Enable**
  1: The hardware trigger selected by the TRGSEL field is enabled.
  0: The hardware triggers are disabled. Starting a conversion is only possible by software.

29.7.3 Channel Enable Register

Name: CHER

Access Type: Write-only

Offset: 0x10

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
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel n Enable**
  Writing a one to these bits will set the corresponding bit in CHSR.
  Writing a zero to these bits has no effect.
  These bits always read a zero.

29.7.4 Channel Disable Register

Name: CHDR

Access Type: Write-only

Offset: 0x14

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
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel n Disable**

  Writing a one to these bits will clear the corresponding bit in CHSR.
  Writing a zero to these bits has no effect.
  These bits always read a zero.

  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 SR are unpredictable.

29.7.5 Channel Status Register

Name: CHSR

Access Type: Read-only

Offset: 0x18

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
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel n Status**

  These bits are set when the corresponding bits in CHER is written to one.

  These bits are cleared when the corresponding bits in CHDR is written to one.

  1: The corresponding channel is enabled.

  0: The corresponding channel is disabled.

29.7.6 Status Register

Name: SR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x000C0000

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

- **RXBUFF: RX Buffer Full**
  This bit is set when the Buffer Full signal from the Peripheral DMA is active.
  This bit is cleared when the Buffer Full signal from the Receive Peripheral DMA is inactive.
- **ENDRX: End of RX Buffer**
  This bit is set when the End Receive signal from the Peripheral DMA is active.
  This bit is cleared when the End Receive signal from the Peripheral DMA is inactive.
- **GOVRE: General Overrun Error**
  This bit is set when a General Overrun Error has occurred.
  This bit is cleared when the SR register is read.
  1: At least one General Overrun Error has occurred since the last read of the SR register.
  0: No General Overrun Error occurred since the last read of the SR register.
- **DRDY: Data Ready**
  This bit is set when a data has been converted and is available in the LCDR register.
  This bit is cleared when the LCDR register is read.
  0: No data has been converted since the last read of the LCDR register.
  1: At least one data has been converted and is available in the LCDR register.
- **OVREn: Overrun Error n**
  These bits are set when an overrun error on the corresponding channel has occurred (if implemented).
  These bits are cleared when the SR register is read.
  0: No overrun error on the corresponding channel (if implemented) since the last read of SR.
  1: There has been an overrun error on the corresponding channel (if implemented) since the last read of SR.
- **EOCn: End of Conversion n**
  These bits are set when the corresponding conversion is complete.
  These bits are cleared when the corresponding CDR or LCDR registers are read.
  0: Corresponding analog channel (if implemented) is disabled, or the conversion is not finished.
  1: Corresponding analog channel (if implemented) is enabled and conversion is complete.

29.7.7 Last Converted Data Register

Name: LCDR

Access Type: Read-only

Offset: 0x20

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
–	–	–	–	–	–	LDATA[9:8]	

7	6	5	4	3	2	1	0
LDATA[7:0]							

- **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.

29.7.8 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x24

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

29.7.9 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x28

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

29.7.10 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x2C

Reset Value: 0x00000000

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

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in IDR is written to one.

A bit in this register is cleared when the corresponding bit in IER is written to one.

29.7.11 Channel Data Register

Name: CDRx

Access Type: Read-only

Offset: 0x2C-0x4C

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
–	–	–	–	–	–	DATA[9:8]	

7	6	5	4	3	2	1	0
DATA[7:0]							

- **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.

29.7.12 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: –

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

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

15	14	13	12	11	10	9	8
–	–	–	–	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

29.8 Module Configuration

The specific configuration for the ADC instance is listed in the following tables.The modPower Manager section.

Table 29-3. Module configuration

Feature	ADC0
ADC_NUM_CHANNELS	8
Internal Trigger 0	TIOA Ouput A of the Timer Counter Channel 0
Internal Trigger 1	TIOB Ouput B of the Timer Counter Channel 0
Internal Trigger 2	TIOA Ouput A of the Timer Counter Channel 1
Internal Trigger 3	TIOB Ouput B of the Timer Counter Channel 1
Internal Trigger 4	TIOA Ouput A of the Timer Counter Channel 2
Internal Trigger 5	TIOBOuput B of the Timer Counter Channel 2

Table 29-4. Module Clock Name

Module name	Clock name
ADC0	CLK_ADC0

Table 29-5. Register Reset Values

Module name	Reset Value
VERSION	0x00000200

30. HSB Bus Performance Monitor (BUSMON)

Rev 1.0.0.0

30.1 Features

- **Allows performance monitoring of High Speed Bus master interfaces**
  - **Up to 4 masters can be monitored**
  - **Peripheral Bus access to monitor registers**
- **The following is monitored**
  - **Data transfer cycles**
  - **Bus stall cycles**
  - **Maximum access latency for a single transfer**
- **Automatic handling of event overflow**

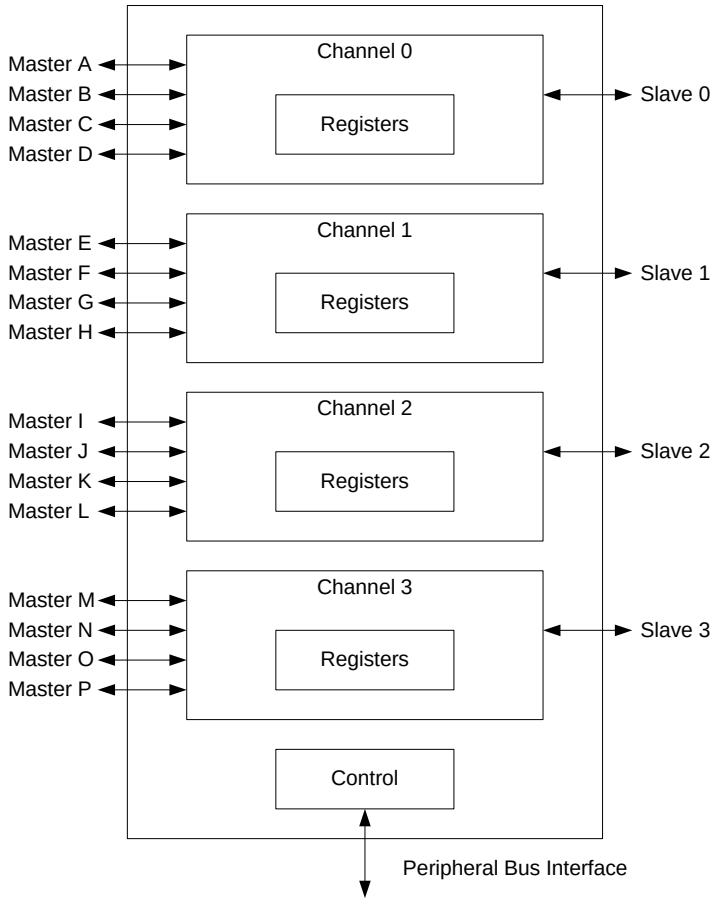
30.2 Overview

BUSMON allows the user to measure the activity and stall cycles on the High Speed Bus (HSB).

Up to 4 device-specific masters can be measured. Each of these masters is part of a measurement channel. Which masters that are connected to a channel is device-specific. Devices may choose not to implement all channels.

30.3 Block Diagram

Figure 30-1. BUSMON Block Diagram



30.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

30.4.1 Clocks

The clock for the BUSMON bus interface (CLK_BUSMON) is generated by the Power Manager. This clock is enabled at reset and can be disabled in the Power Manager. It is recommended to disable the BUSMON before disabling the clock, to avoid freezing the BUSMON in an undefined state.

30.5 Functional Description

Three different parameters can be measured by each channel:

- The number of data transfer cycles since last channel reset
- The number of stall cycles since last channel reset
- The maximum continuous number of stall cycles since last channel reset (This approximates the max latency in the transfers.)

These measurements can be extracted by software and used to generate indicators for bus latency, bus load and maximum bus latency.

Each of the counters have a fixed width, and may therefore overflow. When overflow is encountered in either the Channel n Data Cycles (DATAn) register or the Channel n Stall Cycles (STALLn) register of a channel, all registers in the channel are reset. This behavior is altered if the Channel n Overflow Freeze (CHnOF) bit is set in the Control (CONTROL) register. If this bit is written to one, the channel registers are frozen when either DATAn or STALLn reaches its maximum value. This simplifies one-shot readout of the counter values.

The registers can also be manually reset by writing to the CONTROL register. The Channeln Max Initiation Latency (LATn) register is saturating, when its max count is reached, it will be set to its maximum value. The LATn register is reset whenever DATAn and STALLn are reset.

A counter must manually be enabled by writing to the CONTROL register.

30.6 User interface

Table 30-1. BUSMON Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control register	CONTROL	Read/Write	0x00000000
0x10	Channel0 Data Cycles register	DATA0	Read	0x00000000
0x14	Channel0 Stall Cycles register	STALL0	Read	0x00000000
0x18	Channel0 Max Initiation Latency register	LAT0	Read	0x00000000
0x20	Channel1 Data Cycles register	DATA1	Read	0x00000000
0x24	Channel1 Stall Cycles register	STALL1	Read	0x00000000
0x28	Channel1 Max Initiation Latency register	LAT1	Read	0x00000000
0x30	Channel2 Data Cycles register	DATA2	Read	0x00000000
0x34	Channel2 Stall Cycles register	STALL2	Read	0x00000000
0x38	Channel2 Max Initiation Latency register	LAT2	Read	0x00000000
0x40	Channel3 Data Cycles register	DATA3	Read	0x00000000
0x44	Channel3 Stall Cycles register	STALL3	Read	0x00000000
0x48	Channel3 Max Initiation Latency register	LAT3	Read	0x00000000
0x50	Parameter register	PARAMETER	Read	-(1)
0x54	Version register	VERSION	Read	-(1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

30.6.1 Control Register

Name: CONTROL

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	CH3RES	CH2RES	CH1RES	CH0RES

15	14	13	12	11	10	9	8
-	-	-	-	CH3OF	CH2OF	CH1OF	CH0OF

7	6	5	4	3	2	1	0
-	-	-	-	CH3EN	CH2EN	CH1EN	CH0EN

- **CHnRES: Channel Counter Reset**
  Writting a one to this bit will reset the counter in the channel n.
  Writting a zero to this bit has no effect.
  This bit always reads as zero.
- **CHnOF: Channel Overflow Freeze**
  1: All channel n registers are frozen just before DATA or STALL overflows.
  0: The channel n registers are reset if DATA or STALL overflows.
- **CHnEN: Channel Enabled**
  1: The channel n is enabled.
  0: The channel n is disabled.

30.6.2 Channel n Data Cycles Register

Name: DATAn

Access Type: Read-Only

Offset: 0x10 + n*0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA:**
  Data cycles counted since the last reset.

30.6.3 Channel n Stall Cycles Register

Name: STALLn

Access Type: Read-Only

Offset: 0x14 + n*0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL:**
  Stall cycles counted since the last reset.

30.6.4 Channel n Max Transfer Initiation Cycles Register

Name: LATn

Access Type: Read-Only

Offset: 0x18 + n*0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
LAT[31:24]							

23	22	21	20	19	18	17	16
LAT[23:16]							

15	14	13	12	11	10	9	8
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT:**
  This field is cleared whenever the DATA or STALL register is reset.
  Maximum transfer initiation cycles counted since the last reset.
  This counter is saturating.

30.6.5 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x50

Reset Value: -

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	CH3IMPL	CH2IMPL	CH1IMPL	CH0IMPL

- **CHnIMP: Channel Implementation**
  1: The corresponding channel is implemented.
  0: The corresponding channel is not implemented.

30.6.6 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x54

Reset Value: -

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

30.7 Module Configuration

Table 30-2. Register Reset Values

Register	Reset Value
VERSION	0x00000100
PARAMETER	0x0000000F

31. MultiMedia Card Interface (MCI)

Rev. 3.0.3.6

31.1 Features

- **Compatible with Multimedia Card specification version 4.2**
- **Compatible with SD Memory Card specification version 2.0**
- **Compatible with SDIO specification version 1.1**
- **Compatible with CE-ATA specification 1.1**
- **Cards clock rate up to master clock divided by two**
- **High Speed mode support**
- **Embedded power management to slow down clock rate when not used**
- **Supports 2 Slots**
  - **Each slot for either a MultiMediaCard bus (up to 30 cards) or an SD Memory Card**
- **Support for stream, block and multi-block data read and write**
- **Supports connection to DMA Controller**
  - **Minimizes processor intervention for large buffer transfers**
- **Built in FIFO (from 16 to 256 bytes) with large memory aperture supporting incremental access**
- **Support for CE-ATA completion cignal disable command**
- **Protection against unexpected modification on-the-Fly of the configuration registers**

31.2 Overview

The Multimedia Card Interface (MCI) supports the MultiMedia Card (MMC) specification V4.2, the SD Memory Card specification V2.0, the SDIO V1.1 specificationand CE-ATA specification V1.1.

The MCI includes a Command Register (CMDR), Response Registers (RSPRn), data registers, time-out counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The MCI supports stream, block and multi block data read and write, and is compatible with the DMA Controller, minimizing processor intervention for large buffers transfers.

The MCI operates at a rate of up to CLK_MCI divided by 2 and supports the interfacing of 2 Slots. Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with a SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). The SDCard/SDIO Slot Selection field in the SDCard/SDIO Register (SDCR.SDCSEL) performs this selection.

The SD Memory Card communication is based on a nine-pin interface (clock, command, four data and three power lines) and the MultiMedia Card on a seven-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports MultiMedia Card operations. The main differences between SD and MultiMedia Cards are the initialization process and the bus topology.

MCI fully supports CE-ATA Revision 1.1, built on the MMC System specification V4.0. The module includes dedicated hardware to issue the command completion signal and capture the host command completion signal disable.

31.3 Block Diagram

Figure 31-1. MCI Block Diagram

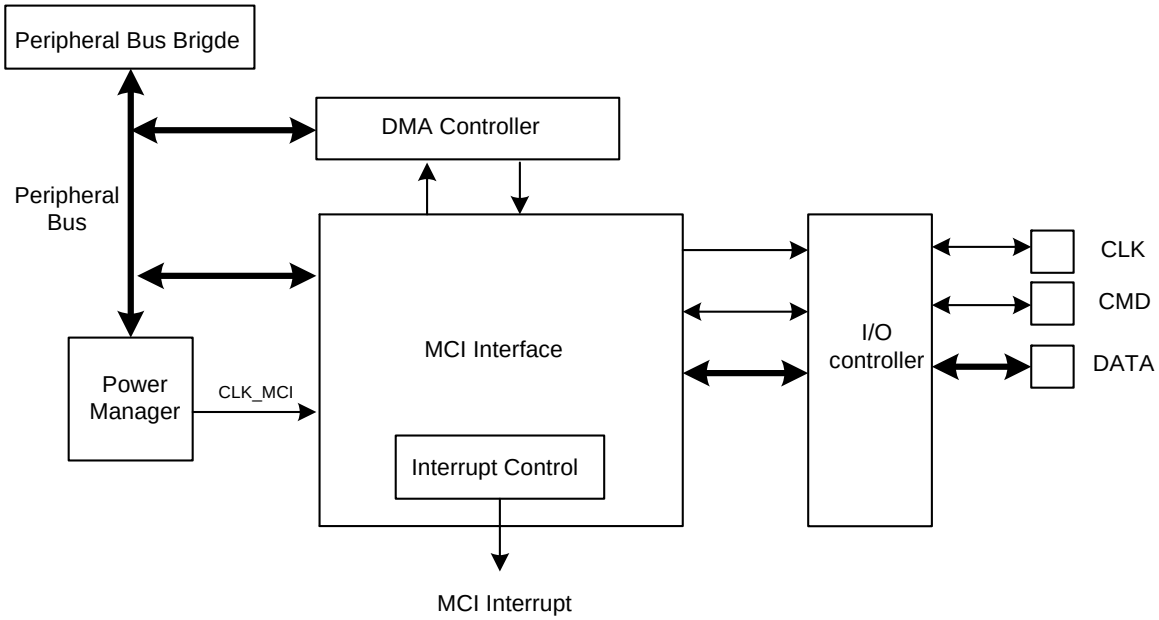
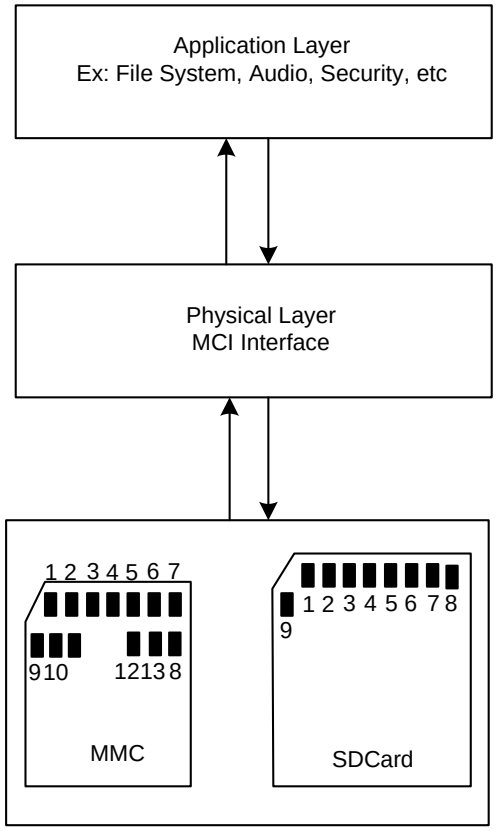


Figure 31-2. Application Block Diagram



31.4 I/O Lines Description

Table 31-1. I/O Lines Description

Pin Name	Pin Description	Type (1)	Comments
CMD[1:0]	Command/Response	Input/Output/ PP/OD	CMD of a MMC or SDCard/SDIO
CLK	Clock	Input/Output	CLK of a MMC or SD Card/SDIO
DATA[7:0]	Data 0..7 of Slot A	Input/Output/PP	DAT[0..7] of a MMC DAT[0..3] of a SD Card/SDIO
DATA[15:8]	Data 0..7 of Slot B	Input/Output/PP	DAT[0..7] of a MMC DAT[0..3] of a SD Card/SDIO

1. PP: Push/Pull, OD: Open Drain

31.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

31.5.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the MCI, the MCI will stop functioning and resume operation after the system wakes up from sleep mode.

31.5.2 I/O Lines

The pins used for interfacing the MultiMedia Cards or SD Cards may be multiplexed with GPIO lines. User must first program the I/O controller to assign the peripheral functions to MCI pins.

31.5.3 Clocks

The clock for the MCI bus interface (CLK_MCI) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the MCI before disabling the clock, to avoid freezing the MCI in an undefined state.

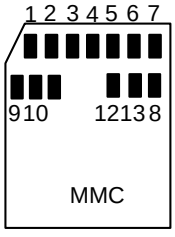
31.5.4 Interrupt

The MCI interrupt request line is connected to the interrupt controller. Using the MCI interrupt requires the interrupt controller to be programmed first.

31.6 Functional Description

31.6.1 Bus Topology

Figure 31-3. Multimedia Memory Card Bus Topology



The MultiMedia Card communication is based on a 13-pin serial bus interface. It has three communication lines and four supply lines.

Table 31-2. Bus Topology

Pin Number	Name	Type[1]	Description	MCI Pin Name[2] (Slot z)
1	DAT[3]	I/O/PP	Data	DATAz[3]
2	CMD	I/O/PP/OD	Command/response	CMDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	CLK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	DATAz[0]
8	DAT[1]	I/O/PP	Data 1	DATAz[1]
9	DAT[2]	I/O/PP	Data 2	DATAz[2]
10	DAT[4]	I/O/PP	Data 4	DATAz[4]
11	DAT[5]	I/O/PP	Data 5	DATAz[5]
12	DAT[6]	I/O/PP	Data 6	DATAz[6]
13	DAT[7]	I/O/PP	Data 7	DATAz[7]

Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

Figure 31-4. MMC Bus Connections (One Slot)

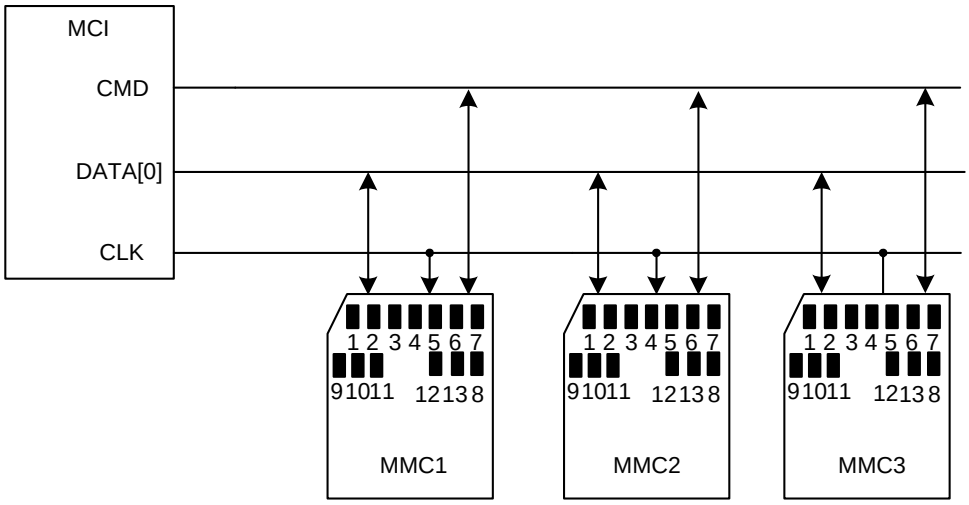
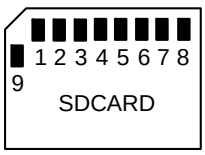


Figure 31-5. SD Memory Card Bus Topology



The SD Memory Card bus includes the signals listed in Table 31-3 on page 813.

Table 31-3. SD Memory Card Bus Signals

Pin Number	Name	Type[1]	Description	MCI Pin Name[2] (Slot z)
1	CD/DAT[3]	I/O/PP	Card detect/ Data line Bit 3	DATAz[3]
2	CMD	PP	Command/response	CMDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	CLK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data line Bit 0	DATAz[0]
8	DAT[1]	I/O/PP	Data line Bit 1 or Interrupt	DATAz[1]
9	DAT[2]	I/O/PP	Data line Bit 2	DATAz[2]

Notes: 1. I: input, O: output, PP: Push Pull, OD: Open Drain.

Figure 31-6. SD Card Bus Connections with One Slot

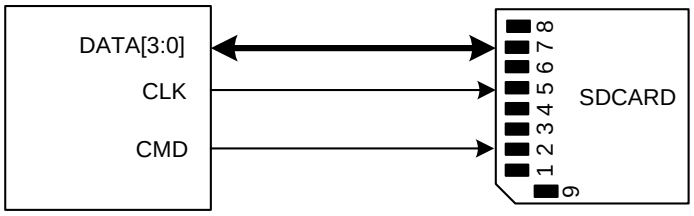


Figure 31-7. SD Card Bus Connections with Two Slots

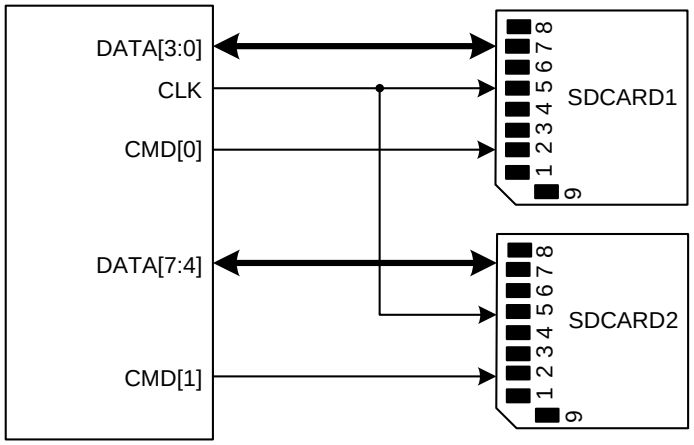
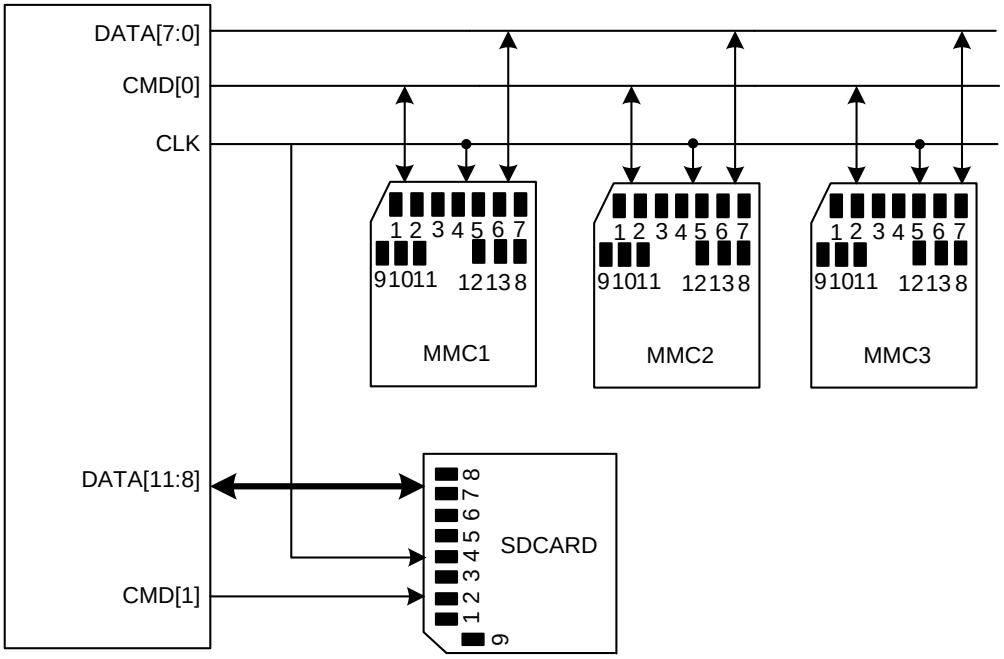


Figure 31-8. Mixing MultiMedia and SD Memory Cards with Two Slots



When the MCI is configured to operate with SD memory cards, the width of the data bus can be selected in the SDCard /SDIO Bus Width field in the SDCR register (SDCR.SDCBUS). See Section “31.7.4” on page 834. for details.

In the case of multimedia cards, only the data line 0 is used. The other data lines can be used as independent GPIOs.

31.6.2 MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- Command: a command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). a command is transferred serially on the CMD line.
- Response: a response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- Data: data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the MultiMedia-Card System Specification. Refer also to Table 31-4 on page 816.

MultiMediaCard bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the MCI clock (CLK).

Two types of data transfer commands are defined:

- Sequential commands: these commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: these commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read or when a multiple block transmission has a pre-defined block count (See Section “31.6.3” on page 817.).

The MCI provides a set of registers to perform the entire range of MultiMedia Card operations.

31.6.2.1 Command - Response Operation

After reset, the MCI is disabled and becomes valid after setting the Multi-Media Interface Enable bit in the Control Register (CR.MCIEN).

The Power Save Mode Enable bit in the CR register (CR.PWEN) saves power by dividing the MCI clock (CLK) by $2^{PWSDIV} + 1$ when the bus is inactive. The Power Saving Divider field locates in the Mode Register (MR.PWSDIV).

The two bits, Read Proof Enable and Write Proof Enable in the MR register (MR.RDPROOF and MR.WRPROOF) allow stopping the MCI Clock (CLK) during read or write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

All the timings for MultiMedia Card are defined in the MultiMediaCard System Specification.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the Command Register (CMDR). The CMDR register allows a command to be carried out.

For example, to perform an ALL_SEND_CID command:

	Host Command					**N_{ID} Cycles**					**CID**			
CMD	S	T	Content	CRC	E	Z	******	Z	S	T	Content	Z	Z	Z

The command ALL_SEND_CID and the fields and values for CMDR register are described in Table 31-4 on page 816 and Table 31-5 on page 816.

Table 31-4. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Resp	Abbreviation	Command Description
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: bcr means broadcast command with response.

Table 31-5. Fields and Values for the CMDR register

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The Argument Register (ARGR) contains the argument field of the command.

To send a command, the user must perform the following steps:

- Set the ARGR register with the command argument.
- Set the CMDR register (see Table 31-5 on page 816).

The command is sent immediately after writing the command register.

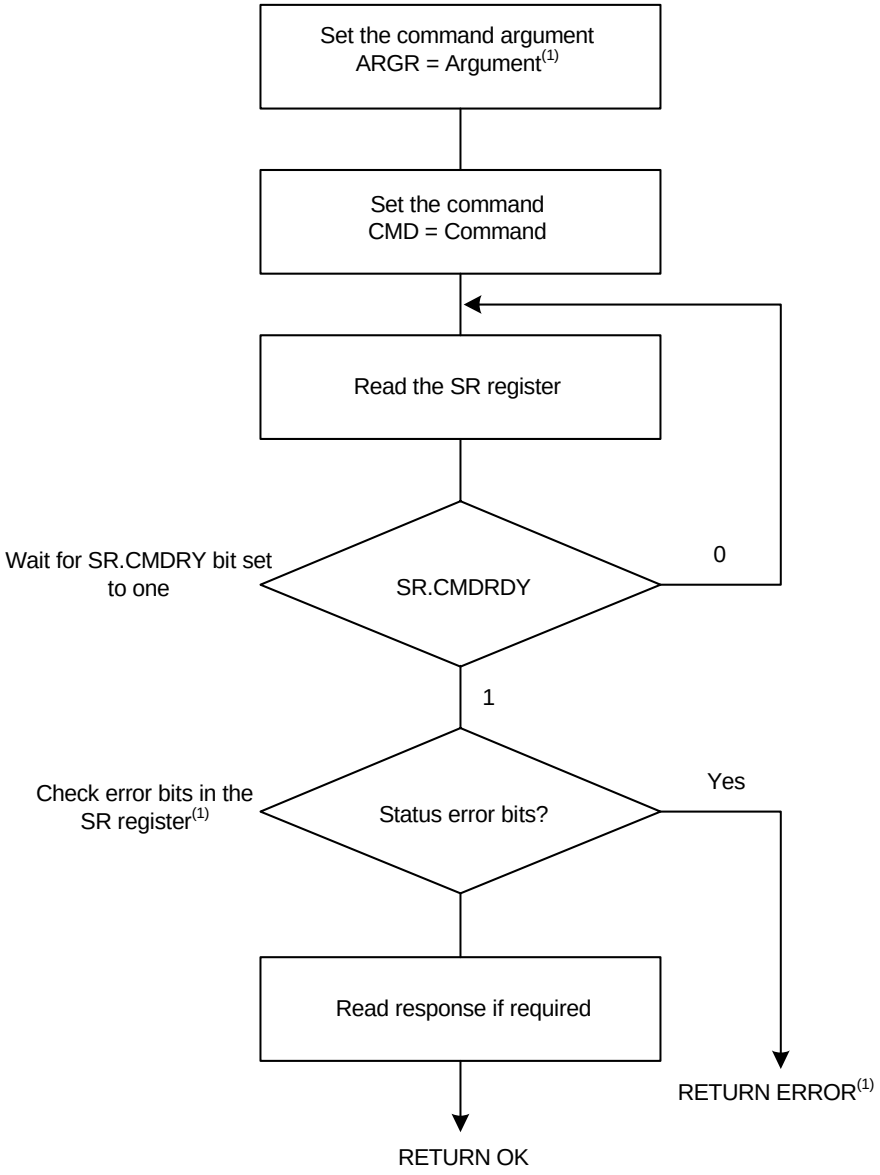
As soon as the command register is written, then the Command Ready bit in the Status Register (SR.CMDRDY) is cleared.

It is released and the end of the card response.

If the command requires a response, it can be read in the Response Registers (RSPRn). The response size can be from 48 bits up to 136 bits depending on the command. The MCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the Interrupt Enable Register (IER) allows using an interrupt method.

Figure 31-9. Command/Response Functional Flow Diagram



Note: 1. If the command is SEND_OP_COND, the CRC error bit is always present (refer to R3 response in the MultiMedia Card specification).

31.6.3 Data Transfer Operation

The MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kind of transfers can be selected setting the Transfer Type field in the CMDR register (CMDR.TRTYP).

These operations can be done using the features of the DMA Controller.

In all cases, the Data Block Length must be defined either in the Data Block Length field in the MR register (MR.BLKLEN)), or in the Block Register (BLKR). This field determines the size of the data block.

Consequent to MMC Specification 3.1, two types of multiple block read (or write) transactions are defined (the host can use either one at any time):

- Open-ended/Infinite Multiple block read (or write):

  The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.

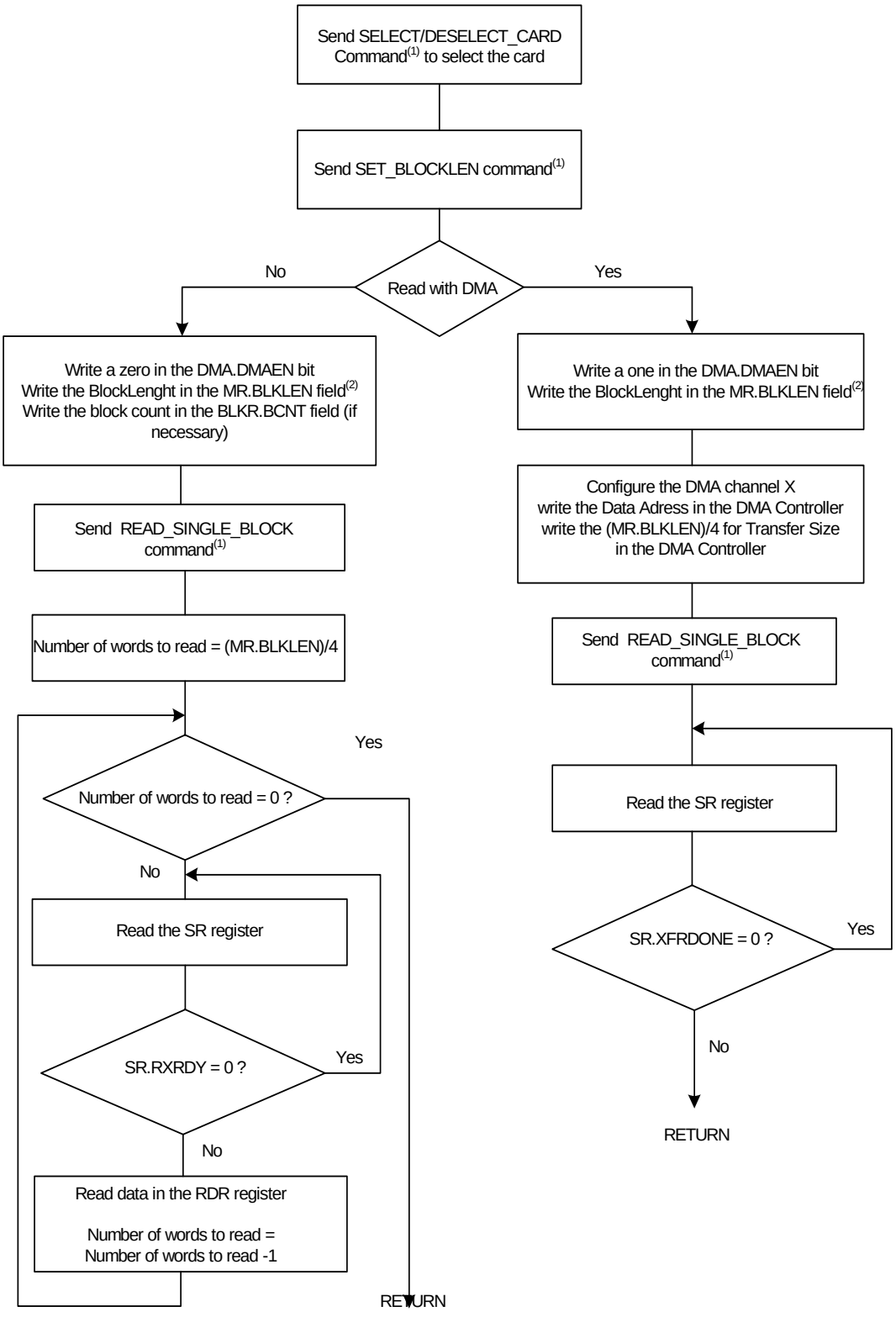
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

The card will transfer (or program) the requested number of data blocks and terminate the transaction. The stop command is not required at the end of this type of multiple block read (or write), unless terminated with an error. In order to start a multiple block read (or write) with pre-defined block count, the host must correctly set the BLKR register. Otherwise the card will start an open-ended multiple block read. The MMC/SDIO Block Count - SDIO Byte Count field in the BLKR register (BLKR.BCNT) defines the number of blocks to transfer (from 1 to 65535 blocks). Writing zero to this field corresponds to an infinite block transfer.

31.6.4 Read/Write Operation

The following flowchart shows how to read a single block with or without use of DMA Controller facilities. In this example (see Figure 31-10 on page 819), a polling method is used to wait for the end of read. Similarly, the user can configure the IER register to trigger an interrupt at the end of read.

Figure 31-10. Read Functional Flow Diagram



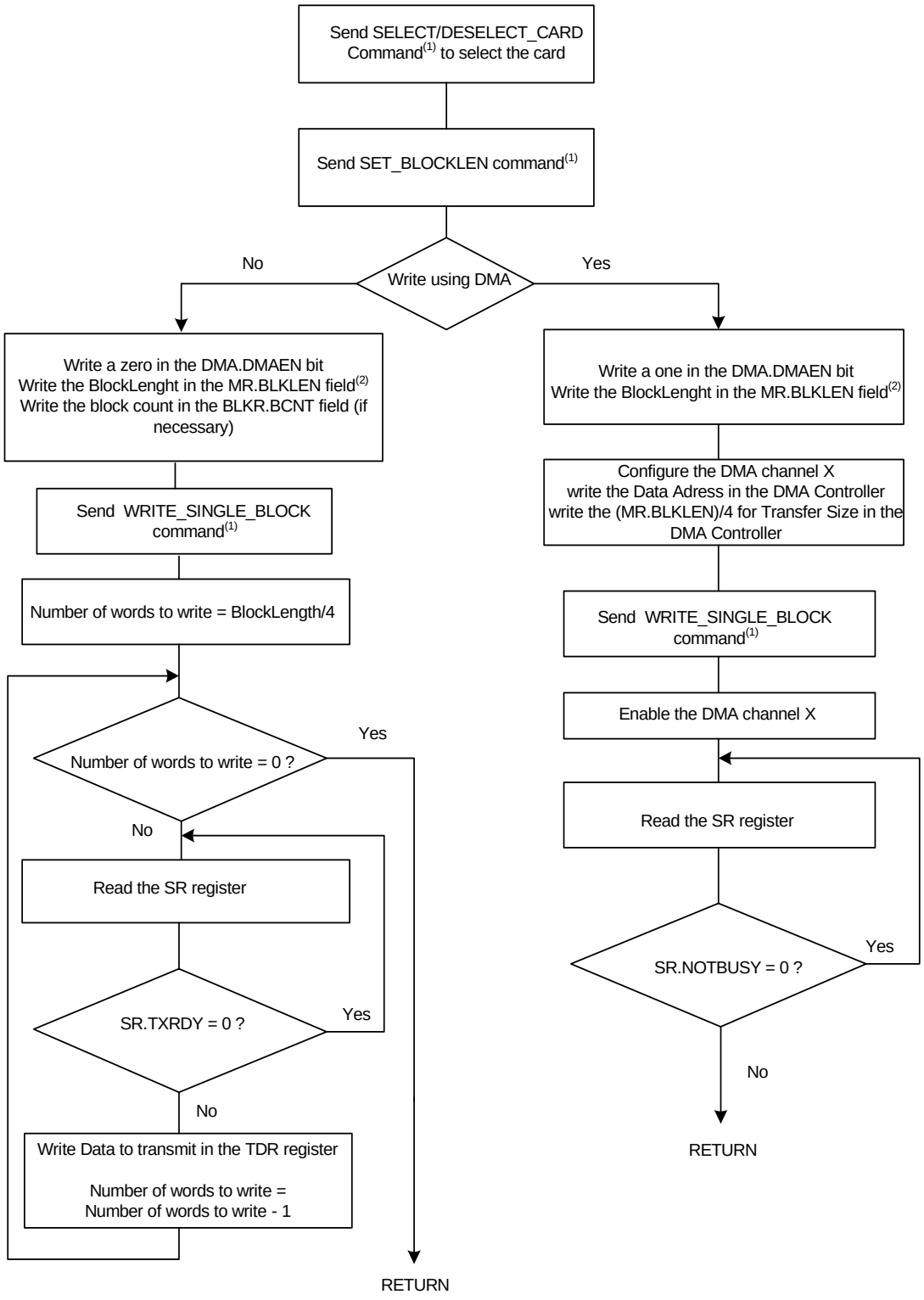
Note: 1. It is assumed that this command has been correctly sent (see Figure 31-9 on page 817).
2. This field is also accessible in the BLKR register.

In write operation, the Padding Value bit in the MR register (MR.PADV) is used to define the padding value when writing non-multiple block size. When the MR.PADV is zero, then 0x00 value is used when padding data, otherwise 0xFF is used.

Write a one in the DMA Hardware Handshaking Enable bit in the DMA Configuration Register (DMA.DMAEN) enables DMA transfer.

The following flowchart shows how to write a single block with or without use of DMA facilities (see Figure 31-11 on page 821). Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (IMR).

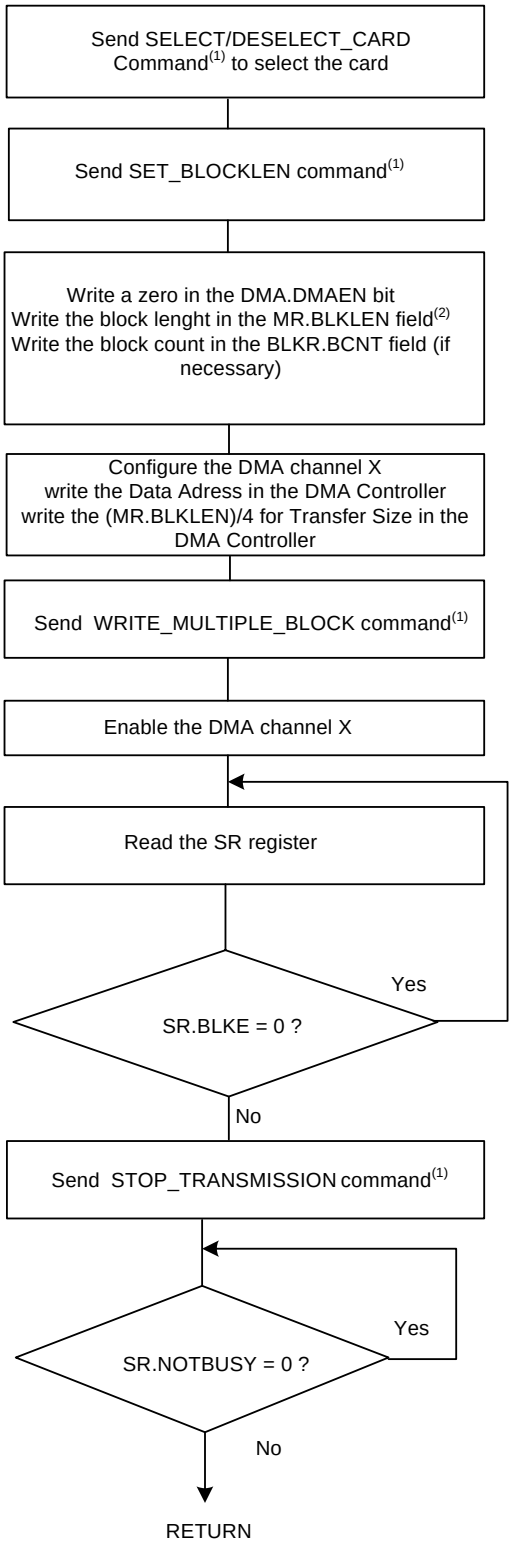
Figure 31-11. Write Functional Flow Diagram



Note: 1. It is assumed that this command has been correctly sent (see Figure 31-9 on page 817).
2. This field is also accessible in BLKR register.

The following flowchart shows how to manage a multiple write block transfer with the DMA Controller (see Figure 31-12 on page 823). Polling or interrupt method can be used to wait for the end of write according to the contents of the IMR register.

Figure 31-12. Multiple Write Functional Flow Diagram



Note: 1. It is assumed that this command has been correctly sent (see Figure 31-9 on page 817).
2. This field is also accessible in BLKR register.

31.6.4.1 WRITE_SINGLE_BLOCK operation using DMA Controller

1. Wait until the current command execution has successfully terminated.
   c. Check that the Transfer Done bit in the SR register (SR.XFRDONE) is set
2. Write the block length in the card. This value defines the value *block_lenght.*
3. Write the MR.BLKLEN with *block_lenght* value.
4. Configure the DMA Channel in the DMA Controller.
5. Write the DMA register with the following fields:
   – Write the *dma_offset* to the DMA Write Buffer Offset field (DMA.OFFSET).
   – Write the DMA Channel Read and Write Chunk Size field (DMA.CHKSIZE).
   – Write a one to he DMA.DMAEN bit to enable DMA hardware handshaking in the MCI.
6. Write a one to the DMA Transfer done bit in IER register (IER.DMADONE).
7. Issue a WRITE_SINGLE_BLOCK command.
8. Wait for DMA Transfer done bit in SR register (SR.DMADONE) is set.

31.6.4.2 READ_SINGLE_BLOCK operation using DMA Controller

1. Wait until the current command execution has successfully terminated.
   d. Check that the SR.XFRDONE bit is set.
2. Write the block length in the card. This value defines the value *block_lenght.*
3. Write the MR.BLKLEN with *block_lenght* value.
4. Configure the DMA Channel in the DMA Controller.
5. Write the DMA register with the following fields:
   – Write zero to the DMA.OFFSET field.
   – Write the DMA.CHKSIZE field.
   – Write to one the DMA.DMAEN bit to enable DMA hardware handshaking in the MCI.
6. Write a one to the IER.DMADONE bit.
7. Issue a READ_SINGLE_BLOCK command.
8. Wait for SR.DMADONE bit is set.

31.6.4.3 WRITE_MULTIPLE_BLOCK

1. Wait until the current command execution has successfully terminated.
   a. Check that the SR.XFRDONE bit is set.
2. Write the block length in the card. This value defines the value *block_lenght.*
3. Write the MR.BLKLEN with *block_lenght* value.
4. Program the DMA Controller to use a list of descriptors. Each descriptor transfers one block of data.
5. Program the DMA register with the following fields:
   – Write the *dma_offset* in the DMA.OFFSET field.
   – Write the DMA.CHKSIZE field.
   – Write a one to the DMA.DMAEN bit to enable DMA hardware handshaking in the MCI.
6. Write a one to the IER.DMADONE bit.
7. Issue a WRITE_MULTIPLE_BLOCK command.
8. Wait for DMA chained buffer transfer complete interrupt.

31.6.4.4 READ_MULTIPLE_BLOCK

1. Wait until the current command execution has successfully terminated.
   a. Check that the SR.CMDRDY and the SR.NOTBUSY are set.
2. Write the block length in the card. This value defines the value *block_lenght.*
3. Write the MR.BLKLEN with *block_lenght* value.
4. Program the DMA Controller to use a list of descriptors.
5. Write the DMA register with the following fields:
   – Write zero to the DMA.OFFSET.
   – Write the DMA.CHKSIZE.
   – Write a one to the DMA.DMAEN bit to enable DMA hardware handshaking in the MCI.
6. Write a one to the IER.DMADONE bit.
7. Issue a READ_MULTIPLE_BLOCK command.
8. Wait for DMA end of chained buffer transfer interrupt.

31.6.5 SD/SDIO Card Operation

The MCI allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the MultiMedia Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the MultiMedia Card with some additions. SD slots can actually be used for more than flash memory cards. Devices that support SDIO can use small devices designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth adapters, modems, barcode readers, IrDA adapters, FM radio tuners, RFID readers, digital cameras and more.

SD/SDIO is covered by numerous patents and trademarks, and licensing is only available through the Secure Digital Card Association.

The SD/SDIO Card communication is based on a nine-pin interface (Clock, Command, four Data and three Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD/SDIO Card and the MultiMedia Card is the initialization process.

The SD/SDIO Card Register (SDCR) allows selection of the Card Slot (SDCSEL) and the data bus width (SDCBUS).

The SD/SDIO Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD/SDIO Card uses only DAT[0] for data transfer. After initialization, the host can change the bus width (number of active data lines).

31.6.5.1 SDIO Data Transfer Type

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format (1 to 511 blocks), while the SD memory cards are fixed in the block transfer mode. The CMDR.TRTYP field allows to choose between SDIO Byte or SDIO Block transfer.

The number of bytes/blocks to transfer is set through the BCNT field in the BLKR register. In SDIO Block mode, the field BLKLEN must be set to the data block size while this field is not used in SDIO Byte mode.

An SDIO Card can have multiple I/O or combined I/O and memory (called Combo Card). Within a multi-function SDIO or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume (Refer to the SDIO Specification for more details). To send a suspend or a resume command, the host must set the SDIO Special Command field in CMDR register (CMDR.IOSPCMD).

31.6.5.2 SDIO Interrupts

Each function within an SDIO or Combo card may implement interrupts (Refer to the SDIO Specification for more details). In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the DAT[1] line to signal the card's interrupt to the host. An SDIO interrupt on each slot can be enabled in the IER register. The SDIO interrupt is sampled regardless of the currently selected slot.

31.6.6 CE-ATA Operation

CE-ATA maps the streamlined ATA command set onto the MMC interface. The ATA task file is mapped onto MMC register space.

CE-ATA utilizes five MMC commands:

- GO_IDLE_STATE (CMD0): used for hard reset.
- STOP_TRANSMISSION (CMD12): causes the ATA command currently executing to be aborted.
- FAST_IO (CMD39): Used for single register access to the ATA taskfile registers, eight bit access only.
- RW_MULTIPLE_REGISTERS (CMD60): used to issue an ATA command or to access the control/status registers.
- RW_MULTIPLE_BLOCK (CMD61): used to transfer data for an ATA command.

CE-ATA utilizes the same MMC command sequences for initialization as traditional MMC devices.

31.6.6.1 Executing an ATA Polling Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for eight kB of DATA.
2. Read the ATA status register until DRQ is set.
3. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
4. Read the ATA status register until DRQ && BSY are set to 0.

31.6.6.2 Executing an ATA Interrupt Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for eight kB of DATA with the IEN field written to zero to enable the command completion signal in the device.
2. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
3. Wait for Completion Signal Received Interrupt.

31.6.6.3 Aborting an ATA Command

If the host needs to abort an ATA command prior to the completion signal it must send a special command to avoid potential collision on the command line. The Special Command field of

CMDR register (CMDR.SPCMD) must be set to three to issue the CE-ATA completion Signal Disable Command.

31.6.6.4 CE-ATA Error Recovery

Several methods of ATA command failure may occur, including:

- No response to an MMC command, such as RW_MULTIPLE_REGISTER (CMD60).
- CRC is invalid for an MMC command or response.
- CRC16 is invalid for an MMC data packet.
- ATA Status register reflects an error by setting the ERR bit to one.
- The command completion signal does not arrive within a host specified time out period.

Error conditions are expected to happen infrequently. Thus, a robust error recovery mechanism may be used for each error event. The recommended error recovery procedure after a time-out is:

- Issue the command completion signal disable if IEN was cleared to zero and the RW_MULTIPLE_BLOCK (CMD61) response has been received.
- Issue STOP_TRANSMISSION (CMD12) and successfully receive the R1 response.
- Issue a software reset to the CE-ATA device using FAST_IO (CMD39).

If STOP_TRANMISSION (CMD12) is successful, then the device is again ready for ATA commands. However, if the error recovery procedure does not work as expected or there is another time-out, the next step is to issue GO_IDLE_STATE (CMD0) to the device. GO_IDLE_STATE (CMD0) is a hard reset to the device and completely resets all device states.

Note that after issuing GO_IDLE_STATE (CMD0), all device initialization needs to be completed again. If the CE-ATA device completes all MMC commands correctly but fails the ATA command with the ERR bit set in the ATA Status register, no error recovery action is required. The ATA command itself failed implying that the device could not complete the action requested, however, there was no communication or protocol failure. After the device signals an error by setting the ERR bit to one in the ATA Status register, the host may attempt to retry the command.

31.6.7 MCI Transfer Done Timings

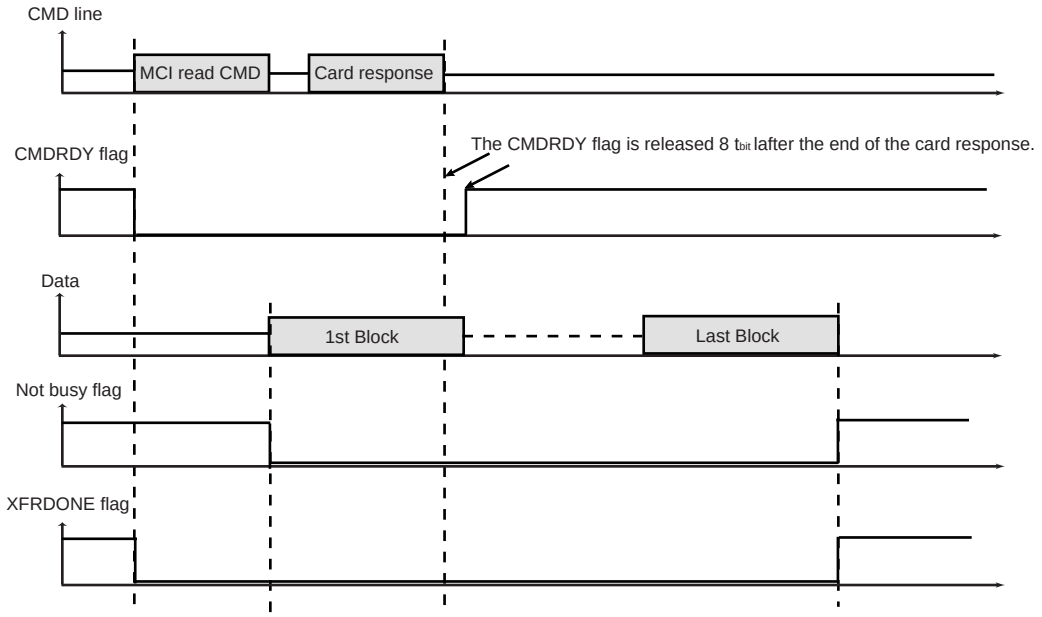
31.6.7.1 Definition

The SR.XFRDONE bit indicates exactly when the read or write sequence is finished.

31.6.7.2 Read Access

During a read access, the SR.XFRDONE bit behaves as shown in Figure 31-13 on page 828.

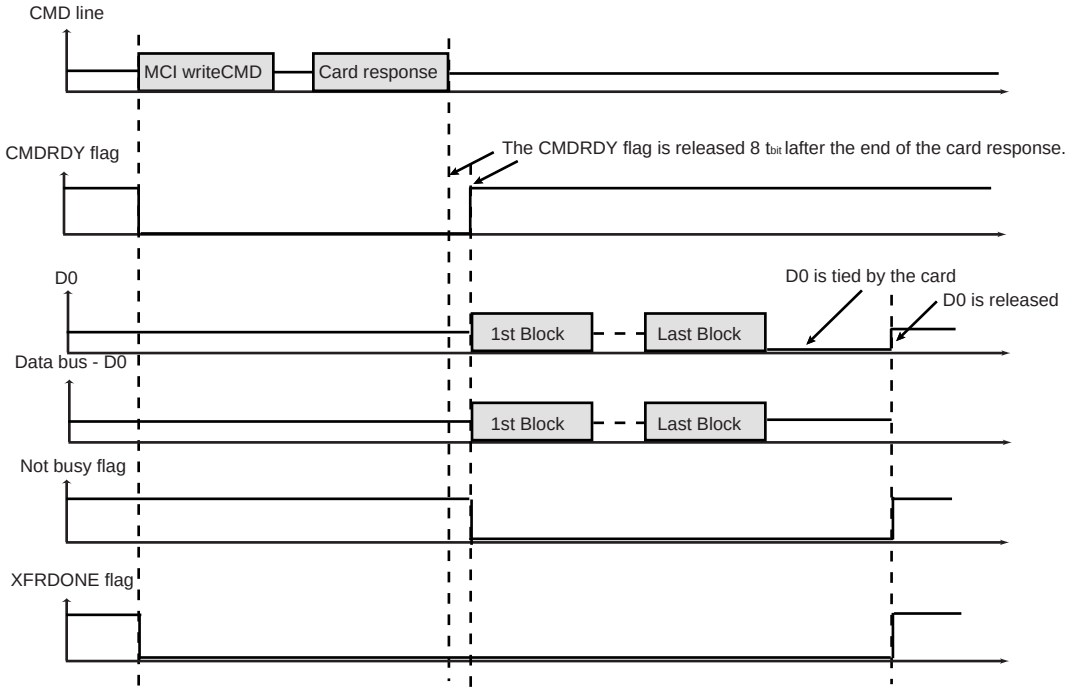
Figure 31-13. SR.XFRDONE During a Read Access



31.6.7.3 Write Access

During a write access, the SR.XFRDONE bit behaves as shown in Figure 31-14 on page 828.

Figure 31-14. SR.XFRDONE During a Write Access



31.7 User Interface

Table 31-6. MCI Register Memory Map

Offset	Register	Name	Access	Reset
0x000	Control Register	CR	Write-only	0x00000000
0x004	Mode Register	MR	Read-write	0x00000000
0x008	Data Time-out Register	DTOR	Read-write	0x00000000
0x00C	SD/SDIO Card Register	SDCR	Read-write	0x00000000
0x010	Argument Register	ARGR	Read-write	0x00000000
0x014	Command Register	CMDR	Write-only	0x00000000
0x018	Block Register	BLKR	Read-write	0x00000000
0x01C	Completion Signal Time-out Register	CSTOR	Read-write	0x00000000
0x020	Response Register	RSPR	Read-only	0x00000000
0x024	Response Register	RSPR1	Read-only	0x00000000
0x028	Response Register	RSPR2	Read-only	0x00000000
0x02C	Response Register	RSPR3	Read-only	0x00000000
0x030	Receive Data Register	RDR	Read-only	0x00000000
0x034	Transmit Data Register	TDR	Write-only	0x00000000
0x040	Status Register	SR	Read-only	0x0C000025
0x044	Interrupt Enable Register	IER	Write-only	0x00000000
0x048	Interrupt Disable Register	IDR	Write-only	0x00000000
0x04C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x050	DMA Configuration Register	DMA	Read-write	0x00000000
0x054	Configuration Register	CFG	Read-write	0x00000000
0x0E4	Write Protection Mode Register	WPMR	Read-write	0x00000000
0x0E8	Write Protection Status Register	WPSR	Read-only	0x00000000
0x0FC	Version Register	VERSION	Read-only	- (1)
0x200-0x3FFC	FIFO Memory Aperture	–	Read-write	0x00000000

1. The reset value are device specific. Please refer to the Module Configuration section at the end of this chapter.

31.7.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x000

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	-	IOWAITDIS	IOWAITEN	PWSDIS	PWSEN	MCIDIS	MCIEN

- **SWRST: Software Reset**
  Writing a one to this bit will reset the MCI interface.
  Writing a zero to this bit has no effect.
- **IOWAITDIS: SDIO Read Wait Disable**
  Writing a one to this bit will disable the SDIO Read Wait Operation.
  Writing a zero to this bit has no effect.
- **IOWAITEN: SDIO Read Wait Enable**
  Writing a one to this bit will enable the SDIO Read Wait Operation.
  Writing a zero to this bit has no effect.
- **PWSDIS: Power Save Mode Disable**
  Writing a one to this bit will disable the Power Saving Mode.
  Writing a zero to this bit has no effect.
- **PWSEN: Power Save Mode Enable**
  Writing a one to this bit and a zero to PWSDIS will enable the Power Saving Mode.
  Writing a one to this bit and a one to PWSDIS will disable the Power Saving Mode.
  Writing a zero to this bit has no effect.
  Warning: Before enabling this mode, the user must write a value different from 0 to the PWSDIV field.
- **MCIDIS: Multi-Media Interface Disable**
  Writing a one to this bit will disable the Multi-Media Interface.
  Writing a zero to this bit has no effect.
- **MCIEN: Multi-Media Interface Enable**
  Writing a one to this bit and a zero to MCIDIS will enable the Multi-Media Interface.
  Writing a one to this bit and a one to MCIDIS will disable the Multi-Media Interface.
  Writing a zero to this bit has no effect.

31.7.2 Mode Register

Name: MR

Access Type: Read-write

Offset: 0x004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
BLKLEN[15:8]							

23	22	21	20	19	18	17	16
BLKLEN[7:0]							

15	14	13	12	11	10	9	8
-	PADV	FBYTE	WRPROOF	RDPROOF	PWSDIV		

7	6	5	4	3	2	1	0
CLKDIV							

- **BLKLEN[15:0]: Data Block Length**
  This field determines the size of the data block.
  This field is also accessible in the BLKR register.
  If FBYTE bit is zero, the BLKEN[1:0] field must be written to 0b00

  Notes:
  1. In SDIO Byte mode, BLKLEN field is not used.
  2. BLKLEN should be written to one before sending the data transfer command. Otherwise, Overrun may occur even if RDPROOF bit is one.
- **PADV: Padding Value**
  0: 0x00 value is used when padding data in write transfer.
  1: 0xFF value is used when padding data in write transfer.
  PADV is used only in manual transfer.
- **FBYTE: Force Byte Transfer**
  Enabling Force Byte Transfer allows byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported.
  Warning: BLKLEN value depends on FBYTE.
  Writing a one to this bit will enable the Force Byte Transfer.
  Writing a zero to this bit will disable the Force Byte Transfer.
- **WRPROOF Write Proof Enable**
  Enabling Write Proof allows to stop the MCI Clock (CLK) during write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.
  Writing a one to this bit will enable the Write Proof mode.
  Writing a zero to this bit will disable the Write Proof mode.
- **RDPROOF Read Proof Enable**
  Enabling Read Proof allows to stop the MCI Clock (CLK) during read access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.
  Writing a one to this bit will enable the Read Proof mode.
  Writing a zero to this bit will disable the Read Proof mode.

- **PWSDIV: Power Saving Divider**
  Multimedia Card Interface clock is divided by $2^{(PWSDIV)}$ + 1 when entering Power Saving Mode.
  Warning: This value must be different from zero before enabling the Power Save Mode in the CR register (CR.PWSEN).
- **CLKDIV: Clock Divider**
  The Multimedia Card Interface Clock (CLK) is CLK_MCI divided by (2*(CLKDIV+1)).

31.7.3 Data Time-out Register

Name: DTOR

Access Type: Read/Write

Offset: 0x008

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
-	DTOMUL			DTOCYC			

These two fields determine the maximum number of CLK_MCI cycles that the MCI waits between two data block transfers. It is equal to (DTOCYC x Multiplier).

If the data time-out defined by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error bit in the SR register (SR.DTOE) is set.

- **DTOMUL: Data Time-out Multiplier**

  Multiplier is defined by DTOMUL as shown in the following table

DTOMUL	Multiplier
0	1
1	16
2	128
3	256
4	1024
5	4096
6	65536
7	1048576

- **DTOCYC: Data Time-out Cycle Number**

31.7.4 SDCard/SDIO Register

Name: SDCR

Access Type: Read/Write

Offset: 0x00C

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
SDCBUS		–	–	–	–	SDCSEL	

- **SDCBUS: SDCard/SDIO Bus Width**

SDCBUS	BUS WIDTH
0	1 bit
1	Reserved
2	4 bits
3	8 bits

- **SDCSEL: SDCard/SDIO Slot**

SDCSEL	SDCard/SDIO Slot
0	Slot A is selected.
1	Slot B is selected.
2	Reserved.
3	Reserved.

31.7.5 Argument Register

Name: ARGR

Access Type: Read/Write

Offset: 0x010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ARG[31:24]							

23	22	21	20	19	18	17	16
ARG[23:16]							

15	14	13	12	11	10	9	8
ARG[15:8]							

7	6	5	4	3	2	1	0
ARG[7:0]							

- **ARG[31:0]: Command Argument**
  this field contains the argument field of the command.

31.7.6 Command Register

Name: CMDR

Access Type: Write-only

Offset: 0x014

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	ATACS	IOSPCMD	

23	22	21	20	19	18	17	16
-	-	TRTYP			TRDIR	TRCMD	

15	14	13	12	11	10	9	8
-	-	-	MAXLAT	OPDCMD	SPCMD		

7	6	5	4	3	2	1	0
RSPTYP		CMDNB					

This register is write-protected while SR.CMDRDY is zero. If an interrupt command is sent, this register is only writable by an interrupt response (SPCMD field). This means that the current command execution cannot be interrupted or modified.

- **ATACS: ATA with Command Completion Signal**
  Writing a one to this bit will configure ATA completion signal within a programmed amount of time in Completion Signal Time-out Register (CSTOR).
  Writing a zero to this bit will configure no ATA completion signal.
- **IOSPCMD: SDIO Special Command**

IOSPCMD	SDIO Special Command Type
0	Not a SDIO Special Command
1	SDIO Suspend Command
2	SDIO Resume Command
3	Reserved

- **TRTYP: Transfer Type**

TRTYP	Transfer Type
0	MMC/SDCard Single Block
1	MMC/SDCard Multiple Block
2	MMC Stream
3	Reserved
4	SDIO Byte
5	SDIO Block
others	Reserved

- **TRDIR: Transfer Direction**
  Writing a zero to this bit will configure the transfer direction as write transfer.
  Writing a one to this bit will configure the transfer direction as read transfer.
- **TRCMD: Transfer Command**

TRCMD	Transfer Type
0	No data transfer
1	Start data transfer
2	Stop data transfer
3	Reserved

- **MAXLAT: Max Latency for Command to Response**
  Writing a zero to this bit will configure a 5-cycle max latency.
  Writing a one to this bit will configure a 64-cycle max latency.
- **OPDCMD: Open Drain Command**
  Writing a zero to this bit will configure the push-pull command.
  Writing a one to this bit will configure the open-drain command.
- **SPCMD: Special Command**

SPCMD	Command
0	Not a special CMD.
1	Initialization CMD: 74 clock cycles for initialization sequence.
2	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
3	CE-ATA Completion Signal disable Command. The host cancels the ability for the device to return a command completion signal on the command line.
4	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
5	Interrupt response: Corresponds to the Interrupt Mode (CMD40).
others	Reserved

- **RSPTYP: Response Type**

RSP	Response Type
0	No response.
1	48-bit response.
2	136-bit response.
3	R1b response type

- **CMDNB: Command Number**
  The Command Number to transmit.

31.7.7 Block Register

Name: BLKR

Access Type: Read/Write

Offset: 0x018

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
BLKLEN[15:8]							

23	22	21	20	19	18	17	16
BLKLEN[7:0]							

15	14	13	12	11	10	9	8
BCNT[15:8]							

7	6	5	4	3	2	1	0
BCNT[7:0]							

- **BLKLEN: Data Block Length**

  This field determines the size of the data block.

  This field is also accessible in the MR register.

  If MR.FBYTE bit is zero, the BLKEN[17:16] field must be written to 0b00

  Notes:
  1. In SDIO Byte mode, BLKLEN field is not used.
  2. BLKLEN should be specified before sending the data transfer command. Otherwise, Overrun may occur (even if MR.RDPROOF bit is set).

- **BCNT: MMC/SDIO Block Count - SDIO Byte Count**

  This field determines the number of data byte(s) or block(s) to transfer.

  The transfer data type and the authorized values for BCNT field are determined by CMDR.TRTYP field:

TRTYP	Type of Transfer	BCNT Authorized Values
0	MMC/SDCard Multiple Block	From 1 to 65535: Value 0 corresponds to an infinite block transfer.
2	SDIO Byte	From 1 to 512 bytes: Value 0 corresponds to a 512-byte transfer. Values from 0x200 to 0xFFFF are forbidden.
3	SDIO Block	From 1 to 511 blocks: Value 0 corresponds to an infinite block transfer. Values from 0x200 to 0xFFFF are forbidden.
Others	-	Reserved.

Warning: In SDIO Byte and Block modes, writing to the seven last bits of BCNT field is forbidden and may lead to unpredictable results.

31.7.8 Completion Signal Time-out Register

Name: CSTOR

Access Type: Read-write

Offset: 0x01C

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
-	CSTOMUL			CSTOCYC			

These two fields determines the maximum number of CLK_MCI cycles that the MCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

These two fields also determine the maximum number of CLK_MCI cycles that the MCI waits between the end of the data transfer and the assertion of the completion signal. The data transfer comprises data phase and the optional busy phase. If a non-DATA ATA command is issued, the MCI starts waiting immediately after the end of the response until the completion signal. If the data time-out defined by CSTOCYC and CSTOMUL has been exceeded, the Completion Signal Time-out Error bit in the SR register (SR.CSTOE) is set.

- **CSTOMUL: Completion Signal Time-out Multiplier**

  Multiplier is defined by CSTOMUL as shown in the following table:

CSTOMUL	Multiplier
0	1
1	16
2	128
3	256
4	1024
5	4096
6	65536
7	1048576

- **CSTOCYC: Completion Signal Time-out Cycle Number**

31.7.9 Response Register n

Name: RSPRn

Access Type: Read-only

Offset: 0x020 + 0*0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RSP[31:24]							

23	22	21	20	19	18	17	16
RSP[23:16]							

15	14	13	12	11	10	9	8
RSP[15:8]							

7	6	5	4	3	2	1	0
RSP[7:0]							

- **RSP[31:0]: Response**

  The response register can be read by N access(es) at the same RSPRn or at consecutive addresses (0x20 + n*0x04). N depends on the size of the response.

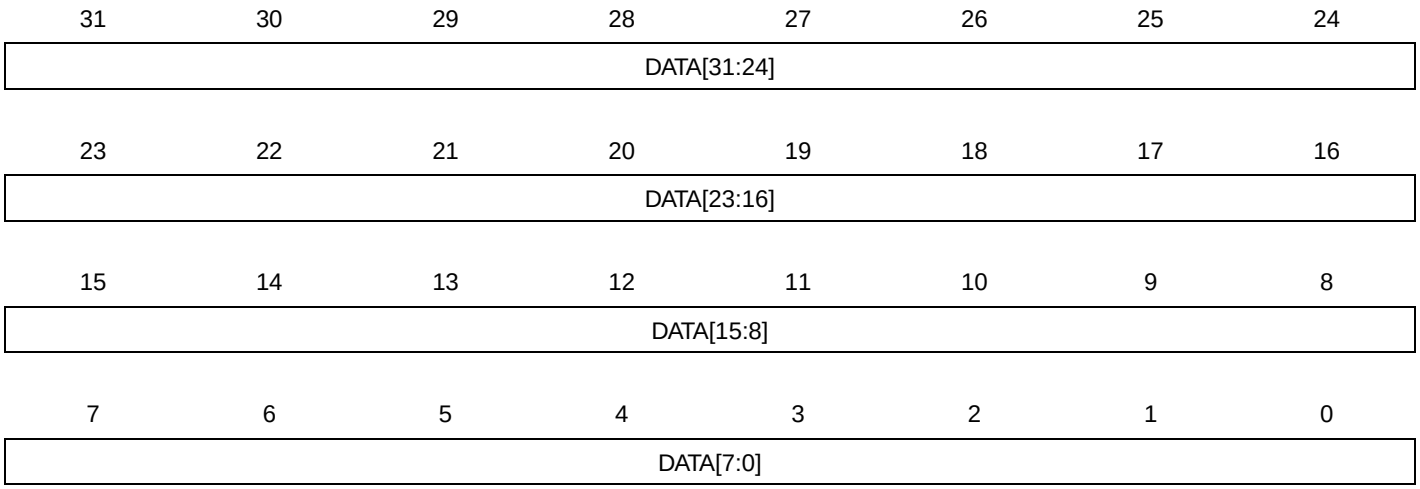
31.7.10 Receive Data Register

Name: RDR

Access Type: Read-only

Offset: 0x030

Reset Value: 0x00000000



- **DATA[31:0]: Data to Read**
  The last data received.

31.7.11 Transmit Data Register

Name: TDR

Access Type: Write-only

Offset: 0x034

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:8]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA[31:0]: Data to Write**
  The data to send.

31.7.12 Status Register

Name: SR

Access Type: Read-only

Offset: 0x040

Reset Value: 0x0C000025

31	30	29	28	27	26	25	24
UNRE	OVRE	-	-	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	CSRCV	SDIOWAIT	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **UNRE: Underrun Error**
  This bit is set when at least one eight-bit data has been sent without valid information (not written).
  This bit is cleared when sending a new data transfer command if the Flow Error bit reset control mode in Configuration Register (CFG.FERRCTRL) is zero or when reading the SR register if CFG.FERRCTRL is one.
- **OVRE: Overrun Error**
  This bit is set when at least one 8-bit received data has been lost (not read).
  This bit is cleared when sending a new data transfer command if CFG.FERRCTRL is zero, or when reading the SR register if CFG.FERRCTRL is one.
- **XFRDONE: Transfer Done**
  This bit is set when the CR register is ready to operate and the data bus is in the idle state.
  This bit is cleared when a transfer is in progress.
- **FIFOEMPTY: FIFO empty**
  This bit is set when the FIFO is empty.
  This bit is cleared when the FIFO contains at least one byte.
- **DMADONE: DMA Transfer done**
  This bit is set when the DMA buffer transfer is completed.
  This bit is cleared when reading the SR register.
- **BLKOVRE: DMA Block Overrun Error**
  This bit is set when a new block of data is received and the DMA controller has not started to move the current pending block.
  This bit is cleared when reading the SR register.
- **CSTOE: Completion Signal Time-out Error**
  This bit is set when the completion signal time-out defined by the CSTOR.CSTOCYC field and the CSTOR.CSTOMUL field is reached.
  This bit is cleared when reading the SR register.
- **DTOE: Data Time-out Error**
  This bit is set when the data time-out defined by the DTOR.DTOCYC field and the DTOR.DTOMUL field is reached.
  This bit is cleared when reading the SR register.

- **DCRCE: Data CRC Error**
  This bit is set when a CRC16 error is detected in the last data block.
  This bit is cleared when reading the SR register.
- **RTOE: Response Time-out Error**
  This bit is set when the response time-out defined by the CMDR.MAXLAT bit is reached.
  This bit is cleared when writing the CMDR register.
- **RENDE: Response End Bit Error**
  This bit is set when the end bit of the response is not detected.
  This bit is cleared when writing the CMDR register.
- **RCRCE: Response CRC Error**
  This bit is set when a CRC7 error is detected in the response.
  This bit is cleared when writing the CMDR register.
- **RDIRE: Response Direction Error**
  This bit is set when the direction bit from card to host in the response is not detected.
  This bit is cleared when writing the CMDR register.
- **RINDE: Response Index Error**
  This bit is set when a mismatch is detected between the command index sent and the response index received.
  This bit is cleared when writing the CMDR register.
- **TXBUFE: TX Buffer Empty Status**
  This bit is set when the DMA Tx Buffer is empty.
  This bit is cleared when the DMA Tx Buffer is not empty.
- **RXBUFF: RX BUffer Full Status**
  This bit is set when the DMA Rx Buffer is full.
  This bit is cleared when the DMA Rx Buffer is not full.
- **CSRCV: CE-ATA Completion Signal Received**
  This bit is set when the device issues a command completion signal on the command line.
  This bit is cleared when reading the SR register.
- **SDIOWAIT: SDIO Read Wait Operation Status**
  This bit is set when the data bus has entered IO wait state.
  This bit is cleared when normal bus operation.
- **SDIOIRQB: SDIO Interrupt for Slot B**
  This bit is cleared when reading the SR register.
  This bit is set when a SDIO interrupt on Slot B occurs.
- **SDIOIRQA: SDIO Interrupt for Slot A**
  This bit is set when a SDIO interrupt on Slot A occurs.
  This bit is cleared when reading the SR register.
- **ENDTX: End of RX Buffer**
  This bit is set when the DMA Controller transmission is finished.
  This bit is cleared when the DMA Controller transmission is not finished.
- **ENDRX: End of RX Buffer**
  This bit is set when the DMA Controller reception is finished.
  This bit is cleared when the DMA Controller reception is not finished.
- **NOTBUSY: MCI Not Busy**
  This bit must be used only for write operations.
  A block write operation uses a simple busy signalling of the write operation duration on the data (DAT[0]) line: during a data transfer block, if the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line (DAT[0]) to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free.
  The NOTBUSY bit allows to deal with these different states.
  1: MCI is ready for new data transfer.
  0: MCI is not ready for new data transfer.
  This bit is cleared at the end of the card response.

This bit is set when the busy state on the data line is ended. This corresponds to a free internal data receive buffer of the card.
Refer to the MMC or SD Specification for more details concerning the busy behavior.

- **DTIP: Data Transfer in Progress**
  This bit is set when the current data transfer is in progress.
  This bit is cleared at the end of the CRC16 calculation
  1: The current data transfer is still in progress.
  0: No data transfer in progress.
- **BLKE: Data Block Ended**
  This bit must be used only for Write Operations.
  This bit is set when a data block transfer has ended.
  This bit is cleared when reading SR.
  1: a data block transfer has ended, including the CRC16 Status transmission, the bit is set for each transmitted CRC Status.
  0: A data block transfer is not yet finished.
  Refer to the MMC or SD Specification for more details concerning the CRC Status.
- **TXRDY: Transmit Ready**
  This bit is set when the last data written in the TDR register has been transferred.
  This bit is cleared the last data written in the TDR register has not yet been transferred.
- **RXRDY: Receiver Ready**
  This bit is set when the data has been received since the last read of the RDR register.
  This bit is cleared when the data has not yet been received since the last read of the RDR register.
- **CMDRDY: Command Ready**
  This bit is set when the last command has been sent.
  This bit is cleared when writing the CMDR register

31.7.13 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x044

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
UNRE	OVRE	-	-	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFF	RXBUFF	CSRCV	SDIOWAIT	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

31.7.14 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x048

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
UNRE	OVRE	-	-	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFF	RXBUFF	CSRCV	SDIOWAIT	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

31.7.15 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x04C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
UNRE	OVRE	-	-	XFRDONE	FIFOEMPTY	DMADONE	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFF	RXBUFF	CSRCV	SDIOWAIT	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

31.7.16 DMA Configuration Register

Name: DMA

Access Type: Read/Write

Offset: 0x050

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
-	-	-	-	-	-	-	DAMEN

7	6	5	4	3	2	1	0
-	CHKSIZE			-	-	OFFSET	

- **DMAEN: DMA Hardware Handshaking Enable**
  1: DMA Interface is enabled.
  0: DMA interface is disabled.
  To avoid unpredictable behavior, DMA hardware handshaking must be disabled when CPU transfers are performed.
  To avoid data losses, the DMA register should be initialized before sending the data transfer command. This is also illustrated in Figure 31-10 on page 819 or Figure 31-11 on page 821
- **CHKSIZE: DMA Channel Read and Write Chunk Size**
  The CHKSIZE field indicates the number of data available when the DMA chunk transfer request is asserted.

CHKSIZE value	Number of data transferred	
0	1	Only available if FIFO_SIZE>= 16 bytes
1	4	Only available if FIFO_SIZE>= 32 bytes
2	8	Only available if FIFO_SIZE>= 64 bytes
3	16	Only available if FIFO_SIZE>= 128 bytes
4	32	Only available if FIFO_SIZE>= 256 bytes
others	-	Reserved

- **OFFSET: DMA Write Buffer Offset**
  This field indicates the number of discarded bytes when the DMA writes the first word of the transfer.

31.7.17 Configuration Register

Name: CFG

Access Type: Read/Write

Offset: 0x054

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
-	-	-	LSYNC	-	-	-	HSMODE

7	6	5	4	3	2	1	0
-	-	-	FERRCTRL	-	-	-	FIFOMODE

- **LSYNC: Synchronize on the last block**
  1: The pending command is sent at the end of the block transfer when the transfer length is not infinite. (block count shall be different from zero)
  0: The pending command is sent at the end of the current data block.
  This register needs to configured before sending the data transfer command.
- **HSMODE: High Speed Mode**
  1: The host controller outputs command line and data lines on the rising edge of the card clock. The Host driver shall check the high speed support in the card registers.
  0: Default bus timing mode.
- **FERRCTRL: Flow Error bit reset control mode**
  1: When an underflow/overflow condition bit is set, reading SR resets the bit.
  0: When an underflow/overflow condition bit is set, a new Write/Read command is needed to reset the bit.
- **FIFOMODE: MCI Internal FIFO control mode**
  1: A write transfer starts as soon as one data is written into the FIFO.
  0: A write transfer starts when a sufficient amount of data is written into the FIFO.
  When the block length is greater than or equal to 3/4 of the MCI internal FIFO size, then the write transfer starts as soon as half the FIFO is filled. When the block length is greater than or equal to half the internal FIFO size, then the write transfer starts as soon as one quarter of the FIFO is filled. In other cases, the transfer starts as soon as the total amount of data is written in the internal FIFO.

31.7.18 Write Protect Mode Register

Name: WPMR

Access Type: Read/Write

Offset: 0x0E4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
WPKEY[23:16]							

23	22	21	20	19	18	17	16
WPKEY[15:8]							

15	14	13	12	11	10	9	8
WPKEY[7:0]							

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	WPEN

- **WPKEY[23:0]: Write Protection Key password**
  This field should be written at value **0x4D4349** (ASCII code for “MCI”).
  Writing any other value in this field has no effect.
- **WPEN: Write Protection Enable**
  1: This bit enables the Write Protection if WPKEY corresponds.
  0: This bit disables the Write Protection if WPKEY corresponds.

31.7.19 Write Protect Status Register

Name: WPSR

Access Type: Read-only

Offset: 0x0E8

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
WPVSRC[15:8]							

15	14	13	12	11	10	9	8
WPVSRC[7:0]							

7	6	5	4	3	2	1	0
-	-	-	-	WPVS			

- **WPVSRC[15:0]: Write Protection Violation Source**
  This field contains address where the violation access occurs.
- **WPVS: Write Protection Violation Status**

WPVS	Definition
0	No Write Protection Violation occurred since the last read of this register (WPSR)
1	Write Protection detected unauthorized attempt to write a control register had occurred (since the last read.)
2	Software reset had been performed while Write Protection was enabled (since the last read).
3	Both Write Protection violation and software reset with Write Protection enabled had occurred since the last read.
others	Reserved

31.7.20 Version Register

Name: VERSION

Access: Read-only

Offset: 0x0FC

Reset Value: -

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

23	22	21	20	19	18	17	16
-	-	-	-	-	VARIANT		

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associate

31.7.21 FIFO Memory Aperture

Name: -

Access: Read/Write

Offset: 0x200 - 0x3FFC

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA[31:0]:Data to read or Data to write**

31.8 Module Configuration

The specific configuration for the MCI instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 31-7. Module Clock Name

Module name	Clock name
MCI	**CLK_MCI**

Table 31-8. Register Reset Values

Register	Reset Value
VERSION	0x00000300

32. Advanced Encryption Standard (AES)

Rev: 1.2.3.0

32.1 Features

- **Compliant with *FIPS Publication 197, Advanced Encryption Standard (AES)***
- **128-bit/192-bit/256-bit cryptographic key**
- **12/14/16 clock cycles encryption/decryption processing time with a 128-bit/192-bit/256-bit cryptographic key**
- **Support of the five standard modes of operation specified in the *NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation - Methods and Techniques:***
  - **Electronic Code Book (ECB)**
  - **Cipher Block Chaining (CBC)**
  - **Cipher Feedback (CFB)**
  - **Output Feedback (OFB)**
  - **Counter (CTR)**
- **8-, 16-, 32-, 64- and 128-bit data size possible in CFB mode**
- **Last output data mode allows optimized Message Authentication Code (MAC) generation**
- **Hardware counter measures against differential power analysis attacks**
- **Connection to DMA Controller capabilities optimizes data transfers for all operating modes**

32.2 Overview

The Advanced Encryption Standard (AES) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 197* specification.

The AES supports all five confidentiality modes of operation for symmetrical key block cipher algorithms (ECB, CBC, OFB, CFB and CTR), as specified in the *NIST Special Publication 800-38A Recommendation.* It is compatible with all these modes via DMA Controller, minimizing processor intervention for large buffer transfers.

The 128-bit/192-bit/256-bit key is stored in write-only four/six/eight 32-bit KEY Word Registers (KEYWnR) which are all write-only registers.

The 128-bit input data and initialization vector (for some modes) are each stored in 32-bit Input Data Registers (IDATAnR) and in Initialization Vector Registers (VnR) which are all write-only registers.

As soon as the initialization vector, the input data and the key are configured, the encryption/decryption process may be started. Then the encrypted/decrypted data is ready to be read out on the four 32-bit Output Data Registers (ODATAnR) or through the DMA Controller.

32.3 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

32.3.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the AES, the AES will stop functioning and resume operation after the system wakes up from sleep mode.

32.3.2 Clocks

The clock for the AES bus interface (CLK_AES) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the AES before disabling the clock, to avoid freezing the AES in an undefined state.

32.3.3 Interrupts

The AES interrupt request line is connected to the interrupt controller. Using the AES interrupt requires the interrupt controller to be programmed first.

32.4 Functional Description

The AES specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information.

Encryption converts data to an unintelligible form called ciphertext. Decrypting the ciphertext converts the data back into its original form, called plaintext. The Processing Mode bit in the Mode Register (MR.CIPHER) allows selection between the encryption and the decryption processes.

The AES is capable of using cryptographic keys of 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. This 128-bit/192-bit/256-bit key is defined in the KEYWnR Registers (KEYWnR).

The input to the encryption processes of the CBC, CFB, and OFB modes includes, in addition to the plaintext, a 128-bit data block called the initialization vector, which must be writing in the Initialization Vector Registers (IVnR). The initialization vector is used in an initial step in the encryption of a message and in the corresponding decryption of the message. The IVRnR registers are also used in the CTR mode to set the counter value.

32.4.1 Operation Modes

The AES supports the following modes of operation:

- ECB: Electronic Code Book
- CBC: Cipher Block Chaining
- OFB: Output Feedback
- CFB: Cipher Feedback
  - CFB8 (CFB where the length of the data segment is 8 bits)
  - CFB16 (CFB where the length of the data segment is 16 bits)
  - CFB32 (CFB where the length of the data segment is 32 bits)
  - CFB64 (CFB where the length of the data segment is 64 bits)
  - CFB128 (CFB where the length of the data segment is 128 bits)
- CTR: Counter

The data pre-processing, post-processing and chaining for the concerned modes are automatically performed. Refer to the *NIST Special Publication 800-38A Recommendation* for more complete information.

These modes are selected by writing the Operation Mode field in the Mode Register (MR.OPMOD).

In CFB mode, five data size are possible (8 bits, 16 bits, 32 bits, 64 bits or 128 bits).

These sizes are selected by writing the Cipher Feedback Data Size field in the MR register (MR.CFDS).

32.4.2 Start Modes

The Start Mode field in the MR register (MR.SMOD) allows selection of the encryption (or decryption) start mode.

32.4.2.1 Manual mode

The sequence is as follows:

- Write the 128-bit/192-bit/256-bit key in the KEYWnR registers.
- Write the initialization vector (or counter) in the IVnR registers.

Note: The Initialization Vector Registers concern all modes except ECB.

- Write the Data Ready bit in the Interrupt Enable Register (IER.DATRDY), depending on whether an interrupt is required or not at the end of processing.
- Write the data to be encrypted/decrypted in the authorized Input Data Registers (IDATAnR).

Table 32-1. Authorized Input Data Registers

Operation Mode	IDATAnR to Write
ECB	All
CBC	All
OFB	All
128-bit CFB	All
64-bit CFB	IDATA1R and IDATA2R
32-bit CFB	IDATA1R
16-bit CFB	IDATA1R
8-bit CFB	IDATA1R
CTR	All

Note: In 64-bit CFB mode, writing to IDATA3R and IDATA4R registers is not allowed and may lead to errors in processing.

Note: In 32-bit, 16-bit and 8-bit CFB modes, writing to IDATA2R, IDATA3R and IDATA4R registers is not allowed and may lead to errors in processing.

- Write the START bit in the Control Register (CR.START) to begin the encryption or the decryption process.
- When the processing completes, the DATRDY bit in the Interrupt Status Register (ISR.DATRDY) is set.
- If an interrupt has been enabled by writing the IER.DATRDY bit, the interrupt line of the AES is activated.
- When the software reads one of the Output Data Registers (ODATAxR), the ISR.DATRDY bit is cleared.

32.4.2.2 Automatic mode

The automatic mode is similar to the manual one, except that in this mode, as soon as the correct number of IDATAnR Registers is written, processing is automatically started without any action in the CR register.

32.4.2.3 DMA mode

The DMA Controller can be used in association with the AES to perform an encryption/decryption of a buffer without any action by the software during processing.

In this starting mode, the type of the data transfer (byte, halfword or word) depends on the operation mode.

Table 32-2. Data Transfer Type for the Different Operation Modes

Operation Mode	Data Transfer Type (DMA)
ECB	word
CBC	word
OFB	word
CFB 128-bit	word
CFB 64-bit	word
CFB 32-bit	word
CFB 16-bit	halfword
CFB 8-bit	byte
CTR	word

The sequence is as follows:

- Write the 128-bit/192-bit/256-bit key in the KEYWnR registers.
- Write the initialization vector (or counter) in the IVnR registers.

Note: The Initialization Vector Registers concern all modes except ECB.

- Initialize the DMA Controller with address where the data buffer to encrypt/decrypt is stored and with the address where encrypted/decrypted data must be stored.

Note: Transmit and receive buffers can be identical.

- Enable the DMA Controller in transmission and reception to start the processing.
- The processing completion should be monitored with the DMA Controller.

32.4.3 Last Output Data Mode

This mode is used to generate cryptographic checksums on data (MAC) by means of cipher block chaining encryption algorithm (CBC-MAC algorithm for example).

After each end of encryption/decryption, the output data is available either on the ODATAnR registers for manual and automatic mode or at the address specified in the receive buffer pointer for DMA mode.

The Last Output Data bit in the Mode Register (MR.LOD) allows retrieval of only the last data of several encryption/decryption processes.

Therefore, there is no need to define a read buffer in DMA mode.

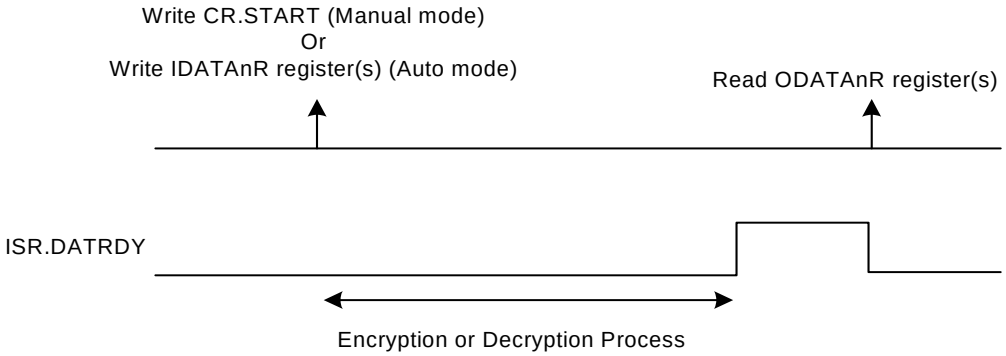
This data is only available on the Output Data Registers (ODATAnR).

32.4.3.1 Manual and automatic modes

- When MR.LOD is zero

The ISR.DATRDY bit is cleared when at least one of the ODATAnR registers is read.

Figure 32-1. Manual and Automatic Modes when MR.LOD is zero

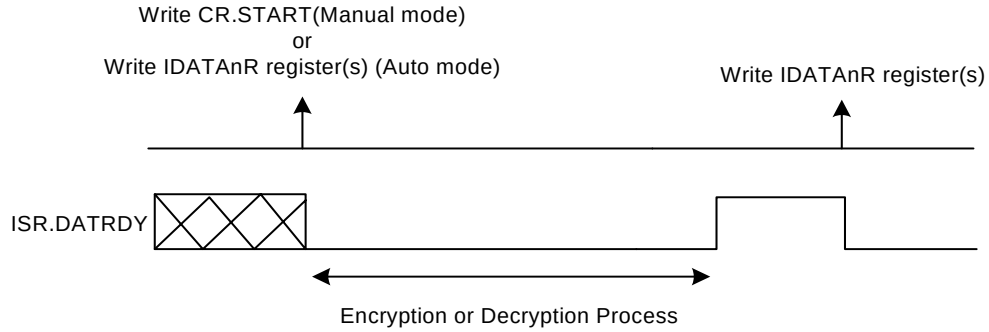


If the user does not want to read the output data registers between each encryption/decryption, the ISR.DATRDY bit will not be cleared. If the ISR.DATRDY bit is not cleared, the user cannot know the end of the following encryptions/decryptions.

- When MR.LOD is one

The ISR.DATRDY bit is cleared when at least one IDATAnR register is written, so before the start of a new transfer. No more ODATAnR register reads are necessary between consecutive encryptions/decryptions.

Figure 32-2. Manual and Automatic Modes when MR.LOD is one

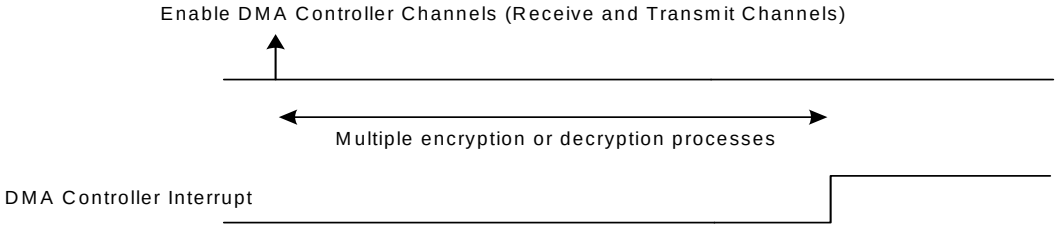


32.4.3.2 DMA mode

- when MR.LOD is zero

The end of the encryption/decryption should be monitored with the DMA Controller.

Figure 32-3. DMA Mode when MR.LOD is zero



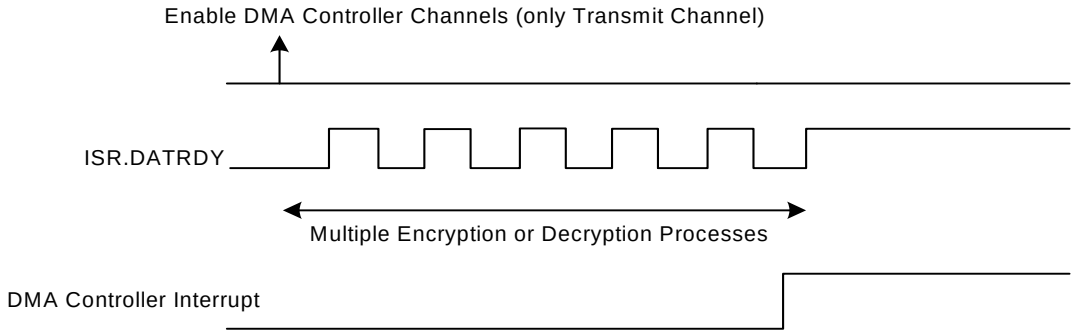
- when MR.LOD is one

The user must first wait for the DMA Controller Interrupt, then for ISR.DATRDY to ensure that the encryption/decryption is completed.

In this case, no receive buffers are required.

The output data is only available on the ODATAnR registers.

Figure 32-4. DMA Mode when MR.LOD is one



Following table summarizes the different cases.

Table 32-3. Last Output Mode Behavior versus Start Modes

	Manual and Automatic Modes		DMA Mode	
	MR.LOD = 0	**MR.LOD = 1**	**MR.LOD = 0**	**MR.LOD = 1**
ISR.DATRDY bit Clearing Condition(1)	At least one ODATAnR register must be read	At least one IDATAnR register must be written	Not used	Managed by the DMA Controller
Encrypted/Decrypted Data Result Location	In ODATAnR registers	In ODATAnR registers	At the address specified in the configuration of DMA Controller	In ODATAnR registers
End of Encryption/Decryption	ISR.DATRDY	ISR.DATRDY	DMA Controller Interrupt	DMA Controller Interrupt then DATRDY.ISR

Note: 1. Depending on the mode, there are other ways of clearing the DATRDY.ISR bit. See the Interrupt Status Register (ISR) definition.

Warning: In DMA mode, reading to the ODATAnR registers before the last data transfer may lead to unpredictable results.

32.4.4 Security Features

32.4.4.1 Countermeasures

The AES also features hardware countermeasures that can be useful to protect data against Differential Power Analysis (DPA) attacks.

These countermeasures can be enabled through the Countermeasure Type field in the MR register (MR.CTYPE). This field is write-only, and all changes to it are taken into account if, at the same time, the Countermeasure Key field in the Mode Register (MR.CKEY) is correctly written (see the Mode Register (MR) description in Section 32.5.2).

Note: Enabling countermeasures has an impact on the AES encryption/decryption throughput.

By default, all the countermeasures are enabled.

The best throughput is achieved with all the countermeasures disabled. On the other hand, the best protection is achieved with all of them enabled.

The Random Number Generator Seed Loading bit in the CR register (CR.LOADSEED) allows a new seed to be loaded in the embedded random number generator used for the different countermeasures.

32.4.4.2 *Unspecified register access detection*

When an unspecified register access occurs, the Unspecified Register Detection Status bit in the ISR register (ISR.URAD) is set to one. Its source is then reported in the Unspecified Register Access Type field in the ISR register (ISR.URAT). Only the last unspecified register access is available through the ISR.URAT field.

Several kinds of unspecified register accesses can occur when:

- Writing the IDATAnR registers during the data processing in DMA mode
- Reading the ODATAnR registers during data processing
- Writing the MR register during data processing
- Reading the ODATAnR registers during sub-keys generation
- Writing the MR register during sub-keys generation
- Reading an write-only register

The ISR.URAD bit and the ISR.URAT field can only be reset by the Software Reset bit in the CR register (CR.SWRST).

32.5 User Interface

Table 32-4. AES Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Mode Register	MR	Read/Write	0x00000000
0x10	Interrupt Enable Register	IER	Write-only	0x00000000
0x14	Interrupt Disable Register	IDR	Write-only	0x00000000
0x18	Interrupt Mask Register	IMR	Read-only	0x00000000
0x1C	Interrupt Status Register	ISR	Read-only	0x0000001E
0x20	Key Word 1 Register	KEYW1R	Write-only	0x00000000
0x24	Key Word 2 Register	KEYW2R	Write-only	0x00000000
0x28	Key Word 3 Register	KEYW3R	Write-only	0x00000000
0x2C	Key Word 4 Register	KEYW4R	Write-only	0x00000000
0x30	Key Word 5 Register	KEYW5R	Write-only	0x00000000
0x34	Key Word 6 Register	KEYW6R	Write-only	0x00000000
0x38	Key Word 7 Register	KEYW7R	Write-only	0x00000000
0x3C	Key Word 8 Register	KEYW8R	Write-only	0x00000000
0x40	Input Data 1 Register	IDATA1R	Write-only	0x00000000
0x44	Input Data 2 Register	IDATA2R	Write-only	0x00000000
0x48	Input Data 3 Register	IDATA3R	Write-only	0x00000000
0x4C	Input Data 4 Register	IDATA4R	Write-only	0x00000000
0x50	Output Data 1 Register	ODATA1R	Read-only	0x00000000
0x54	Output Data 2 Register	ODATA2R	Read-only	0xC01F0000
0x58	Output Data 3 Register	ODATA3R	Read-only	0x00000000
0x5C	Output Data 4 Register	ODATA4R	Read-only	0x00000000
0x60	Initialization Vector 1 Register	IV1R	Write-only	0x00000000
0x64	Initialization Vector 2 Register	IV2R	Write-only	0x00000000
0x68	Initialization Vector 3 Register	IV3R	Write-only	0x00000000
0x6C	Initialization Vector 4 Register	IV4R	Write-only	0x00000000
0xFC	Version Register	VR	Read-only	-(1)

Note: 1. The reset value are device specific. Please refer to the Module Configuration section at the end of this chapter.

32.5.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	LOADSEED

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	SWRST

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	START

- **LOADSEED: Random Number Generator Seed Loading**
  Writing a one to this bit will load a new seed in the embedded random number generator used for the different countermeasures.
  writing a zero to this bit has no effect.
- **SWRST: Software Reset**
  Writing a one to this bit will reset the AES.
  writing a zero to this bit has no effect.
- **START: Start Processing**
  Writing a one to this bit will start manual encryption/decryption process.
  writing a zero to this bit has no effect.

32.5.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	CTYPE				

23	22	21	20	19	18	17	16
CKEY				-	CFBS		

15	14	13	12	11	10	9	8
LOD	OPMOD			KEYSIZE		SMOD	

7	6	5	4	3	2	1	0
PROCDLY				-	-	-	CIPHER

- **CTYPE: Countermeasure Type**

CTYPE					Description
X	X	X	X	0	Countermeasure type 1 is disabled
X	X	X	X	1	Add random spurious power consumption during some configuration settings
X	X	X	0	X	Countermeasure type 2 is disabled
X	X	X	1	X	Add randomly 1 cycle to processing.
X	X	0	X	X	Countermeasure type 3 is disabled
X	X	1	X	X	Add randomly 1 cycle to processing (other version)
X	0	X	X	X	Countermeasure type 4 is disabled
X	1	X	X	X	Add randomly up to /13/15 cycles (for /192/256-bit key) to processing
0	X	X	X	X	Countermeasure type 5 is disabled
1	X	X	X	X	Add random spurious power consumption during processing (recommended with DMA access)

All the countermeasures are enabled by default.
CTYPE field is write-only and can only be modified if CKEY is correctly set.

- **CKEY: Countermeasure Key**

  Writing the value 0xE to this field allows the CTYPE field to be modified.

  Writing another value to this field has no effect.

  This bit always reads as zero.

- **CFBS: Cipher Feedback Data Size**

CFBS	Description
0	128-bit
1	64-bit
2	32-bit
3	16-bit
4	8-bit
Others	Reserved

- **LOD: Last Output Data Mode**

  Writing a one to this bit will enabled the LOD mode.

  Writing a zero to this bit will disabled the LOD mode.

  These mode is described in the Table 32-3 on page 862.

- **OPMOD: Operation Mode**

OPMOD	Description
0	ECB: Electronic Code Book mode
1	CBC: Cipher Block Chaining mode
2	OFB: Output Feedback mode
3	CFB: Cipher Feedback mode
4	CTR: Counter mode
Others	Reserved

- **KEYSIZE: Key Size**

KEYSIZE	Description
0	AES Key Size is 128 bits
1	AES Key Size is 192 bits
Others	AES Key Size is 256 bits

- **SMOD: Start Mode**

SMOD	Description
0	Manual mode
1	Automatic mode
2	DMA mode • LOD = 0: The encrypted/decrypted data are available at the address specified in the configuration of DMA Controller. • LOD = 1: The encrypted/decrypted data are available in the ODATAnR registers.
3	Reserved

- **PROCDLY: Processing Delay**

  The processing time represents the number of clock cycles that the AES needs in order to perform one encryption/decryption with no countermeasures activated:

$$\text{Processing Time} = 12 \times (\text{PROCDLY} + 1)$$

  The best performance is achieved with PROCDLY equal to 0.
  Writing a value to this field will update the processing time.
  Reading this field will give the current processing delay.

- **CIPHER: Processing Mode**

  0: Decrypts data is enabled.
  1: Encrypts data is enabled.

32.5.3 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

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
-	-	-	-	-	-	-	URAD

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DATRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

32.5.4 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

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
-	-	-	-	-	-	-	URAD

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DATRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

32.5.5 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

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
-	-	-	-	-	-	-	URAD

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DATRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

32.5.6 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x0000001E

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

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

15	14	13	12	11	10	9	8
URAT				-	-	-	URAD

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DATRDY

- **URAT: Unspecified Register Access Type:**

URAT	Description
0	The IDATAnR register during the data processing in DMA mode.
1	The ODATAnR register read during the data processing.
2	The MR register written during the data processing.
3	The ODATAnR register read during the sub-keys generation.
4	The MR register written during the sub-keys generation.
5	Write-only register read access.
Others	Reserved

Only the last Unspecified Register Access Type is available through the URAT field.

This field is reset to 0 when SWRST bit in the Control Register is written to one.

- **URAD: Unspecified Register Access Detection Status**

  This bit is set when at least one unspecified register access has been detected since the last software reset.

  This bit is cleared when SWRST bit in the Control Register is set to one.

- **DATRDY: Data Ready**

  This bit is set/clear as described in the Table 32-3 on page 862.

  This bit is also cleared when SWRST bit in the Control Register is set to one.

32.5.7 Key Word n Register

Name: KEYWnR

Access Type: Write-only

Offset: 0x20 +(n-1)*0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEYWn[31:24]							

23	22	21	20	19	18	17	16
KEYWn[23:16]							

15	14	13	12	11	10	9	8
KEYWn[15:8]							

7	6	5	4	3	2	1	0
KEYWn[7:0]							

- **KEYWn[31:0]: Key Word n**

  Writing the 128-bit/192-bit/256-bit cryptographic key used for encryption/decryption in the four/six/eight 32-bit Key Word registers.

  KEYW1 corresponds to the first word of the key and respectively KEYW4/KEYW6/KEYW8 to the last one.

  This field always read as zero to prevent the key from being read by another application.

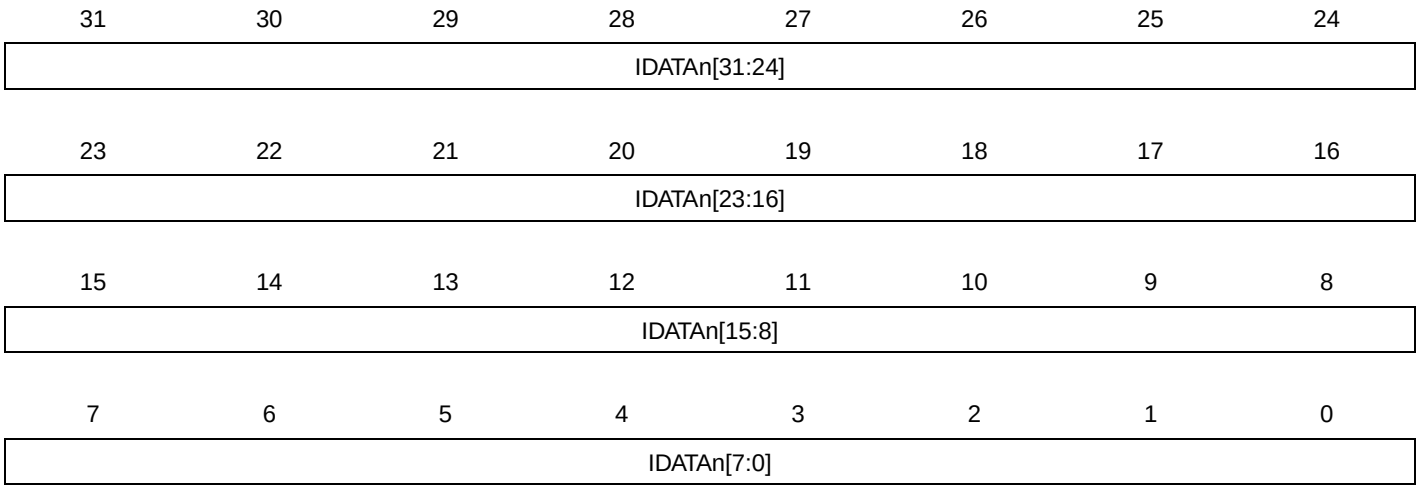
32.5.8 Input Data n Register

Name: IDATAnR

Access Type: Write-only

Offset: 0x40 + (n-1)*0x04

Reset Value: 0x00000000



- **IDATAn[31:0]: Input Data Word n**

  Writing the 128-bit data block used for encryption/decryption in the four 32-bit Input Data registers.
  IDATA1 corresponds to the first word of the data to be encrypted/decrypted, and IDATA4 to the last one.
  This field always read as zero to prevent the input data from being read by another application.

32.5.9 Output Data n Register

Name: ODATAnR

Access Type: Read-only

Offset: 0x50 + (n-1)*0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ODATAn[31:24]							

23	22	21	20	19	18	17	16
ODATAn[23:16]							

15	14	13	12	11	10	9	8
ODATAn[15:8]							

7	6	5	4	3	2	1	0
ODATAn[7:0]							

- **ODATAn[31:0]: Output Data n**
  Reading the four 32-bit ODATAnR give the 128-bit data block that has been encrypted/decrypted. ODATA1 corresponds to the first word, ODATA4 to the last one.

32.5.10 Initialization Vector n Register

Name: IVnR

Access Type: Write-only

Offset: 0x60 + (n-1)*0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
IVn[31:24]							

23	22	21	20	19	18	17	16
IVn[23:16]							

15	14	13	12	11	10	9	8
IVn[15:8]							

7	6	5	4	3	2	1	0
IVn[7:0]							

- **IVn[31:0]: Initialization Vector n**

  The four 32-bit Initialization Vector registers set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input:

MODE(OPMODE.	Description
CBC,OFB, CFB	initialization vector
CTR	counter value
ECB	not used, must not be written

  IV1 corresponds to the first word of the Initialization Vector, IV4 to the last one.

  This field is always read as zero to prevent the Initialization Vector from being read by another application.

32.5.11 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

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

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION[11:0]**
  Version number of the module. No functionality associated.

32.6 Module Configuration

The specific configuration for each AES instance is listed in the following tables.The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 32-5. Module clock name

Module name	Clock name
AES	**CLK_AES**

Table 32-6. Register Reset Values

Register	Reset Value
VERSION	0x00000123

33. Audio Bitstream DAC (ABDAC)

Rev: 1.0.1.1

33.1 Features

- **Digital Stereo DAC**
- **Oversampled D/A conversion architecture**
  - **Oversampling ratio fixed 128x**
  - **FIR equalization filter**
  - **Digital interpolation filter: Comb4**
  - **3rd Order Sigma-Delta D/A converters**
- **Digital bitstream outputs**
- **Parallel interface**
- **Connected to DMA Controller for background transfer without CPU intervention**

33.2 Overview

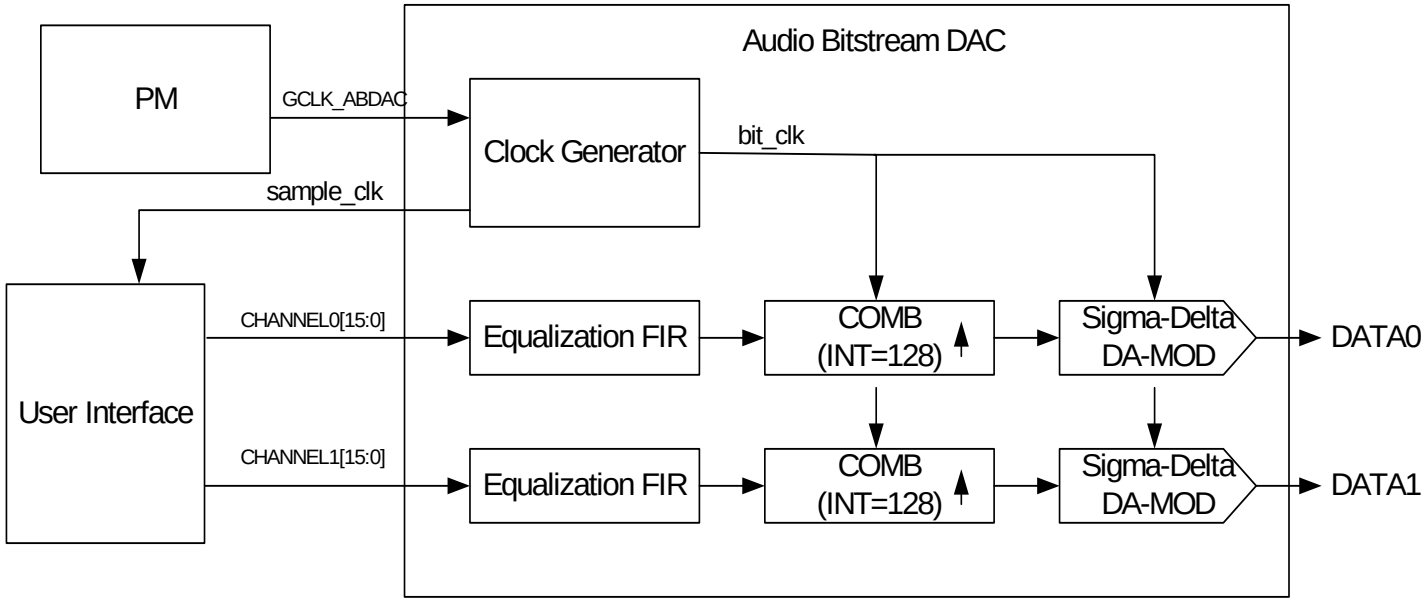
The Audio Bitstream DAC converts a 16-bit sample value to a digital bitstream with an average value proportional to the sample value. Two channels are supported, making the Audio Bitstream DAC particularly suitable for stereo audio. Each channel has a pair of complementary digital outputs, DATAn and DATANn, which can be connected to an external high input impedance amplifier.

The Audio Bitstream DAC is compromised of two 3rd order Sigma-Delta D/A converter with an oversampling ratio of 128. The samples are upsampled with a 4th order Sinc interpolation filter (Comb4) before being input to the Sigma-Delta Modulator. In order to compensate for the pass band frequency response of the interpolation filter and flatten the overall frequency response, the input to the interpolation filter is first filtered with a simple 3-tap FIR filter.The total frequency response of the Equalization FIR filter and the interpolation filter is given in Figure 33-2 on page 883. The digital output bitstreams from the Sigma-Delta Modulators should be low-pass filtered to remove high frequency noise inserted by the Modulation process.

The output DATAn and DATANn should be as ideal as possible before filtering, to achieve the best SNR quality. The output can be connected to a class D amplifier output stage, or it can be low pass filtered and connected to a high input impedance amplifier. A simple 1st order or higher low pass filter that filters all the frequencies above 50kHz should be adequate.

33.3 Block Diagram

Figure 33-1. ABDAC Block Diagram



33.4 I/O Lines Description

Table 33-1. I/O Lines Description

Pin Name	Pin Description	Type
DATA0	Output from Audio Bitstream DAC Channel 0	Output
DATA1	Output from Audio Bitstream DAC Channel 1	Output
DATAN0	Inverted output from Audio Bitstream DAC Channel 0	Output
DATAN1	Inverted output from Audio Bitstream DAC Channel 1	Output

33.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

33.5.1 I/O Lines

The output pins used for the output bitstream from the Audio Bitstream DAC may be multiplexed with IO lines.

Before using the Audio Bitstream DAC, the I/O Controller must be configured in order for the Audio Bitstream DAC I/O lines to be in Audio Bitstream DAC peripheral mode.

33.5.2 Clocks

The CLK_ABDAC to the Audio Bitstream DAC is generated by the Power Manager (PM). Before using the Audio Bitstream DAC, the user must ensure that the Audio Bitstream DAC clock is enabled in the Power Manager.

The ABDAC needs a separate clock for the D/A conversion operation. This clock, GCLK_ABDAC should be set up in the Generic Clock register in the Power Manager and its frequency must as follow:

$$f_{GCLK} = 256 \times f_S$$

For f_s= 48kHz this means that the GCLK_ABDAC clock must have a frequency of 12.288MHz.

33.5.3 Interrupts

The ABDAC interrupt request line is connected to the interrupt controller. Using the ABDAC interrupt requires the interrupt controller to be programmed first.

33.6 Functional Description

33.6.1 How to Initialize the Module

In order to use the Audio Bitstream DAC the product dependencies given in Section 33.5 on page 881 must be resolved. Particular attention should be given to the configuration of clocks and I/O lines in order to ensure correct operation of the Audio Bitstream DAC.

The Audio Bitstream DAC is enabled by writing a one to the enable bit in the Audio Bitstream DAC Control Register (CR.EN). The two 16-bit sample values for channel 0 and 1 can then be written to the least and most significant halfword of the Sample Data Register (SDR), respectively. The Transmit Ready Interrupt Status bit in the Interrupt Status Register (ISR.TXREADY) will be set whenever the ABDAC is ready to receive a new sample. A new sample value should be written to SDR before 256 ABDAC clock cycles, or an underrun will occur, as indicated by the Underrun Interrupt Status bit in ISR (ISR.UNDERRUN). ISR is cleared when read, or when writing one to the corresponding bits in the Interrupt Clear Register (ICR).

The Audio Bitstream DAC can also be configured for Peripheral DMA access, in which case only the CR.EN bit needs to be set in the Audio Bitstream DAC module.

33.6.2 Equalization Filter

The equalization filter is a simple 3-tap FIR filter. The purpose of this filter is to compensate for the pass band frequency response of the sinc interpolation filter. The equalization filter makes the pass band response more flat and moves the -3dB corner a little higher.

33.6.3 Interpolation Filter

The interpolation filter interpolates from f_s to 128f_s. This filter is a 4th order Cascaded Integrator-Comb filter, and the basic building blocks of this filter is a comb part and an integrator part.

33.6.4 Sigma-Delta Modulator

This part is a 3rd order Sigma-Delta Modulator consisting of three differentiators (delta blocks), three integrators (sigma blocks) and a one bit quantizer. The purpose of the integrators is to shape the noise, so that the noise is reduced in the band of interest and increased at the higher frequencies, where it can be filtered.

33.6.5 Data Format

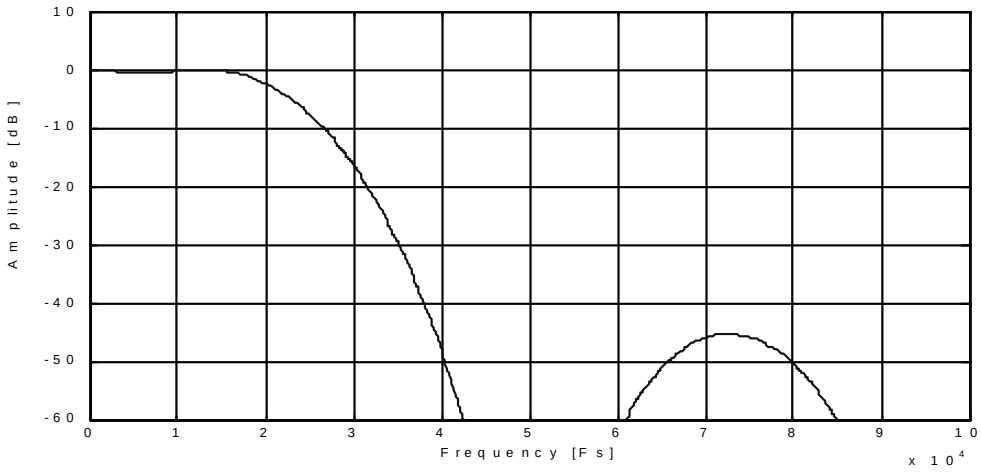
Input data is on two's complement format.

33.6.6 Data Swapping

When the SWAP bit in the ABDAC Control Register (CR.SWAP) is written to one, writing to the Sample Data Register (SDR) will cause the values written to the CHANNEL0 and CHANNEL1 fields to be swapped.

33.6.7 Frequency Response

Figure 33-2. Frequency Response, EQ-FIR+COMB[4]



33.6.8 Peripheral DMA Controller

The Audio Bitstream DAC is connected to the Peripheral DMA Controller. The Peripheral DMA Controller can be programmed to automatically transfer samples to the Audio Bitstream DAC Sample Data Register (SDR) when the Audio Bitstream DAC is ready for new samples. This enables the Audio Bitstream DAC to operate without any CPU intervention such as polling the Interrupt Status Register (ISR) or using interrupts. See the Peripheral DMA Controller documentation for details on how to setup Peripheral DMA transfers.

33.7 User Interface

Table 33-2. ABDAC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Sample Data Register	SDR	Read/Write	0x00000000
0x08	Control Register	CR	Read/Write	0x00000000
0x0C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x10	Interrupt Enable Register	IER	Write-only	0x00000000
0x14	Interrupt Disable Register	IDR	Write-only	0x00000000
0x18	Interrupt Clear Register	ICR	Write-only	0x00000000
0x1C	Interrupt Status Register	ISR	Read-only	0x00000000

33.7.1 Sample Data Register

Name: SDR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHANNEL1[15:8]							
23	22	21	20	19	18	17	16
CHANNEL1[7:0]							
15	14	13	12	11	10	9	8
CHANNEL0[15:8]							
7	6	5	4	3	2	1	0
CHANNEL0[7:0]							

- **CHANNEL1: Sample Data for Channel 1**
  signed 16-bit Sample Data for channel 1.
- **CHANNEL0: Signed 16-bit Sample Data for Channel 0**
  signed 16-bit Sample Data for channel 0.

33.7.2 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

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

- **EN: Enable Audio Bitstream DAC**
  1: The module is enabled.
  0: The module is disabled.
- **SWAP: Swap Channels**
  1: The swap of CHANNEL0 and CHANNEL1 samples is enabled.
  0: The swap of CHANNEL0 and CHANNEL1 samples is disabled.

33.7.3 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0C

Reset Value: 0x00000000

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

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in IDR is written to one.

A bit in this register is set when the corresponding bit in IER is written to one.

33.7.4 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

33.7.5 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

33.7.6 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in ISR and the corresponding interrupt request.

33.7.7 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

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

- **TXREADY: TX Ready Interrupt Status**

  This bit is cleared when the Audio Bitstream DAC is not ready to receive a new data in SDR.

  This bit is set when the Audio Bitstream DAC is ready to receive a new data in SDR.

- **UNDERRUN: Underrun Interrupt Status**

  This bit is cleared when no Audio Bitstream DAC Underrun has occurred since the last time this bit has been cleared (by reset or by writing in ICR).

  This bit is set when at least one Audio Bitstream DAC Underrun has occurred since the last time this bit has been cleared (by reset or by writing in ICR).

34. Programming and Debugging

34.1 Overview

General description of programming and debug features, block diagram and introduction of main concepts.

34.2 Service Access Bus

The AVR32 architecture offers a common interface for access to On-Chip Debug, programming, and test functions. These are mapped on a common bus called the Service Access Bus (SAB), which is linked to the JTAG port through a bus master module, which also handles synchronization between the debugger and SAB clocks.

When accessing the SAB through the debugger there are no limitations on debugger frequency compared to chip frequency, although there must be an active system clock in order for the SAB accesses to complete. If the system clock is switched off in sleep mode, activity on the debugger will restart the system clock automatically, without waking the device from sleep. Debuggers may optimize the transfer rate by adjusting the frequency in relation to the system clock. This ratio can be measured with debug protocol specific instructions.

The Service Access Bus uses 36 address bits to address memory or registers in any of the slaves on the bus. The bus supports sized accesses of bytes (8 bits), halfwords (16 bits), or words (32 bits). All accesses must be aligned to the size of the access, i.e. halfword accesses must have the lowest address bit cleared, and word accesses must have the two lowest address bits cleared.

34.2.1 SAB address map

The Service Access Bus (SAB) gives the user access to the internal address space and other features through a 36 bits address space. The 4 MSBs identify the slave number, while the 32 LSBs are decoded within the slave's address space. The SAB slaves are shown in Table 34-1 on page 892.

Table 34-1. SAB Slaves, addresses and descriptions.

Slave	Address [35:32]	Description
Unallocated	0x0	Intentionally unallocated
OCD	0x1	OCD registers
HSB	0x4	HSB memory space, as seen by the CPU
HSB	0x5	Alternative mapping for HSB space, for compatibility with other AVR32 devices.
Memory Service Unit	0x6	Memory Service Unit registers
Reserved	Other	Unused

34.2.2 SAB security restrictions

The Service Access bus can be restricted by internal security measures. A short description of the security measures are found in the table below.

34.2.2.1 Security measure and control location

A security measure is a mechanism to either block or allow SAB access to a certain address or address range. A security measure is enabled or disabled by one or several control signals. This is called the control location for the security measure.

These security measures can be used to prevent an end user from reading out the code programmed in the flash, for instance.

Table 34-2. SAB Security measures.

Security measure	Control Location	Description
Security bit	FLASHC security bit set	Programming and debugging not possible, very restricted access.
User code programming	FLASHC UPROT + security bit set	Restricts all access except parts of the flash and the flash controller for programming user code. Debugging is not possible unless an OS running from the secure part of the flash supports it.

Below follows a more in depth description of what locations are accessible when the security measures are active.

Table 34-3. Security bit SAB restrictions

Name	Address start	Address end	Access
OCD DCCPU, OCD DCEMU, OCD DCSR	0x100000110	0x100000118	Read/Write
User page	0x580800000	0x581000000	Read
Other accesses	-	-	Blocked

Table 34-4. User code programming SAB restrictions

Name	Address start	Address end	Access
OCD DCCPU, OCD DCEMU, OCD DCSR	0x100000110	0x100000118	Read/Write
User page	0x580800000	0x581000000	Read
FLASHC PB interface	0x5FFFE0000	0x5FFFE0400	Read/Write
FLASH pages outside BOOTPROT	0x580000000 + BOOTPROT size	0x580000000 + Flash size	Read/Write
Other accesses	-	-	Blocked

34.3 On-Chip Debug (OCD)

Rev: 1.4.2.1

34.3.1 Features

- **Debug interface in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 2+**
- **JTAG access to all on-chip debug functions**
- **Advanced program, data, ownership, and watchpoint trace supported**
- **NanoTrace JTAG-based trace access**
- **Auxiliary port for high-speed trace information**
- **Hardware support for 6 program and 2 data breakpoints**
- **Unlimited number of software breakpoints supported**
- **Automatic CRC check of memory regions**

34.3.2 Overview

Debugging on the AT32UC3A3 is facilitated by a powerful On-Chip Debug (OCD) system. The user accesses this through an external debug tool which connects to the JTAG port and the Auxiliary (AUX) port. The AUX port is primarily used for trace functions, and a JTAG-based debugger is sufficient for basic debugging.

The debug system is based on the Nexus 2.0 standard, class 2+, which includes:

- Basic run-time control
- Program breakpoints
- Data breakpoints
- Program trace
- Ownership trace
- Data trace

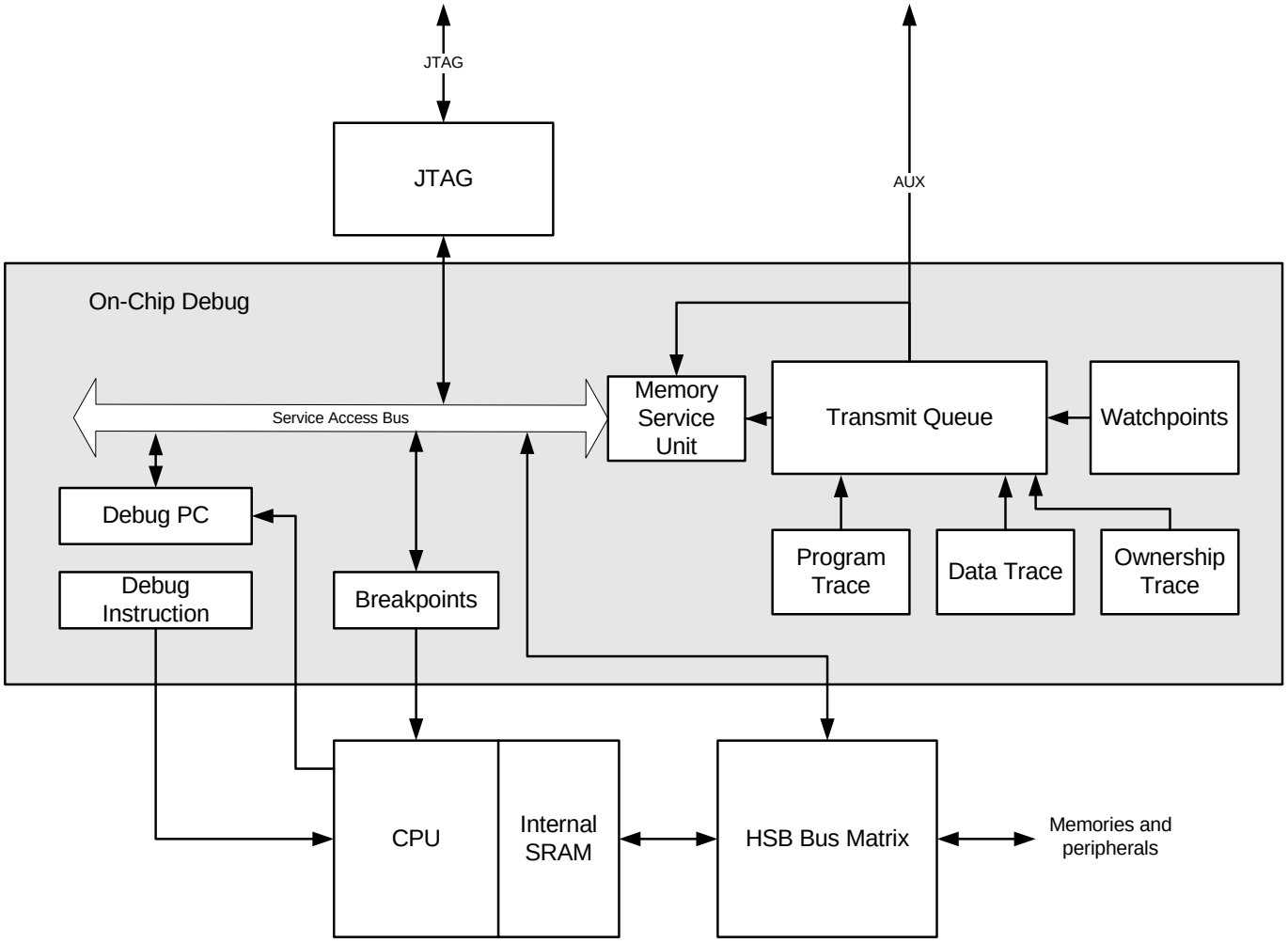
In addition to the mandatory Nexus debug features, the AT32UC3A3 implements several useful OCD features, such as:

- Debug Communication Channel between CPU and JTAG
- Run-time PC monitoring
- CRC checking
- NanoTrace
- Software Quality Assurance (SQA) support

The OCD features are controlled by OCD registers, which can be accessed by JTAG when the NEXUS_ACCESS JTAG instruction is loaded. The CPU can also access OCD registers directly using mtdr/mfdr instructions in any privileged mode. The OCD registers are implemented based on the recommendations in the Nexus 2.0 standard, and are detailed in the AVR32UC Technical Reference Manual.

34.3.3 Block Diagram

Figure 34-1. On-Chip Debug Block Diagram

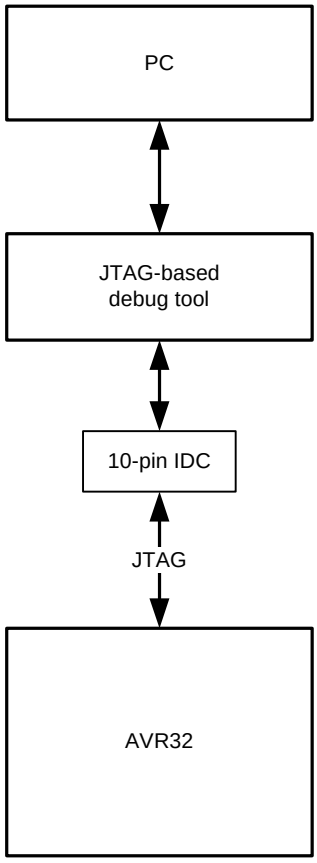


34.3.4 JTAG-based Debug Features

A debugger can control all OCD features by writing OCD registers over the JTAG interface. Many of these do not depend on output on the AUX port, allowing a JTAG-based debugger to be used.

A JTAG-based debugger should connect to the device through a standard 10-pin IDC connector as described in the AVR32UC Technical Reference Manual.

Figure 34-2. JTAG-based Debugger



34.3.4.1 Debug Communication Channel

The Debug Communication Channel (DCC) consists of a pair OCD registers with associated handshake logic, accessible to both CPU and JTAG. The registers can be used to exchange data between the CPU and the JTAG master, both runtime as well as in debug mode.

34.3.4.2 breakpoints

One of the most fundamental debug features is the ability to halt the CPU, to examine registers and the state of the system. This is accomplished by breakpoints, of which many types are available:

- Unconditional breakpoints are set by writing OCD registers by JTAG, halting the CPU immediately.
- Program breakpoints halt the CPU when a specific address in the program is executed.
- Data breakpoints halt the CPU when a specific memory address is read or written, allowing variables to be watched.
- Software breakpoints halt the CPU when the breakpoint instruction is executed.

When a breakpoint triggers, the CPU enters debug mode, and the D bit in the Status Register is set. This is a privileged mode with dedicated return address and return status registers. All privileged instructions are permitted. Debug mode can be entered as either OCD mode, running instructions from JTAG, or monitor mode, running instructions from program memory.

34.3.4.3 OCD mode

When a breakpoint triggers, the CPU enters OCD mode, and instructions are fetched from the Debug Instruction OCD register. Each time this register is written by JTAG, the instruction is executed, allowing the JTAG to execute CPU instructions directly. The JTAG master can e.g. read out the register file by issuing mtdr instructions to the CPU, writing each register to the Debug Communication Channel OCD registers.

34.3.4.4 monitor mode

Since the OCD registers are directly accessible by the CPU, it is possible to build a software-based debugger that runs on the CPU itself. Setting the Monitor Mode bit in the Development Control register causes the CPU to enter monitor mode instead of OCD mode when a breakpoint triggers. Monitor mode is similar to OCD mode, except that instructions are fetched from the debug exception vector in regular program memory, instead of issued by JTAG.

34.3.4.5 program counter monitoring

Normally, the CPU would need to be halted for a JTAG-based debugger to examine the current PC value. However, the AT32UC3A3 provides a Debug Program Counter OCD register, where the debugger can continuously read the current PC without affecting the CPU. This allows the debugger to generate a simple statistic of the time spent in various areas of the code, easing code optimization.

34.3.5 Memory Service Unit

The Memory Service Unit (MSU) is a block dedicated to test and debug functionality. It is controlled through a dedicated set of registers addressed through the MEMORY_SERVICE JTAG command.

34.3.5.1 Cyclic Redundancy Check (CRC)

The MSU can be used to automatically calculate the CRC of a block of data in memory. The OCD will then read out each word in the specified memory block and report the CRC32-value in an OCD register.

34.3.5.2 NanoTrace

The MSU additionally supports NanoTrace. This is an AVR32-specific feature, in which trace data is output to memory instead of the AUX port. This allows the trace data to be extracted by JTAG MEMORY_ACCESS, enabling trace features for JTAG-based debuggers. The user must write MSU registers to configure the address and size of the memory block to be used for NanoTrace. The NanoTrace buffer can be anywhere in the physical address range, including internal and external RAM, through an EBI, if present. This area may not be used by the application running on the CPU.

34.3.6 AUX-based Debug Features

Utilizing the Auxiliary (AUX) port gives access to a wide range of advanced debug features. Of prime importance are the trace features, which allow an external debugger to receive continuous information on the program execution in the CPU. Additionally, Event In and Event Out pins allow external events to be correlated with the program flow.

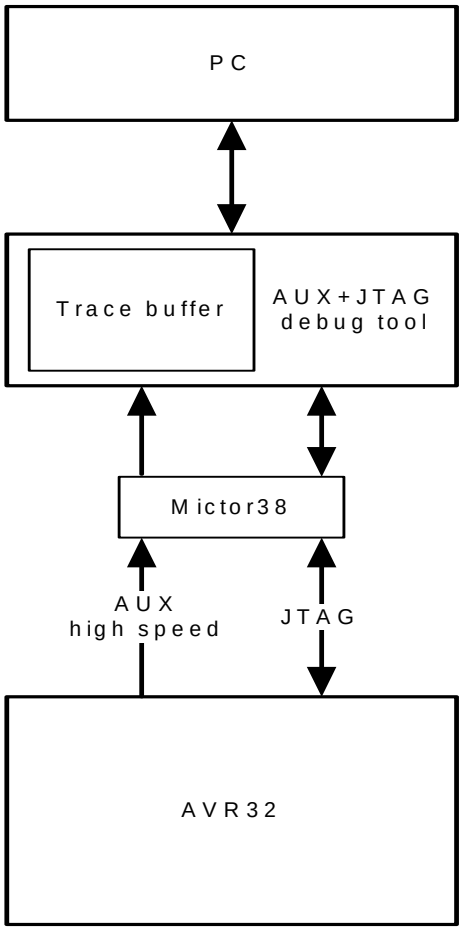
The AUX port contains a number of pins, as shown in Table 34-5 on page 898. These are multiplexed with I/O Controller lines, and must explicitly be enabled by writing OCD registers before the debug session starts. The AUX port is mapped to two different locations, selectable by OCD Registers, minimizing the chance that the AUX port will need to be shared with an application.

Debug tools utilizing the AUX port should connect to the device through a Nexus-compliant Mictor-38 connector, as described in the AVR32UC Technical Reference manual. This connector includes the JTAG signals and the RESET_N pin, giving full access to the programming and debug features in the device.

Table 34-5. Auxiliary Port Signals

Signal	Direction	Description
MCKO	Output	Trace data output clock
MDO[5:0]	Output	Trace data output
MSEO[1:0]	Output	Trace frame control
EVTI_N	Input	Event In
EVTO_N	Output	Event Out

Figure 34-3. AUX+JTAG based Debugger



34.3.6.1 trace operation

Trace features are enabled by writing OCD registers by JTAG. The OCD extracts the trace information from the CPU, compresses this information and formats it into variable-length messages according to the Nexus standard. The messages are buffered in a 16-frame transmit queue, and are output on the AUX port one frame at a time.

The trace features can be configured to be very selective, to reduce the bandwidth on the AUX port. In case the transmit queue overflows, error messages are produced to indicate loss of data. The transmit queue module can optionally be configured to halt the CPU when an overflow occurs, to prevent the loss of messages, at the expense of longer run-time for the program.

34.3.6.2 *program trace*

Program trace allows the debugger to continuously monitor the program execution in the CPU. Program trace messages are generated for every branch in the program, and contains compressed information, which allows the debugger to correlate the message with the source code to identify the branch instruction and target address.

34.3.6.3 *data trace*

Data trace outputs a message every time a specific location is read or written. The message contains information about the type (read/write) and size of the access, as well as the address and data of the accessed location. The AT32UC3A3 contains two data trace channels, each of which are controlled by a pair of OCD registers which determine the range of addresses (or single address) which should produce data trace messages.

34.3.6.4 *ownership trace*

Program and data trace operate on virtual addresses. In cases where an operating system runs several processes in overlapping virtual memory segments, the Ownership Trace feature can be used to identify the process switch. When the O/S activates a process, it will write the process ID number to an OCD register, which produces an Ownership Trace Message, allowing the debugger to switch context for the subsequent program and data trace messages. As the use of this feature depends on the software running on the CPU, it can also be used to extract other types of information from the system.

34.3.6.5 *watchpoint messages*

The breakpoint modules normally used to generate program and data breakpoints can also be used to generate Watchpoint messages, allowing a debugger to monitor program and data events without halting the CPU. Watchpoints can be enabled independently of breakpoints, so a breakpoint module can optionally halt the CPU when the trigger condition occurs. Data trace modules can also be configured to produce watchpoint messages instead of regular data trace messages.

34.3.6.6 *Event In and Event Out pins*

The AUX port also contains an Event In pin (EVTI_N) and an Event Out pin (EVTO_N). EVTI_N can be used to trigger a breakpoint when an external event occurs. It can also be used to trigger specific program and data trace synchronization messages, allowing an external event to be correlated to the program flow.

When the CPU enters debug mode, a Debug Status message is transmitted on the trace port. All trace messages can be timestamped when they are received by the debug tool. However, due to the latency of the transmit queue buffering, the timestamp will not be 100% accurate. To improve this, EVTO_N can toggle every time a message is inserted into the transmit queue, allowing trace messages to be timestamped precisely. EVTO_N can also toggle when a breakpoint module triggers, or when the CPU enters debug mode, for any reason. This can be used to measure precisely when the respective internal event occurs.

34.3.6.7 Software Quality Analysis (SQA)

Software Quality Analysis (SQA) deals with two important issues regarding embedded software development. *Code coverage* involves identifying untested parts of the embedded code, to improve test procedures and thus the quality of the released software. *Performance analysis* allows the developer to precisely quantify the time spent in various parts of the code, allowing bottlenecks to be identified and optimized.

Program trace must be used to accomplish these tasks without instrumenting (altering) the code to be examined. However, traditional program trace cannot reconstruct the current PC value without correlating the trace information with the source code, which cannot be done on-the-fly. This limits program trace to a relatively short time segment, determined by the size of the trace buffer in the debug tool.

The OCD system in AT32UC3A3 extends program trace with SQA capabilities, allowing the debug tool to reconstruct the PC value on-the-fly. Code coverage and performance analysis can thus be reported for an unlimited execution sequence.

34.3.7 Nexus OCD AUX port connections

If the OCD trace system is enabled, the trace system will take control over a number of pins, irrespectively of the GPIO configuration. Two different OCD trace pin mappings are possible, depending on the configuration of the OCD AXC register (AXO and AXS bitfileds). For details, see the AVR32UC Technical Reference Manual.

Table 34-6. Nexus OCD AUX port connections

Pin	AXO=0 & AXS=0	AXO=0 & AXS=1 or AXO=1	AXO=0 & AXS=2
EVTI_N	PB05	PA08	PX00
MDO[5]	PA00	PX56	PX06
MDO[4]	PA01	PX57	PX05
MDO[3]	PA03	PX58	PX04
MDO[2]	PA16	PA24	PX03
MDO[1]	PA13	PA23	PX02
MDO[0]	PA12	PA22	PX01
EVTO_N	PB06	PB06	PB06
MCKO	PB07	PA00	PB09
MSEO[1]	PA10	PA07	PX08
MSEO[0]	PA11	PX55	PX07

34.4 JTAG and Boundary Scan (JTAG)

Rev.: 2.0.0.3

34.4.1 Features

- **IEEE1149.1 compliant JTAG Interface**
- **Boundary-Scan Chain for board-level testing**
- **Direct memory access and programming capabilities through JTAG interface**
- **On-Chip Debug access in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0)**

34.4.2 Overview

The JTAG Interface offers a four pin programming and debug solution, including boundary scan support for board- level testing. The JTAG interface supports memory access, programming capabilities, and On-Chip Debug access.

Figure 34-4 on page 903 shows how the JTAG is connected in an AVR32 device. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (shift register) between the TDI-input and TDO-output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

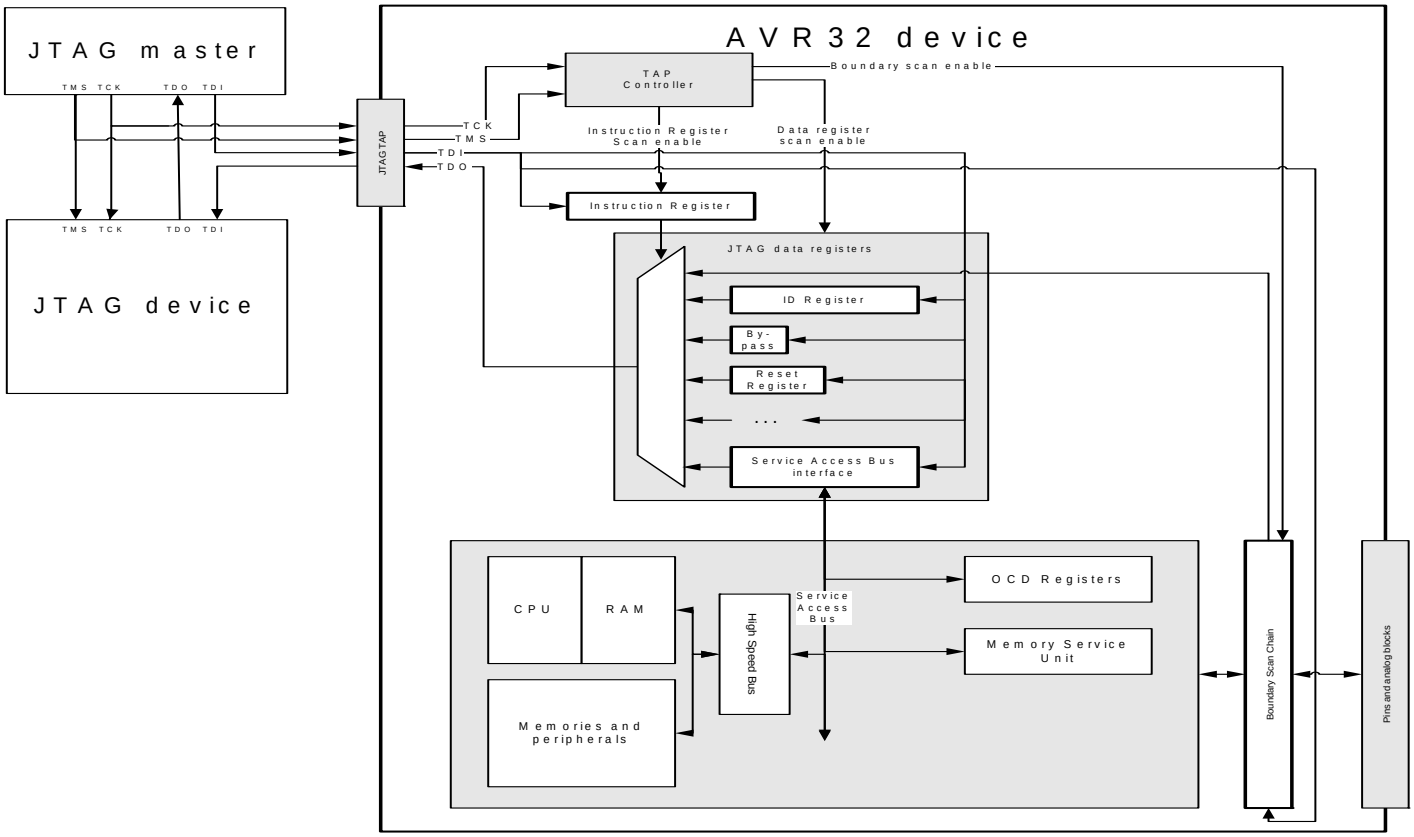
The ID Register, Bypass Register, and the Boundary-Scan Chain are the Data Registers used for board-level testing. The Reset Register can be used to keep the device reset during test or programming.

The Service Access Bus (SAB) interface contains address and data registers for the Service Access Bus, which gives access to On-Chip Debug, programming, and other functions in the device. The SAB offers several modes of access to the address and data registers, as discussed in Section 34.4.10.

Section 34.4.11 lists the supported JTAG instructions, with references to the description in this document.

34.4.3 Block Diagram

Figure 34-4. JTAG and Boundary Scan Access



34.4.4 I/O Lines Description

Table 34-7. I/O Lines Description

Pin Name	Pin Description	Type
TCK	Test Clock Input. Fully asynchronous to system clock frequency.	Input
TMS	Test Mode Select, sampled on rising TCK	Input
TDI	Test Data In, sampled on rising TCK.	Input
TDO	Test Data Out, driven on falling TCK.	Output

34.4.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

34.4.5.1 Clocks

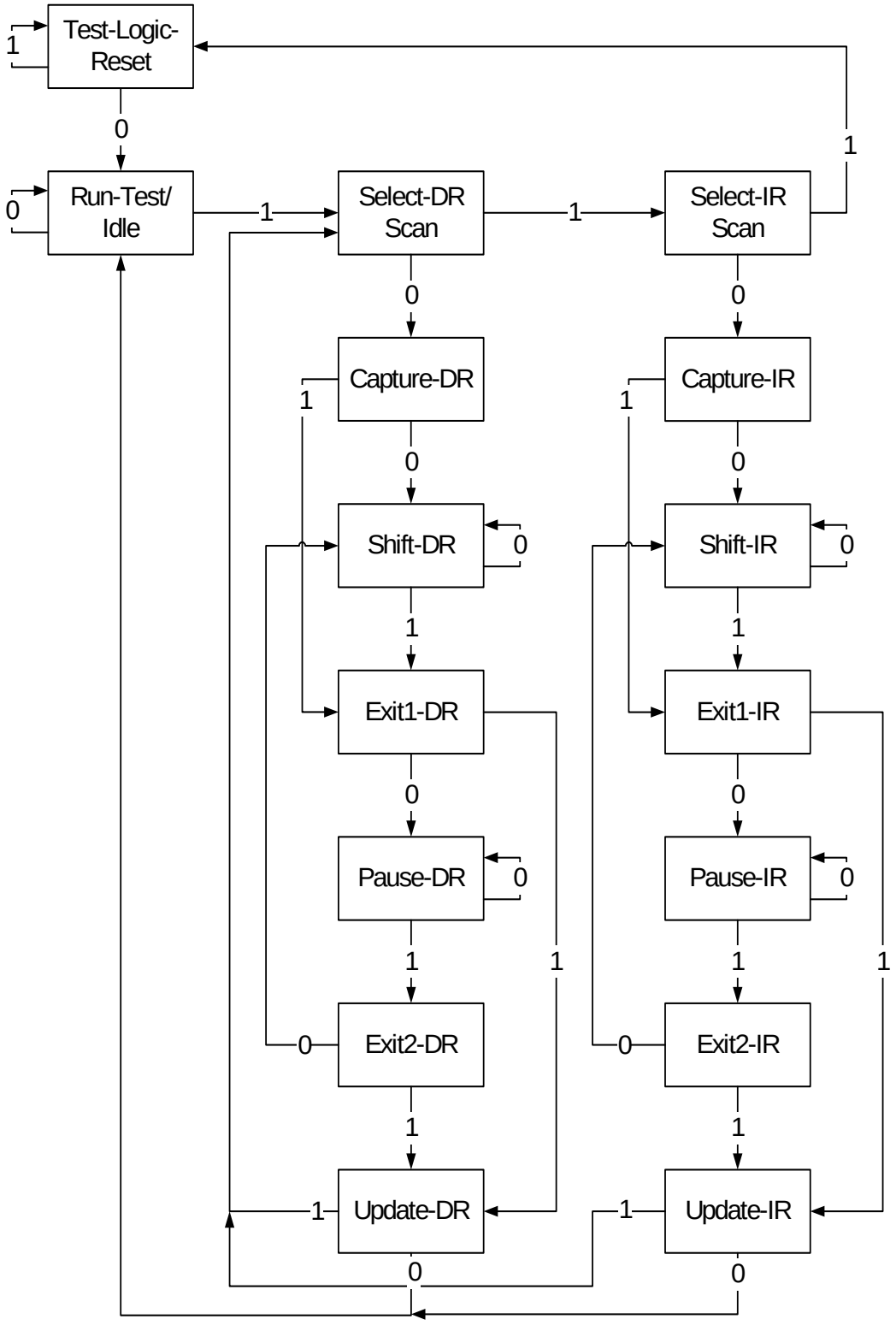
The JTAG Interface is using the external TCK pin as clock source. This clock must be provided by the user.

34.4.6 JTAG Interface

The JTAG Interface is accessed through the dedicated JTAG pins shown in Table 34-7 on page 903. The TMS control line navigates the TAP controller, as shown in Figure 34-5 on page 905. The TAP controller manages the serial access to the JTAG Instruction and Data registers. Data is scanned into the selected instruction or data register on TDI, and out of the register on TDO, in the Shift-IR and Shift-DR states, respectively. The LSB is shifted in and out first. TDO is high-Z in other states than Shift-IR and Shift-DR.

The device implements a 5-bit Instruction Register (IR). A number of public JTAG instructions defined by the JTAG standard are supported, as described in Section 34.4.12, as well as a number of AVR32-specific private JTAG instructions described in Section 34.4.13. Each instruction selects a specific data register for the Shift-DR path, as described for each instruction.

Figure 34-5. TAP Controller State Diagram



34.4.7 How to Initialize the Module

Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for 5 TCK clock periods. This sequence should always be applied at the start of a JTAG session to bring the TAP Controller into a defined state before applying JTAG commands. Applying a 0 on TMS for 1 TCK period brings the TAP Controller to the Run-Test/Idle state, which is the starting point for JTAG operations.

34.4.8 Typical Sequence

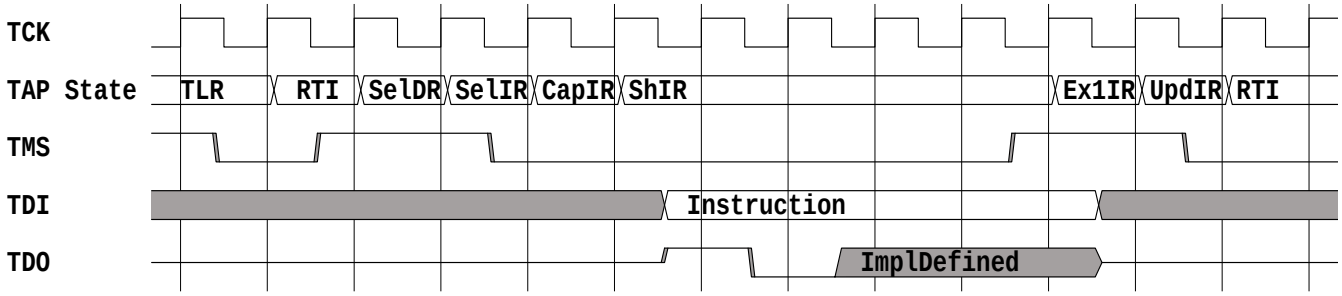
Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG Interface is:

34.4.8.1 Scanning in JTAG Instruction

At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register (Shift-IR) state. While in this state, shift the 5 bits of the JTAG instructions into the JTAG instruction register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 4 LSBs in order to remain in the Shift-IR state. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the shift register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.

Figure 34-6. Scanning in JTAG Instruction



34.4.8.2 Scanning in/out Data

At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register (Shift-DR) state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.

Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers.

34.4.9 Boundary-Scan

The Boundary-Scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by

the TDI/TDO signals to form a long shift register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-Scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the 4 TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST can be used for testing the Printed Circuit Board. Initial scanning of the data register path will show the ID-code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR32 device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the test mode. Entering reset, the outputs of any Port Pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external RESETn pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

When using the JTAG Interface for Boundary-Scan, the JTAG TCK clock is independent of the internal chip clock, which is not required to run.

34.4.10 Service Access Bus

The AVR32 architecture offers a common interface for access to On-Chip Debug, programming, and test functions. These are mapped on a common bus called the Service Access Bus (SAB), which is linked to the aWire through a bus master module, which also handles synchronization between the aWire and SAB clocks.

For more information about the SAB and a list of SAB slaves see the Service Access Bus chapter.

34.4.10.1 SAB Address Mode

The MEMORY_SIZED_ACCESS instruction allows a sized read or write to any 36-bit address on the bus. MEMORY_WORD_ACCESS is a shorthand instruction for 32-bit accesses to any 36-bit address, while the NEXUS_ACCESS instruction is a Nexus-compliant shorthand instruction for accessing the 32-bit OCD registers in the 7-bit address space reserved for these. These instructions require two passes through the Shift-DR TAP state: one for the address and control information, and one for data.

34.4.10.2 Block Transfer

To increase the transfer rate, consecutive memory accesses can be accomplished by the MEMORY_BLOCK_ACCESS instruction, which only requires a single pass through Shift-DR for data transfer only. The address is automatically incremented according to the size of the last SAB transfer.

34.4.10.3 Canceling a SAB Access

It is possible to abort an ongoing SAB access by the CANCEL_ACCESS instruction, to avoid hanging the bus due to an extremely slow slave.

34.4.10.4 Busy Reporting

As the time taken to perform an access may vary depending on system activity and current chip frequency, all the SAB access JTAG instructions can return a busy indicator. This indicates whether a delay needs to be inserted, or an operation needs to be repeated in order to be successful. If a new access is requested while the SAB is busy, the request is ignored.

The SAB becomes busy when:

- Entering Update-DR in the address phase of any read operation, e.g. after scanning in a NEXUS_ACCESS address with the read bit set.
- Entering Update-DR in the data phase of any write operation, e.g. after scanning in data for a NEXUS_ACCESS write.
- Entering Update-DR during a MEMORY_BLOCK_ACCESS.
- Entering Update-DR after scanning in a counter value for SYNC.
- Entering Update-IR after scanning in a MEMORY_BLOCK_ACCESS if the previous access was a read and data was scanned after scanning the address.

The SAB becomes ready again when:

- A read or write operation completes.
- A SYNC countdown completed.
- A operation is cancelled by the CANCEL_ACCESS instruction.

What to do if the busy bit is set:

- During Shift-IR: The new instruction is selected, but the previous operation has not yet completed and will continue (unless the new instruction is CANCEL_ACCESS). You may continue shifting the same instruction until the busy bit clears, or start shifting data. If shifting data, you must be prepared that the data shift may also report busy.
- During Shift-DR of an address: The new address is ignored. The SAB stays in address mode, so no data must be shifted. Repeat the address until the busy bit clears.
- During Shift-DR of read data: The read data are invalid. The SAB stays in data mode. Repeat scanning until the busy bit clears.
- During Shift-DR of write data: The write data are ignored. The SAB stays in data mode. Repeat scanning until the busy bit clears.

34.4.10.5 Error Reporting

The Service Access Bus may not be able to complete all accesses as requested. This may be because the address is invalid, the addressed area is read-only or cannot handle byte/halfword accesses, or because the chip is set in a protected mode where only limited accesses are allowed.

The error bit is updated when an access completes, and is cleared when a new access starts.

What to do if the error bit is set:

- During Shift-IR: The new instruction is selected. The last operation performed using the old instruction did not complete successfully.
- During Shift-DR of an address: The previous operation failed. The new address is accepted. If the read bit is set, a read operation is started.
- During Shift-DR of read data: The read operation failed, and the read data are invalid.
- During Shift-DR of write data: The previous write operation failed. The new data are accepted and a write operation started. This should only occur during block writes or stream writes. No error can occur between scanning a write address and the following write data.
- While polling with CANCEL_ACCESS: The previous access was cancelled. It may or may not have actually completed.

34.4.11 JTAG Instruction Summary

The implemented JTAG instructions in the AVR32 are shown in the table below.

Table 34-8. JTAG Instruction Summary

Instruction OPCODE	Instruction	Description	Page
0x01	IDCODE	Select the 32-bit ID register as data register.	910
0x02	SAMPLE_PRELOAD	Take a snapshot of external pin values without affecting system operation.	910
0x03	EXTEST	Select boundary scan chain as data register for testing circuitry external to the device.	910
0x04	INTEST	Select boundary scan chain for internal testing of the device.	911
0x06	CLAMP	Bypass device through Bypass register, while driving outputs from boundary scan register.	911
0x0C	AVR_RESET	Apply or remove a static reset to the device	918
0x0F	CHIP_ERASE	Erase the device	918
0x10	NEXUS_ACCESS	Select the SAB Address and Data registers as data register for the TAP. The registers are accessed in Nexus mode.	912
0x11	MEMORY_WORD_ACCESS	Select the SAB Address and Data registers as data register for the TAP.	915
0x12	MEMORY_BLOCK_ACCESS	Select the SAB Data register as data register for the TAP. The address is auto-incremented.	916
0x13	CANCEL_ACCESS	Cancel an ongoing Nexus or Memory access.	917
0x14	MEMORY_SERVICE	Select the SAB Address and Data registers as data register for the TAP. The registers are accessed in Memory Service mode.	913
0x15	MEMORY_SIZED_ACCESS	Select the SAB Address and Data registers as data register for the TAP.	914
0x17	SYNC	Synchronization counter	918
0x1C	HALT	Halt the CPU for safe programming.	919
0x1F	BYPASS	Bypass this device through the bypass register.	911
Others	N/A	Acts as BYPASS	

34.4.11.1 Security Restrictions

When the security fuse in the Flash is programmed, the following JTAG instructions are restricted:

- NEXUS_ACCESS
- MEMORY_WORD_ACCESS
- MEMORY_BLOCK_ACCESS
- MEMORY_SIZED_ACCESS

For description of what memory locations remain accessible, please refer to the SAB address map.

Full access to these instructions is re-enabled when the security fuse is erased by the CHIP_ERASE JTAG instruction.

Note that the security bit will read as programmed and block these instructions also if the Flash Controller is statically reset.

Other security mechanisms can also restrict these functions. If such mechanisms are present they are listed in the SAB address map section.

34.4.12 Public JTAG Instructions

34.4.12.1 IDCODE

This instruction selects the 32 bit ID register as Data Register. The ID register consists of a version number, a device number, and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: The static IDCODE value is latched into the shift register.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

34.4.12.2 SAMPLE_PRELOAD

JTAG instruction for taking a snap-shot of the input/output pins without affecting the system operation, and pre-loading the scan chain without updating the DR-latch. The Boundary-Scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-Scan Chain.
- Shift-DR: The Boundary-Scan Chain is shifted by the TCK input.

34.4.12.3 EXTEST

JTAG instruction for selecting the Boundary-Scan Chain as Data Register for testing circuitry external to the AVR32 package. The contents of the latched outputs of the Boundary-Scan chain is driven out as soon as the JTAG IR-register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins is sampled into the Boundary-Scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

34.4.12.4 INTEST

This instruction selects the Boundary-Scan Chain as Data Register for testing internal logic in the device. The logic inputs are determined by the Boundary-Scan Chain, and the logic outputs are captured by the Boundary-Scan chain. The device output pins are driven from the Boundary-Scan Chain.

The active states are:

- Capture-DR: Data from the internal logic is sampled into the Boundary-Scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to internal logic inputs.

34.4.12.5 CLAMP

This instruction selects the Bypass register as Data Register. The device output pins are driven from the Boundary-Scan Chain.

The active states are:

- Capture-DR: Loads a logic ‘0’ into the Bypass Register.
- Shift-DR: Data is scanned from TDI to TDO through the Bypass register.

34.4.12.6 BYPASS

JTAG instruction selecting the 1-bit Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic ‘0’ into the Bypass Register.
- Shift-DR: Data is scanned from TDI to TDO through the Bypass register.

34.4.13 Private JTAG Instructions

34.4.13.1 Notation

The AVR32 defines a number of private JTAG instructions. Each instruction is briefly described in text, with details following in table form.

Table 34-10 on page 912 shows bit patterns to be shifted in a format like "**peb01**". Each character corresponds to one bit, and eight bits are grouped together for readability. The rightmost bit is always shifted first, and the leftmost bit shifted last. The symbols used are shown in Table 34-9.

Table 34-9. Symbol Description

Symbol	Description
0	Constant low value - always reads as zero.
1	Constant high value - always reads as one.
a	An address bit - always scanned with the least significant bit first
b	A busy bit. Reads as one if the SAB was busy, or zero if it was not. See Section 34.4.10.4 for details on how the busy reporting works.
d	A data bit - always scanned with the least significant bit first.
e	An error bit. Reads as one if an error occurred, or zero if not. See Section 34.4.10.5 for details on how the error reporting works.

Table 34-9. Symbol Description

p	The chip protected bit. Some devices may be set in a protected state where access to chip internals are severely restricted. See the documentation for the specific device for details. On devices without this possibility, this bit always reads as zero.
r	A direction bit. Set to one to request a read, set to zero to request a write.
s	A size bit. The size encoding is described where used.
x	A don't care bit. Any value can be shifted in, and output data should be ignored.

In many cases, it is not required to shift all bits through the data register. Bit patterns are shown using the full width of the shift register, but the suggested or required bits are emphasized using **bold** text. I.e. given the pattern "**aaaaaaar** xxxxxxxx xxxxxxxx xxxxxxxx xx", the shift register is 34 bits, but the test or debug unit may choose to shift only 8 bits "**aaaaaaar**".

The following describes how to interpret the fields in the instruction description tables:

Table 34-10. Instruction Description

Instruction	Description
IR input value	Shows the bit pattern to shift into IR in the Shift-IR state in order to select this instruction. The pattern is show both in binary and in hexadecimal form for convenience. Example: **10000** (0x10)
IR output value	Shows the bit pattern shifted out of IR in the Shift-IR state when this instruction is active. Example: peb01
DR Size	Shows the number of bits in the data register chain when this instruction is active. Example: 34 bits
DR input value	Shows which bit pattern to shift into the data register in the Shift-DR state when this instruction is active. Multiple such lines may exist, e.g. to distinguish between reads and writes. Example: aaaaaaar xxxxxxxx xxxxxxxx xxxxxxxx xx
DR output value	Shows the bit pattern shifted out of the data register in the Shift-DR state when this instruction is active. Multiple such lines may exist, e.g. to distinguish between reads and writes. Example: xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

34.4.13.2 NEXUS_ACCESS

This instruction allows Nexus-compliant access to On-Chip Debug registers through the SAB. OCD registers are addressed by their register index, as listed in the AVR32 Technical Reference Manual. The 7-bit register index, a read/write control bit, and the 32-bit data is accessed through the JTAG port.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the NEXUS_ACCESS instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

Starting in Run-Test/Idle, OCD registers are accessed in the following way:

1. Select the DR Scan path.
2. Scan in the 7-bit address for the OCD register and a direction bit (1=read, 0=write).

3. Go to Update-DR and re-enter Select-DR Scan.
4. For a read operation, scan out the contents of the addressed register. For a write operation, scan in the new contents of the register.
5. Return to Run-Test/Idle.

For any operation, the full 7 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 34-11. NEXUS_ACCESS Details

Instructions	Details
IR input value	**10000** (0x10)
IR output value	peb01
DR Size	34 bits
DR input value (Address phase)	**aaaaaaar** xxxxxxxx xxxxxxxx xxxxxxxx xx
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xx
DR output value (Address phase)	xx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb
DR output value (Data read phase)	eb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

34.4.13.3 MEMORY_SERVICE

This instruction allows access to registers in an optional Memory Service unit. Memory Service registers are addressed by their register index, as listed in the Memory Service documentation. The 7-bit register index, a read/write control bit, and the 32-bit data is accessed through the JTAG port.

The Memory Service unit may offer features such as CRC calculation of memory, debug trace support, and test features. Please refer to the Memory Service Unit documentation and the part specific documentation for details.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the MEMORY_SERVICE instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

Starting in Run-Test/Idle, Memory Service registers are accessed in the following way:

1. Select the DR Scan path.
2. Scan in the 7-bit address for the Memory Service register and a direction bit (1=read, 0=write).
3. Go to Update-DR and re-enter Select-DR Scan.
4. For a read operation, scan out the contents of the addressed register. For a write operation, scan in the new contents of the register.
5. Return to Run-Test/Idle.

For any operation, the full 7 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 34-12. MEMORY_SERVICE Details

Instructions	Details
IR input value	**10100** (0x14)
IR output value	peb01
DR Size	34 bits
DR input value (Address phase)	**aaaaaaar** xxxxxxxx xxxxxxxx xxxxxxxx xx
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xx
DR output value (Address phase)	xx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb
DR output value (Data read phase)	eb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

34.4.13.4 MEMORY_SIZED_ACCESS

This instruction allows access to the entire Service Access Bus data area. Data are accessed through a 36-bit byte index, a 2-bit size, a direction bit, and 8, 16, or 32 bits of data. Not all units mapped on the SAB bus may support all sizes of accesses, e.g. some may only support word accesses.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the MEMORY_SIZED_ACCESS instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

The size field is encoded as i Table 34-13.

Table 34-13. Size Field Semantics

Size field value	Access size	Data alignment
00	Byte (8 bits)	Address modulo 4 : data alignment 0: **dddddddd** xxxxxxxx xxxxxxxx xxxxxxxx 1: xxxxxxxx **dddddddd** xxxxxxxx xxxxxxxx 2: xxxxxxxx xxxxxxxx **dddddddd** xxxxxxxx 3: xxxxxxxx xxxxxxxx xxxxxxxx **dddddddd**

Table 34-13. Size Field Semantics

Size field value	Access size	Data alignment
01	Halfword (16 bits)	Address modulo 4 : data alignment 0: **dddddddd dddddddd** xxxxxxxx xxxxxxxx 1: Not allowed 2: xxxxxxxx xxxxxxxx **dddddddd dddddddd** 3: Not allowed
10	Word (32 bits)	Address modulo 4 : data alignment 0: **dddddddd dddddddd dddddddd dddddddd** 1: Not allowed 2: Not allowed 3: Not allowed
11	Reserved	N/A

Starting in Run-Test/Idle, SAB data are accessed in the following way:

1. Select the DR Scan path.
2. Scan in the 36-bit address of the data to access, a 2-bit access size, and a direction bit (1=read, 0=write).
3. Go to Update-DR and re-enter Select-DR Scan.
4. For a read operation, scan out the contents of the addressed area. For a write operation, scan in the new contents of the area.
5. Return to Run-Test/Idle.

For any operation, the full 36 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 34-14. MEMORY_SIZED_ACCESS Details

Instructions	Details
IR input value	**10101** (0x15)
IR output value	peb01
DR Size	39 bits
DR input value (Address phase)	aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaassr
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xxxxxxx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xxxxxxx
DR output value (Address phase)	xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb
DR output value (Data read phase)	xxxxxeb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xxxxxxx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

34.4.13.5 MEMORY_WORD_ACCESS

This instruction allows access to the entire Service Access Bus data area. Data are accessed through a 34-bit word index, a direction bit, and 32 bits of data. This instruction is identical to MEMORY_SIZED_ACCESS except that it always does word sized accesses. The size field is implied, and the two lowest address bits are removed.

Note: This instruction was previously known as MEMORY_ACCESS, and is provided for backwards compatibility.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the MEMORY_WORD_ACCESS instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

Starting in Run-Test/Idle, SAB data are accessed in the following way:

1. Select the DR Scan path.
2. Scan in the 34-bit address of the data to access, and a direction bit (1=read, 0=write).
3. Go to Update-DR and re-enter Select-DR Scan.
4. For a read operation, scan out the contents of the addressed area. For a write operation, scan in the new contents of the area.
5. Return to Run-Test/Idle.

For any operation, the full 34 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 34-15. MEMORY_WORD_ACCESS Details

Instructions	Details
IR input value	**10001** (0x11)
IR output value	peb01
DR Size	35 bits
DR input value (Address phase)	aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aar
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xxx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xxx
DR output value (Address phase)	xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xeb
DR output value (Data read phase)	xeb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xxx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

34.4.13.6 MEMORY_BLOCK_ACCESS

This instruction allows access to the entire SAB data area. Up to 32 bits of data are accessed at a time, while the address is sequentially incremented from the previously used address.

In this mode, the SAB address, size, and access direction is not provided with each access. Instead, the previous address is auto-incremented depending on the specified size and the previous operation repeated. The address must be set up in advance with MEMORY_SIZE_ACCESS or MEMORY_WORD_ACCESS. It is allowed, but not required, to shift data after shifting the address.

This instruction is primarily intended to speed up large quantities of sequential word accesses. It is possible to use it also for byte and halfword accesses, but the overhead in this is case much larger as 32 bits must still be shifted for each access.

The following sequence should be used:

1. Use the MEMORY_SIZE_ACCESS or MEMORY_WORD_ACCESS to read or write the first location.
2. Apply MEMORY_BLOCK_ACCESS in the IR Scan path.
3. Select the DR Scan path. The address will now have incremented by 1, 2, or 4 (corresponding to the next byte, halfword, or word location).
4. For a read operation, scan out the contents of the next addressed location. For a write operation, scan in the new contents of the next addressed location.
5. Go to Update-DR.
6. If the block access is not complete, return to Select-DR Scan and repeat the access.
7. If the block access is complete, return to Run-Test/Idle.

For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 34-16. MEMORY_BLOCK_ACCESS Details

Instructions	Details
IR input value	**10010** (0x12)
IR output value	peb01
DR Size	34 bits
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xx
DR output value (Data read phase)	eb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

The overhead using block word access is 4 cycles per 32 bits of data, resulting in an 88% transfer efficiency, or 2.1 MBytes per second with a 20 MHz TCK frequency.

34.4.13.7 CANCEL_ACCESS

If a very slow memory location is accessed during a SAB memory access, it could take a very long time until the busy bit is cleared, and the SAB becomes ready for the next operation. The CANCEL_ACCESS instruction provides a possibility to abort an ongoing transfer and report a timeout to the user.

When the CANCEL_ACCESS instruction is selected, the current access will be terminated as soon as possible. There are no guarantees about how long this will take, as the hardware may not always be able to cancel the access immediately. The SAB is ready to respond to a new command when the busy bit clears.

Table 34-17. CANCEL_ACCESS Details

Instructions	Details
IR input value	**10011** (0x13)
IR output value	peb01
DR Size	1
DR input value	x
DR output value	0

34.4.13.8 SYNC

This instruction allows external debuggers and testers to measure the ratio between the external JTAG clock and the internal system clock. The SYNC data register is a 16-bit counter that counts down to zero using the internal system clock. The busy bit stays high until the counter reaches zero.

Starting in Run-Test/Idle, SYNC instruction is used in the following way:

1. Select the DR Scan path.
2. Scan in an 16-bit counter value.
3. Go to Update-DR and re-enter Select-DR Scan.
4. Scan out the busy bit, and retry until the busy bit clears.
5. Calculate an approximation to the internal clock speed using the elapsed time and the counter value.
6. Return to Run-Test/Idle.

The full 16-bit counter value must be provided when starting the synch operation, or the result will be undefined. When reading status, shifting may be terminated once the required number of bits have been acquired.

Table 34-18. SYNC_ACCESS Details

Instructions	Details
IR input value	**10111** (0x17)
IR output value	peb01
DR Size	16 bits
DR input value	dddddddd dddddddd
DR output value	xxxxxxxx xxxxxxeb

34.4.13.9 AVR_RESET

This instruction allows a debugger or tester to directly control separate reset domains inside the chip. The shift register contains one bit for each controllable reset domain. Setting a bit to one resets that domain and holds it in reset. Setting a bit to zero releases the reset for that domain.

See the device specific documentation for the number of reset domains, and what these domains are.

For any operation, all bits must be provided or the result will be undefined.

Table 34-19. AVR_RESET Details

Instructions	Details
IR input value	**01100** (0x0C)
IR output value	p0001
DR Size	Device specific.
DR input value	Device specific.
DR output value	Device specific.

34.4.13.10 CHIP_ERASE

This instruction allows a programmer to completely erase all nonvolatile memories in a chip. This will also clear any security bits that are set, so the device can be accessed normally. In

devices without non-volatile memories this instruction does nothing, and appears to complete immediately.

The erasing of non-volatile memories starts as soon as the CHIP_ERASE instruction is selected. The CHIP_ERASE instruction selects a 1 bit bypass data register.

A chip erase operation should be performed as:

1. Scan in the HALT instruction
2. Scan in the value 1 to halt the CPU
3. Stay in Run-Test/Idle for 10 TCK cycles to let the halt command propagate properly
4. Scan in the CHIP_ERASE instruction
5. Keep scanning the CHIP_ERASE instruction until the busy bit is cleared and the protection bit is cleared.
6. Scan in the HALT instruction
7. Scan in the value 0 to release the CPU
8. Return to Run-Test/Idle
9. Stay in Run-Test/Idle for 10 TCK cycles to let the halt command propagate properly.

Table 34-20. CHIP_ERASE Details

Instructions	Details
IR input value	**01111** (0x0F)
IR output value	p0b01 Where b is the *busy* bit.
DR Size	1 bit
DR input value	x
DR output value	0

34.4.13.11 HALT

This instruction allows a programmer to easily stop the CPU to ensure that it does not execute invalid code during programming.

This instruction selects a 1-bit halt register. Setting this bit to one resets the device and halts the CPU. Setting this bit to zero resets the device and releases the CPU to run normally. The value shifted out from the data register is one if the CPU is halted.

The HALT instruction can be used in the following way:

1. Scan in the HALT instruction
2. Scan in the value 1 to halt the CPU
3. Stay in Run-Test/Idle for 10 TCK cycles to let the command propagate properly
4. Use any MEMORY_* instructions to program the device
5. Scan in the HALT instruction
6. Scan in the value 0 to release the CPU
7. Return to Run-Test/Idle

8. Stay in Run-Test/Idle for 10 TCK cycles to let the command propagate properly - the device now runs with the new code.

Table 34-21. HALT Details

Instructions	Details
IR input value	**11100** (0x1C)
IR output value	p0001
DR Size	1 bit
DR input value	d
DR output value	d

34.4.14 JTAG Data Registers

The following device specific registers can be selected as JTAG scan chain depending on the instruction loaded in the JTAG Instruction Register. Additional registers exist, but are implicitly described in the functional description of the relevant instructions.

34.4.14.1 Device Identification Register

The Device Identification Register contains a unique identifier for each product. The register is selected by the IDCODE instruction, which is the default instruction after a JTAG reset.

	MSB																			LSB
Bit	31			28	27							12	11						1	0
Device ID	**Revision**				**Part Number**								**Manufacturer ID**							**1**
	4 bits				16 bits								11 bits							1 bit

Revision — **This is a 4 bit number identifying the revision of the component. Rev A = 0x0, B = 0x1, etc.**

Part Number — **The part number is a 16 bit code identifying the component.**

Manufacturer ID — **The Manufacturer ID is a 11 bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is 0x01F.**

- *Device specific ID codes*

The different device configurations have different JTAG ID codes, as shown in Table 34-22. Note that if the flash controller is statically reset, the ID code will be undefined.

Table 34-22. Device and JTAG ID

Device name	JTAG ID code (r is the revision number)
AT32UC3A3256S	0xr202003F
AT32UC3A3128S	0xr202103F
AT32UC3A364S	0xr202203F
AT32UC3A3256	0xr202603F
AT32UC3A3128	0xr202703F
AT32UC3A364	0xr202803F

34.4.14.2 Reset register

The reset register is selected by the AVR_RESET instruction and contains one bit for each reset domain in the device. Setting each bit to one will keep that domain reset until the bit is cleared.

										LSB
Bit	4				3		2		1	0
Device ID	**OCD**				**APP**		**RESERVED**		**RESERVED**	**CPU**

CPU	**CPU**
APP	**HSB and PB buses**
OCD	**On-Chip Debug logic and registers**
RSERVED	**No effect**

Note: This register is primarily intended for compatibility with other AVR32 devices. Certain operations may not function correctly when parts of the system are reset. It is generally recommended to only write 0x11111 or 0x00000 to these bits to ensure no unintended side effects occur.

34.4.14.3 Boundary-Scan Chain

The Boundary-Scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as driving and observing the logic levels between the digital I/O pins and the internal logic. Typically, output value, output enable, and input data are all available in the boundary scan chain.

The boundary scan chain is described in the BDSL (Boundary Scan Description Language) file available at the Atmel web site.

35. Electrical Characteristics

35.1 Absolute Maximum Ratings*

Operating Temperature ..... -40°C to +85°C

Storage Temperature ....-60°C to +150°C

Voltage on Input Pin
with respect to Ground .....-0.3V to 3.6V

Maximum Operating Voltage (VDDCORE) 1.95V

Maximum Operating Voltage (VDDIO)..... 3.6V

Total DC Output Current on all I/O Pin
for TQFP144 packag ..... 370 mA
for TBGA144 package ..... 370 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.

35.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are certified for a junction temperature up to T_J = 100°C.

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{VDDIO}	DC Supply Peripheral I/Os			3.0		3.6	V
V_{IL}	Input Low-level Voltage			-0.3		+0.8	V
V_{IH}	Input High-level Voltage			2.0		V_{VDDIO}+0.3	V
V_{OL}	Output Low-level Voltage	I_{OL}=-2mA for Pin drive x1 I_{OL}=-4mA for Pin drive x2 I_{OL}=-8mA for Pin drive x3				0.4	V
V_{OH}	Output High-level Voltage	I_{OL}=2mA for Pin drive x1 I_{OL}=4mA for Pin drive x2 I_{OL}=8mA for Pin drive x3		V_{VDDIO}-0.4			V
I_{LEAK}	Input Leakage Current	Pullup resistors disabled					µA
C_{IN}	Input Capacitance				7		pF
R_{PULLUP}	Pull-up Resistance			9	15	25	Ohm
I_O	Output Current Pin drive x1 Pin drive x2 Pin drive x3 See Table 35-1					2.0 4.0 8.0	mA
I_{SC}	Static Current	On V_{VDDIN} = 3.3V, CPU in static mode	T_A =25°C		TBD		µA
			T_A =85°C		TBD		µA

Table 35-1. Pins drive capabilities

PIN	Drive	PIN	Drive	PIN	Drive
PA00	x3	PB05	x1	PX24	x2
PA01	x1	PB06	x1	PX25	x2
PA02	x1	PB07	x3	PX26	x2
PA03	x1	PB08	x2	PX27	x2
PA04	x1	PB09	x2	PX28	x2
PA05	x1	PB10	x2	PX29	x2
PA06	x1	PB11	x1	PX30	x2
PA07	x1	PC00	x1	PX31	x2
PA08	x3	PC01	x1	PX32	x2
PA09	x2	PC02	x1	PX33	x2
PA10	x2	PC03	x1	PX34	x2
PA11	x2	PC04	x1	PX35	x2
PA12	x1	PC05	x1	PX36	x2
PA13	x1	PX00	x2	PX37	x2
PA14	x1	PX01	x2	PX38	x2
PA15	x1	PX02	x2	PX39	x2
PA16	x1	PX03	x2	PX40	x2
PA17	x1	PX04	x2	PX41	x2
PA18	x1	PX05	x2	PX42	x2
PA19	x1	PX06	x2	PX43	x2
PA20	x1	PX07	x2	PX44	x2
PA21	x1	PX08	x2	PX45	x3
PA22	x1	PX09	x2	PX46	x2
PA23	x1	PX10	x2	PX47	x2
PA24	x1	PX11	x2	PX48	x2
PA25	x1	PX12	x2	PX49	x2
PA26	x1	PX13	x2	PX50	x2
PA27	x2	PX14	x2	PX51	x2
PA28	x1	PX15	x2	PX52	x2
PA29	x1	PX16	x2	PX53	x2
PA30	x1	PX17	x2	PX54	x2
PA31	x1	PX18	x2	PX55	x2
PB00	x1	PX19	x2	PX56	x2
PB01	x1	PX20	x2	PX57	x2
PB02	x1	PX21	x2	PX58	x2
PB03	x1	PX22	x2	PX59	x2
PB04	x1	PX23	x2		

35.3 Regulator characteristics

35.3.1 Electrical characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{VDDIN}	Supply voltage (input)		2.7	3.3	3.6	V
$V_{VDDCORE}$	Supply voltage (output)		1.81	1.85	1.89	V
I_{OUT}	Maximum DC output current with $V_{VDDIN = 3.3V}$				100	mA
	Maximum DC output current with $V_{VDDIN = 2.7V}$				90	mA

35.3.2 Decoupling requirements

Symbol	Parameter	Condition	Typ.	Techno.	Units
C_{IN1}	Input Regulator Capacitor 1		1	NPO	nF
C_{IN2}	Input Regulator Capacitor 2		4.7	X7R	uF
C_{OUT1}	Output Regulator Capacitor 1		470	NPO	pF
C_{OUT2}	Output Regulator Capacitor 2		2.2	X7R	uF

35.3.3 BOD

Table 35-2. BODLEVEL Values

BODLEVEL Value	Typ.	Units.
111111b	1.58	V
101000b	1.62	V
100000b	1.67	V
011000b	1.77	V
000000b	1.92	V

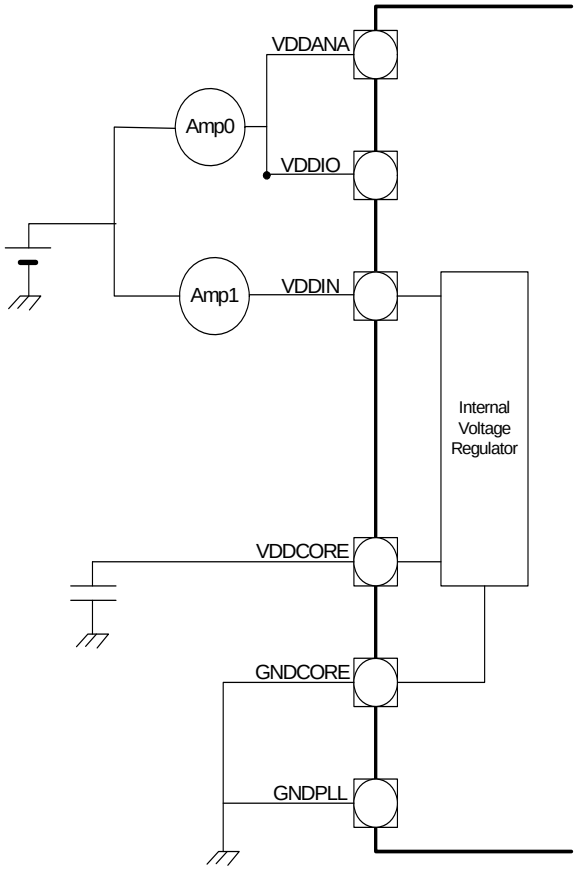
The values in Table 35-2 describes the values of the BODLEVEL in the flash FGPFRL register.

35.4 Power Consumption

The values in Table 35-3 and Table 35-4 on page 927 are measured values of power consumption with operating conditions as follows:

- V_{DDIO} = 3.3V
- T_A = 25°C, T_A = 85°C
- I/Os are configured in input, pull-up enabled.

Figure 35-1. Measurement setup



These figures represent the power consumption measured on the power supplies.

Table 35-3. Power Consumption for Different Modes(1)

Mode	Conditions		Consumption Typ.	Unit
Active	CPU running from flash. CPU clocked from PLL0 at f MHz Voltage regulator is on. XIN0 : external clock. (1) XIN1 stopped. XIN32 stopped PLL0 running All peripheral clocks activated. GPIOs on internal pull-up. JTAG unconnected with ext pull-up.	f = 12 MHz	10	mA
		f = 24 MHz	18	mA
		f = 36MHz	27	mA
		f = 50 MHz	34	mA
		f = 60 MHz	42	mA
Static	Typ : Ta = 25 °C CPU is in static mode GPIOs on internal pull-up. All peripheral clocks de-activated. DM and DP pins connected to ground. XIN0,Xin1 and XIN2 are stopped	on Amp0	0	uA
		on Amp1	<100	uA

1. Core frequency is generated from XIN0 using the PLL so that 140 MHz < fpll0 < 160 MHz and 10 MHz < fxin0 < 12MHz

Table 35-4. Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
GPIO	37	µA/MHz
SMC	10	
SDRAMC	4	
ADC	18	
EBI	31	
INTC	25	
TWI	14	
PDCA	30	
RTC	7	
SPI	13	
SSC	13	
TC	10	
USART	35	

35.5 Clock Characteristics

These parameters are given in the following conditions:

- V_{DDCORE} = 1.8V

- Ambient Temperature = 25°C

35.5.1 CPU/HSB Clock Characteristics

Table 35-5. Core Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPCPU})$	CPU Clock Frequency			66	MHz
t_{CPCPU}	CPU Clock Period		15.5		ns

35.5.2 PBA Clock Characteristics

Table 35-6. PBA Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPBA})$	PBA Clock Frequency			66	MHz
t_{CPPBA}	PBA Clock Period		15.5		ns

35.5.3 PBB Clock Characteristics

Table 35-7. PBB Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPBB})$	PBB Clock Frequency			66	MHz
t_{CPPBB}	PBB Clock Period		15.5		ns

35.5.4 XIN Clock Characteristics

Table 35-8. XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency		3	24	MHz
t_{CPXIN}	XIN Clock Period		20.0		ns
t_{CHXIN}	XIN Clock High Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
t_{CLXIN}	XIN Clock Low Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN Input Capacitance	(1)		TBD	pF
R_{IN}	XIN Pulldown Resistor	(1)		TBD	kΩ

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.)

35.6 Crystal Oscillator Characteristis

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

35.6.1 32 KHz Oscillator Characteristics

Table 35-9. 32 KHz Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency			32 768		Hz
C_L	Equivalent Load Capacitance		6		12.5	pF
	Duty Cycle		TBD		TBD	%
t_{ST}	Startup Time	R_S = TBD kΩ, C_L = TBD pF(1)			TBD	ms

Note: 1. R_S is the equivalent series resistance, C_L is the equivalent load capacitance.

35.6.2 Main Oscillators Characteristics

Table 35-10. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		3		16	MHz
C_{L1}, C_{L2}	Internal Load Capacitance ($C_{L1} = C_{L2}$)			12		pF
C_L	Equivalent Load Capacitance			TBD		pF
	Duty Cycle		40	50	60	%
t_{ST}	Startup Time				TBD	ms
I_{OSC}	Current Consumption	Active mode @TBD MHz			TBD	µA
		Standby mode @TBD V			TBD	µA

Notes: 1. C_S is the shunt capacitance

35.6.3 PLL Characteristics

Table 35-11. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{OUT}	Output Frequency		80		240	MHz
F_{IN}	Input Frequency		TBD		TBD	MHz
I_{PLL}	Current Consumption	active mode			TBD	mA
		standby mode			TBD	µA

Note: 1. Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

35.7 ADC Characteristics

Table 35-12. Channel Conversion Time and ADC Clock

Parameter	Conditions	Min	Typ	Max	Units
ADC Clock Frequency	10-bit resolution mode			5	MHz
ADC Clock Frequency	8-bit resolution mode			8	MHz
Startup Time	Return from Idle Mode			20	µs
Track and Hold Acquisition Time		600			ns
Conversion Time	ADC Clock = 5 MHz			2	µs
Conversion Time	ADC Clock = 8 MHz			1.25	µs
Throughput Rate	ADC Clock = 5 MHz			384(1)	kSPS
Throughput Rate	ADC Clock = 8 MHz			533(2)	kSPS

Notes: 1. Corresponds to 13 clock cycles at 5 MHz: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

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 35-13. Analog Inputs

Parameter	Min	Typ	Max	Units
Input Voltage Range	0		V_{DDANA}	
Input Leakage Current		TBD		µA
Input Capacitance		17		pF

Table 35-14. Transfer Characteristics

Parameter	Conditions	Min	Typ	Max	Units
Resolution			10		Bit
Absolute Accuracy	f=5MHz			0.8	LSB
Integral Non-linearity	f=5MHz		0.35	0.5	LSB
Differential Non-linearity	f=5MHz		0.3	0.5	LSB
Offset Error	f=5MHz	-0.5		0.5	LSB
Gain Error	f=5MHz	-0.5		0.5	LSB

35.8 USB Transceiver Characteristics

35.8.1 Electrical Characteristics

Table 35-15. Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level				TBD	V
V_{IH}	High Level		TBD			V
V_{DI}	Differential Input Sensivity	|(D+) - (D-)|	TBD			V
V_{CM}	Differential Input Common Mode Range		TBD		TBD	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line			TBD	pF
I	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	TBD		TBD	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		TBD		Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	TBD		TBD	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	TBD		TBD	V
V_{CRS}	Output Signal Crossover Voltage		TBD		TBD	V

35.8.2 Switching Characteristics

Table 35-16. In Low Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 400 pF	TBD		TBD	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 400 pF	TBD		TBD	ns
t_{FRFM}	Rise/Fall time Matching	C_{LOAD} = 400 pF	TBD		TBD	%

Table 35-17. In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 50 pF	TBD		TBD	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 50 pF	TBD		TBD	ns
t_{FRFM}	Rise/Fall time Matching		TBD		TBD	%

35.9 EBI Timings

These timings are given for worst case process, T = 85·C, VDDCORE = 1.65V, VDDIO = 3V and 40 pF load capacitance.

Table 35-18. SMC Clock Signal.

Symbol	Parameter	Max[1]	Units
$1/(t_{CPSMC})$	SMC Controller Clock Frequency	$1/(t_{cpcpu})$	MHz

Note: 1. The maximum frequency of the SMC interface is the same as the max frequency for the HSB.

Table 35-19. SMC Read Signals with Hold Settings

Symbol	Parameter	Min	Units
NRD Controlled (READ_MODE = 1)			
SMC_1	Data Setup before NRD High	12	ns
SMC_2	Data Hold after NRD High	0	
SMC_3	NRD High to NBS0/A0 Change[1]	nrd hold length * t_{CPSMC} - 1.3	
SMC_4	NRD High to NBS1 Change[1]	nrd hold length * t_{CPSMC} - 1.3	
SMC_5	NRD High to NBS2/A1 Change[1]	nrd hold length * t_{CPSMC} - 1.3	
SMC_7	NRD High to A2 - A23 Change[1]	nrd hold length * t_{CPSMC} - 1.3	
SMC_8	NRD High to NCS Inactive[1]	(nrd hold length - ncs rd hold length) * t_{CPSMC} - 2.3	
SMC_9	NRD Pulse Width	nrd pulse length * t_{CPSMC} - 1.4	
NRD Controlled (READ_MODE = 0)			
SMC_{10}	Data Setup before NCS High	11.5	ns
SMC_{11}	Data Hold after NCS High	0	
SMC_{12}	NCS High to NBS0/A0 Change[1]	ncs rd hold length * t_{CPSMC} - 2.3	
SMC_{13}	NCS High to NBS0/A0 Change[1]	ncs rd hold length * t_{CPSMC} - 2.3	
SMC_{14}	NCS High to NBS2/A1 Change[1]	ncs rd hold length * t_{CPSMC} - 2.3	
SMC_{16}	NCS High to A2 - A23 Change[1]	ncs rd hold length * t_{CPSMC} - 4	
SMC_{17}	NCS High to NRD Inactive[1]	ncs rd hold length - nrd hold length)* t_{CPSMC} - 1.3	
SMC_{18}	NCS Pulse Width	ncs rd pulse length * t_{CPSMC} - 3.6	

Note: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs rd hold length” or “nrd hold length”.

Table 35-20. SMC Read Signals with no Hold Settings

Symbol	Parameter	Min	Units
NRD Controlled (READ_MODE = 1)			
SMC_{19}	Data Setup before NRD High	13.7	ns
SMC_{20}	Data Hold after NRD High	1	
NRD Controlled (READ_MODE = 0)			
SMC_{21}	Data Setup before NCS High	13.3	ns
SMC_{22}	Data Hold after NCS High	0	

Table 35-21. SMC Write Signals with Hold Settings

Symbol	Parameter	Min	Units
NRD Controlled (READ_MODE = 1)			
SMC_{23}	Data Out Valid before NWE High	(nwe pulse length - 1) * t_{CPSMC} - 0.9	ns
SMC_{24}	Data Out Valid after NWE High(1)	nwe hold length * t_{CPSMC} - 6	
SMC_{25}	NWE High to NBS0/A0 Change(1)	nwe hold length * t_{CPSMC} - 1.9	
SMC_{26}	NWE High to NBS1 Change(1)	nwe hold length * t_{CPSMC} - 1.9	
SMC_{29}	NWE High to A1 Change(1)	nwe hold length * t_{CPSMC} - 1.9	
SMC_{31}	NWE High to A2 - A23 Change(1)	nwe hold length * t_{CPSMC} - 1.7	
SMC_{32}	NWE High to NCS Inactive(1)	(nwe hold length - ncs wr hold length)* t_{CPSMC} - 2.9	
SMC_{33}	NWE Pulse Width	nwe pulse length * t_{CPSMC} - 0.9	
NRD Controlled (READ_MODE = 0)			
SMC_{34}	Data Out Valid before NCS High	(ncs wr pulse length - 1)* t_{CPSMC} - 4.6	ns
SMC_{35}	Data Out Valid after NCS High(1)	ncs wr hold length * t_{CPSMC} - 5.8	
SMC_{36}	NCS High to NWE Inactive(1)	(ncs wr hold length - nwe hold length)* t_{CPSMC} - 0.6	

Note: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “nwe hold length"

Table 35-22. SMC Write Signals with No Hold Settings (NWE Controlled only).

Symbol	Parameter	Min	Units
SMC_{37}	NWE Rising to A2-A25 Valid	5.4	ns
SMC_{38}	NWE Rising to NBS0/A0 Valid	5	
SMC_{39}	NWE Rising to NBS1 Change	5	
SMC_{40}	NWE Rising to A1/NBS2 Change	5	
SMC_{41}	NWE Rising to NBS3 Change	5	
SMC_{42}	NWE Rising to NCS Rising	5.1	
SMC_{43}	Data Out Valid before NWE Rising	(nwe pulse length - 1) * t_{CPSMC} - 1.2	
SMC_{44}	Data Out Valid after NWE Rising	5	
SMC_{45}	NWE Pulse Width	nwe pulse length * t_{CPSMC} - 0.9	

Figure 35-2. SMC Signals for NCS Controlled Accesses.

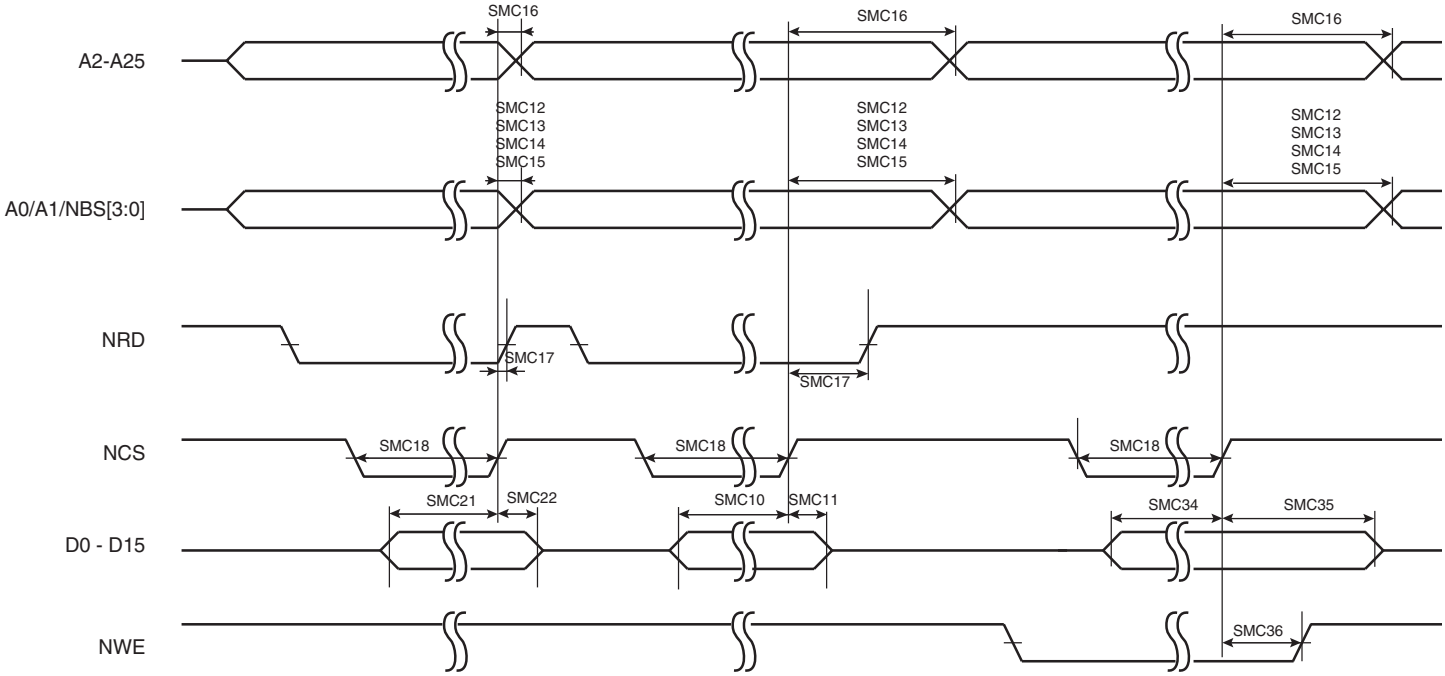
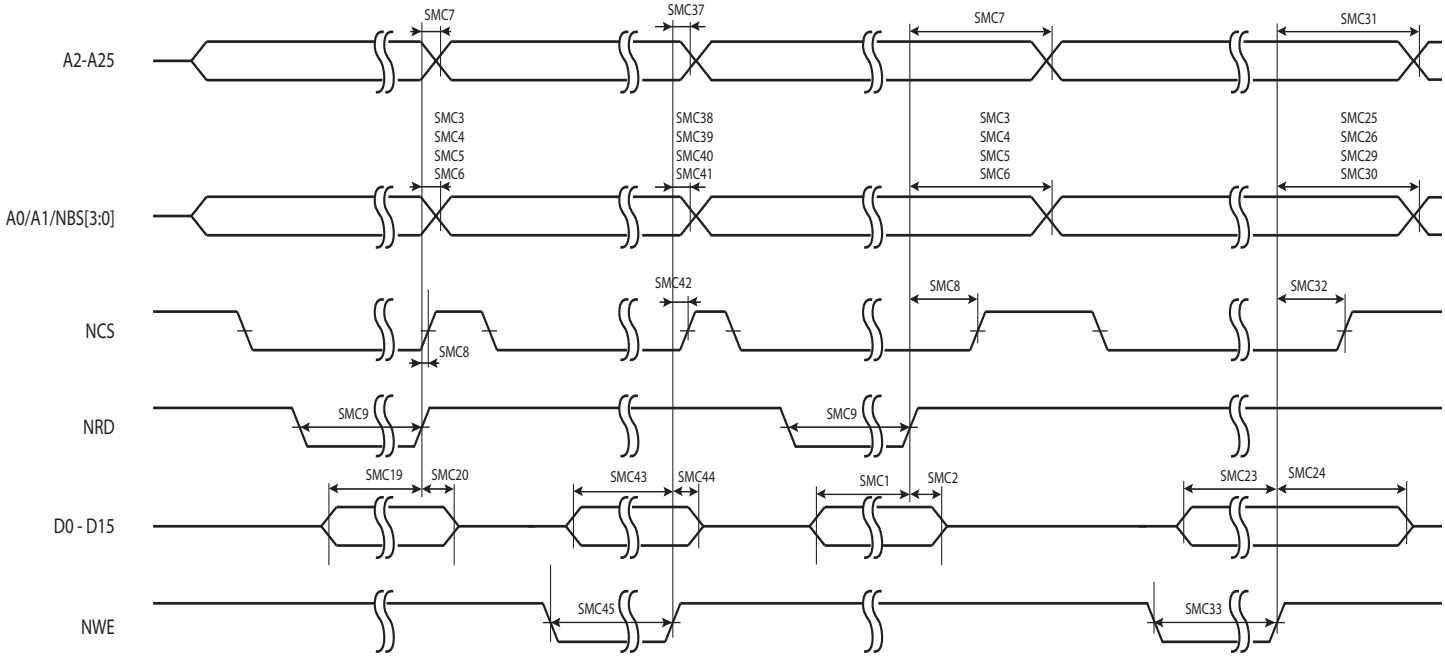


Figure 35-3. SMC Signals for NRD and NRW Controlled Accesses.



35.9.1 SDRAM Signals

These timings are given for 10 pF load on SDCK and 40 pF on other signals.

Table 35-23. SDRAM Clock Signal.

Symbol	Parameter	Max[1]	Units
$1/(t_{CPSDCK})$	SDRAM Controller Clock Frequency	$1/(t_{cpcpu})$	MHz

Note: 1. The maximum frequency of the SDRAMC interface is the same as the max frequency for the HSB.

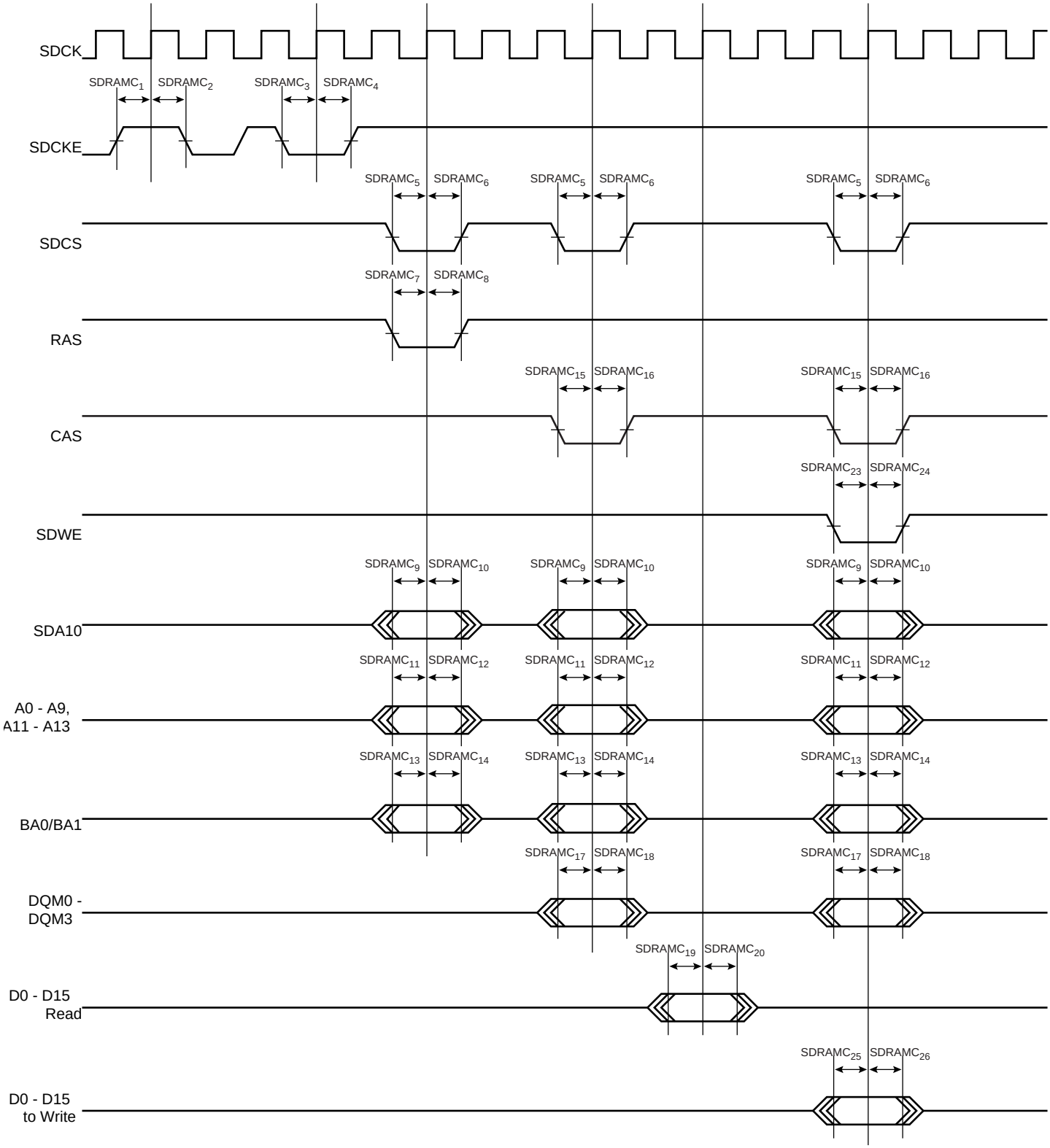
Table 35-24. SDRAM Clock Signal.

Symbol	Parameter	Min	Units
$SDRAMC_1$	SDCKE High before SDCK Rising Edge	7.4	ns
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge	3.2	
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge	7	
$SDRAMC_4$	SDCKE High after SDCK Rising Edge	2.9	
$SDRAMC_5$	SDCS Low before SDCK Rising Edge	7.5	
$SDRAMC_6$	SDCS High after SDCK Rising Edge	1.6	
$SDRAMC_7$	RAS Low before SDCK Rising Edge	7.2	
$SDRAMC_8$	RAS High after SDCK Rising Edge	2.3	
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge	7.6	
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge	1.9	

Table 35-24. SDRAM Clock Signal.

Symbol	Parameter	Min	Units
$SDRAMC_{11}$	Address Change before SDCK Rising Edge	6.2	ns
$SDRAMC_{12}$	Address Change after SDCK Rising Edge	2.2	
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge	6.3	
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge	2.4	
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge	7.4	
$SDRAMC_{16}$	CAS High after SDCK Rising Edge	1.9	
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge	6.4	
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge	2.2	
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge	9	
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge	0	
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge	7.6	
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge	1.8	
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge	7.1	
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge	1.5	

Figure 35-4. SDRAMC Signals relative to SDCK.



35.10 JTAG Timings

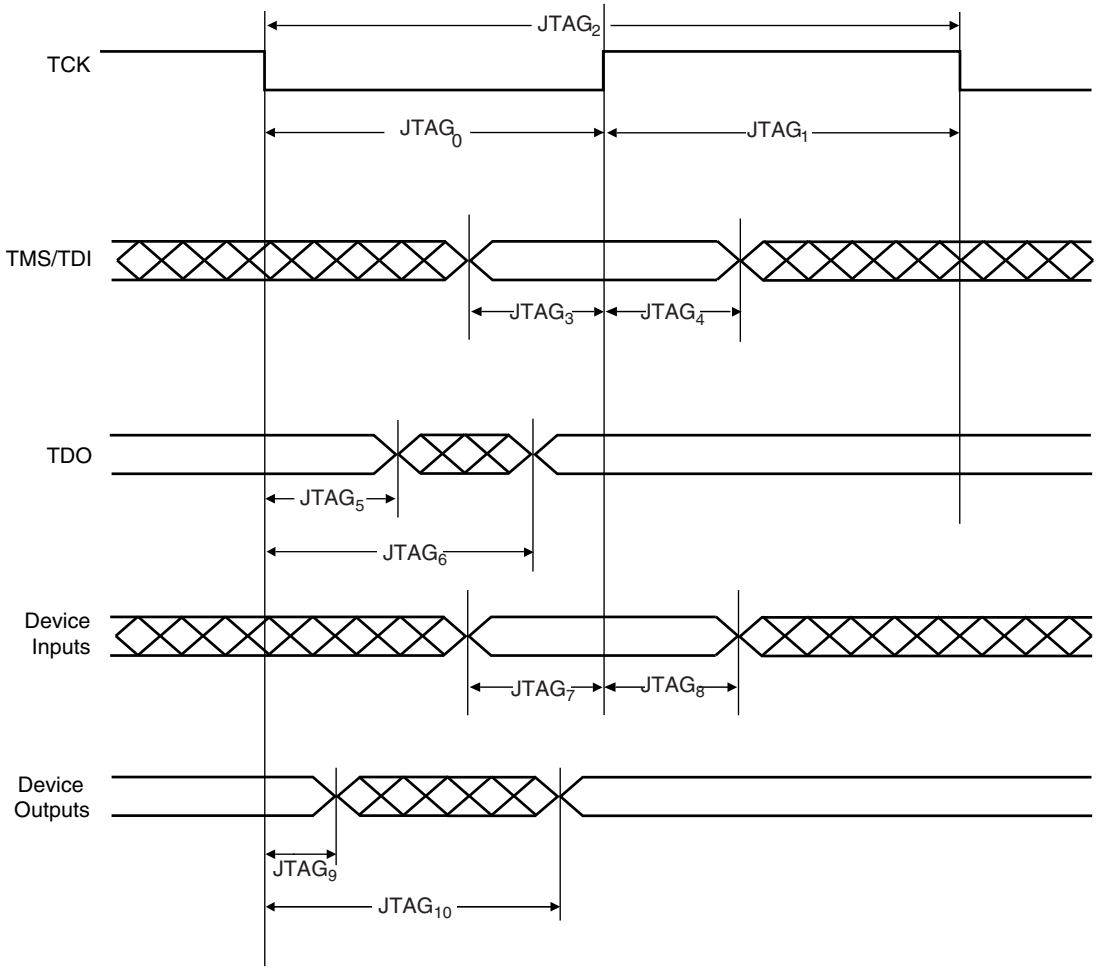
35.10.1 JTAG Interface Signals

Table 35-25. JTAG Interface Timing specification

Symbol	Parameter	Conditions	Min	Max	Units
$JTAG_0$	TCK Low Half-period	(1)	6		ns
$JTAG_1$	TCK High Half-period	(1)	3		ns
$JTAG_2$	TCK Period	(1)	9		ns
$JTAG_3$	TDI, TMS Setup before TCK High	(1)	1		ns
$JTAG_4$	TDI, TMS Hold after TCK High	(1)	0		ns
$JTAG_5$	TDO Hold Time	(1)	4		ns
$JTAG_6$	TCK Low to TDO Valid	(1)		6	ns
$JTAG_7$	Device Inputs Setup Time	(1)			ns
$JTAG_8$	Device Inputs Hold Time	(1)			ns
$JTAG_9$	Device Outputs Hold Time	(1)			ns
$JTAG_{10}$	TCK to Device Outputs Valid	(1)			ns

Note: 1. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF

Figure 35-5. JTAG Interface Signals



35.11 SPI Characteristics

Figure 35-6. SPI Master mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

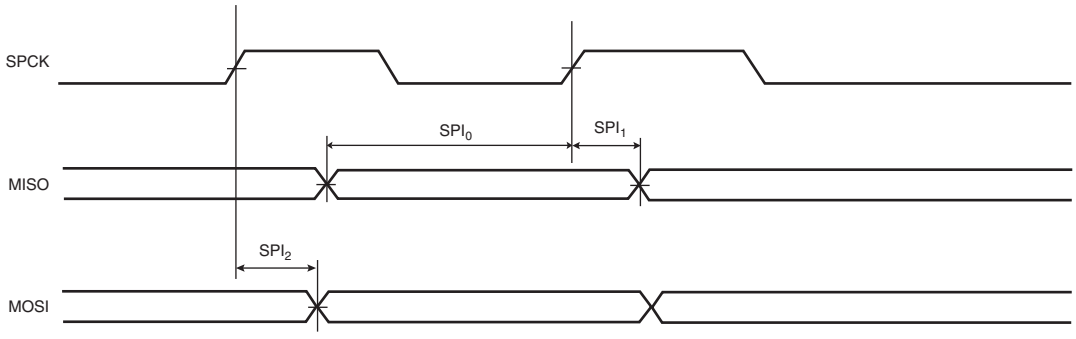


Figure 35-7. SPI Master mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

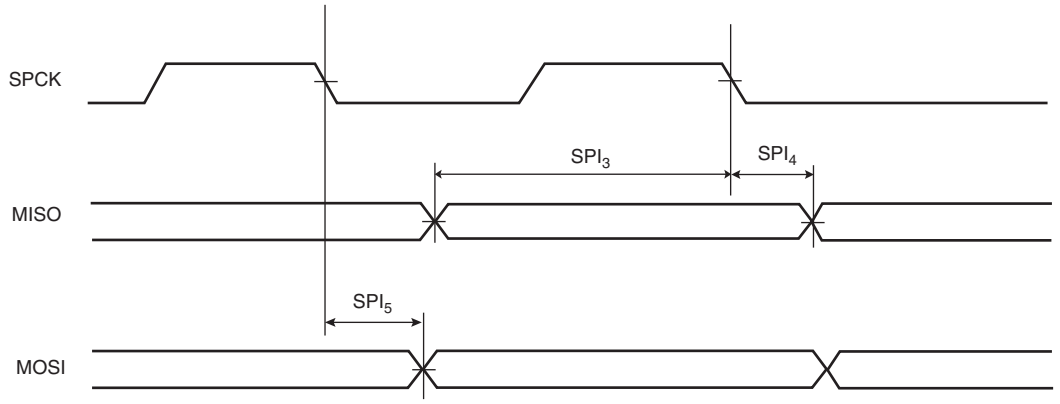


Figure 35-8. SPI Slave mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

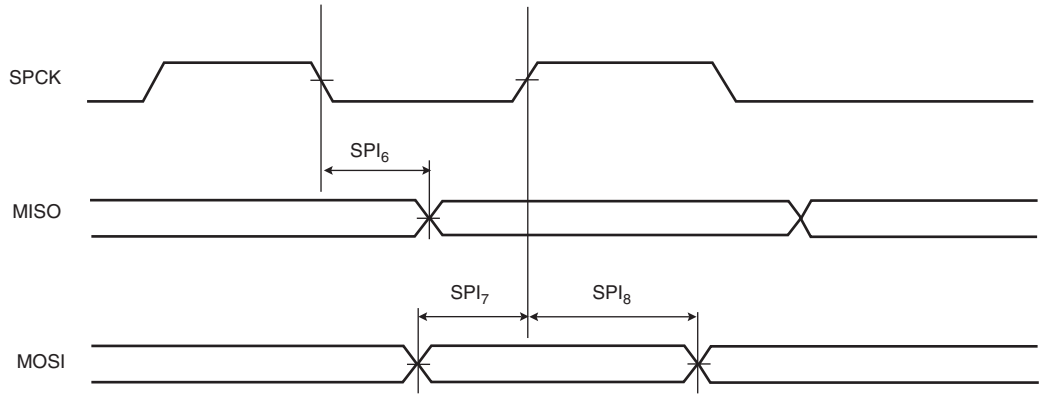


Figure 35-9. SPI Slave mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

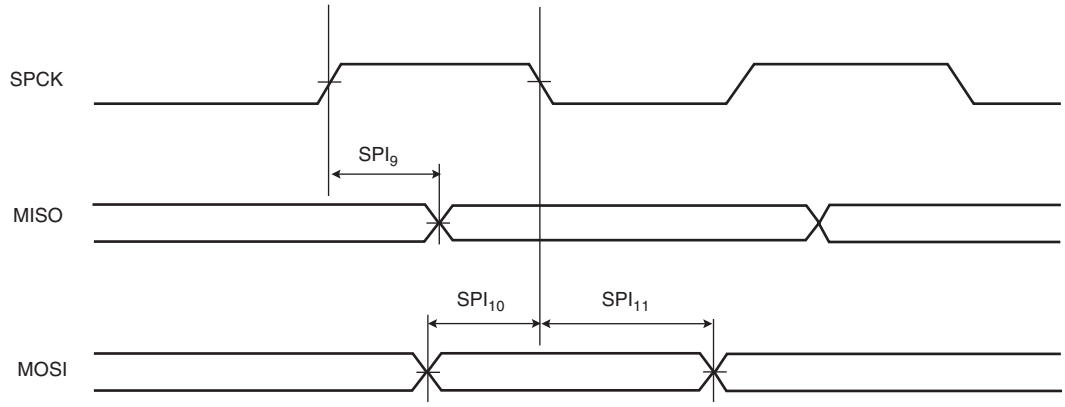


Table 35-26. SPI Timings

Symbol	Parameter	Conditions	Min	Max	Units
SPI_0	MISO Setup time before SPCK rises (master)	3.3V domain(1)	22 + (t_{CPMCK})/2(2)		ns
SPI_1	MISO Hold time after SPCK rises (master)	3.3V domain(1)	0		ns
SPI_2	SPCK rising to MOSI Delay (master)	3.3V domain(1)		7	ns
SPI_3	MISO Setup time before SPCK falls (master)	3.3V domain(1)	22 + (t_{CPMCK})/2(2)		ns
SPI_4	MISO Hold time after SPCK falls (master)	3.3V domain (1)	0		ns
SPI_5	SPCK falling to MOSI Delay (master)	3.3V domain (1)		7	ns
SPI_6	SPCK falling to MISO Delay (slave)	3.3V domain (1)		26.5	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	3.3V domain (1)	0		ns
SPI_8	MOSI Hold time after SPCK rises (slave)	3.3V domain (1)	1.5		ns
SPI_9	SPCK rising to MISO Delay (slave)	3.3V domain (1)		27	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	3.3V domain (1)	0		ns
SPI_{11}	MOSI Hold time after SPCK falls (slave)	3.3V domain (1)	1		ns

Notes: 1. 3.3V domain: V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40 pF.
2. t_{CPMCK}: Master Clock period in ns.

35.12 MACB Characteristics

Table 35-27. Ethernet MAC Signals

Symbol	Parameter	Conditions	Min (ns)	Max (ns)
$EMAC_1$	Setup for EMDIO from EMDC rising	Load: 20pF(2)		
$EMAC_2$	Hold for EMDIO from EMDC rising	Load: 20pF(2)		
$EMAC_3$	EMDIO toggling from EMDC falling	Load: 20pF(2)		

Notes: 1. f: MCK frequency (MHz)
2. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 20 pF

Table 35-28. Ethernet MAC MII Specific Signals

Symbol	Parameter	Conditions	Min (ns)	Max (ns)
$EMAC_4$	Setup for ECOL from ETXCK rising	Load: 20pF (1)	3	
$EMAC_5$	Hold for ECOL from ETXCK rising	Load: 20pF (1)	0	
$EMAC_6$	Setup for ECRS from ETXCK rising	Load: 20pF (1)	3	
$EMAC_7$	Hold for ECRS from ETXCK rising	Load: 20pF (1)	0	
$EMAC_8$	ETXER toggling from ETXCK rising	Load: 20pF (1)		15
$EMAC_9$	ETXEN toggling from ETXCK rising	Load: 20pF (1)		15
$EMAC_{10}$	ETX toggling from ETXCK rising	Load: 20pF (1)		15
$EMAC_{11}$	Setup for ERX from ERXCK	Load: 20pF (1)	1	

Table 35-28. Ethernet MAC MII Specific Signals

Symbol	Parameter	Conditions	Min (ns)	Max (ns)
$EMAC_{12}$	Hold for ERX from ERXCK	Load: 20pF (1)	1.5	
$EMAC_{13}$	Setup for ERXER from ERXCK	Load: 20pF (1)	1	
$EMAC_{14}$	Hold for ERXER from ERXCK	Load: 20pF (1)	0.5	
$EMAC_{15}$	Setup for ERXDV from ERXCK	Load: 20pF (1)	1.5	
$EMAC_{16}$	Hold for ERXDV from ERXCK	Load: 20pF (1)	1	

Note: 1. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 20 pF

Figure 35-10. Ethernet MAC MII Mode

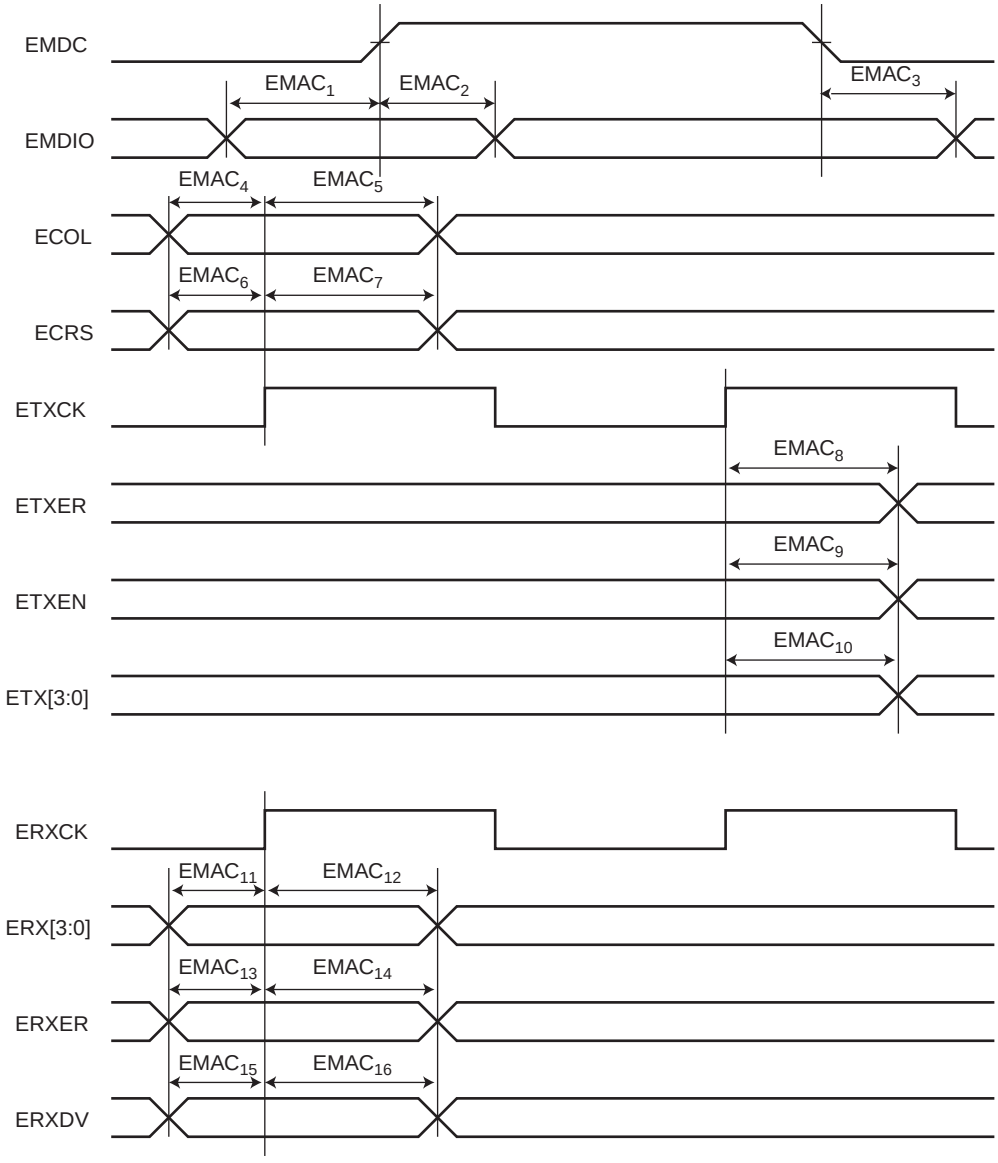
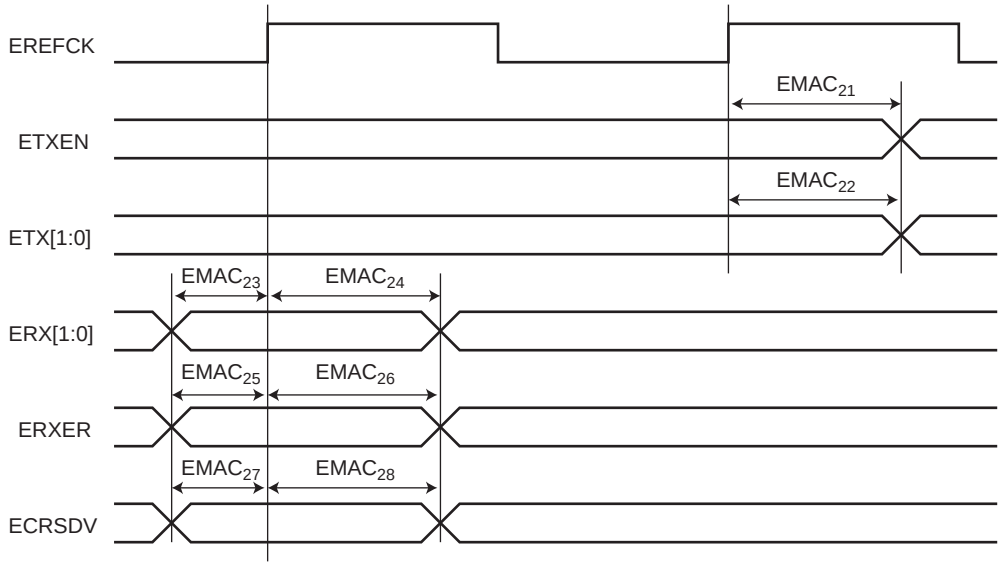


Table 35-29. Ethernet MAC RMII Specific Signals

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_{21}$	ETXEN toggling from EREFCK rising	7	14.5
$EMAC_{22}$	ETX toggling from EREFCK rising	7	14.7
$EMAC_{23}$	Setup for ERX from EREFCK	1.5	
$EMAC_{24}$	Hold for ERX from EREFCK	0	
$EMAC_{25}$	Setup for ERXER from EREFCK	1.5	
$EMAC_{26}$	Hold for ERXER from EREFCK	0	
$EMAC_{27}$	Setup for ECRSDV from EREFCK	1.5	
$EMAC_{28}$	Hold for ECRSDV from EREFCK	0	

Figure 35-11. Ethernet MAC RMII Mode



35.13 Flash Characteristics

The following table gives the device maximum operating frequency depending on the field FWS of the Flash FSR register. This field defines the number of wait states required to access the Flash Memory.

Table 35-30. Flash Wait States

FWS	Read Operations	Maximum Operating Frequency (MHz)
0	1 cycle	36
1	2 cycles	66

36. Mechanical Characteristics

36.1 Thermal Considerations

36.1.1 Thermal Data

Table 36-1 summarizes the thermal resistance data depending on the package.

Table 36-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TQFP144	TBD	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP144	TBD	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TBGA144	TBD	°C/W
θ_{JC}	Junction-to-case thermal resistance		TBGA144	TBD	

36.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

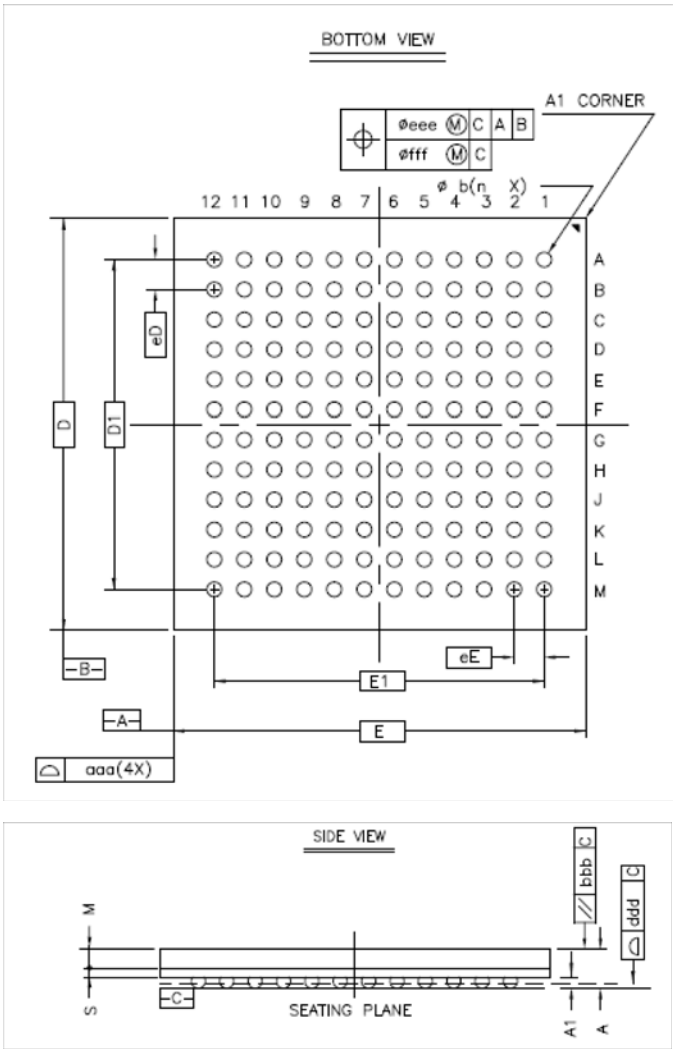
where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 36-1 on page 944.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 36-1 on page 944.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section "Regulator characteristics" on page 925.
- T_A = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T_J in °C.

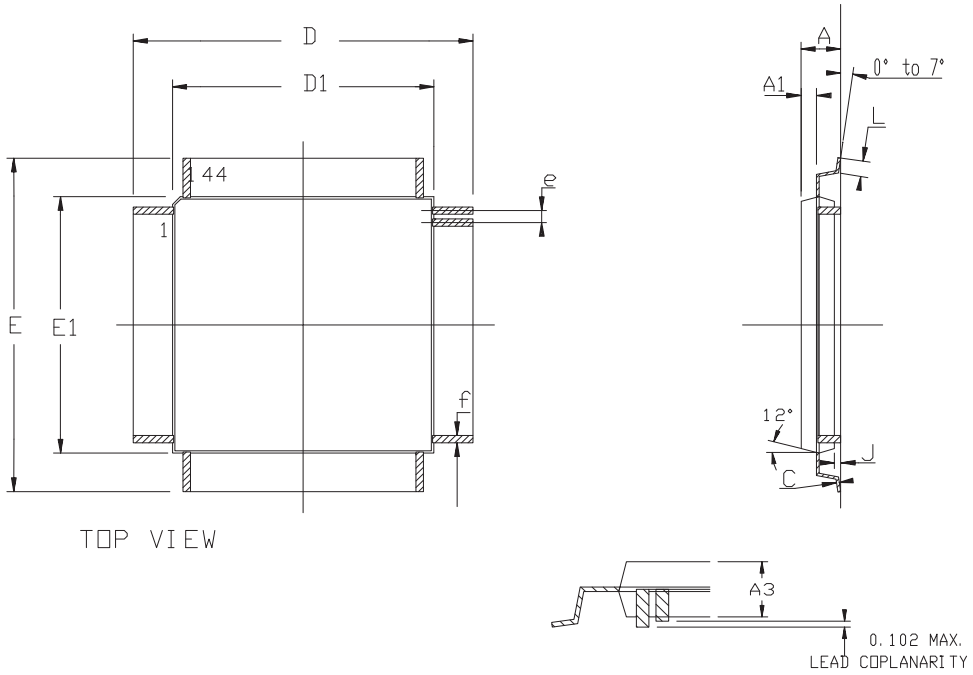
36.2 Package Drawings

Figure 36-1. TBGA 144 package drawing



		Symbol	Common Dimensions
Package :			TFBGA
Body Size:	X	E	11
	Y	D	11
Ball Pitch :	X	eE	0.800
	Y	eD	0.800
Total Thickness :		A	1.200 Max.
Mold Thickness :		M	0.530 Ref.
Substrate Thickness :		S	0.260 Ref.
Ball Diameter :			0.400
Stand Off :		A1	0.270 ~ 0.370
Ball Width :		b	0.380 ~ 0.480
Package Edge Tolerance :		aaa	0.150
Mold Flatness :		bbb	0.200
Coplanarity:		ddd	0.120
Ball Offset (Package) :		eee	0.150
Ball Offset (Ball) :		fff	0.080
Ball Count :		n	144
Edge Ball Center to Center :	X	E1	8.800
	Y	D1	8.800

Figure 36-2. LQFP-144 package drawing



	MM		INCH	
	Min	Max	Min	Max
A	-	1.60	-	.063
C	0.09	0.20	.004	.008
A3	1.35	1.45	.053	.057
D	21.90	22.10	.862	.870
D1	19.90	20.10	.783	.791
E	21.90	22.10	.862	.870
E1	19.90	20.10	.783	.791
J	0.05	0.15	.002	.006
L	0.45	0.75	.018	.030
e	0.50 BSC		.0197 BSC	
f	0.22 BSC		.009 BSC	

Table 36-2. Device and Package Maximum Weight

TBD	mg

Table 36-3. Package Characteristics

Moisture Sensitivity Level	TBD

Table 36-4. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Table 36-5. Device and Package Maximum Weight

TBD	mg

36.3 Soldering Profile

Table 36-6 gives the recommended soldering profile from J-STD-20.

Table 36-6. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	TBD
Preheat Temperature 175°C ±25°C	TBD
Temperature Maintained Above 217°C	TBD
Time within 5°C of Actual Peak Temperature	TBD
Peak Temperature Range	TBD
Ramp-down Rate	TBD
Time 25°C to Peak Temperature	TBD

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

37. Ordering Information

Device	Ordering Code	Package	Conditioning	Temperature Operating Range
AT32UC3A3256S	**AT32UC3A3256S-ALUT**	**144 lead LQFP**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3256S-ALUR	**144 lead LQFP**	Reels	Industrial (-40·C to 85·C)
	AT32UC3A3256S-CTUT	**144 balls TBGA**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3256S-CTUR	**144 balls TBGA**	Reels	Industrial (-40·C to 85·C)
AT32UC3A3256	**AT32UC3A3256-ALUT**	**144 lead LQFP**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3256-ALUR	**144 lead LQFP**	Reels	Industrial (-40·C to 85·C)
	AT32UC3A3256-CTUT	**144 balls TBGA**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3256-CTUR	**144 balls TBGA**	Reels	Industrial (-40·C to 85·C)
AT32UC3A3128S	**AT32UC3A3128S-ALUT**	**144 lead LQFP**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3128S-ALUR	**144 lead LQFP**	Reels	Industrial (-40·C to 85·C)
	AT32UC3A3128S-CTUT	**144 balls TBGA**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3128S-CTUR	**144 balls TBGA**	Reels	Industrial (-40·C to 85·C)
AT32UC3A3128	**AT32UC3A3128-ALUT**	**144 lead LQFP**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3128-ALUR	**144 lead LQFP**	Reels	Industrial (-40·C to 85·C)
	AT32UC3A3128-CTUT	**144 balls TBGA**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A3128-CTUR	**144 balls TBGA**	Reels	Industrial (-40·C to 85·C)
AT32UC3A364S	**AT32UC3A364S-ALUT**	**144 lead LQFP**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A364S-ALUR	**144 lead LQFP**	Reels	Industrial (-40·C to 85·C)
	AT32UC3A364S-CTUT	**144 balls TBGA**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A364S-CTUR	**144 balls TBGA**	Reels	Industrial (-40·C to 85·C)
AT32UC3A364	**AT32UC3A364-ALUT**	**144 lead LQFP**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A364-ALUR	**144 lead LQFP**	Reels	Industrial (-40·C to 85·C)
	AT32UC3A364-CTUT	**144 balls TBGA**	Tray	Industrial (-40·C to 85·C)
	AT32UC3A364-CTUR	**144 balls TBGA**	Reels	Industrial (-40·C to 85·C)

38. Errata

38.1 Rev. E

38.1.1 Processor and Architecture

1. **LDM instruction with PC in the register list and without ++ increments Rp**
   For LDM with PC in the register list: the instruction behaves as if the ++ field is always set, ie the pointer is always updated. This happens even if the ++ field is cleared. Specifically, the increment of the pointer is done in parallel with the testing of R12.
   Fix/Workaround
   None.

38.1.2 ADC

1. **Sleep Mode activation needs additional A to D conversion**
   If the ADC sleep mode is activated when the ADC is idle the ADC will not enter sleep mode before after the next AD conversion.

   Fix/Workaround
   Activate the sleep mode in the mode register and then perform an AD conversion.

38.1.3 SPI

1. **SPI Bad Serial Clock Generation on 2nd chip_select when SCBR = 1, CPOL=1 and NCPHA=0**
   When multiple CS are in use, if one of the baudrate equals to 1 and one of the others doesn't equal to 1, and CPOL=1 and CPHA=0, then an additional pulse will be generated on SCK.
   Fix/workaround
   When multiple CS are in use, if one of the baudrate equals 1, the other must also equal 1 if CPOL=1 and CPHA=0.

2. **SPI Disable does not work in Slave mode**
   Fix/workaround
   Read the last received data then perform a Software reset.

38.2 Rev. D

38.2.1 Processor and Architecture

1. **LDM instruction with PC in the register list and without ++ increments Rp**
   For LDM with PC in the register list: the instruction behaves as if the ++ field is always set, ie the pointer is always updated. This happens even if the ++ field is cleared. Specifically, the increment of the pointer is done in parallel with the testing of R12.
   Fix/Workaround
   None.

2. **RETE instruction does not clear SREG[L] from interrupts.**
   The RETE instruction clears SREG[L] as expected from exceptions.
   Fix/Workaround

When using the STCOND instruction, clear SREG[L] in the stacked value of SR before returning from interrupts with RETE.

3. **Exceptions when system stack is protected by MPU**
RETS behaves incorrectly when MPU is enabled and MPU is configured so that system stack is not readable in unprivileged mode.
Fix/Workaround
Workaround 1: Make system stack readable in unprivileged mode,
or
Workaround 2: Return from supervisor mode using rete instead of rets. This requires :
1. Changing the mode bits from 001b to 110b before issuing the instruction.
Updating the mode bits to the desired value must be done using a single mtsr instruction so it is done atomically. Even if this step is described in general as not safe in the UC technical reference guide, it is safe in this very specific case.
2. Execute the RETE instruction.

4. **Multiply instructions do not work on RevD.**
All the multiply instructions do not work.
Fix/Workaround
Do not use the multiply instructions.

38.2.2 ADC

1. **Sleep Mode activation needs additional A to D conversion**
If the ADC sleep mode is activated when the ADC is idle the ADC will not enter sleep mode before after the next AD conversion.

Fix/Workaround
Activate the sleep mode in the mode register and then perform an AD conversion.

38.2.3 SPI

1. **SPI Bad Serial Clock Generation on 2nd chip_select when SCBR = 1, CPOL=1 and NCPHA=0**
When multiple CS are in use, if one of the baudrate equals to 1 and one of the others doesn't equal to 1, and CPOL=1 and CPHA=0, then an additional pulse will be generated on SCK.
Fix/workaround
When multiple CS are in use, if one of the baudrate equals 1, the other must also equal 1 if CPOL=1 and CPHA=0.

2. **SPI Disable does not work in Slave mode**
Fix/workaround
Read the last received data then perform a Software reset.

38.2.4 TWI

1. **TWIM Version Register is zero**
TWIM Version Register (VR) is zero instead of 0x100.
Fix/Workaround
None.

39. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

39.1 Rev. A – 03/09

1. Initial revision.

Table of Contents

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature