

Mobile multimedia application processor

Features

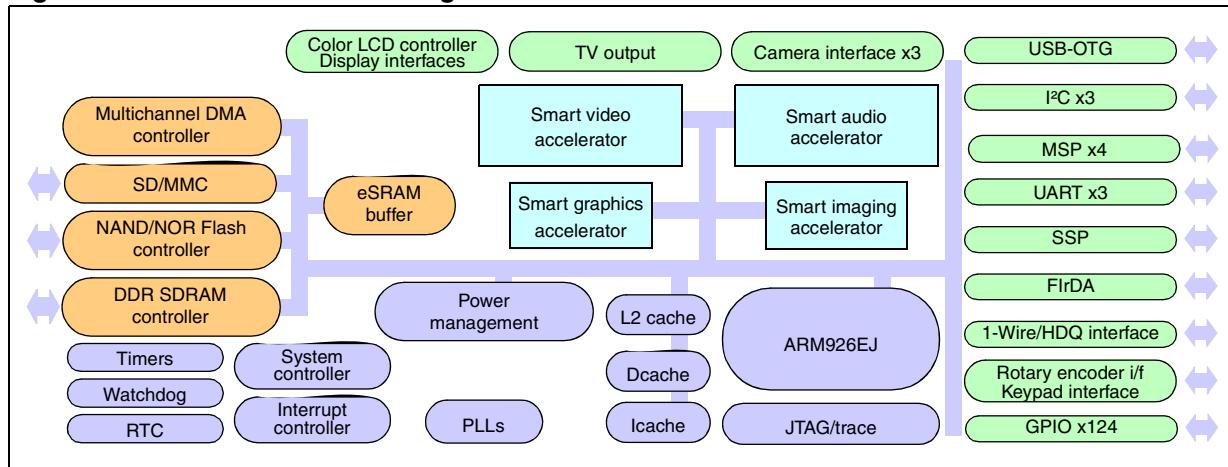
- Smart video accelerator
 - MPEG-4 Simple Profile L5 encoding/decoding up to SDTV 30 fps
 - DVB-H class C H264 BP L2 2Mbps CIF 30fps
 - DVB-H class B VC1 SP 384kbps QVGA 20fps
 - H.264 decode resolution up to VGA bit-rate up to 4Mbps
 - H263 codec profile 0 level 30 384kbps CIF 30fps
 - JPEG encode up to 40 Mpixel/s, decode up to 10 Mpixel/s
- Smart audio accelerator with extensive digital-audio software library
- Smart imaging accelerator
 - 5 Mpixel camera support
 - 2 SMIA CCP2 camera interfaces and parallel camera CCIR-656 interface up to 80 MHz
 - Real-time image reconstruction up to 80 Mpixel/s and 10-bit raw Bayer interface
- Smart graphics accelerator

- Ultra low-power implementation
- TV output
- Advanced power management unit
- ARM926EJ® 32-bit RISC CPU, up to 332 MHz
 - 16-Kbyte instruction cache, 16-Kbyte data cache, 128-Kbyte level 2 cache
- On-chip SRAM: 512 Kbytes
- On-chip ROM: 32 Kbytes for boot
- 16-bit DDR/SDR-SDRAM memory controller (up to 166 MHz)
- NOR Flash/NAND Flash/CompactFlash/CF+ controller
- TFBGA 12mm x 12mm x 1.2mm package

Description

The STn8815 application processor enables smart phones, mobile multimedia, internet appliances and in-car entertainment systems to playback media content, record video clips and pictures, receive mobile-TV and perform real-time bidirectional audio-visual communication.

Figure 1. STn8815A12 block diagram



Contents

1	STn8815 overview	17
1.1	Key benefits	20
1.2	Main features	20
1.3	Low power consumption	21
2	Architecture overview	22
2.1	Smart video accelerator (SVA)	23
2.2	Smart audio accelerator (SAA)	23
2.3	Smart imaging accelerator (SIA)	24
2.4	Advanced power management unit (PMU)	24
2.5	ARM926EJ processor	24
2.6	Embedded memory units	25
2.7	Flexible static memory controller (FSMC)	25
2.8	SDRAM memory controller (SDMC)	25
2.9	Real time clock (RTC)	25
2.10	Timers	25
2.11	Watchdog timer	25
2.12	Vectored interrupt controller (VIC)	25
2.13	System and reset controller (SRC)	26
2.14	Direct memory access (DMA) controllers	26
2.15	Universal asynchronous receivers-transmitters (UARTs)	26
2.16	Synchronous serial port (SSP)	26
2.17	Camera interfaces	26
2.18	TV interface	27
2.19	Liquid crystal display controller (LCDC)	27
2.20	Master display interface (MDIF)	27
2.21	Pulse width light modulator (PWL)	27
2.22	General purpose inputs/outputs (GPIOs)	27
2.23	Memory card interface (MMC/SD)	28
2.24	USB-OTG high-speed interface	28
2.25	I ² C bus interface	28

2.26	Fast IrDA interface (FIrDA)	29
2.27	Multichannel serial ports (MSP)	29
2.28	Scroll key and keypad encoder (SKE) interface	29
3	Pin information	30
3.1	Signal definitions and assignments	32
3.2	Pin characteristics	59
3.3	GPIO alternate functions	69
4	Clocks	80
4.1	Input clocks	81
4.2	Output clocks	81
4.2.1	PLL1 System clocks	81
4.2.2	PLL2 Peripheral clocks	81
5	System and reset controller (SRC)	83
5.1	System mode control	83
5.2	System mode control state machine	84
5.3	Crystal oscillator and PLL control	85
5.4	PLL1 frequency control	86
5.4.1	Restrictions on changing the PLL1 frequency	86
5.4.2	PLL unlocking	86
5.4.3	Restrictions on the PLL1 output frequency	86
5.5	PLL2 frequency control	89
5.6	PLL1 and PLL2 interrupts	89
5.7	Interrupt response mode	90
5.8	Wait-for-interrupt control	90
5.9	HCLK to CLK relationship	90
5.10	SDRAM refresh rates	90
5.11	Watchdog and timer clock enable signals	91
5.12	CLKOUT0/1 programmable clock signals	92
5.13	Clocks and power supplies	93
5.14	Clocks	94
5.15	Power supplies	95
5.16	Reset controller	96

5.16.1	Power-on reset	96
5.16.2	Software reset	96
5.16.3	Watchdog reset	96
6	Power management unit (PMU)	98
6.1	PMU features	98
6.2	Power-supply domains	99
6.3	Power-state switching	100
6.3.1	Sleep / deep-sleep mode entry	101
6.3.2	Sleep / deep-sleep mode	101
6.3.3	Deep-sleep mode exit	101
6.3.4	Sleep mode exit	102
6.3.5	Reviving DRAMs from self-refresh mode	102
6.4	MSP bypass	103
6.5	Tempo feature	104
6.6	Bypassing PLL2 and subsequent dividers	105
6.7	PMU signals	105
7	ARM926EJ processor	106
7.1	Features	106
7.2	Architecture	107
7.3	Addresses	107
7.4	Instruction cache	107
7.4.1	ICache features	108
7.4.2	ICache operation	108
7.4.3	Caching behavior	108
7.4.4	ICache validity	109
7.5	Data cache	109
7.5.1	DCache features	109
7.5.2	DCache operation	109
7.5.3	Caching and buffering behavior	110
7.5.4	Validity	110
7.6	Write buffer	110
7.7	Memory management unit (MMU)	111
7.7.1	MMU features	111
7.7.2	Access permissions and domains	112

7.7.3	Translated entries	112
7.7.4	Address translation	112
7.8	MMU faults and CPU aborts	120
7.8.1	Fault status and fault address	120
7.8.2	Domain access control	122
7.8.3	Fault checking sequence	122
7.8.4	External aborts	124
7.8.5	TLB structure	124
7.9	System control processor (CP15)	125
7.10	Embedded trace module (ETM)	125
7.10.1	ETM features	125
7.10.2	ETM interface	125
7.10.3	Additional reference material	127
8	Level 2 Cache Controller (L2CC)	128
8.1	L2CC features	128
8.1.1	L2CC buffers	128
8.1.2	L2CC replacement strategy	130
8.1.3	Uses of lockdown format C	130
8.1.4	External abort support for System (L3) memory	131
8.2	Using L2CC with ARM926EJ cores	132
8.3	L2CC Event Monitor (L2EM)	133
9	Hardware semaphore unit (HSEM)	134
10	LCD controller (LCDC)	135
10.1	Features	135
10.2	Overview	136
10.2.1	LCD panel resolution	136
10.2.2	Types of LCD panel supported	136
10.2.3	Number of colors supported	136
10.2.4	Requirements for different display types	138
10.2.5	HR-TFT signals	139
10.2.6	Interrupts	139
10.3	Functional description	140
10.3.1	Data transfer	140

10.3.2	LCD power-up and power-down sequence	141
10.3.3	Dual DMA FIFOs and associated control logic	141
10.3.4	Pixel serializer	141
10.3.5	Palette RAM	145
10.3.6	Gray scaler	145
10.3.7	True color support for 12-, 15-, 16- and 18-bit panels	147
10.3.8	Color enhancement from 16-bpp to 18-bpp (for 18-bit TFT panels)	150
10.3.9	Upper and lower panel formatters	150
10.3.10	Panel clock generator	150
10.3.11	Timing controller	150
10.3.12	Vertical compare event	150
10.4	Data bus usage for STN panels	151
10.5	Data bus usage for TFT panels	152
10.5.1	Data bus usage for different graphics modes	152
10.5.2	TFT panel connections for different data bus widths	155
10.6	Timing waveforms	159
10.6.1	STN timings	160
10.6.2	Horizontal timing restrictions for STN panels	160
10.6.3	TFT timings	161
10.6.4	HR-TFT timings	162
10.7	LCDC signals	164
11	Multi-timer units (MTUs)	165
11.1	Overview	165
11.2	Interrupts	165
11.3	Functional description	165
11.3.1	Timer clock	166
11.3.2	Timer modes	167
11.3.3	FRC	167
11.4	MTU signals	167
12	Watchdog timer (WDT)	169
12.1	Overview	169
12.2	Functional description	169
12.3	WDT signals	170

13	Real-time clock (RTC)	171
13.1	Functional description	171
13.2	RTC signals	172
13.2.1	RTC interrupt	172
13.2.2	RTC wake-up signal	172
13.3	Oscillator frequency calibration	172
13.3.1	Trimming calculations	172
13.3.2	Examples	173
14	Real-time timer (RTT)	175
14.1	Features	175
14.2	Functional description	175
14.2.1	RTT modes	175
14.2.2	RTT signals	176
15	SDRAM memory controller (SDMC)	177
15.1	Features	177
15.2	Functional description	177
15.2.1	AHB slave memory interfaces	177
15.2.2	Low power operation	179
15.2.3	Memory chip-selects	179
15.3	Supported memory devices	180
15.4	Address mapping	180
15.5	Commands	201
15.6	SDMC signals	202
16	Flexible static memory controller (FSMC)	203
16.1	Features	203
16.2	Overview	203
16.3	SRAM/NOR Flash controller	204
16.3.1	Supported memory types	204
16.3.2	SRAM/NOR Flash controller timing parameters	205
16.3.3	Wait management by the SRAM/NOR Flash controller	209
16.3.4	Burst wrap support	211
16.3.5	Bus turn around	212

16.4	PC card/CompactFlash/NAND Flash controller	213
16.4.1	NAND Flash controller connection to PC card	215
16.4.2	NAND Flash controller timing parameters	215
16.4.3	Wait management by NAND Flash controller	217
16.4.4	NAND Flash support	218
16.5	FSMC address mapping	222
16.6	FSMC signals	223
17	DMA controllers (DMAC)	225
17.1	Features	225
17.2	Overview	226
17.3	Functional description	227
17.3.1	AHB master interfaces	227
17.3.2	DMA request priority	229
17.3.3	Channel hardware	229
17.3.4	System considerations	229
17.4	DMA request and response connectivity	229
17.5	Peripheral request lines	230
18	General-purpose inputs/outputs (GPIOs)	233
18.1	Features	233
18.2	Functional description	234
18.3	GPIO states after reset	235
18.3.1	Operation of input/output pins	235
18.3.2	Wake-up	235
18.3.3	Alternate function control	235
18.4	GPIO signals	235
19	Asynchronous serial ports	236
19.1	Features	236
19.2	Overview	236
19.3	Functional description	237
19.3.1	Data transmission	237
19.3.2	Data reception	237
19.3.3	Transmit and receive FIFOs	238
19.3.4	Baud rate divisor	238

19.3.5	System and diagnostic loopback testing	240
19.3.6	Hardware flow control	241
19.3.7	Software flow control	242
19.3.8	Automatic format extractor (autobaud)	243
19.3.9	DMA interface	245
19.4	UART signals	247
19.4.1	Interface signals	247
19.4.2	Interrupt signals	247
20	Scroll key and keypad encoder (SKE)	249
20.1	Features	249
20.2	Overview	249
20.2.1	Scroll key debounce interval	250
20.2.2	Scroll key operation	250
20.3	Keypad encoder	251
20.3.1	Keypad debounce interval	251
20.3.2	Keypad manual scan	252
20.3.3	Keypad automatic scan	253
20.4	SKE signals	253
20.4.1	Interface signals	253
20.4.2	Interrupts	253
21	USB On-The-Go interface (USB-OTG)	254
21.1	Introduction	254
21.2	USB-OTG features	255
21.3	Architecture	256
21.3.1	CPU and DMA	256
21.3.2	OTG transceiver	257
21.4	Functional description	257
21.4.1	OTG controller modes	257
21.4.2	Clocks and resets	258
21.4.3	OTG transceiver interface	258
21.4.4	AHB bridge	258
21.4.5	OTG controller system	259
21.4.6	UTM synchronization block	259
21.4.7	Packet encoding and decoding	259

21.4.8	Endpoint controller	259
21.4.9	CPU Interface	260
21.4.10	RAM controller	260
21.4.11	Bit/byte ordering block	260
21.5	OTG transceiver interfaces	260
21.6	Operation in peripheral mode	261
21.6.1	USB reset	261
21.6.2	Peripheral mode IN transactions	261
21.6.3	Peripheral mode OUT transactions	263
21.6.4	Additional actions	266
21.7	Operation in host mode	267
21.7.1	Host mode device set-up	267
21.7.2	Host mode IN transactions	267
21.7.3	Host mode OUT transactions	268
21.7.4	Host mode transaction scheduling	268
21.7.5	Host mode reset	269
21.7.6	Host mode suspend	269
21.8	OTG session request (SRP)	269
21.8.1	SRP starting a session	270
21.8.2	SRP detecting activity	270
21.8.3	Host negotiation (HNP)	271
21.8.4	DMA channels	271
21.8.5	Dynamic FIFO sizing	272
21.8.6	USB-OTG signals	273
21.8.7	GPIO mapping	273
22	Pulse width light modulator (PWL)	274
23	Synchronous serial port (SSP)	275
23.1	Features	275
23.2	Overview	275
23.3	Functional description	276
23.3.1	Supported frame formats	276
23.3.2	SPI clock configuration	287
23.3.3	SSP bit rate generation	287
23.3.4	SSP data endianness	288

23.4	DMA interface	288
23.4.1	Transmit DMA signals	288
23.4.2	Receive DMA signals	288
23.5	SSP signals	290
23.5.1	Interrupt signals	290
23.5.2	Interface signals	291
24	Vectored interrupt controller (VIC)	292
24.1	Features	292
24.2	Functional description	292
24.2.1	Fast interrupt request (FIQ)	293
24.2.2	Interrupt request (IRQ)	293
25	Multichannel serial ports (MSPx)	294
25.1	Features	294
25.2	Overview	295
25.3	Functional description	296
25.3.1	Transmitting data	296
25.3.2	Receiving data	297
25.3.3	Transmitter/receiver resets	297
25.3.4	Sample rate generator	298
25.4	Clock operation	300
25.4.1	Transmit clock selection	300
25.4.2	Receive clock selection	300
25.5	Frame operation	301
25.5.1	Frame and element length	301
25.5.2	Frame synchronization signals	301
25.5.3	Frame synchronization signal generation	301
25.5.4	Frame synchronization signal polarity	302
25.5.5	Frame phase (single or dual)	302
25.5.6	Data delay	303
25.5.7	Maximum frame frequency	305
25.5.8	Unexpected frame synchronization ignore	305
25.5.9	Data packing	306
25.5.10	Data swapping	306
25.5.11	Loopback mode	306

25.5.12	Data companding	306
25.5.13	Using companding hardware as a co-processor	307
25.5.14	Data endianness	308
25.5.15	Multichannel operations	308
25.5.16	SPI clock modes for SPI protocols	308
25.5.17	MSP error detection	312
25.6	DMA interface	314
25.7	MSP configuration examples	315
25.8	MSP signals	318
25.8.1	Interrupt signals	318
25.8.2	Interface signals	321
26	Fast IrDA controller (FIrDA)	322
26.1	Features	322
26.2	Hardware overview	323
26.2.1	Wrapper unit	323
26.2.2	Modulation unit	323
26.2.3	Synchronization unit	323
26.2.4	Demodulation unit	323
26.2.5	Baud rate generation unit	323
26.2.6	FIFO unit	324
26.3	Data flow overview	324
26.3.1	Data transfer and communication	324
26.3.2	IrDA protocol stack layers	324
26.3.3	IrLAP frames	325
26.4	Functional description	326
26.4.1	Operating states	326
26.4.2	Clock domains	327
26.5	Wrapper unit	327
26.5.1	Transmission state	327
26.5.2	Reception state	331
26.6	Modulation unit	332
26.6.1	Serial infrared (SIR)	332
26.6.2	Medium infrared (MIR)	333
26.6.3	Fast infrared (FIR)	333
26.7	Synchronization unit	335

26.8	Demodulation unit	335
26.8.1	Serial and medium infrared (SIR and MIR)	335
26.8.2	Fast infrared (FIR)	336
26.9	Baud rate generation unit	336
26.9.1	Serial infrared (SIR)	337
26.9.2	Medium infrared (MIR)	337
26.9.3	Fast infrared (FIR)	338
26.10	FIFO unit	338
26.10.1	Transmit state	339
26.10.2	Receive state	339
26.11	Interaction with the DMA controller	340
26.12	IrDA signals	340
26.12.1	Interface signals	340
26.12.2	Interrupts	341
27	I2C high speed controller	342
27.1	Introduction	342
27.2	Features	343
27.3	I2C protocol overview	343
27.3.1	General	344
27.3.2	Bit transfer	344
27.3.3	Data transfer	345
27.3.4	Arbitration and clock generation	346
27.3.5	Formats with 7-bit addressing mode	347
27.3.6	Address byte definition	349
27.3.7	General call address	350
27.3.8	Start byte	351
27.3.9	CBUS compatibility	351
27.3.10	Fast mode	352
27.4	High-speed controller	352
27.4.1	High speed transfer	352
27.4.2	Serial data transfer format in Hs-mode	353
27.4.3	Switching from F/S to Hs mode and back	354
27.4.4	Hs-mode devices at lower speed modes	355
27.4.5	Mixed speed modes on one serial bus system	355
27.4.6	Formats with 10-bit addressing mode	355

27.4.7	Address format	355
27.4.8	Data transfer format	355
27.4.9	General call address and start-byte with 10 bit-addressing	358
27.5	I ² C signals	359
28	1-wire master (OWM)	360
28.1	Features	360
28.2	Overview	360
28.3	Functional description	361
28.3.1	I/O signaling	361
28.3.2	OWM bit time slots: TSLOT in OWM_CR	363
28.3.3	Data endianness	363
28.4	OWM signals	363
28.4.1	Interface signals	363
28.4.2	Interrupts	364
29	SD-Card host interface (SDI)	365
29.1	Features	365
29.2	Overview	365
29.3	Functionality	369
29.4	SDI interface signals	371
30	AC/DC characteristics	373
30.1	Absolute maximum ratings	373
30.2	Electrostatic discharge (ESD) standards	373
30.3	Recommended DC operating conditions	374
30.4	DC characteristics	375
30.5	AC characteristics	378
30.6	Internal clock cycle definition	378
30.7	Oscillator electrical specifications	379
30.7.1	32.768 kHz oscillator specifications	379
30.7.2	13 to 26 MHz oscillator specifications	380
30.7.3	Reset timings	381
30.7.4	Timing requirements for device power-on reset	382
30.8	Flexible static memory controller (FSMC) timings	383

30.8.1	Switching characteristics for muxed memory read and write cycles	383
30.8.2	Timing requirements for muxed memory read cycles	384
30.8.3	Switching characteristics for CF/NAND Flash read and write cycles	387
30.8.4	Timing requirements for CF/NAND Flash memory read cycles	387
30.9	SDRAM memory controller (SDMC) timings	394
30.9.1	Switching characteristics for SDR-SDRAM access	394
30.9.2	Timing requirements for low-power/mobile SDR-SDRAM read access	395
30.9.3	Specific AC specification for mobile DDR-SDRAM access	395
30.9.4	Switching characteristics for DDR-SDRAM access	395
30.9.5	Timing requirements for mobile DDR-SDRAM read access	397
30.10	Color LCD controller (CLCD) timings	400
30.11	Display interface (DIF) timings	402
30.12	Serial camera interface (CSI) timings	404
30.12.1	Timing requirements for CSI inputs in data/strobe mode	405
30.13	CCIR-656 video input port (CCIRI) timings	406
30.14	CCIR-656 video output port (CCIRO) timings	407
30.14.1	Switching characteristics for CCIRO outputs	407
30.14.2	Timing requirements for CCIRO output	408
30.15	SD-Card/MMC interface (SDI) timings	409
30.16	Synchronous serial port (SSP)	411
30.16.1	Switching characteristics as master or slave	411
30.16.2	Timing requirements as master or slave	411
30.17	Multichannel serial port (MSP) timings	413
30.17.1	Transmitter switching characteristics (in non-SPI mode)	413
30.17.2	Transmitter timing requirements (in non-SPI mode)	414
30.17.3	Receiver switching characteristics (in non-SPI mode)	415
30.17.4	Receiver timing requirements (in non-SPI mode)	415
30.17.5	Switching characteristics as SPI master or slave	417
30.17.6	Timing requirements as SPI master or slave	417
30.17.7	Baud rate generator timing requirements (all modes)	419
30.18	I2C timings	420
30.19	1-Wire Master interface (OWM) timings	421
30.20	USB timing	423
30.20.1	Full speed mode timings	423
30.20.2	ULPI interface timings (single data rate)	424

30.21	Embedded Trace Module (ETM) timings	425
30.21.1	Switching characteristics for ETM outputs	425
30.22	Asynchronous serial interface (UART) timings	425
30.22.1	Switching characteristics for UART input/output	425
30.23	IRDA serial interface (IRDA) timings	427
30.23.1	Switching characteristics for serial infrared (SIR), medium infrared (MIR) mode, fast infrared (FIR) mode	427
30.24	JTAG timings	428
30.24.1	Switching characteristics for JTAG inputs/outputs	428
31	Package mechanical data	429
32	Ordering information	431
33	Revision history	431

1 STn8815 overview

- Smart video accelerator
 - MPEG4 encoder/decoder Simple Profile L5 SDTV 30fps
 - DVB-H class C H264 baseline L2 2Mbps CIF 30fps
 - DVB-H class B VC1 SP Main-Level 384kbps QVGA 20fps / CIF 15fps
 - T-DMB profile 1 and 2: H264 baseline L1.3 768kbps CIF 30fps
 - T-DMB two streams H264 baseline L1.3 768kbps QVGA 30fps each
 - H264 decode L2.2 2.5Mbps: VGA 30fps
 - H263 codec profile 0 level 30 384kbps CIF 30fps
 - VC-1 Main Profile Low-Level 2Mbps QVGA 24fps
 - JPEG encode up to 40 Mpixel/s, decode up to 10 Mpixel/s
 - Ultra low-power implementation
- Smart audio accelerator
 - Extensive digital-audio software library
 - Ultra low-power implementation
- Smart imaging accelerator
 - 5 Mpixel camera support
 - Two SMIA CCP2 camera interfaces
 - Parallel camera CCIR-656 interface up to 80 MHz
 - Real-time image reconstruction up to 80 Mpixel/s
 - 10-bit raw Bayer interface
 - Ultra low-power implementation
- Smart graphics accelerator for 2-D and 3-D OSD
- TV output
- Advanced power management unit
 - Run, idle, doze and sleep modes
 - CPU clock with programmable frequency
 - Enhanced dynamic power-domain management
 - Dynamic voltage scaling
- ARM926EJ® 32-bit RISC CPU, up to 332 MHz
 - 16-Kbyte instruction cache, 16-Kbyte data cache
 - 128-Kbyte level 2 cache
 - Three instruction sets: 32-bit for high performance, 16-bit (Thumb) for efficient code density, byte Java mode (Jazelle™) for direct execution of Java code
 - Embedded medium trace module (ETM Medium+)
- On-chip SRAM: 512 Kbytes
- On-chip ROM: 32 Kbytes for boot
- 16-bit DDR/SDR-SDRAM memory controller (up to 166 MHz)
- NOR Flash/NAND Flash/CompactFlash/CF+ controller
- High-speed MMC/SD Card

- Color LCD controller for STN or TFT panels or display interface for display module
 - 24-bpp true color
 - MIPI™ legacy DBI and DPI
- One high speed USB 2.0 On-The-Go controller interfaces (12 and 480 Mbit/s)
 - ULPI v1.1 compliance
 - ULPI SDR support, DDR not supported
- I/O peripherals
 - Three autobaud UARTs (one with modem control signals) up to 6 Mbit/s
 - One IrDA (SIR/MIR/FIR) interface up to 4 Mbit/s
 - One synchronous serial port (SSP) up to 24 Mbit/s
 - Four multichannel serial ports (MSP) up to 48 Mbit/s
 - Three I²C master/slave interfaces, 1 dedicated to SIA
 - Rotary encoder interface; keypad matrix interface
 - 1-Wire®/HDQ interface ⁽¹⁾
- 124 general-purpose I/Os (muxed with peripheral I/Os)
- System and peripheral controller
 - Multichannel DMA controller
 - 64-source interrupt controller
 - Eight 32-bit timers/counters
 - Real-time clock (RTC)
 - Real-time timer (RTT)
 - Watchdog timer
- Programmable PLL for CPU and system clocks
- Two crystal oscillators:
32 kHz and 13/19.2 MHz
- JTAG IEEE 1149.1 boundary scan
- Supply voltages
 - 1.2 V to 1.4 V logic; 1.8 V to 2.5 V I/O, PLL analog; 2.5 V OTP
- TFBGA 12mm x 12mm x 1.2mm, pitch 0.5 mm, ball 0.3mm

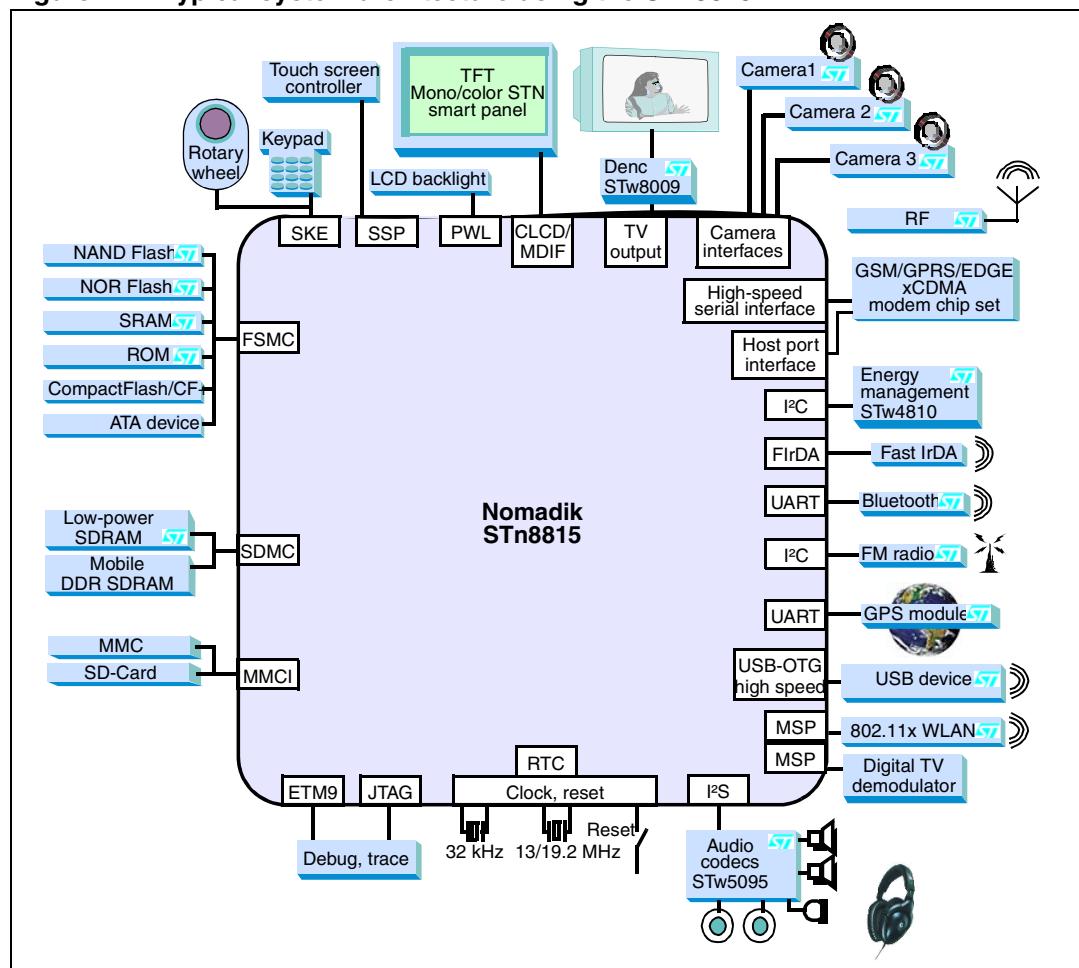
1. 1-Wire is a registered trademark of Dallas Semiconductor.

The convergence of computing, multimedia and mobile communications is well underway. Already the familiar voice phone is being transformed into a personal device with a wide range of multimedia capabilities. Soon mobile users will be able to benefit from a broad spectrum of multimedia features and services, to include capturing, sending and receiving images, videos and music. To deliver such data-heavy, processing-intensive services, portable handheld systems must be optimized for high performance but low power, space and cost.

In response to this need, the STn8815 processor platform from ST-Ericsson is a culmination of breakthroughs in video coding efficiency, inventive algorithms and chip implementation schemes. It will enable smart phones, wireless PDAs, internet appliances and car entertainment systems to play back media content, record pictures and video clips, and perform bidirectional audio-visual communication with other systems in real time.

The STn8815 focuses on the essential features to meet the future needs of mobile products and services: a high-performance multimedia capability coupled with low power consumption, and based on an open platform strategy.

Figure 2. Typical system architecture using the STn8815



1.1 Key benefits

The STn8815 brings the following key benefits to mobile manufacturers and consumers:

- Unsurpassed audio, video and imaging quality,
- Ultra-low power consumption for longer battery operation,
- Easier application development for shorter time-to-market,
- Scalability for multiple market segments and future multimedia applications.

1.2 Main features

The STn8815 processor platform enables compelling multimedia applications by means of its unique distributed-processing architecture.

The application processor features low-power smart accelerators which handle all audio, video and graphics functions. These free the main CPU for control and program flow tasks, or allow the CPU to enter power-saving modes to prolong battery life. The smart accelerators operate independently and concurrently to ensure the lowest absolute system power and deterministic high-performance.

The main features of the platform are:

- A smart video accelerator for SDTV video encoding and decoding, with MIPI and SMIA camera interfaces.
- A smart audio accelerator containing a comprehensive set of digital audio decoders and encoders, and offering a large number of 3-D surround effects.
- A smart imaging accelerator, providing real-time, programmable image reconstruction engine.
- A smart graphics accelerator.
- A dynamic, multi-mode power management unit.
- The ARM926EJ processor, a powerful industry-standard CPU with Java acceleration.
- On-chip ROM and SRAM memory devices, including a 3-Mbit frame buffer.
- Multichannel DMA controller for efficient data transfer without CPU intervention.
- A multi-layer AMBA crossbar interconnect for optimized data transfers between the CPU, accelerators, memory devices and peripherals.
- Hardware semaphores for flexible inter-process management.
- A wide range of peripheral interfaces (GPIO, USB-OTG high speed, UART, I²C, FIRDA, SD/high-speed MMC fast serial ports, TV output, color LCD and camera interfaces, scroll-key encoder, key-pad scanner).
- Direct support for high-level operating system such as SymbianTM, Linux and WinCE[®] operating systems (OSs).

1.3 Low power consumption

The new multimedia functionality of mobile products brings with it an increase in power consumption that is outpacing advances in battery technology. The STn8815 chip saves on power by avoiding the need for high clock speeds wherever possible, but its extremely low power consumption results from a systematic effort at all design levels to reduce power requirements. These include:

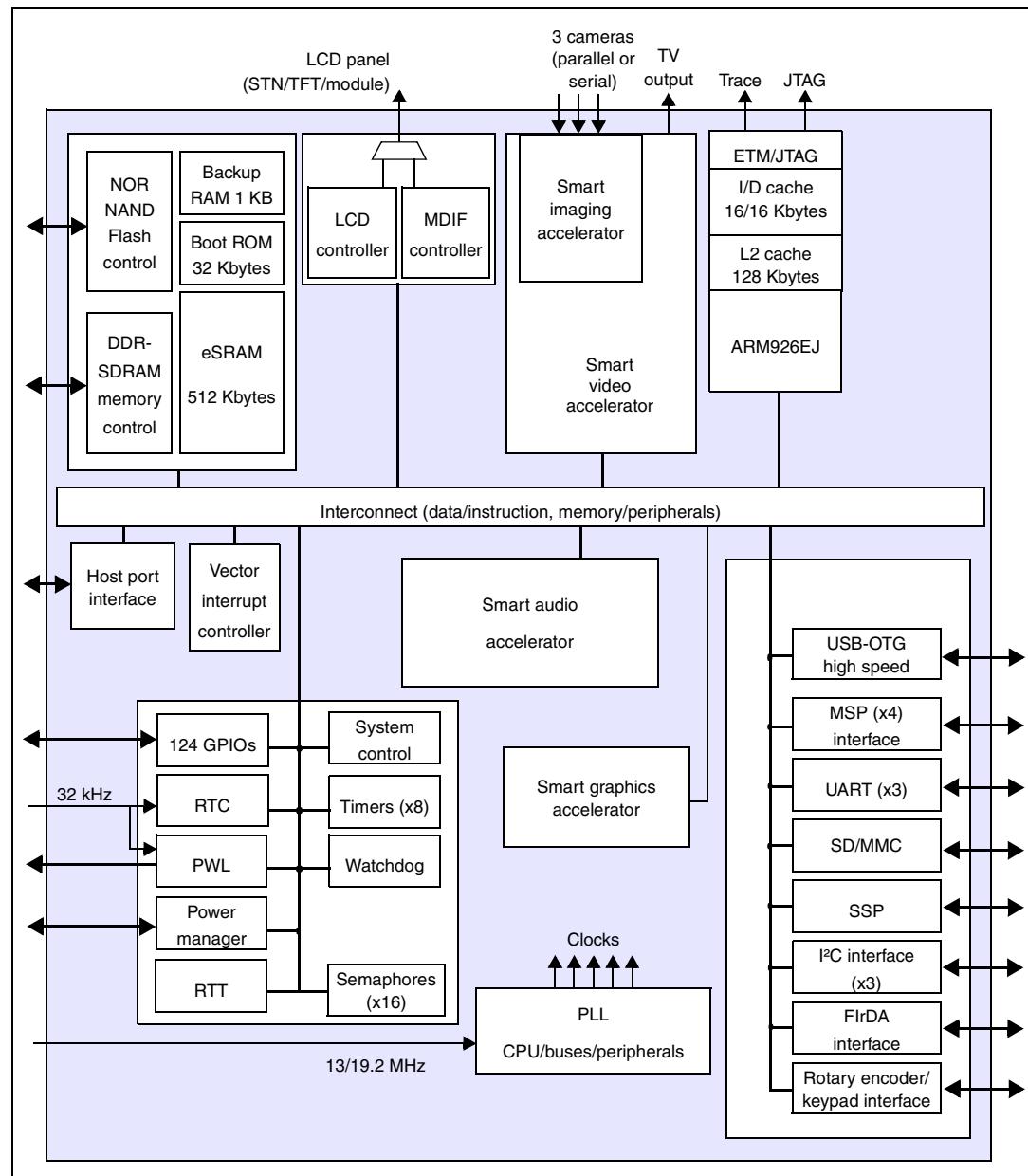
- The use of smart accelerators and distributed processing to off load from the CPU,
- Efficient code execution by means of innovative algorithms, energy-efficient instruction set architectures and Java acceleration,
- The efficient use of bandwidth for on-chip data transport, achieved by data compression, buffering and image scaling,
- Aggressive power management which includes turning off inactive parts of the chip and keeping the CPU in power-saving modes as much as possible.

2 Architecture overview

The STn8815 platform comprises an industry-standard ARM CPU supported by smart audio, video, imaging and graphics accelerators, on-chip memory and controllers, a rich set of peripheral interfaces, and a power management system. The processors, controllers, memory and peripheral interfaces are connected by a multi-layer advanced microcontroller bus architecture (AMBA) for efficient data transport between the components. The overall STn8815 architecture is illustrated in *Figure 3*.

The main hardware components of the STn8815 are listed and outlined in the sections below.

Figure 3. STn8815 block diagram



2.1 Smart video accelerator (SVA)

Using leading-edge technology, this block is a low-power, high-performance video accelerator that supports the following features:

- MPEG4 encoder/decoder Simple Profile L5 SDTV 30fps
- DVB-H class C H264 baseline L2 2Mbps CIF 30fps
- DVB-H class B VC1 SP Main-Level 384kbps QVGA 20fps / CIF 15fps
- T-DMB profile 1 and 2: H264 baseline L1.3 768kbps CIF 30fps
- T-DMB two streams H264 baseline L1.3 768kbps QVGA 30fps each
- H264 decode L2.2 2.5Mbps: VGA 30fps
- H263 codec profile 0 level 30 384kbps CIF 30fps
- VC-1 Main Profile Low-Level 2Mbps QVGA 24fps
- JPEG baseline encoder acceleration up to 40 Mpixel/s.
- JPEG baseline and extended DCT-based decoder acceleration up to 10 Mpixel/s.
- Programmable DSP (MMDSP+) for intermediate level processing, clocked at up to 166 MHz.
- Picture pre-/post-processing.
- Ultra-low power implementation.

This list of codecs is non-exhaustive, and is a baseline for the SVA performances.

2.2 Smart audio accelerator (SAA)

This high-performance block is a flexible sophisticated audio accelerator based on the MMDSP+ programmable audio DSP, clocked up to 166 MHz, and features:

- 24-bit data path,
- ultra-low power implementation.

The audio accelerator handles:

- Speech and audio codecs: AMR (WB, NR), MP3, AAC, WMA and more,
- Sample-rate conversion to 8, 16, 32, 44.1 and 48 kHz,
- Extension of sound field and 3-D surround effects,
- Noise reduction and echo cancelling.

2.3 Smart imaging accelerator (SIA)

The smart imaging accelerator is provided by the image preprocessing module of the SVA. This flexible imaging engine provides real-time, programmable image processing, and features:

- Shot-to-shot performance (with CMOS sensor):
 - 3 Mpixel, up to 15 image/s,
 - 5 Mpixel, up to 9 image/s.
- Direct support for smart sensors (CMOS/CCD modules), unlimited resolution (YUV input with JPEG compression).
- Image reconstruction up to 80 Mpixel/s with raw Bayer input.
- With shutter, maximum sensor resolution depends on external SDRAM size and on target shot-to-shot delay.
- Auto-focus control, auto exposure control.
- Automatic white balance, contrast enhancement, brightness control.
- On-the-fly zoom, flash-gun control.
- Noise reduction, gamma correction, image sharpening.
- Ultra-low power implementation.

2.4 Advanced power management unit (PMU)

The dynamic PMU optimizes power consumption of the STn8815. It delivers all the platform clocks, and handles reset management. It also manages GPIO levels during sleep mode and emergency self-refresh of SDRAM.

The PMU controls the external voltage regulator, in order to change its settings in different modes. In deep-sleep mode, only GPIOs, real-time clock (RTC), system and reset controller (SRC) and PMU remain in operation.

Voltage scaling supports two modes: standard 1.2 V and overdrive 1.4 V.

The family of power manager ICs, STw481x companion chips, seamlessly interface with the Nomadik STn8815 and optimize global system power consumption leveraging on the PMU.

2.5 ARM926EJ processor

The STn8815 CPU is an ARM926EJ reduced instruction set computer (RISC) processor. This 32-bit processor core supports 32-bit ARM and 16-bit Thumb instruction sets, enabling the user to trade off between high performance and high code density.

The cached ARM CPU features a memory management unit (MMU) and is clocked at a frequency up to 350 MHz. It has a 16-Kbyte instruction cache, a 16-Kbyte data cache, and a 128-Kbyte level 2 cache, and supports the Jazelle™ extensions for Java acceleration.

It also includes an embedded trace module (ETM Medium+) for real-time CPU activity tracing and debugging. It supports 4-bit and 8-bit normal trace mode and 4-bit demultiplexed trace mode, with normal or half-rate clock.

2.6 Embedded memory units

- 32 Kbytes of public ROM (for boot purposes),
- 512 Kbytes of public RAM,
- 1 Kbyte backup.

2.7 Flexible static memory controller (FSMC)

The FSMC interfaces to off-chip multiplexed burst NOR Flash memory devices and NAND Flash memory devices, and to CompactFlash/CF+ cards. The FSMC manages up to 4 chip-selects of NOR Flash memories, and up to 2 chip-selects of NAND Flash memories or CompactFlash/CF+ devices. The memory controller features error code correction accelerated by hardware that reduces host CPU workload to support NAND Flash very fast read/write transfers.

2.8 SDRAM memory controller (SDMC)

The SDMC is used to interface with simple-data rate synchronous dynamic random access memory (SDR-SDRAM) and double-data rate (DDR) SDRAM (133 MHz in standard mode and 166 MHz in overdrive mode). The SDMC manages up to two chip-selects of 16-bit wide SDR-SDRAM or DDR-SDRAM. It can address up to 1 Gbits of SDRAM.

2.9 Real time clock (RTC)

The RTC provides a 1-second resolution clock. This keeps time when the system is inactive and can be used to wake the system up when a programmed alarm time is reached. It has a clock trimming feature to compensate for the accuracy of the 32.768 kHz crystal.

2.10 Timers

The STn8815 provides eight 16- or 32-bit (configurable) timers, as two groups of four. They generate the periodic and timed interrupts required by OS services.

2.11 Watchdog timer

This OS resource is used to trigger a system reset in the event of software failure.

2.12 Vectored interrupt controller (VIC)

The VIC allows the OS interrupt handler to quickly dispatch interrupt service routines in response to peripheral interrupts. There are 64 interrupt lines and the VIC uses a separate bit position for each interrupt source. Software controls each request line to generate software interrupts.

2.13 System and reset controller (SRC)

The SRC provides a control interface for clock generation components external to the subsystem. It also controls system-wide and peripheral-specific energy management features.

2.14 Direct memory access (DMA) controllers

Direct memory accesses can be employed for data transfers involving DMA peripherals. A DMA controller services FIFO fill/empty requests from these peripherals immediately without CPU interaction. Peripheral-to-peripheral and memory-to-memory DMAs are also supported. A multichannel DMA controller is provided for efficient and concurrent data transfers.

2.15 Universal asynchronous receivers-transmitters (UARTs)

The STn8815 provides three autobaud UARTs, one of which offers all modem control/status signals. They are enhanced versions of the industry-standard 16C550 UART with a high data rate up to 6 Mbit/s and an embedded hardware Xon/Xoff handshake.

2.16 Synchronous serial port (SSP)

The STn8815 provides one SSP for synchronous serial communication with external peripherals. SPI, MicroWire and TI protocols are supported, with programmable word length of up to 32 bits and data rate up to 24 Mbit/s.

The SSP has the following features in both master and slave configurations:

- Parallel-to-serial conversion of data written to an internal, 32-bit wide, 32-location deep, transmit FIFO.
- Serial-to-parallel conversion of received data, which is buffered in a 32-bit wide, 32-location deep, receive FIFO.
- Programmable data frame size from 4 to 32 bits.
- Programmable clock bit rate and prescaler.
- Programmable clock phase and polarity in SPI mode.
- Support for direct memory accesses.
- Support for serial LCD smart panels.

2.17 Camera interfaces

- 3 camera interfaces that can be selected alternately:
 - 2 serial SMIA CCP2 camera interfaces (mutually exclusive)
 - Parallel camera CCIR-656 interface up to 80-MHz with embedded/external sync.
- Direct support of 8-bit and 10-bit raw Bayer RGB data formats.
- High resolution up to 5 Mpixels for camera sensors (raw Bayer data).
- Unlimited resolution for camera modules with JPEG compression.

2.18 TV interface

The STn8815 interfaces seamlessly with the STw8009 companion-chip, which performs signal conversion and connects directly to a TV set.

The TV interface is compatible with video and S-video.

2.19 Liquid crystal display controller (LCDC)

This interface drives LCD panels, and supports the following displays:

- STN displays: single- or dual-panel with 8-bit color and 4- or 8-bit monochrome,
- TFT displays: 12-, 16-, 18- and 24-bit color.

The resolution can be set as follows:

- 1-, 2- or 4-bits-per-pixel (bpp) palettized for mono STN,
- 1-, 2-, 4- or 8-bpp palettized for color STN and TFT,
- 16-bpp true-color non-palettized for color STN and TFT,
- 24-bpp packed and non-packed true-color (non-palettized) for color TFT.

The interface supports frame modulation and directly supports Sharp HR-TFT panels.

2.20 Master display interface (MDIF)

This interface drives LCD display modules, that is, panels that include their own display memory and perform LCD panel refresh themselves. The MDIF is a parallel bidirectional interface that can send commands or data to or read data from the display panel logic. It has a DMA engine to automatically fetch data/commands from main memory without CPU intervention.

In addition, the MDIF includes a submode to drive serial LCD smart panels.

2.21 Pulse width light modulator (PWL)

The PWL provides control of LCD backlighting. It produces a series of pulses that are fed to the backlighting, where the width (or duty cycle) of the pulses determines the perceived lighting level. An 8-bit random sequence generator decreases the spectral power at the modulator harmonic frequencies.

2.22 General purpose inputs/outputs (GPIOs)

The STn8815 provides 124 programmable inputs or outputs that have switchable pull-up and pull-down resistors and are controllable in two modes:

- Software mode through an APB bus interface,
- Hardware mode through a hardware control interface.

The GPIO interface provides the following individually programmable functions.

- Any number of pins may be configured as interrupt sources.
- Debouncing logic can be enabled for each GPIO to filter out glitches on I/Os.
- Any GPIO may be used to wake up the device from sleep mode independent of interrupt programming, and the input level that triggers wake-up is definable for each enabled GPIO.

2.23 Memory card interface (MMC/SD)

This interface can directly control one of the following memory cards:

- SD card (without encryption/decryption logic),
- MultiMediaCard (high-speed rate 96 Mbit/s),

It also supports several of each card type using the GPIOs for card selection.

2.24 USB-OTG high-speed interface

The STn8815 provides one USB-OTG high-speed interface. The USB-OTG features:

- High-speed signalling rate at 480 Mbit/s.
- Support for full-speed (12 Mbit/s) signaling bit rate.
- Support for session request protocol (SRP) and host negotiation protocol (HNP).
- Up to 16 bidirectional endpoints plus control endpoint 0.
- A digital interface to external PHY, such as the AOC018 companion chip.
- ULPI v1.1 compliant interface and ULPI DDR support.
- Backward compatibility with the STw481x family of power-manager ICs at full speed.

2.25 I²C bus interface

The STn8815 provides three I²C bus interfaces, one of which is dedicated to the smart imaging accelerator. The I²C interfaces support the following features:

- Slave transmitter/receiver and master transmitter/receiver modes.
- 10-bit addressing.
- Standard (100 kHz) and fast (400 kHz) speeds.
- Compliance with I²C standards.

In addition to receiving and transmitting data, the interface converts data from serial to parallel format and vice-versa using an interrupt or polled handshake.

2.26 Fast IrDA interface (FIrDA)

This interface supports IrDA half-duplex communications. It performs modulation and demodulation of infrared signals, and the wrapping of IrLAP frames. The IrDA interface supports the following infrared modes and baud rates:

- Serial infrared (SIR): 9.6 Kbit/s, 19.2 Kbit/s, 38.4 Kbit/s, 57.6 Kbit/s and 115.2 Kbit/s,
- Medium infrared (MIR): 576 Kbit/s and 1.152 Mbit/s,
- Fast infrared (FIR): 4 Mbit/s.

2.27 Multichannel serial ports (MSP)

The STn8815 includes 4 MSP synchronous receive and transmit serial interfaces. The MSPs support a data rate of up to 48 Mbit/s with the following features:

- Philips I²S format: left aligned with one cycle between leading edge of frame synchronization and first data bit, 16 or 24 bits per sample.
- Sony format: right aligned, 48 cycles per frame, 16 or 24 bits per sample.
- Matsushita format: right aligned, 64 cycles per frame, 16 or 24 bits per sample.
- Programmable number of bit-clock cycles per frame: 32, 48 or 64.
- Programmable polarity of bit-clock and frame synchronization.
- Programmable number of bits per sample: 16, 18, 20 or 24 bits.

They also provide:

- µ-Law and A-Law compressing/expanding,
- Independent framing and clocking for receive and transmit.
- External shift clock or an internal, programmable frequency shift clock for data transfer.
- Support for DMA transfers.

2.28 Scroll key and keypad encoder (SKE) interface

- This interface supports up to 2 scroll-key inputs such as jog-dials or thumb wheels.
- The keypad interface supports auto-scanning with debouncing of a keypad matrix up to 8 x 8.

3 Pin information

Signals are input to and output from the STn8815 via 324 balls (referred to as *pins* in this document) arranged in 22 rows and 22 columns on the underside of the chip.

The function of each signal is described in the appropriate chapter of this book, and in the tables in [Section 3.1](#).

The electrical characteristics of the pins are described in [Table 5](#).

The GPIO pins can have up to three STn8815 functions (denoted A, B and C). If a GPIO pin is not being used for any of the functions, it can be used by customer software. See [Chapter 18 on page 233](#) for more information.

Table 1. Pin locations and signal names (left side)

	1	2	3	4	5	6	7	8	9	10	11
A	DEC1	DEC1	TDO	GPIO3	RCOMP	RCOMPSTL	CSICKP	GPIO78	GPIO81	GPIO9	GPIO12
B	DEC1	TCK	TDI	GPIO0	GPIO4	GPIO7	CSIDAP	GPIO77	GPIO80	GPIO8	GPIO11
C	GNDANA	HRDY RTCK	TRSTn	TMS	GPIO2	GPIO6	CSIDAN	CSICKN	GPIO79	GPIO82	GPIO10
D	VDDANA	GPIO63	GPIO62	VOTP	GPIO1	GPIO5	VDDIOF	VDDIOF	VDD12	VDDIOF	VDDIOE
E	GPIO58	GPIO60	GPIO59	GPIO61							
F	GPIO95	GPIO57	GPIO56	VDDIOA							
G	GPIO92	GPIO94	GPIO93	VDD12							
H	GPIO89	GPIO91	GPIO90	VDDIOA							
J	GPIO86	GPIO88	GPIO87	VDD12							
K	GPIO83	GPIO85	GPIO84	VDDIOA							
L	GPIO52	GPIO55	GPIO54	GPIO53							
M	GPIO49	GPIO51	GPIO50	VDD12							
N	GPIO46	GPIO47	GPIO48	VDDIOA							
P	GPIO43	GPIO44	GPIO45	VDD12							
R	GPIO40	GPIO41	GPIO42	VDDIOB							
T	CLPWR DIFCS1n	CLPS DIFRDn	CLPCK DIFCD	VDD12							
U	CLACDE DIFWRn	CLLPHS DIFHS	CLLCD0 DIFD0	VDDIOB							
V	CLLCD1 DIFD1	CLLCD3 DIFD3	CLLCD5 DIFD5	CLLCD2 DIFD2							
W	CLLCD4 DIFD4	CLLCD7 DIFD7	CLLCD11 DIFD11	CLLCD12 DIFD12	CLFPVS DIFVS	VDD12	VDDIOB	SDRCS1n	VDDIOC	VDDIOC	VDDIOC
Y	CLLCD6 DIFD6	CLLCD8 DIFD8	CLLCD14 DIFD14	GPIO39	GPIO35	GPIO34	SDRDQ11	SDRDQ15	VDDIOC	SDRDQ9	SDRDQSU
AA	DEC2	CLLCD9 DIFD9	CLLCD13 DIFD13	CLLCD15 DIFD15	GPIO36	GPIO33	SDRDQ13	SDRDQ12	SDRDQ14	SDRDQ10	SDRDQ8
AB	DEC2	DEC2	CLLCD10 DIFD10	GPIO38	GPIO37	GPIO32	SDRAD4	SDRAD6	SDRAD9	SDRCKE0	SDRAD7

1 2 3 4 5 6 7 8 9 10 11

Table 2. Pin locations and signal names (right side)

12	13	14	15	16	17	18	19	20	21	22	A
GPIO15	GPIO18	GPIO22	GPIO26	GPIO29	SXTALI	SXTALO	MXTALI	MXTALO	DEC4	DEC4	B
GPIO14	GPIO17	GPIO21	GPIO25	GPIO28	VDDOK	PORn	BATOK	GPIO76	SCANMOD	DEC4	C
GPIO13	GPIO16	GPIO20	GPIO24	GPIO27	GPIO31	PWREN	GPIO75	GPIO74	GPIO73	GPIO70	D
VDDIOE	GPIO19	VDD12	GPIO23	VDD12	GPIO30	VDDIOE	DEC4	GPIO72	GPIO71	GPIO67	E
GND	GND	GND				VDD12	GPIO69	GPIO68	GPIO66		F
GND	GND	GND				VDDIOD	GPIO65	GPIO64	TAPSEL		G
GND	GND	GND				GPIO120	GPIO109	GPIO116	GPIO118		H
GND	GND	GND				VDDIOD	GPIO108	GPIO111	GPIO113		J
GND	GND	GND				VDD12	GPIO117	GPIO112	GPIO123		K
GND	GND	GND				VDDIOD	GPIO114	GPIO107	GPIO106		L
GND	GND	GND				VDDIOD	SCANEN	GPIO110	GPIO115		M
GND	GND	GND				GPIO122	SMADQ7	SMADQ6	GPIO121		N
GND	GND	GND				VDD12	GPIO101	GPIO100	GPIO103		P
GND	GND	GND				VDDIOD	SMADQ3	SMADQ2	GPIO102		R
GND	GND	GND				VDDIOD	GPIO98	GPIO97	SMADQ5		T
GND	GND	GND				VDD12	GPIO96	SMADQ1	SMADQ4		U
GND	GND	GND				GPIO119	SMADQ0	VDDIOC	GPIO99		V
GND	GND	GND				VDDIOC	VDDIOC	GPIO105	GPIO104		W
VDDIOC	VDDIOC	VDDIOC	VDD12	VDDIOC	SDRCS0n	VDDIOC	VDDIOC	SDRAD13	SDRAD1	GND	Y
SDRDQM_U	SDRFBCK	SDRDQML	SDRDQ7	VDDIOC	SDRDQ6	VDDIOC	SDRDQ0	SDRRASn	SDRAD0	VDD12	AA
SDRCKP	SDRCKN	SDRCKE1	SDRDQSL	SDRDQ5	SDRDQ3	SDRDQ1	SDRDQ4	SDRDQ2	SDRAD14	DEC3	AB
SDRAD11	SDRAD12	SDRAD8	SDRAD5	SDRAD3	SDRAD2	SDRAD10	SDRWn	SDRCASn	DEC3	DEC3	
12	13	14	15	16	17	18	19	20	21	22	

3.1 Signal definitions and assignments

The following tables detail the signals carried on and off the chip, and provide:

- the signal names (and alternate function if a GPIO pin),
- the pins on which the signals are carried,
- the direction of the signals; input only (I), output only (O) or bidirectional (I/O),
- the nature of the signals.

Table 3. Signal definitions

Signal	Pin	Type	Description
Clocks and reset, system control and configuration			
MXTALI	A19	I	Main crystal input or external oscillator Input
MXTALO	A20	O	Main MHz Crystal Output
SXTALI	A17	I	32 kHz RTC Crystal Input
SXTALO	A18	O	32 kHz RTC Crystal Output
PORn	B18	I	System Power-Up Reset. Glitches of less than 50 ns are filtered.
CLKOUT0 (GPIO25 GPIO32 GPIO76)	B15 AB6 B20	O	Clock Output 0: Programmable clock output 0 signal. To enable this clock output, the GPIO25 alternate A function or GPIO32 alternate C function or GPIO76 alternate C function must be enabled. When the CLKOUT0 signal is not used, this pin can be configured as GPIO or other alternate function signal. Upon reset, this pin is pulled low and GPIO mode.
CLKOUT1 (GPIO55)	L2	O	Clock Output 1: Programmable clock output 1 signal. To enable this clock output, the GPIO55 alternate B function must be enabled. When the CLKOUT1 signal is not used, this pin can be configured as GPIO or other alternate function signal. Upon reset, this pin is pulled-up and GPIO mode.
CLK32K (GPIO26) (GPIO75)	A15 C19	O	32 kHz Raw Clock Output: Deliver the 32 kHz clock signal. Will toggle in deep-sleep mode. When the 32 kHz clock output signal is not used, this pin can be configured as GPIO or other alternate function signal. Upon reset, this pin is pulled low and GPIO mode.
BATOK	B19	I	Battery OK Signal: This signal is continuously monitored by System and Reset Controller to trigger an interrupt (FIQ or IRQ) or reply automatically the chip in DEEP-SLEEP when a HIGH to LOW transition occurs. This signal must be HIGH to move the device from DEEP-SLEEP to SLEEP mode (i.e. in DEEP-SLEEP mode, a wake-up event is ignored if BATOK is LOW). BATOK can be used to signal to the device that the battery is low or removed from the system.
VDDOK	B17	I	VDD OK Signal: This signal is continuously monitored by System and Reset Controller to trigger an interrupt (FIQ or IRQ) or reply automatically the chip in DEEP-SLEEP when a HIGH to LOW transition occurs. This signal must be HIGH to re-enable the clocks in the device when exiting from DEEP-SLEEP or SLEEP modes. VDDOK can be used to signal to the device that the VDD12 regulator is going out of regulation (VDDOK LOW, i.e. an overcurrent condition occurs, like a shorted card is inserted in the system) or is ready to deliver full current (VDDOK HIGH, after external regulator is switched from low current mode to normal mode by PWREN signal).
PWREN	C18	O	Power Enable output: This signal is under control of the Power Management Unit (PMU), and goes LOW when device is in DEEP-SLEEP mode. A wake-up event will reassert this signal if BATOK is HIGH. PWREN can be used to signal external 1.2V voltage regulator that it can enter a low-current mode (PWREN LOW) or provide normal current (PWREN HIGH).
DBGCFG (GPIO27)	C16	I	MM-DSP debugger TAP source configuration: This pin is latched at the end of a power-on reset, to determine which IOs are used to drive the MM-DSP debugger TAP: Latched LOW: HTCK on GPIO37, HTMS on GPIO36, HTDO on GPIO35, HTDI on GPIO34, HTRSTn on GPIO33. Latched HIGH: ARM926 and MM-DSPs debugger TAPs are chained and use the JTAG port pins. After reset, this pin can be used as GPIO or other alternate function signal.
Test, JTAG			

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
TAPSEL	F22	I	TAP Selection: When HIGH, JTAG pins are directed to ARM926 CPU TAP controller (for CPU debugger connection). When LOW, JTAG pins are directed to chip TAP controller (for boundary-scan and logic scan test).
SCANEN	L20	I	SCAN Test Enable: When LOW, SCAN test is disable, chip is functional. When HIGH, SCAN test mode is enabled. To be grounded in users application.
SCANMOD	B21	I	Test Mode Enable: When HIGH, the core clock comes from TSTCLK input. To be grounded in users application.
TSTCLK	J9	I	Test Clock Input: Used to provide the core clock CLK for test purpose only. Must be grounded in normal application.
TMS	C4	I	JTAG port mode select
TDO	A3	O/Z	JTAG port data out
TDI	B3	I	JTAG port data in
TCK	B2	I	JTAG port clock
RTCK	C2	O	JTAG port return clock, delivered by ARM926 TAP controller, used by Multi-ICE. Consists of TCK clock resynchronized on the ARM926 CPU clock. Is delivered only when pin TAPSEL is HIGH.
TRSTn	C3	I	JTAG port reset. If JTAG/Debug is used, TRSTn must transit from LOW to HIGH before or at same time as PORn. If JTAG/Debug is not used, TRSTn can be tied to ground or remain unconnected (internal pull-down)
ARM926 Embedded trace module port (ETM)			
ETMCLK (GPIO95)	F1	O	ARM926 ETM Clock: Has the same frequency as the ARM926 processor clock in normal (i.e. non demultiplexed trace) mode, or half the CPU frequency in demultiplexed trace mode. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, this pin is pulled-down and in GPIO mode.
ETMPSTA2 (GPIO94)	G2	O	ARM926 ETM Pipeline Status: These are broadcast on the pipeline status pins in normal trace mode and A part in demultiplexed mode. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-down and in GPIO mode.
ETMPSTA1 (GPIO93)	G3	O	ARM926 ETM Pipeline Status: These are broadcast on the pipeline status pins in normal trace mode and A part in demultiplexed mode. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-down and in GPIO mode.
ETMPSTA0 (GPIO92)	G1	O	ARM926 ETM Pipeline Status: These are broadcast on the pipeline status pins in normal trace mode and A part in demultiplexed mode. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-down and in GPIO mode.
ETMPSTB2 (GPIO54)	L3	O	ARM926 ETM Pipeline Status, B part in demux trace mode: Pipeline status B part pins in demultiplexed trace mode. These balls are switched in this mode when the ETM is enabled (ETMEN = HIGH) and the trace port mode is demux (PORTMODE = 10b), else the balls are GPIOs. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-up (I2CSDA1/I2CSCL1) or down (RTS1n) and in GPIO mode.
ETMPSTB1 (GPIO53)	L4	O	ARM926 ETM Pipeline Status, B part in demux trace mode: Pipeline status B part pins in demultiplexed trace mode. These balls are switched in this mode when the ETM is enabled (ETMEN = HIGH) and the trace port mode is demux (PORTMODE = 10b), else the balls are GPIOs. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-up (I2CSDA1/I2CSCL1) or down (RTS1n) and in GPIO mode.
ETMPSTB0 (GPIO52)	L1	O	ARM926 ETM Pipeline Status, B part in demux trace mode: Pipeline status B part pins in demultiplexed trace mode. These balls are switched in this mode when the ETM is enabled (ETMEN = HIGH) and the trace port mode is demux (PORTMODE = 10b), else the balls are GPIOs. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-up (I2CSDA1/I2CSCL1) or down (RTS1n) and in GPIO mode.
ETMSYNCA (GPIO91)	H2	O	ARM926 ETM Synchronization: Trace synchronization signal in normal trace mode, A part in demultiplexed trace mode. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, this pin is pulled-down and in GPIO mode.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
ETMSYNCB (GPIO51)	M2	O	ARM926 ETM Synchronization, B part in demux trace mode: Trace synchronization signal, B part in demultiplexed trace mode. This ball is switched in this mode when the ETM is enabled (ETMEN = HIGH) and the trace port mode is demux (PORTMODE = 10b). When ETM is disabled or not in demux port mode, this pin can be used as GPIO or other alternate function signal. Upon reset, this pin is pulled-down and in GPIO mode.
ETMPKTA7B3 (GPIO90)	H3	O	
ETMPKTA6B2 (GPIO89)	H1	O	
ETMPKTA5B1 (GPIO88)	J2	O	
ETMPKTA4B0 (GPIO87)	J3	O	ARM926 Trace Packet Port: When ETM is enabled, these pins are delivering the Trace Packet Port output signals. In normal trace mode, these are broadcast on an 8-pin trace packet port. In demultiplexed 4-bit trace mode, A Trace packet is delivered on ETMPKTA0...3, and B packet on ETMPKTA4B0...A7B3. When ETM is disabled, these pins can be used as GPIO or other alternate function signal. Upon reset, these pins are pulled-down and in GPIO mode.
ETMPKTA3 (GPIO86)	J1	O	
ETMPKTA2 (GPIO85)	K2	O	
ETMPKTA1 (GPIO84)	K3	O	
ETMPKTA0 (GPIO83)	K1	O	
ETMTRIGIN (GPIO55)	L2	I	ARM926 ETM External Trigger Input: This ball is providing external trigger for ETM when the ETM is enabled. When ETM is disabled or external trigger signals not enabled, this pin can be used as GPIO or other alternate function signal. Upon reset, this pin is pulled-down and in GPIO mode.
General purpose I/Os (GPIO)			
GPIO123	J22	IO	GPIO line 123
GPIO122	M19	IO	GPIO line 122
GPIO121	M22	IO	GPIO line 121
GPIO120	G19	IO	GPIO line 120
GPIO119	U19	IO	GPIO line 119
GPIO118	G22	IO	GPIO line 118
GPIO117	J20	IO	GPIO line 117
GPIO116	G21	IO	GPIO line 116
GPIO115	L22	IO	GPIO line 115
GPIO114	K20	IO	GPIO line 114
GPIO113	H22	IO	GPIO line 113
GPIO112	J21	IO	GPIO line 112
GPIO111	H21	IO	GPIO line 111

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
GPIO110	L21	IO	GPIO line 110
GPIO109	G20	IO	GPIO line 109
GPIO108	H20	IO	GPIO line 108
GPIO107	K21	IO	GPIO line 107
GPIO106	K22	IO	GPIO line 106
GPIO105	V21	IO	GPIO line 105
GPIO104	V22	IO	GPIO line 104
GPIO103	N22	IO	GPIO line 103
GPIO102	P22	IO	GPIO line 102
GPIO101	N20	IO	GPIO line 101
GPIO100	N21	IO	GPIO line 100
GPIO99	U22	IO	GPIO line 99
GPIO98	R20	IO	GPIO line 98
GPIO97	R21	IO	GPIO line 97
GPIO96	T20	IO	GPIO line 96
GPIO95	F1	IO	GPIO line 95
GPIO94	G2	IO	GPIO line 94
GPIO93	G3	IO	GPIO line 93
GPIO92	G1	IO	GPIO line 92
GPIO91	H2	IO	GPIO line 91
GPIO90	H3	IO	GPIO line 90
GPIO89	H1	IO	GPIO line 89
GPIO88	J2	IO	GPIO line 88
GPIO87	J3	IO	GPIO line 87
GPIO86	J1	IO	GPIO line 86
GPIO85	K2	IO	GPIO line 85
GPIO84	K3	IO	GPIO line 84
GPIO83	K1	IO	GPIO line 83
GPIO82	C10	IO	GPIO line 82
GPIO81	A9	IO	GPIO line 81
GPIO80	B9	IO	GPIO line 80
GPIO79	C9	IO	GPIO line 79
GPIO78	A8	IO	GPIO line 78
GPIO77	B8	IO	GPIO line 77
GPIO76	B20	IO	GPIO line 76
GPIO75	C19	IO	GPIO line 75

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
GPIO74	C20	IO	GPIO line 74
GPIO73	C21	IO	GPIO line 73
GPIO72	D20	IO	GPIO line 72
GPIO71	D21	IO	GPIO line 71
GPIO70	C22	IO	GPIO line 70
GPIO69	E20	IO	GPIO line 69
GPIO68	E21	IO	GPIO line 68
GPIO67	D22	IO	GPIO line 67
GPIO66	E22	IO	GPIO line 66
GPIO65	F20	IO	GPIO line 65
GPIO64	F21	IO	GPIO line 64
GPIO63	D2	IO	GPIO line 63
GPIO62	D3	IO	GPIO line 62
GPIO61	E4	IO	GPIO line 61
GPIO60	E2	IO	GPIO line 60
GPIO59	E3	IO	GPIO line 59
GPIO58	E1	IO	GPIO line 58
GPIO57	F2	IO	GPIO line 57
GPIO56	F3	IO	GPIO line 56
GPIO55	L2	IO	GPIO line 55
GPIO54	L3	IO	GPIO line 54
GPIO53	L4	IO	GPIO line 53
GPIO52	L1	IO	GPIO line 52
GPIO51	M2	IO	GPIO line 51
GPIO50	M3	IO	GPIO line 50
GPIO49	M1	IO	GPIO line 49
GPIO48	N3	IO	GPIO line 48
GPIO47	N2	IO	GPIO line 47
GPIO46	N1	IO	GPIO line 46
GPIO45	P3	IO	GPIO line 45
GPIO44	P2	IO	GPIO line 44
GPIO43	P1	IO	GPIO line 43
GPIO42	R3	IO	GPIO line 42
GPIO41	R2	IO	GPIO line 41
GPIO40	R1	IO	GPIO line 40
GPIO39	Y4	IO	GPIO line 39

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
GPIO38	AB4	IO	GPIO line 38
GPIO37	AB5	IO	GPIO line 37
GPIO36	AA5	IO	GPIO line 36
GPIO35	Y5	IO	GPIO line 35
GPIO34	Y6	IO	GPIO line 34
GPIO33	AA6	IO	GPIO line 33
GPIO32	AB6	IO	GPIO line 32
GPIO31	C17	IO	GPIO line 31
GPIO30	D17	IO	GPIO line 30
GPIO29	A16	IO	GPIO line 29
GPIO28	B16	IO	GPIO line 28
GPIO27	C16	IO	GPIO line 27
GPIO26	A15	IO	GPIO line 26
GPIO25	B15	IO	GPIO line 25
GPIO24	C15	IO	GPIO line 24
GPIO23	D15	IO	GPIO line 23
GPIO22	A14	IO	GPIO line 22
GPIO21	B14	IO	GPIO line 21
GPIO20	C14	IO	GPIO line 20
GPIO19	D13	IO	GPIO line 19
GPIO18	A13	IO	GPIO line 18
GPIO17	B13	IO	GPIO line 17
GPIO16	C13	IO	GPIO line 16
GPIO15	A12	IO	GPIO line 15
GPIO14	B12	IO	GPIO line 14
GPIO13	C12	IO	GPIO line 13
GPIO12	A11	IO	GPIO line 12
GPIO11	B11	IO	GPIO line 11
GPIO10	C11	IO	GPIO line 10
GPIO9	A10	IO	GPIO line 9
GPIO8	B10	IO	GPIO line 8
GPIO7	B6	IO	GPIO line 7
GPIO6	C6	IO/ DI-	GPIO line 6
GPIO5	D6	IO/ DI+	GPIO line 5

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
GPIO4	B5	IO/ DI-	GPIO line 4
GPIO3	A4	IO/ DI+	GPIO line 3
GPIO2	C5	IO	GPIO line 2
GPIO1	D5	IO	GPIO line 1
GPIO0	B4	IO	GPIO line 0
Audio and video accelerator I/Os			
IPGPIO15 (GPIO113) (GPIO107)	H22	IO	Smart Video Accelerator VPIP GPIO line 15. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	K21		
IPGPIO14 (GPIO116) (GPIO106)	G21	IO	Smart Video Accelerator VPIP GPIO line 14. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	K22		
IPGPIO13 (GPIO1) (GPIO114)	D5	IO	Smart Video Accelerator VPIP GPIO line 13. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	K20		
IPGPIO12 (GPIO2) (GPIO111)	C5	IO	Smart Video Accelerator VPIP GPIO line 12. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	H21		
IPGPIO11 (GPIO115) (GPIO105)	L22	IO	Smart Video Accelerator VPIP GPIO line 11. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	V21		
IPGPIO10 (GPIO110) (GPIO104)	L21	IO	Smart Video Accelerator VPIP GPIO line 10. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	V22		
IPGPIO9 (GPIO0) (GPIO108)	B4	IO	Smart Video Accelerator VPIP GPIO line 9. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	H20		
IPGPIO8 (GPIO7) (GPIO112)	B6	IO	Smart Video Accelerator VPIP GPIO line 8. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	J21		
IPGPIO7 (GPIO113) (GPIO107)	H22	IO	Smart Video Accelerator VPIP GPIO line 7. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	K21		
IPGPIO6 (GPIO116) (GPIO106)	G21	IO	Smart Video Accelerator VPIP GPIO line 6. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	K22		
IPGPIO5 (GPIO0) (GPIO108)	B4	IO	Smart Video Accelerator VPIP GPIO line 5. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	H20		

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
IPGPIO4 (GPIO1) (GPIO114)	D5	IO	Smart Video Accelerator VPIP GPIO line 4. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	K20		
IPGPIO3 (GPIO2) (GPIO111)	C5	IO	Smart Video Accelerator VPIP GPIO line 3. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	H21		
IPGPIO2 (GPIO7) (GPIO112)	B6	IO	Smart Video Accelerator VPIP GPIO line 2. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	J21		
IPGPIO1 (GPIO115) (GPIO105)	L22	IO	Smart Video Accelerator VPIP GPIO line 1. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	V21		
IPGPIO0 (GPIO110) (GPIO104)	L21	IO	Smart Video Accelerator VPIP GPIO line 0. Software must ensure that only one single input/ouput is enable at any time to avoid conflict.
	V22		
IPI2C_SDA (GPIO76) (GPIO54)	B20	IOD	Smart Video Accelerator VPIP I2C Data Signal
	L3		
IPI2C_SCL (GPIO109) (GPIO53)	G20	IOD	Smart Video Accelerator VPIP I2C Clock Signal
	L4		
SDRAM memory controller (SDMC)			
SDRCKN	AA13	O	Inverted Clock Out for DDR-SDRAM Memories
SDRCKP	AA12	O	Clock Out for DDR/SDR-SDRAM Memories
SDRFBCK	Y13	I	Feed-Back Clock from SDR-SDRAM Memories
SDRCKE1	AA14	O	Clock Enable for DDR/SDR-SDRAM Memories selected by Chip-Select 1
SDRCKE0	AB10	O	Clock Enable for DDR/SDR-SDRAM Memories selected by Chip-Select 0
SDRCS1n	W8	O	Chip-Select 1 for DDR/SDR-SDRAM Memories
SDRCS0n	W17	O	Chip-Select 0 for DDR/SDR-SDRAM Memories
SDRRASn	Y20	O	RAS Strobe for DDR/SDR-SDRAM Memories
SDRCASn	AB20	O	CAS Strobe for DDR/SDR-SDRAM Memories
SDRWRn	AB19	O	Write Strobe for DDR/SDR-SDRAM Memories
SDRDQSU	Y11	IO	Upper Byte Data Strobe from/to DDR-SDRAM Memories
SDRDQSL	AA15	IO	Lower Byte Data Strobe from/to DDR-SDRAM Memories
SDRDQMU	Y12	O	Upper Byte Lane Enable for DDR/SDR-SDRAM Memories
SDRDQML	Y14	O	Lower Byte Lane Enable for DDR/SDR-SDRAM Memories
SDRAD14	AA21	O	DDR/SDR-SDRAM Address line 14
SDRAD13	W20	O	DDR/SDR-SDRAM Address line 13
SDRAD12	AB13	O	DDR/SDR-SDRAM Address line 12

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
SDRAD11	AB12	O	DDR/SDR-SDRAM Address line 11
SDRAD10	AB18	O	DDR/SDR-SDRAM Address line 10
SDRAD9	AB9	O	DDR/SDR-SDRAM Address line 9
SDRAD8	AB14	O	DDR/SDR-SDRAM Address line 8
SDRAD7	AB11	O	DDR/SDR-SDRAM Address line 7
SDRAD6	AB8	O	DDR/SDR-SDRAM Address line 6
SDRAD5	AB15	O	DDR/SDR-SDRAM Address line 5
SDRAD4	AB7	O	DDR/SDR-SDRAM Address line 4
SDRAD3	AB16	O	DDR/SDR-SDRAM Address line 3
SDRAD2	AB17	O	DDR/SDR-SDRAM Address line 2
SDRAD1	W21	O	DDR/SDR-SDRAM Address line 1
SDRAD0	Y21	O	DDR/SDR-SDRAM Address line 0
SDRDQ15	Y8	IO	DDR/SDR-SDRAM Data line 15
SDRDQ14	AA9	IO	DDR/SDR-SDRAM Data line 14
SDRDQ13	AA7	IO	DDR/SDR-SDRAM Data line 13
SDRDQ12	AA8	IO	DDR/SDR-SDRAM Data line 12
SDRDQ11	Y7	IO	DDR/SDR-SDRAM Data line 11
SDRDQ10	AA10	IO	DDR/SDR-SDRAM Data line 10
SDRDQ9	Y10	IO	DDR/SDR-SDRAM Data line 9
SDRDQ8	AA11	IO	DDR/SDR-SDRAM Data line 8
SDRDQ7	Y15	IO	DDR/SDR-SDRAM Data line 7
SDRDQ6	Y17	IO	DDR/SDR-SDRAM Data line 6
SDRDQ5	AA16	IO	DDR/SDR-SDRAM Data line 5
SDRDQ4	AA19	IO	DDR/SDR-SDRAM Data line 4
SDRDQ3	AA17	IO	DDR/SDR-SDRAM Data line 3
SDRDQ2	AA20	IO	DDR/SDR-SDRAM Data line 2
SDRDQ1	AA18	IO	DDR/SDR-SDRAM Data line 1
SDRDQ0	Y19	IO	DDR/SDR-SDRAM Data line 0
Flexible static memory controller (FSMC)			
SMCKO (GPIO106)	K22	O	FSMC Clock for Burst NOR-Flash Flash: FSMC delivers a clock signal when accessing a burst NOR-Flash memory. When the SMCKO signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMPS1n (GPIO119)	U19	O	FSMC CF/CF+/NAND-Flash Chip-Select 1 (active LOW). When the SMPS1n signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
SMPS0n (GPIO120)	G19	O	FSMC CF/CF+/NAND-Flash Chip-Select 0 (active LOW). When the SMPS0n signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMCS1n (GPIO109)	G20	O	FSMC Chip-Select 1 for muxed SRAM/NOR-Flash (active LOW). When the SMCS1n signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMCS0n (GPIO108)	H20	O	FSMC Chip-Select 0 for muxed SRAM/NOR-Flash (active LOW). When the SMCS0n signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMWEn (GPIO123)	J22	O	FSMC Write Enable for SRAM/NOR-Flash and CF/CF+/NAND-Flash (active LOW): This signal is an output and is used to perform writes to SRAM/NOR-Flash/NAND-Flash memories and to CF/CF+ registers (memory interface mode) or configuration registers (PC Card I/O mode). When the SMWEn signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMOEn (GPIO122)	M19	O	FSMC Output Enable for SRAM/NOR-Flash and CF/CF+/NAND-Flash (active LOW): This signal is an output and is used to enable the output of SRAM/NOR-Flash/NAND-Flash memories and CF/CF+ data, CIS and configuration registers (memory interface mode) or CIS and configuration registers (PC Card I/O mode). When the SMOEn signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMWAITn (GPIO121)	M22	I	FSMC Wait: Active HIGH or active LOW (default) as programmed in the SMC control registers for each chip-select. Allows external Device selected by SMCSxn or SMPSxn to extend I/O cycle when LOW. When the SMWAITn signal is not used, this pin can be used as GPIO or other alternate function signal. Note that in this case, the internal wait# signal of FSMC will be forced LOW, resulting in system hang if wait# signal is expected HIGH to complete a memory access. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMFWPn (GPIO105)	V21	O	FSMC Write Protect for NOR-Flash Memories (active LOW): This signal is an output and is used to protect the flash memory devices from write/erase operations when LOW. This signal is controlled by NOR-Flash bank 0 configuration register. When the SMFWPn signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.
SMFRSTn (GPIO104)	V22	O	FSMC Reset signal for NOR-Flash Memories (active LOW). This signal is an output and is used to reset or control the power-down of the flash memory devices. This signal is controlled by NOR-Flash bank 0 configuration register, and also follows PORn signal LOW level. When the SMFRSTn signal is not used, this pin can be used as GPIO or other alternate function signal. Upon reset, this pins is in alternate function A mode with pull-up/pull-down disabled.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
SMADQ15 (GPIO103)	N22	IO	Multiplexed Address/Data lines for NOR-Flash Memories, upper byte: The address is driven on this bus when SMADVn signal is LOW. Data lines for SRAM/non multiplexed NOR-Flash, and for PC-Card/NAND-Flash controller, upper byte. When the SMADQ15:8 signals are not used, each pin can be used as GPIO or other alternate function signal. Upon reset, these pins are in alternate function A mode, with pull-up/pull-down disabled.
SMADQ14 (GPIO102)	P22	IO	
SMADQ13 (GPIO101)	N20	IO	
SMADQ12 (GPIO100)	N21	IO	
SMADQ11 (GPIO99)	U22	IO	
SMADQ10 (GPIO98)	R20	IO	
SMADQ9 (GPIO97)	R21	IO	
SMADQ8 (GPIO96)	T20	IO	
SMADQ7	M20	IO	
SMADQ6	M21	IO	Multiplexed Address/Data lines for NOR-Flash Memories, lower byte: The address is driven on this bus when SMADVn signal is LOW. Data lines for SRAM/non multiplexed NOR-Flash and for PC-Card/NAND-Flash controller, lower byte.
SMADQ5	R22	IO	
SMADQ4	T22	IO	
SMADQ3	P20	IO	
SMADQ2	P21	IO	
SMADQ1	T21	IO	
SMADQ0	U20	IO	
SMBE1n (GPIO67)	D22	O	FSMC Upper Byte Lane Enable for SRAM Memory (active LOW)
SMPCE2n (GPIO67)	D22	IO	FSMC -CE2n output signal for CF/CF+ (active LOW)
SMBE0n (GPIO66)	E22	O	FSMC Low Byte Lane Enable for SRAM Memory (active LOW)
SMPCE1n (GPIO66)	E22	IO	FSMC -CE1n output signal for CF/CF+ (active LOW)
SMADVn (GPIO107)	K21	O	FSMC Address Valid for Burst NOR-Flash (active LOW): Active LOW for the first access of a burst to burst NOR-Flash memories.
SMPIORn (GPIO65)	F20	O	FSMC -IORD output signal for CF/CF+ (active LOW)
SMAD25 (GPIO65)	F20	O	FSMC address line 25. Software must ensure that only one single output is enabled at any time to avoid conflict.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
SMAD24 (GPIO118)	G22	O	FSMC address line 24. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD23 (GPIO117)	J20	O	FSMC address line 23. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD22 (GPIO116)	G21	O	FSMC address line 22. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD21 (GPIO115)	L22	O	FSMC address line 21. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD20 (GPIO114)	K20	O	FSMC address line 20. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD19 (GPIO113)	H22	O	FSMC address line 19. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD18 (GPIO112)	J21	O	FSMC address line 18. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD17 (GPIO111)	H21	O	FSMC address line 17. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD16 (GPIO110)	L21	O	FSMC address line 16. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD15 (GPIO95)	F1	O	FSMC address line 15. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD14 (GPIO94)	G2	O	FSMC address line 14. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD13 (GPIO93)	G3	O	FSMC address line 13. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD12 (GPIO92)	G1	O	FSMC address line 12. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD11 (GPIO91)	H2	O	FSMC address line 11. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD10 (GPIO90) (GPIO106)	H3 K22	O	FSMC address line 10. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD9 (GPIO89) (GPIO107)	H1 K21	O	FSMC address line 9. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD8 (GPIO118) (GPIO88)	G22 J2	O	FSMC address line 8. Software must ensure that only one single ouput is enable at any time to avoid conflict.
SMAD7 (GPIO117) (GPIO87)	J20 J3	O	FSMC address line 7. Software must ensure that only one single ouput is enable at any time to avoid conflict.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
SMAD6 (GPIO116) (GPIO86)	G21	O	FSMC address line 6. Software must ensure that only one single output is enable at any time to avoid conflict.
	J1		
SMAD5 (GPIO85) (GPIO115)	K2	O	FSMC address line 5. Software must ensure that only one single output is enable at any time to avoid conflict.
	L22		
SMAD4 (GPIO114) (GPIO84)	K20	O	FSMC address line 4. Software must ensure that only one single output is enable at any time to avoid conflict.
	K3		
SMAD3 (GPIO113) (GPIO83)	H22	O	FSMC address line 3. Software must ensure that only one single output is enable at any time to avoid conflict.
	K1		
SMAD2 (GPIO112) (GPIO50)	J21	O	FSMC address line 2. Software must ensure that only one single output is enable at any time to avoid conflict.
	M3		
SMAD1 (GPIO111) (GPIO49)	H21	O	FSMC address line 1. Software must ensure that only one single output is enable at any time to avoid conflict.
	M1		
SMAD0 (GPIO110) (GPIO48)	L21	O	FSMC address line 0. Software must ensure that only one single output is enable at any time to avoid conflict.
	N3		
SMCS2n (GPIO64)	F21	O	FSMC Chip-Select 2 for SRAM/NOR-Flash Memories (active LOW)
SMPIOWn (GPIO64)	F21	O	FSMC -IOWRn output signal for CF/CF+ (active LOW)
SMPREGn (GPIO31)	C17	O	FSMC -REG output signal for CF/CF+ (active LOW)
SMDIRn (GPIO30)	D17	O	FSMC Direction signal for all transactions: LOW for a write, HIGH for a read.
SMPIOIS16n (GPIO27)	C16	I	FSMC -IOIS16n input signal for CF/CF+ (active LOW)
LCD controller (LCDC)			
CLPWR	T1	O/Z	LCD panel power enable (STN + TFT)/MOD signal (HR-TFT) (when CLCD is selected)
CLLPHS	U2	O/Z	LCD panel line synchronization pulse (STN)/horizontal synchronization pulse (TFT)/CLLP horizontal synchro signal (HR-TFT)
CLPCK	T3	O/Z	LCD panel clock (STN, TFT, HR-TFT)
CLFPVS	W5	O/Z	LCD panel frame pulse (STN)/vertical synchronization (TFT)/CLSPS vertical synchro (HR-TFT)
CLACDE	U1	O/Z	LCD panel AC bias drive (STN)/data enable output (TFT)/CLS signal (HR-TFT)
CLPS	T2	O/Z	CLPS signal (HR-TFT only)

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
CLSPLR (GPIO35)	Y5	O/Z	LCD panel CLSPL/R: This signal is CLSPL/R for HR-TFT panel
CLPREV (GPIO34)	Y6	O/Z	LCD panel CLPREV: This signal is CLPREV for HR-TFT panel
CLCD23 (GPIO39)	Y4	O/Z	LCD panel Data 23, Upper bits
CLCD22 (GPIO38)	AB4	O/Z	LCD panel Data 22, Upper bits
CLCD21 (GPIO37)	AB5	O/Z	LCD panel Data 21, Upper bits
CLCD20 (GPIO36)	AA5	O/Z	LCD panel Data 20, Upper bits
CLCD19 (GPIO35)	Y5	O/Z	LCD panel Data 19, Upper bits
CLCD18 (GPIO34)	Y6	O/Z	LCD panel Data 18, Upper bits
CLCD17 (GPIO33)	AA6	O/Z	LCD panel Data 17, Upper bits
CLCD16 (GPIO32)	AB6	O/Z	LCD panel Data 16, Upper bits
CLCD15	AA4	IO/Z	LCD Panel Output Data bit 15
CLCD14	Y3	IO/Z	LCD Panel Output Data bit 14
CLCD13	AA3	IO/Z	LCD Panel Output Data bit 13
CLCD12	W4	IO/Z	LCD Panel Output Data bit 12
CLCD11	W3	IO/Z	LCD Panel Output Data bit 11
CLCD10	AB3	IO/Z	LCD Panel Output Data bit 10
CLCD9	AA2	IO/Z	LCD Panel Output Data bit 9
CLCD8	Y2	IO/Z	LCD Panel Output Data bit 8
CLCD7	W2	IO/Z	LCD Panel Output Data bit 7
CLCD6	Y1	IO/Z	LCD Panel Output Data bit 6
CLCD5	V3	IO/Z	LCD Panel Output Data bit 5
CLCD4	W1	IO/Z	LCD Panel Output Data bit 4
CLCD3	V2	IO/Z	LCD Panel Output Data bit 3
CLCD2	V4	IO/Z	LCD Panel Output Data bit 2
CLCD1	V1	IO/Z	LCD Panel Output Data bit 1
CLCD0	U3	IO/Z	LCD Panel Output Data bit 0
Master display interface (MDIF)			
DIFCSn	T1	O/Z	Display Interface Chip Select 1
DIFHS	U2	I	Display Horizontal (or composite) Synchro Input

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
DIFCD	T3	O/Z	Display Command (LOW)/Data (HIGH) control bit
DIFVS	W5	I	Display Vertical (or composite) Synchro Input
DIFWRn	U1	O/Z	Display Write Enable
DIFRDn	T2	O/Z	Display Read Enable
DIFCS2n (GPIO33)	AA6	O/Z	Display Interface Chip Select 2
DIFRESn (GPIO32)	AB6	O/Z	Display Interface Reset
DIFD15	AA4	IO/Z	Display Interface Data/Command bit 15 (Lower byte, Bidirectional)
DIFD14	Y3	IO/Z	Display Interface Data/Command bit 14 (Lower byte, Bidirectional)
DIFD13	AA3	IO/Z	Display Interface Data/Command bit 13 (Lower byte, Bidirectional)
DIFD12	W4	IO/Z	Display Interface Data/Command bit 12 (Lower byte, Bidirectional)
DIFD11	W3	IO/Z	Display Interface Data/Command bit 11 (Lower byte, Bidirectional)
DIFD10	AB3	IO/Z	Display Interface Data/Command bit 10 (Lower byte, Bidirectional)
DIFD9	AA2	IO/Z	Display Interface Data/Command bit 9 (Lower byte, Bidirectional)
DIFD8	Y2	IO/Z	Display Interface Data/Command bit 8 (Lower byte, Bidirectional)
DIFD7	W2	IO/Z	Display Interface Data/Command bit 7 (Lower byte, Bidirectional)
DIFD6	Y1	IO/Z	Display Interface Data/Command bit 6 (Lower byte, Bidirectional)
DIFD5	V3	IO/Z	Display Interface Data/Command bit 5 (Lower byte, Bidirectional)
DIFD4	W1	IO/Z	Display Interface Data/Command bit 4 (Lower byte, Bidirectional)
DIFD3	V2	IO/Z	Display Interface Data/Command bit 3 (Lower byte, Bidirectional)
DIFD2	V4	IO/Z	Display Interface Data/Command bit 2 (Lower byte, Bidirectional)
DIFD1	V1	IO/Z	Display Interface Data/Command bit 1 (Lower byte, Bidirectional)
DIFD0	U3	IO/Z	Display Interface Data/Command bit 0 (Lower byte, Bidirectional)
CCIR-656 video input port			
CCIRID9 (GPIO92)	G1	I	CCIR-656 Input Port Data Input: These pins are used as optional upper input data bits of the CCIR-656 video input port.
CCIRID8 (GPIO91)	H2	I	

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
CCIRID7 (GPIO90)	H3	I	CCIR-656 Input Port Data Input: These pins are used as input data bits of the CCIR-656 video input port.
CCIRID6 (GPIO89)	H1	I	
CCIRID5 (GPIO88)	J2	I	
CCIRID4 (GPIO87)	J3	I	
CCIRID3 (GPIO86)	J1	I	
CCIRID2 (GPIO85)	K2	I	
CCIRID1 (GPIO84)	K3	I	
CCIRID0 (GPIO83)	K1	I	
CCIRIVS (GPIO50)	M3	I	CCIR-656 Input Port Vertical Synchro Input: This pin is used as vertical synchro input of the CCIR-656 video input port.
CCIRIHS (GPIO49)	M1	I	CCIR-656 Input Port Horizontal Synchro Input: This pin is used as horizontal synchro input of the CCIR-656 video input port.
CCIRICLK (GPIO48)	N3	I	CCIR-656 Clock Input: This pin is used as clock input of the CCIR-656 video input port.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
CCIR-656 video output port			
CCIROD7 (GPIO90/ GPIO106)	H3	O	CCIR-656 Data Output: These pins are the CCIR-656 video output port data signals.
CCIROD6 (GPIO89/ GPIO116)	H1	O	
CCIROD5 (GPIO88/ GPIO115)	J2	O	
CCIROD4 (GPIO87/ GPIO114)	J3	O	
CCIROD3 (GPIO86/ GPIO113)	J1	O	
CCIROD2 (GPIO85/ GPIO112)	K2	O	
CCIROD1 (GPIO84/ GPIO111)	K3	O	
CCIROD0 (GPIO83/ GPIO110)	K1	O	
CCIROCLK (GPIO48/ GPIO107)	N3	IO	CCIR-656 Output Port Clock Input or Output (27 MHz): This pin is used as clock input (external 27 MHz square signal) or output (27 MHz delivered by PLL2) of the CCIR-656 video output port.
Camera serial interface			
CSIDAP	B7	DI+	Camera Port 0 Differential Data+ line
CSIDAN	C7	DI-	Camera Port 0 Differential Data- line
CSICKP	A7	DI+	Camera Port 0 Differential Clock+ line
CSICKN	C8	DI-	Camera Port 0 Differential Clock- line
CSIDAP1 (GPIO3)	A4	DI+	Camera Port 1 Differential Data+ line
CSIDAN1 (GPIO4)	B5	DI-	Camera Port 1 Differential Data- line
CSICKP1 (GPIO5)	D6	DI+	Camera Port 1 Differential Clock+ line
CSICKN1 (GPIO6)	C6	DI-	Camera Port 1 Differential Clock- line

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
Pulse width light modulator			
PWLOUT (GPIO75)	C19	O	Pulse Width Light Modulator Output
I2C interface			
I2CSCL0 (GPIO62)	D3	IOD	I2C0 Bus Clock Signal
I2CSDA0 (GPIO63)	D2	IOD	I2C0 Bus Data Signal
I2CSCL1 (GPIO53)	L4	IOD	I2C1 Bus Clock Signal
I2CSDA1 (GPIO54)	L3	IOD	I2C1 Bus Data Signal
1-Wire / HDQ Interface			
OWMDQ (GPIO76)	B20	IOD	One WIRE TM / HDQ TM
Scroll-key interface			
SKA0 (GPIO3)	A4	I	Scroll Key 0 Signals: These signals are the Scroll Key 0 quadrature digital signals.
SKB0 (GPIO4)	B5	I	Scroll Key 0 Signals: These signals are the Scroll Key 0 quadrature digital signals.
SKA1 (GPIO5) (GPIO34)	D6	I	Scroll Key 1 Signals: These signals are the Scroll Key 1 quadrature digital signals. Software must ensure that only one single input is enable at any time to avoid conflict.
SKB1 (GPIO6) (GPIO35)	C6		Scroll Key 1 Signals: These signals are the Scroll Key 1 quadrature digital signals. Software must ensure that only one single input is enable at any time to avoid conflict.
Keypad interface			
KPI0 (GPIO40)	R1	I	Matrix Key Pad Row Inputs 0
KPI1 (GPIO41)	R2	I	Matrix Key Pad Row Inputs 1
KPI2 (GPIO42)	R3	I	Matrix Key Pad Row Inputs 2
KPI3 (GPIO43)	P1	I	Matrix Key Pad Row Inputs 3
KPI4 (GPIO44)	P2	I	Matrix Key Pad Row Inputs 4
KPI5 (GPIO45)	P3	I	Matrix Key Pad Row Inputs 5

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
KPI6 (GPIO46)	N1	I	Matrix Key Pad Row Inputs 6
KPI7 (GPIO47)	N2	I	Matrix Key Pad Row Inputs 7
KPO0 (GPIO96)	T20	OD	Matrix Key Pad Column Outputs 0
KPO1 (GPIO97)	R21	OD	Matrix Key Pad Column Outputs 1
KPO2 (GPIO98)	R20	OD	Matrix Key Pad Column Outputs 2
KPO3 (GPIO99)	U22	OD	Matrix Key Pad Column Outputs 3
KPO4 (GPIO100)	N21	OD	Matrix Key Pad Column Outputs 4
KPO5 (GPIO101)	N20	OD	Matrix Key Pad Column Outputs 5
KPO6 (GPIO102)	P22	OD	Matrix Key Pad Column Outputs 6
KPO7 (GPIO103)	N22	OD	Matrix Key Pad Column Outputs 7
USB-OTG full speed external PHY interface			
USBOEn (GPIO68)	E21	O	USB Output Enable (active LOW)
USBINTn (GPIO69)	E20	I	USB External Transceiver interrupt input
USBVP (GPIO70)	C22	IO	USB Single-Ended Input/Output from/to D+ receiver
USBRCV (GPIO71)	D21	I	USB Single-Ended Data from Differential Data Receiver
USBVM (GPIO72)	D20	IO	USB Single-Ended Input/Output from/to D- receiver
USBSCL (GPIO73)	C21	IO	USB I2C Clock Line
USBSDA (GPIO74)	C20	O	USB I2C Data Line
USB-OTG high speed and ULPI external PHY interface			
ULPISTP (GPIO68)	E21	O	ULPI Stop signal
ULPIDIR (GPIO69)	E20	I	ULPI Direction signal

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
ULPINXT (GPIO74)	C20	I	ULPI Next signal
ULPICLK (GPIO75)	C19	I	ULPI Clock signal
ULPID0 (GPIO70)	C22	IO	ULPI Data bit 0
ULPID1 (GPIO71)	D21	IO	ULPI Data bit 1
ULPID2 (GPIO72)	D20	IO	ULPI Data bit 2
ULPID3 (GPIO73)	C21	IO	ULPI Data bit 3
ULPID4 (GPIO27)	C16	IO	ULPI Data bit 4
ULPID5 (GPIO26)	A15	IO	ULPI Data bit 5
ULPID6 (GPIO30)	D17	IO	ULPI Data bit 6
ULPID7 (GPIO31)	C17	IO	ULPI Data bit 7
UART signals			
UTXD0 (GPIO0)	B4	O	UART0 Transmitted Serial Data
URXD0 (GPIO1)	D5	I	UART0 Received Serial Data
UCTS0n (GPIO2)	C5	I	UART0 Clear To Send modem status (active LOW)
UDCD0n (GPIO3)	A4	I	UART0 Data Carrier Detect modem status (active LOW)
UDSR0n (GPIO4)	B5	I	UART0 Data Set Ready modem status (active LOW)
URI0n (GPIO5)	D6	I	UART0 Ring Indicator modem status (active LOW)
UDTR0n (GPIO6)	C6	O	UART0 Data Terminal Ready modem status (active LOW)
URTS0n (GPIO7)	B6	O	UART0 Request To Send modem status (active LOW)
UTXD1 (GPIO56/ GPIO28)	F3 B16	O	UART1 Transmitted Serial Data
URXD1 (GPIO57/ GPIO29 GPIO9)	F2 A16 A10	I	UART1 Received Serial Data
UCTS1n (GPIO51)	M2	I	UART1 Clear To Send modem status (active LOW)
URTS1n (GPIO52)	L1	O	UART1 Request To Send modem status (active LOW)

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
UTXD2 (GPIO38/ GPIO72)	AB4	O	UART2 Transmitted Serial Data
URXD2 (GPIO39/ GPIO70)	Y4	I	UART2 Received Serial Data
UCTS2n (GPIO37)	AB5	I	UART2 Clear To Send modem status (active LOW)
URTS2n (GPIO36)	AA5	O	UART2 Request To Send modem status (active LOW)
IrDA SIR/MIR/FIR signals			
FIRTXD (GPIO28)	B16	O	IrDA SIR/MIR/FIR Transmitted Serial Data
FIRRXD (GPIO29)	A16	I	IrDA SIR/MIR/FIR Received Serial Data
Synchronous serial ports			
SSPTXD (GPIO58)	E1	IO	SSP Transmitted Serial Data: It is a bi-directional pin in Half-Duplex MicroWire and Unidirectional Slave, and an output is the other frame modes.
SSPRXD (GPIO59)	E3	I	SSP Received Serial Data
SSPCLK (GPIO60)	E2	IO	SSP Serial Bit Clock: This signal is the bit clock.
SSPFRM (GPIO61)	E4	IO	SSP Frame Synchro: This signal is the Frame Synchro.
Multichannel serial port			
MSPTXD0 (GPIO17)	B13	O/Z	MSP0 Transmitted Serial Data
MSPTFS0 (GPIO18)	A13	IO	MSP0 Transmit Frame Synchro
MSPTCK0 (GPIO19)	D13	IO	MSP0 Transmit Serial Bit Clock
MSPRXD0 (GPIO20)	C14	I	MSP0 Received Serial Data
MSPRFS0 (GPIO21)	B14	IO	MSP0 Receive Frame Synchro
MSPRCK0 (GPIO22)	A14	IO	MSP0 Receive Serial Bit Clock
MSPSCK0 (GPIO23)	D15	I	MSP0 Sample Rate External Clock
MSPTXD1 (GPIO38)	AB4	O/Z	MSP1 Transmitted Serial Data

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
MSPTFS1 (GPIO36)	AA5	IO	MSP1 Transmit Frame Synchro
MSPTCK1 (GPIO37)	AB5	IO	MSP1 Transmit Serial Bit Clock
MSPRXD1 (GPIO39)	Y4	I	MSP1 Received Serial Data
MSPTXD2 (GPIO67)	D22	O/Z	MSP2 Transmitted Serial Data
MSPTFS2 (GPIO66)	E22	IO	MSP2 Transmit Frame Synchro
MSPTCK2 (GPIO65)	F20	IO	MSP2 Transmit Serial Bit Clock
MSPRXD2 (GPIO64)	F21	I	MSP2 Received Serial Data
MSPTXD3 (GPIO56)	F3	O/Z	MSP3 Transmitted Serial Data
MSPTFS3 (GPIO51)	M2	IO	MSP3 Transmit Frame Synchro
MSPTCK3 (GPIO52)	L1	IO	MSP3 Transmit Serial Bit Clock
MSPRXD3 (GPIO57)	F2	I	MSP3 Received Serial Data
MultiMedia Card/SD-card interface			
MCCLK (GPIO8)	B10	O	Multimedia(Plus)/SD-Card Clock;
MCMSFBCLK (GPIO24)	C15	I	Multimedia(Plus)/SD-Card Feedback Clock
MCCMD (GPIO9)	A10	IO	Multimedia(Plus)/SD-Card Command bit
MCCMDDIR (GPIO10)	C11	O	Multimedia(Plus)/SD-Card Command Direction control: When this signal is LOW, MCCMD pin is an input signal of the MM/SD Card controller, when HIGH, MCCMD pin is output signal of the MM(Plus)/SD-Card controller.
MCDAT7 (GPIO31)	C17	IO	High-Speed Multimedia-Card Data bit 7
MCDAT6 (GPIO30)	D17	IO	High-Speed Multimedia-Card Data bit 6
MCDAT5 (GPIO27)	C16	IO	High-Speed Multimedia-Card Data bit 5
MCDAT4 (GPIO26)	A15	IO	High-Speed Multimedia-Card Data bit 4
MCDAT3 (GPIO11)	B11	IO	High-Speed Multimedia/SD-Card Data bit 3

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
MCDAT2 (GPIO12)	A11	IO	High-Speed Multimedia/SD-Card Data bit 2
MCDAT1 (GPIO13)	C12	IO	High-Speed Multimedia/SD-Card Data bit 1
MCDAT0 (GPIO14)	B12	IO	Multimedia(Plus)/SD-Card Data bit 0
MCDAT0DIR (GPIO15)	A12	O	MultiMedia(Plus)/SD-Card Data Direction control for bit 0
MCDATDIR2 (GPIO23)	D15	O	MultiMedia/SD-Card Data Direction control for bit 2
MCDAT31DIR (GPIO16)	C13	O	Multimedia(Plus)/SD-Card Data Direction control for bit 3 and 1 (7...3 and 1 for MM Plus Card)
I/O supply voltage			
VDDIOA	F4 H4 K4 N4	PWR	Power Supply pins for the IO buffers. Supplies the following IO: JTAG port (TDO, TMS, TDI, TRSTn, TCK, RTCK), TSTCLK, GPIO63:40, GPIO95:83
VDDIOB	R4 U4 W7	PWR	Power Supply pins for the IO buffers. Supplies the following IO: CLCD/DIF port, GPIO39:32
VDDIOC	U21 V19 V20 W10 W11 W12 W13 W14 W16 W18 W19 W9 Y16 Y18 Y9	PWR	Power Supply pins for the IO buffers. Supplies the SDMC IOs.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
VDDIOD	F19	PWR	Power Supply pins for the IO buffers. Supplies the following IO: FSMC , GPIO63:56, GPIO123:96
	H19		
	K19		
	L19		
	P19		
	R19		
VDDIOE	D11	PWR	Power Supply pins for the IO buffers. Supplies the following IO: GPIO76:68, GPIO31:8, MXTALI, MXTALO, SXTALI, SXTALO, PORn, VDDOK, BATOK, PWREN, SCANMOD
	D12		
	D18		
VDDIOF	D10	PWR	Power Supply pins for the IO buffers. Supplies the following IO: CSI0/1 ports, GPIO82:77, GPIO7:0
	D7		
	D8		
1.2V supply voltage			
VDD12	D14	PWR	Power Supply pins for the Internal Logic (1.2 V). Must remains supplied in DEEP-SLEEP mode, but with reduced current.
	D16		
	D9		
	E19		
	G4		
	J19		
	J4		
	M4		
	N19		
	P4		
	T19		
	T4		
	W15		
	W6		
	Y22		
Special power			
VDDANA	D1	PWR	PLL Analog supply, 1.8 V +/-5% with noise < 50 mV amplitude.
GNDANA	C1	PWR	PLL Analog ground: must follow VSS voltage (+/-0.1V).
VOTP	D4	PWR	2.5 V +/- 10% voltage supply for OTP polarization (no current).
Special			
RCOMPSSTL	A6	Ref	Pad compensation for SDMC memory controller, connect to GND via a resistor of 121 kohm +/-1%.

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
RCOMP	A5	Ref	Pad compensation, connect to GND via a resistor of 121 kohm +/-1%.
Ground pin			
GND	J10	PWR	Ground pins
	J11		
	J12		
	J13		
	J14		
	K10		
	K11		
	K12		
	K13		
	K14		
	K9		
	L10		
	L11		
	L12		
	L13		
	L14		
	L9		
	M10		
	M11		
	M12		
	M13		
	M14		
	M9		
	N10		
	N11		
	N12		
	N13		
	N14		
	N9		
	P10		
	P11		
	P12		
	P13		

Table 3. Signal definitions (continued)

Signal	Pin	Type	Description
GND	P14	PWR	Ground pins
	P9		
	W22		
Decoupling signals			
DEC1	A1	-	Decoupling of internal 1.2 V logic supply. Connect a capacitor of 100 nF between this signal and GND. Note that the 3 balls are internally connected together.
	A2		
	B1		
DEC2	AA1	-	Decoupling of internal 1.2 V logic supply. Connect a capacitor of 100 nF between this signal and GND. Note that the 3 balls are internally connected together.
	AB1		
	AB2		
DEC3	AA22	-	Decoupling of internal 1.2 V logic supply. Connect a capacitor of 100 nF between this signal and GND. Note that the 3 balls are internally connected together.
	AB21		
	AB22		
DEC4	A21	-	Decoupling of internal 1.2 V logic supply. Connect a capacitor of 100 nF between this signal and GND. Note that the 3 balls are internally connected together.
	A22		
	B22		
	D19		

3.2 Pin characteristics

Table 5 gives the characteristics of each pin. *Table 4* describes the columns in *Table 5*.

Table 4. Pin characteristics table legend

Column	Description
I/O	Terminal type: I: Input, O: Output, IO: Bidirectional.
Drive	Output buffer strength.
Dom	VDDIO power domain that supplies the input/output IO buffer.
Pull	Defines if there is pull-up (typical current sourced 36 mA) and/or pull-down (typical current sinked 36 mA) in the pad buffer, refer to the column 'Comment' to understand if these pull-up and pull-down are switchable by software or permanent pull-up/down. The column PORn state defines for switchable pull-up and pull-down which one is active by default after a Power-On reset
Note	Terminal specific information: – N: Standard low voltage transistor-transistor logic (LVTTL) (1.8 V/2.5 V) input/output, – P: Boundary-scannable terminal, – R: Analog oscillator output, – S: Programmable strength LVTTL (1.8 V) input/output, – T: SubLVDS (low voltage differential signaling) (1.8 V differential) input, – U: SubLVDS biasing input.
Core off	Direction/level of pins when VDDIO is applied but VDD12 is not yet present.
PORn	Direction/level of pins when VDDIO and VDD12 are present and PORn pin is low: – PD: Pin is in input, pulled-down by internal 36 µA pull-down (typical value), – PU: Pin is in input, pulled-up by internal 36 µA pull-up (typical value), – Low: Pin is driven low, – High: Pin is driven high, – HiZ: Pin is floating.
Deepsleep	Direction/level of pins during deep sleep state (embedded 1.2 V switch cut the core supply).

Table 5. Pin characteristics

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
Clocks & Reset, System control and Configuration									
MXTALI	I		E		N	input	input	input	
MXTALO	O		E		R	output	output	output	
SXTALI	I		E		N	input	input	input	
SXTALO	O		E		R	output	output	output	
PORn	I		E		N,P	input	input	input	Need external circuitry to drive this signal
BATOK	I		E		N,P	HiZ	HiZ	HiZ	Need external circuitry to drive this signal
VDDOK	I		E		N,P	HiZ	HiZ	HiZ	Need external circuitry to drive this signal

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
PWREN	O	8 mA	E		N,P	HIGH	HIGH	LOW	
Test, JTAG									
TAPSEL	I		D	Yes	N,P	input	PD	PD	Permanent Pull-down in functional mode
SCANEN	I		D	Yes	N,P	HiZ	PD	PD	To be tied to ground. This pin is for manufacturing purposes only. Permanent pull-down in functional mode.
SCANMOD	I		E	Yes	N,P	HiZ	PD	PD	To be tied to ground. This pin is for manufacturing purposes only. Permanent pull-down in functional mode.
TSTCLK	I		A	Yes	N,P	HiZ	PD	PD	To be tied to ground. This pin is for manufacturing purposes only. Permanent pull-down in functional mode.
TMS	I		A	Yes	N	HiZ	PD	PD	Permanent Pull-down
TDO	O/Z	8 mA	A		N	LOW	LOW	LOW	
TDI	I		A	Yes	N	HiZ	PD	PD	Permanent Pull-down
TCK	I		A	Yes	N	HiZ	PD	PD	Permanent Pull-down
RTCK	O/Z	8 mA	A		N,P	LOW	LOW	LOW	
TRSTn	I		A	Yes	N	HiZ	PD	PD	Permanent Pull-down
General Purpose I/Os (GPIO)									
GPIO123	IO	8 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO122	IO	8 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO121	IO	4 mA	D	Yes	N,P	HiZ	PU	prog.	
GPIO120	IO	4 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO119	IO	4 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO118	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO117	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO116	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO115	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO114	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO113	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO112	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
GPIO111	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO110	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO109	IO	4 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO108	IO	4 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO107	IO	8 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO106	IO	8 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO105	IO	4 mA	D	Yes	N,P	HiZ	LOW	prog.	PORn state: output
GPIO104	IO	4 mA	D	Yes	N,P	HiZ	HIGH	prog.	PORn state: output
GPIO103	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO102	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO101	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO100	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO99	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO98	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO97	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO96	IO	4 mA	D	Yes	N,P	HiZ	PD	prog.	
GPIO95	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO94	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO93	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO92	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO91	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO90	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO89	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO88	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO87	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO86	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO85	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO84	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO83	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO82	IO	8 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO81	IO	4 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO80	IO	4 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO79	IO	4 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO78	IO	8 mA	F	Yes	N,P	HiZ	PD	prog.	

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
GPIO77	IO	8 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO76	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO75	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO74	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO73	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO72	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO71	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO70	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO69	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO68	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO67	IO	4 mA	D	Yes	N,P	HiZ	PU	prog.	
GPIO66	IO	4 mA	D	Yes	N,P	HiZ	PU	prog.	
GPIO65	IO	4 mA	D	Yes	N,P	HiZ	PU	prog.	
GPIO64	IO	4 mA	D	Yes	N,P	HiZ	PU	prog.	
GPIO63	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO62	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO61	IO	4 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO60	IO	4 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO59	IO	4 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO58	IO	4 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO57	IO	4 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO56	IO	4 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO55	IO	8 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO54	IO	8 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO53	IO	8 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO52	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO51	IO	8 mA	A	Yes	N,P	HiZ	PD	prog.	
GPIO50	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO49	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO48	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO47	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO46	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO45	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO44	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
GPIO43	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO42	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO41	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO40	IO	4 mA	A	Yes	N,P	HiZ	PU	prog.	
GPIO39	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO38	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO37	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO36	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO35	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO34	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO33	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO32	IO	4 mA	B	Yes	N,P	HiZ	PD	prog.	
GPIO31	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO30	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO29	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO28	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO27	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO26	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO25	IO	8 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO24	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO23	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO22	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO21	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO20	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO19	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO18	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO17	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO16	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO15	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO14	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO13	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO12	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO11	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO10	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
GPIO9	IO	4 mA	E	Yes	N,P	HiZ	PU	prog.	
GPIO8	IO	4 mA	E	Yes	N,P	HiZ	PD	prog.	
GPIO7	IO	2 mA	F	Yes	N,P	HiZ	PU	prog.	
GPIO6	IO/DI-	2 mA	F	Yes	N,T	HiZ	PU	HiZ	IOs are standard CMOS inputs with Schmitt trigger when CSI1 is disabled, or differential SubLVDS input pair when CSI1 is enabled
GPIO5	IO/DI+	2 mA	F	Yes	N,T	HiZ	PD	HiZ	IOs are standard CMOS inputs with Schmitt trigger when CSI1 is disabled, or differential SubLVDS input pair when CSI1 is enabled
GPIO4	IO/DI-	2 mA	F	Yes	N,T	HiZ	PU	HiZ	IOs are standard CMOS inputs with Schmitt trigger when CSI1 is disabled, or differential SubLVDS input pair when CSI1 is enabled
GPIO3	IO/DI+	2 mA	F	Yes	N,T	HiZ	PD	HiZ	IOs are standard CMOS inputs with Schmitt trigger when CSI1 is disabled, or differential SubLVDS input pair when CSI1 is enabled
GPIO2	IO	2 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO1	IO	2 mA	F	Yes	N,P	HiZ	PD	prog.	
GPIO0	IO	2 mA	F	Yes	N,P	HiZ	PD	prog.	

SDRAM Memory Controller (SDMC)

SDRCKN	O	2-8 mA	C		S,P	output	HIGH	LOW	
SDRCKP	O	2-8 mA	C		S,P	output	LOW	LOW	
SDRFBCK	I		C		S,P	HiZ	HiZ	HiZ	Externally connected to SDRCKP signal.
SDRCKE1	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRCKE0	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRCS1n	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRCS0n	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRRASn	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRCASn	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRWRn	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRDQSU	IO	2-8 mA	C	Yes	S,P	HiZ	PD	PD	Pull-down are always enabled except when device is in DEEP-SLEEP mode.

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
SDRDQSL	IO	2-8 mA	C	Yes	S,P	HiZ	PD	PD	Pull-down are always enabled except when device is in DEEP-SLEEP mode.
SDRDQMU	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRDQML	O	2-8 mA	C		S,P	HIGH	HIGH	HIGH	
SDRAD14	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD13	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD12	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD11	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD10	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD9	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD8	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD7	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD6	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD5	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD4	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD3	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD2	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD1	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRAD0	O	2-8 mA	C		S,P	LOW	LOW	LOW	
SDRDQ15	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ14	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ13	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ12	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ11	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ10	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
SDRDQ9	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ8	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ7	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ6	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ5	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ4	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ3	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ2	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ1	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
SDRDQ0	IO	2-8 mA	C	Yes	S,P	HiZ	HiZ	HiZ or PD	Pull-down are enabled by software (bit SDMCPDE in PMU_CTRL register)
Flexible Static Memory Controller (FSMC)									
SMADQ7	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
SMADQ6	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
SMADQ5	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
SMADQ4	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
SMADQ3	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
SMADQ2	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
SMADQ1	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
SMADQ0	IO	4 mA	D	Yes	N,P	HiZ	HiZ	Low	Pull-down enabled by software (bit FSMCPDE in PMU_CTRL reg)
LCD/DIF controller									
CLPWR/ DIFCSn	O/Z	4 mA	B	NO	N,P	HIGH	LOW	LOW	LCD mode
							HIGH	HIGH/HiZ	DIF mode HiZ if bit DIFCLCDHiZ in PMU_CTRL register is high
CLLPHS/ DIFHS	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	LCD mode
							HiZ	HiZ	DIF mode
CLPCK/ DIFCD	O/Z	4 mA	B	NO	N,P	LOW	LOW	LOW	LCD mode
							LOW	LOW/HiZ	DIF mode HiZ if bit DIFCLCDHiZ in PMU_CTRL register is high
CLFPVS/ DIFVS	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	LCD mode
							HiZ	HiZ	DIF mode
CLACDE/ DIFWRn	O/Z	4 mA	B	NO	N,P	HIGH	LOW	LOW	LCD mode
							HIGH	HIGH/HiZ	DIF mode HiZ if bit DIFCLCDHiZ in PMU_CTRL register is high
CLPS/ DIFRDn	O/Z	4 mA	B	NO	N,P	HIGH	LOW	LOW	LCD mode
							HIGH	HIGH/HiZ	DIF mode HiZ if bit DIFCLCDHiZ in PMU_CTRL register is high
CLCD15/ DIFD15	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD14/ DIFD14	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD13/ DIFD13	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
CLCD12/ DIFD12	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD11/ DIFD11	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD10/ DIFD10	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD9/ DIFD9	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD8/ DIFD8	IO/Z	4 mA	B	NO	N,P	HiZ	LOW / HiZ	LOW	If PORn LOW, HiZ state in DIF mode only
CLCD7/ DIFD7	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD6/ DIFD6	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD5/ DIFD5	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD4/ DIFD4	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD3/ DIFD3	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD2/ DIFD2	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD1/ DIFD1	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
CLCD0/ DIFD0	IO/Z	4 mA	B	NO	N,P	HiZ	LOW	LOW	
Compact Camera Port									
CSIDAP	DI+		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)
CSIDAN	DI-		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)
CSICKP	DI+		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)
CSICKN	DI-		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)
CSIDAP1	DI+		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)

Table 5. Pin characteristics (continued)

Signal	I/O	Drive	Dom	Pull	Note	Core off	PORn	Deep sleep	Comment
CSIDAN1	DI-		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)
CSICKP1	DI+		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)
CSICKN1	DI-		F	No	T	input	input	off	Differential SubLVDS input pair, switched off in SLEEP mode (no DC current)

3.3 GPIO alternate functions

Table 6 summarizes the alternate functions for each GPIO pin. If an alternate function is not available for a pin, selecting it puts the pin in HiZ.

Table 6. GPIO alternate functions

GPIO	Signal	I/O	Description
GPIO0	UTXD0	O	(Alternate A) UART0 Transmitted Serial Data
	IPGPIO5	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 5
	IPGPIO9	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 9
GPIO1	URXD0	I	(Alternate A) UART0 Received Serial Data
	IPGPIO4	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 4
	IPGPIO13	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 13
GPIO2	UCTS0n	I	(Alternate A) UART0 Clear To Send modem status (active LOW)
	IPGPIO3	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 3
	IPGPIO12	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 12
GPIO3	UDCD0n	I	(Alternate A) UART0 Data Carrier Detect modem status (active LOW)
	SKA0	I	(Alternate B) Scroll Key 0 Signals
	CSIDAP1	DI+	(double bonding) CSI1 Data+ line when CSI port 1 is enable
GPIO4	UDSR0n	I	(Alternate A) UART0 Data Set Ready modem status (active LOW)
	SKB0	I	(Alternate B) Scroll Key 0 Signals
	CSIDAN1	DI-	(double bonding) CSI1 Data- line when CSI port 1 is enable
GPIO5	URI0n	I	(Alternate A) UART0 Ring Indicator modem status (active LOW)
	SKA1	I	(Alternate B) Scroll Key 1 Signals
	CSICKP1	DI+	(double bonding) CSI1 Clock+ line when CSI port 1 is enable

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO6	UDTR0n	O	(Alternate A) UART0 Data Terminal Ready modem status (active LOW)
	SKB1	I	(Alternate B) Scroll Key 1 Signals
	CSICKN1	DI-	(double bonding) CSI1 Clock- line when CSI port 1 is enable
GPIO7	URTS0n	O	(Alternate A) UART0 Request To Send modem status (active LOW)
	IPGPIO2	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 2
	IPGPIO8	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 8
GPIO8	MCCLK	O	(Alternate A) Multimedia(Plus)/SD-Card Clock;
GPIO9	MCCMD	IO	(Alternate A) Multimedia(Plus)/SD-Card Command bit
	URXD1	I	(Alternate C) UART1 Received Serial Data
GPIO10	MCCMDDIR	O	(Alternate A) Multimedia(Plus)/SD-Card Command Direction control
GPIO11	MCDAT3	IO	(Alternate A) High-Speed Multimedia/SD-Card Data bit 3
GPIO12	MCDAT2	IO	(Alternate A) High-Speed Multimedia/SD-Card Data bit 2
GPIO13	MCDAT1	IO	(Alternate A) High-Speed Multimedia/SD-Card Data bit 1
GPIO14	MCDAT0	IO	(Alternate A) Multimedia(Plus)/SD-Card Data bit 0
GPIO15	MCDAT0DIR	O	(Alternate A) MultiMedia(Plus)/SD-Card Data Direction control for bit 0
GPIO16	MCDAT31DIR	O	(Alternate A) Multimedia(Plus)/SD-Card Data Direction control for bit 3 and 1 (7...3 and 1 for MM Plus Card)
GPIO17	MSPTXD0	O/Z	(Alternate A) MSP0 Transmitted Serial Data
GPIO18	MSPTFS0	IO	(Alternate A) MSP0 Transmit Frame Synchro
GPIO19	MSPTCK0	IO	(Alternate A) MSP0 Transmit Serial Bit Clock
GPIO20	MSPRXD0	I	(Alternate A) MSP0 Received Serial Data
GPIO21	MSPRFS0	IO	(Alternate A) MSP0 Receive Frame Synchro
GPIO22	MSPRCK0	IO	(Alternate A) MSP0 Receive Serial Bit Clock
GPIO23	MSPSCK0	I	(Alternate A) MSP0 Sample Rate External Clock
	MCDATDIR2	O	(Alternate B) MultiMedia/SD-Card Data Direction control for bit 2
GPIO24	MCMFBCLK	I	(Alternate A) Multimedia(Plus)/SD-Card Feedback Clock
GPIO25	CLKOUT0	O	(Alternate A) Clock Output 0: Programmable clock output 0 signal.
	HCSn	I	(Alternate C) Host Port Chip Select (active LOW)
GPIO26	CLK32K	O	(Alternate A) 32 kHz Raw Clock Output: Deliver the 32 kHz clock signal.
	MCDAT4	IO	(Alternate B) High-Speed Multimedia-Card Data bit 4
	ULPID5	IO	(Alternate C) ULPI Data bit 5
GPIO27	SMPIOIS16n	I	(Alternate A) FSMC -IOIS16n input signal for CF/CF+ (active LOW)
	MCDAT5	IO	(Alternate B) High-Speed Multimedia-Card Data bit 5
	ULPID4	IO	(Alternate C) ULPI Data bit 4
	DBGCFG	I	(Alternate Other) MM-DSP debugger TAP source configuration

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO28	FIRTXD	O	(Alternate A) IrDA SIR/MIR/FIR Transmitted Serial Data
	UTXD1	O	(Alternate B) UART1 Transmitted Serial Data
GPIO29	FIRRXD	I	(Alternate A) IrDA SIR/MIR/FIR Received Serial Data
	URXD1	I	(Alternate B) UART1 Received Serial Data
GPIO30	SMDIRn	O	(Alternate A) FSMC Direction signal for all transactions
	MCDAT6	IO	(Alternate B) High-Speed Multimedia-Card Data bit 6
	ULPID6	IO	(Alternate C) ULPI Data bit 6
GPIO31	SMPREGn	O	(Alternate A) FSMC -REG ouput signal for CF/CF+ (active LOW)
	MCDAT7	IO	(Alternate B) High-Speed Multimedia-Card Data bit 7
	ULPID7	IO	(Alternate C) ULPI Data bit 7
GPIO32	CLCD16	O/Z	(Alternate B) LCD panel Data 16, Upper bits
	DIFRESn	O/Z	(Alternate Other) Display Interface Reset
	CLKOUT0	O	(Alternate C) Clock Output 0: Programmable clock output 0 signal.
GPIO33	CLCD17	O/Z	(Alternate B) LCD panel Data 17, Upper bits
	DIFCS2n	O/Z	(Alternate Other) Display Interface Chip Select 2
	HTRSTn	I	(Debug Alternate2) Smart Accelerator JTAG port reset
GPIO34	CLPREV	O/Z	(Alternate A) This signal is CLPREV for HR-TFT panel
	CLCD18	O/Z	(Alternate B) LCD panel Data 18, Upper bits
	SKA1	I	(Alternate C) Scroll Key 1 Signals
	HVSDA	IOD	(Debug Alternate) Smart Video Accelerator I2C Data Signal (used only in debug)
	HTDI	I	(Debug Alternate2) Smart Accelerator JTAG port data in
GPIO35	CLSPLR	O/Z	(Alternate A) This signal is CLSPL/R for HR-TFT panel
	CLCD19	O/Z	(Alternate B) LCD panel Data 19, Upper bits
	SKB1	I	(Alternate C) Scroll Key 1 Signals
	HVSCL	IOD	(Debug Alternate) Smart Video Accelerator I2C Clock Signal (used only in debug)
	HTDO	O/Z	(Debug Alternate2) Smart Accelerator JTAG port data out
GPIO36	URTS2n	O	(Alternate A) UART2 Request To Send modem status (active LOW)
	CLCD20	O/Z	(Alternate B) LCD panel Data 20, Upper bits
	MSPTFS1	IO	(Alternate A Other) MSP1 Transmit Frame Synchro
	HASDA	IOD	(Debug Alternate) Smart Audio Accelerator I2C Data Signal (used only in debug)
	HTMS	I	(Debug Alternate2) Smart Accelerator JTAG port mode select

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO37	UCTS2n	I	(Alternate A) UART2 Clear To Send modem status (active LOW)
	CLCD21	O/Z	(Alternate B) LCD panel Data 21, Upper bits
	MSPTCK1	IO	(Alternate A Other) MSP1 Transmit Serial Bit Clock
	HASCL	IOD	(Debug Alternate) Smart Audio Accelerator I2C Clock Signal (used only in debug)
	HTCK	I	(Debug Alternate2) Smart Accelerator JTAG port clock
GPIO38	UTXD2	O	(Alternate A) UART2 Transmitted Serial Data
	CLCD22	O/Z	(Alternate B) LCD panel Data 22, Upper bits
	MSPTXD1	O/Z	(Alternate A Other) MSP1 Transmitted Serial Data
	BRKIN	I	(Debug Alternate) ARM ICE, Smart Audio & Video Accelerator external debugger BreakIn input
GPIO39	URXD2	I	(Alternate A) UART2 Received Serial Data
	CLCD23	O/Z	(Alternate B) LCD panel Data 23, Upper bits
	MSPRXD1	I	(Alternate A Other) MSP1 Received Serial Data
	BRKOUT	O	(Debug Alternate) ARM ICE, Smart Audio & Video Accelerator external debugger BreakOut output
GPIO40	KPI0	I	(Alternate A) Matrix Key Pad Row Inputs 0
	HAGPIO0	IO	(Alternate C) Smart Audio Accelerator GPIO0 / Nexus 0
	HVGPIO0	IO	(Alternate Other) Smart Video Accelerator GPIO0 / Nexus 0
GPIO41	KPI1	I	(Alternate A) Matrix Key Pad Row Inputs 1
	HAGPIO1	IO	(Alternate C) Smart Audio Accelerator GPIO1 / Nexus 1
	HVGPIO1	IO	(Alternate Other) Smart Video Accelerator GPIO1 / Nexus 1
GPIO42	KPI2	I	(Alternate A) Matrix Key Pad Row Inputs 2
	HAGPIO2	IO	(Alternate C) Smart Audio Accelerator GPIO2 / Nexus 2
	HVGPIO2	IO	(Alternate Other) Smart Video Accelerator GPIO2 / Nexus 2
GPIO43	KPI3	I	(Alternate A) Matrix Key Pad Row Inputs 3
	HAGPIO3	IO	(Alternate C) Smart Audio Accelerator GPIO3 / Nexus 3
	HVGPIO3	IO	(Alternate Other) Smart Video Accelerator GPIO3 / Nexus 3
GPIO44	KPI4	I	(Alternate A) Matrix Key Pad Row Inputs 4
	HAGPIO4	IO	(Alternate C) Smart Audio Accelerator GPIO4 / Nexus 4
	HVGPIO4	IO	(Alternate Other) Smart Video Accelerator GPIO4 / Nexus 4
GPIO45	KPI5	I	(Alternate A) Matrix Key Pad Row Inputs 5
	HAGPIO5	IO	(Alternate C) Smart Audio Accelerator GPIO5 / Nexus 5
	HVGPIO5	IO	(Alternate Other) Smart Video Accelerator GPIO5 / Nexus 5

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO46	KPI6	I	(Alternate A) Matrix Key Pad Row Inputs 6
	HAGPIO6	IO	(Alternate C) Smart Audio Accelerator GPIO6 / Nexus 6
	HVGPIO6	IO	(Alternate Other) Smart Video Accelerator GPIO6 / Nexus 6
GPIO47	KPI7	I	(Alternate A) Matrix Key Pad Row Inputs 7
	HAGPIO7	IO	(Alternate C) Smart Audio Accelerator GPIO7 / Nexus 7
	HVGPIO7	IO	(Alternate Other) Smart Video Accelerator GPIO7 / Nexus 7
GPIO48	CCIRICLK	I	(Alternate A) CCIR656 Clock Input
	CCIROCLK	IO	(Alternate B) CCIR656 Output Port Clock Input or Output (27 MHz)
	SMAD0	O	(Alternate C) FSMC address line 0
GPIO49	CCIRIHS	I	(Alternate A) CCIR656 Input Port Horizontal Synchro Input
	SMAD1	O	(Alternate C) FSMC address line 1
GPIO50	CCIRIVS	I	(Alternate A) CCIR656 Input Port Vertical Synchro Input
	SMAD2	O	(Alternate C) FSMC address line 2
GPIO51	UCTS1n	I	(Alternate A) UART1 Clear To Send modem status (active LOW)
	MSPTFS3	IO	(Alternate C) MSP3 Transmit Frame Synchro
	ETMSYNCB	O	(Debug Alternate) ARM926 ETM Synchronization, B part in demux trace mode
	IPTDI	I	(Debug Alternate2) Smart Video Accelerator VPIP JTAG port data in
GPIO52	URTS1n	O	(Alternate A) UART1 Request To Send modem status (active LOW)
	MSPTCK3	IO	(Alternate C) MSP3 Transmit Clock (alternate)
	ETMPSTB0	O	(Debug Alternate) ARM926 ETM Pipeline Status, B part in demux trace mode
	IPTDO	O/Z	(Debug Alternate2) Smart Video Accelerator VPIP JTAG port data out
GPIO53	I2CSCL1	IOD	(Alternate A) I2C1 Bus Clock Signal
	IPI2C_SCL	IOD	(Alternate B) Smart Video Accelerator VPIP I2C Clock Signal
	ETMPSTB1	O	(Debug Alternate) ARM926 ETM Pipeline Status, B part in demux trace mode
	IPTMS	I	(Debug Alternate2) Smart Video Accelerator VPIP JTAG port mode select
GPIO54	I2CSDA1	IOD	(Alternate A) I2C1 Bus Data Signal
	IPI2C_SDA	IOD	(Alternate B) Smart Video Accelerator VPIP I2C Data Signal
	ETMPSTB2	O	(Debug Alternate) ARM926 ETM Pipeline Status, B part in demux trace mode
	IPTCK	I	(Debug Alternate2) Smart Video Accelerator VPIP JTAG port clock
GPIO55	CLKOUT1	O	(Alternate A) Clock Output 1: Programmable clock output 1 signal.
	ETMTRIGIN	I	(Debug Alternate) ARM926 ETM External Trigger Input
	IPTRIGOUT	O	(Debug Alternate2) Smart Video Accelerator VPIP JTAG port control
GPIO56	UTXD1	O	(Alternate A) UART1 Transmitted Serial Data
	MSPTXD3	IO	(Alternate C) MSP3 Transmitted Serial Data

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO57	URXD1	I	(Alternate A) UART1 Received Serial Data
	MSPRXD3	IO	(Alternate C) MSP3 Received Serial Data
GPIO58	SSPTXD	IO	(Alternate A) SSP Transmitted Serial Data
GPIO59	SSPRXD	I	(Alternate A) SSP Received Serial Data
GPIO60	SSPCLK	IO	(Alternate A) SSP Serial Bit Clock
GPIO61	SSPFRM	IO	(Alternate A) SSP Frame Synchro
	HWRn	I	(Alternate B) Host Port Write Enable (active LOW)
GPIO62	I2CSCL0	IOD	(Alternate A) I2C0 Bus Clock Signal
	HANXCK	O	(Alternate C) Smart Accelerator Audio debug Nexus Clock
	HVNXCK	O	(Alternate Other) Smart Accelerator Video debug Nexus Clock
GPIO63	I2CSDA0	IOD	(Alternate A) I2C0 Bus Data Signal
GPIO64	SMPIONn	O	(Alternate A) FSMC -IOWRn output signal for CF/CF+ (active LOW)
	SMCS2n	O	(Alternate B) FSMC Chip-Select 2 for SRAM/NOR-Flash Memories (active LOW)
	MSPRXD2	I	(Alternate C) MSP2 Received Serial Data
GPIO65	SMPIORn	O	(Alternate A) FSMC -IORD output signal for CF/CF+ (active LOW)
	SMAD25	O	(Alternate B) FSMC address line 25
	MSPTCK2	IO	(Alternate C) MSP2 Transmit Serial Bit Clock
GPIO66	SMPCE1n	IO	(Alternate A) FSMC -CE1n output signal for CF/CF+ (active LOW)
	SMBE0n	O	(Alternate B) FSMC Low Byte Lane Enable for SRAM Memory (active LOW)
	MSPTFS2	IO	(Alternate C) MSP2 Transmit Frame Synchro
GPIO67	SMPCE2n	IO	(Alternate A) FSMC -CE2n output signal for CF/CF+ (active LOW)
	SMBE1n	O	(Alternate B) FSMC Upper Byte Lane Enable for SRAM Memory (active LOW)
	MSPTXD2	O/Z	(Alternate C) MSP2 Transmitted Serial Data
GPIO68	USBOEn	O	(Alternate B) USB Output Enable (active LOW)
	ULPISTP	O	(Alternate C) ULPI Stop signal
GPIO69	USBINTn	I	(Alternate B) USB External Transceiver interrupt input
	ULPIDIR	I	(Alternate C) ULPI Direction signal
GPIO70	URXD2	I	(Alternate A) UART2 Received Serial Data
	USBVP	IO	(Alternate B) USB Single-Ended Input/Output from/to D+ receiver
	ULPID0	IO	(Alternate C) ULPI Data bit 0
GPIO71	USBRCV	I	(Alternate B) USB Single-Ended Data from Differential Data Receiver
	ULPID1	IO	(Alternate C) ULPI Data bit 1

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO72	UTXD2	O	(Alternate A) UART2 Transmitted Serial Data
	USBVM	IO	(Alternate B) USB Single-Ended Input/Output from/to D- receiver
	ULPID2	IO	(Alternate C) ULPI Data bit 2
GPIO73	USBSCL	IO	(Alternate B) USB I2C Clock Line
	ULPID3	IO	(Alternate C) ULPI Data bit 3
GPIO74	USBSDA	O	(Alternate B) USB I2C Data Line
	ULPINXT	I	(Alternate C) ULPI Next signal
GPIO75	PWLOUT	O	(Alternate A) Pulse Width Light Modulator Output
	CLK32K	O	(Alternate B) 32 kHz Raw Clock Output: Deliver the 32 kHz clock signal.
	ULPICLK	I	(Alternate C) ULPI Clock signal
GPIO76	OWMDQ	IOD	(Alternate A) One WIRE TM / HDQ TM
	IPI2C_SDA	IOD	(Alternate B) Smart Video Accelerator VPIP I2C Data Signal
	CLKOUT0	O	(Alternate C) Clock Output 0: Programmable clock output 0 signal.
GPIO83	CCIRID0	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD0	O	(Alternate B) CCIR656 Data Output
	SMAD3	O	(Alternate C) FSMC address line 3
	ETMPKTA0	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO84	CCIRID1	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD1	O	(Alternate B) CCIR656 Data Output
	SMAD4	O	(Alternate C) FSMC address line 4
	ETMPKTA1	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO85	CCIRID2	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD2	O	(Alternate B) CCIR656 Data Output
	SMAD5	O	(Alternate C) FSMC address line 5
	ETMPKTA2	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO86	CCIRID3	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD3	O	(Alternate B) CCIR656 Data Output
	SMAD6	O	(Alternate C) FSMC address line 6
	ETMPKTA3	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO87	CCIRID4	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD4	O	(Alternate B) CCIR656 Data Output
	SMAD7	O	(Alternate C) FSMC address line 7
	ETMPKTA4B0	O	(Debug Alternate) ARM926 Trace Packet Port

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO88	CCIRID5	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD5	O	(Alternate B) CCIR656 Data Output
	SMAD8	O	(Alternate C) FSMC address line 8
	ETMPKTA5B1	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO89	CCIRID6	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD6	O	(Alternate B) CCIR656 Data Output
	SMAD9	O	(Alternate C) FSMC address line 9
	ETMPKTA6B2	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO90	CCIRID7	I	(Alternate A) CCIR656 Input Port Data Input
	CCIROD7	O	(Alternate B) CCIR656 Data Output
	SMAD10	O	(Alternate C) FSMC address line 10
	ETMPKTA7B3	O	(Debug Alternate) ARM926 Trace Packet Port
GPIO91	CCIRID8	I	(Alternate A) CCIR656 Input Port Data Optional Upper Input
	SMAD11	O	(Alternate C) FSMC address line 11
	ETMSYNCA	O	(Debug Alternate) ARM926 ETM Synchronization
GPIO92	CCIRID9	I	(Alternate A) CCIR656 Input Port Data Optional Upper Input
	SMAD12	O	(Alternate C) FSMC address line 12
	ETMPSTA0	O	(Debug Alternate) ARM926 ETM Pipeline Status
GPIO93	SMAD13	O	(Alternate C) FSMC address line 13
	ETMPSTA1	O	(Debug Alternate) ARM926 ETM Pipeline Status
GPIO94	SMAD14	O	(Alternate C) FSMC address line 14
	ETMPSTA2	O	(Debug Alternate) ARM926 ETM Pipeline Status
GPIO95	SMAD15	O	(Alternate C) FSMC address line 15
	ETMCLK	O	(Debug Alternate) ARM926 ETM Clock
GPIO96	SMADQ8	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO0	OD	(Alternate C) Matrix Key Pad Column Outputs 0
GPIO97	SMADQ9	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO1	OD	(Alternate C) Matrix Key Pad Column Outputs 1
GPIO98	SMADQ10	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO2	OD	(Alternate C) Matrix Key Pad Column Outputs 2
GPIO99	SMADQ11	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO3	OD	(Alternate C) Matrix Key Pad Column Outputs 3

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO100	SMADQ12	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO4	OD	(Alternate C) Matrix Key Pad Column Outputs 4
GPIO101	SMADQ13	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO5	OD	(Alternate C) Matrix Key Pad Column Outputs 5
GPIO102	SMADQ14	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO6	OD	(Alternate C) Matrix Key Pad Column Outputs 6
GPIO103	SMADQ15	IO	(Alternate A) Multiplexed Address/Data lines for NOR-Flash Memories, upper byte
	KPO7	OD	(Alternate C) Matrix Key Pad Column Outputs 7
GPIO104	SMFRSTn	O	(Alternate A) FSMC Reset signal for NOR-Flash Memories (active LOW)
	IPGPIO0	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 0
	IPGPIO10	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 10
GPIO105	SMFWPn	O	(Alternate A) FSMC Write Protect for NOR-Flash Memories (active LOW)
	IPGPIO1	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 1
	IPGPIO11	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 11
GPIO106	SMCKO	O	(Alternate A) FSMC Clock for Burst NOR-Flash Flash
	IPGPIO6	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 6
	IPGPIO14	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 14
	SMAD10	O	(Alternate Other) FSMC address line 10
	CCIROD7	O	(Alternate Other) CCIR-656 Output data line 7
GPIO107	SMADVn	O	(Alternate A) FSMC Address Valid for Burst NOR-Flash (active LOW)
	IPGPIO7	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 7
	IPGPIO15	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 15
	SMAD9	O	(Alternate Other) FSMC address line 9
	CCIROCLK	O	(Alternate Other) CCIR-656 Output clock
GPIO108	SMCS0n	O	(Alternate A) FSMC Chip-Select 0 for muxed SRAM/NOR-Flash (active LOW)
	IPGPIO5	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 5
	IPGPIO9	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 9
GPIO109	SMCS1n	O	(Alternate A) FSMC Chip-Select 1 for muxed SRAM/NOR-Flash (active LOW)
	IPI2C_SCL	IOD	(Alternate B) Smart Video Accelerator VPIP I2C Clock Signal

Table 6. GPIO alternate functions (continued)

GPIO	Signal	I/O	Description
GPIO110	SMAD16	O	(Alternate A) FSMC address line 16
	IPGPIO0	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 0
	IPGPIO10	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 10
	SMAD0	O	(Alternate Other) FSMC address line 0
	CCIROD0	O	(Alternate Other) CCIR-656 Output data line 0
GPIO111	SMAD17	O	(Alternate A) FSMC address line 17
	IPGPIO3	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 3
	IPGPIO12	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 12
	SMAD1	O	(Alternate Other) FSMC address line 1
	CCIROD1	O	(Alternate Other) CCIR-656 Output data line 1
GPIO112	SMAD18	O	(Alternate A) FSMC address line 18
	IPGPIO2	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 2
	IPGPIO8	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 8
	SMAD2	O	(Alternate Other) FSMC address line 2
	CCIROD2	O	(Alternate Other) CCIR-656 Output data line 2
GPIO113	SMAD19	O	(Alternate A) FSMC address line 19
	IPGPIO7	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 7
	IPGPIO15	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 15
	SMAD3	O	(Alternate Other) FSMC address line 3
	CCIROD3	O	(Alternate Other) CCIR-656 Output data line 3
GPIO114	SMAD20	O	(Alternate A) FSMC address line 20
	IPGPIO4	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 4
	IPGPIO13	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 13
	SMAD4	O	(Alternate Other) FSMC address line 4
	CCIROD4	O	(Alternate Other) CCIR-656 Output data line 4
GPIO115	SMAD21	O	(Alternate A) FSMC address line 21
	IPGPIO1	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 1
	IPGPIO11	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 11
	SMAD5	O	(Alternate Other) FSMC address line 5
	CCIROD5	O	(Alternate Other) CCIR-656 Output data line 5
GPIO116	SMAD22	O	(Alternate A) FSMC address line 22
	IPGPIO6	IO	(Alternate B) Smart Video Accelerator VPIP GPIO line 6
	IPGPIO14	IO	(Alternate C) Smart Video Accelerator VPIP GPIO line 14
	SMAD6	O	(Alternate Other) FSMC address line 6
	CCIROD6	O	(Alternate Other) CCIR-656 Output data line 6

Table 6. GPIO alternate functions (continued)

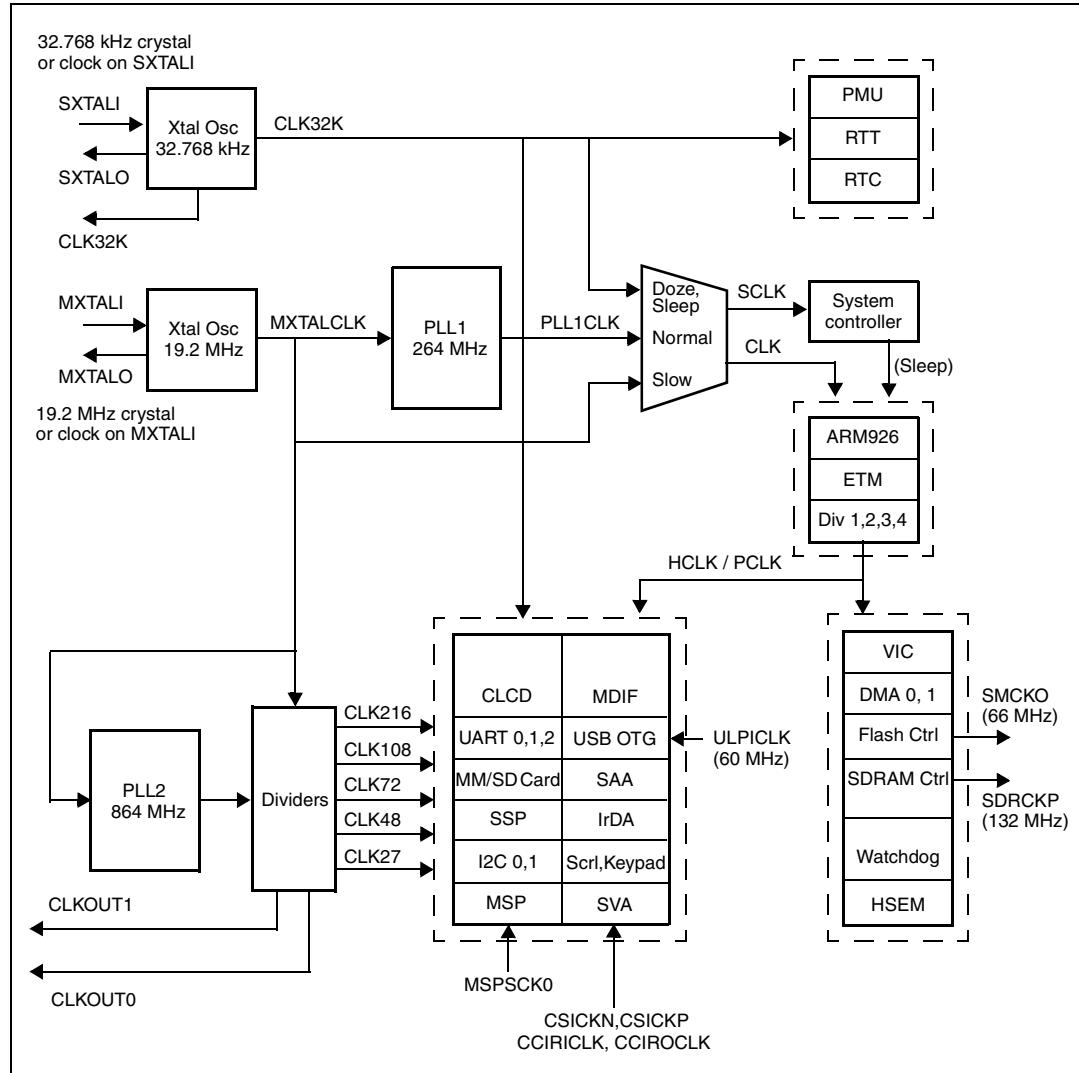
GPIO	Signal	I/O	Description
GPIO117	SMAD23	O	(Alternate A) FSMC address line 23
	SMAD7	O	(Alternate Other) FSMC address line 7
GPIO118	SMAD24	O	(Alternate A) FSMC address line 24
	SMAD8	O	(Alternate Other) FSMC address line 8
GPIO119	SMPS1n	O	(Alternate A) FSMC CF/CF+/NAND-Flash Chip-Select 1 (active LOW)
GPIO120	SMPS0n	O	(Alternate A) FSMC CF/CF+/NAND-Flash Chip-Select 0 (active LOW)
GPIO121	SMWAITn	I	(Alternate A) FSMC Wait
GPIO122	SMOEn	O	(Alternate A) FSMC Output Enable for SRAM/NOR-Flash and CF/CF+/NAND-Flash (active LOW)
GPIO123	SMWEn	O	(Alternate A) FSMC Write Enable for SRAM/NOR-Flash and CF/CF+/NAND-Flash (active LOW)

4 Clocks

This chapter summarizes the clocks used by the STn8815A12 platform. The individual clocks are described in more detail in later chapters.

Figure 4 illustrates the overall clock scheme of the platform.

Figure 4. Clock scheme



4.1 Input clocks

The STn8815A12 requires two external input clocks:

- **SXTALI: 32.768 kHz clock**

SXTALI provides (in normal mode) the real-time clock at a nominal rate of 1 Hz (CLK1Hz).

- **MXTALI: 19.2 MHz clock**

MXTALI (typically driven at 19.2 MHz) drives the internal phase locked loops (PLLs) and some peripherals as follows:

- PLL1: programmable output frequency, in the range 20.4 to 332 MHz, for the system clocks (CLK/HCLK) and system controller clock (SCLK); see [Section 4.2.1](#).
- PLL2: Fixed output frequencies, for peripheral clocks; see [Section 4.2.2](#).

4.2 Output clocks

The STn8815A12 uses two phase locked loops (PLLs) to produce the clocks, PLL1 and PLL2. The clocks they produce are described in the subsections below.

4.2.1 PLL1 System clocks

PLL1 takes the 19.2 MHz MXTALI clock and delivers the system clocks CLK and HCLK, with a programmable output frequency in the range 20.4 MHz to 332 MHz.

The system clocks are defined as follows:

- CLK. This clock:
 - is the main system clock,
 - is used to clock the ARM926 processor and the ETM extended trace module,
 - has a programmable frequency.
- HCLK. This clock:
 - is equal to CLK divided by 1, 2, 3 or 4,
 - is used to clock the AHB and APB (PCLK) buses, the vectored interrupt controller (VIC), timers and the watchdog.

Other AHB masters and slaves, and the PCLK domain, use gated versions of HCLK.

4.2.2 PLL2 Peripheral clocks

The PLL2 takes the 19.2 MHz MXTALI clock and delivers clocks at fixed frequencies:

- CLK216 with frequency 216 MHz,
- CLK108 with frequency 108 MHz (CLK216 divided by 2),
- CLK72 with frequency 72 MHz,
- CLK48 with frequency 48 MHz,
- CLK27 with frequency 27 MHz (CLK108 divided by 4)

These clocks are then used by most of the platform's peripherals.

Having a separate PLL2 to generate these peripheral clock frequencies allows the peripherals to continue running at their standard frequencies without having to reprogram their internal clock prescalers when the PLL1 output frequency is changed or stopped.

Two clock outputs are used to drive the peripheral clocks, as described below.

CLK72: The CLK72 output drives the peripheral clocks listed in [Table 7](#).

Table 7. Peripheral clocks sourced from CLK72 clock

Clock	Frequency	Peripheral
DIFCLK	72 or 78 MHz	DIFCLK clocks the master display interface (MDIF) logic.
CLCDCLK	72 or 78 MHz	CLCDCLK clocks the color LCD controller (CLCD) kernel logic.

CLK48: The CLK48 output drives the 48 MHz peripheral clocks listed in [Table 8](#).

Table 8. Peripheral clocks sourced from CLK48 MHz clock

Clock	Peripheral
UARTxCLK	UARTxCLK clocks the UARTx kernel logic. Baud rates of up to 3 Mbit/s can be attained.
CLCDCLK	CLCDCLK clocks the color LCD controller (CLCD) kernel logic.
SSPCLK	SSPCLK clocks the synchronous serial port (SSP) kernel logic.
I2CxCLK	I2CxCLK clocks the I2C kernel logic.
MCCLK	MCCLK clocks the SD-card/MM-card kernel logic.
IRDACLK	IRDACLK clocks the IrDA interface kernel logic.
MSPxCLK	MSPxCLK clocks the MSPx baud-rate generator logic.
USBCLK	USBCLK clocks the USB OTG kernel logic in FS mode.

5 System and reset controller (SRC)

The system and reset controller (SRC) provides an interface to control the operation of the system. The SRC is clocked from the system controller clock (SCLK), which continues to toggle when the rest of the system is suspended, thus the SRC is always active.

5.1 System mode control

The SRC controls the system modes. A system mode control state machine is provided to define the source of the system clock and the system controller clock inputs for the different system modes. The available modes are:

- normal
- slow
- doze
- sleep/deep-sleep

The four system modes are described in [Table 9](#).

The state machine can be configured to define the required system operating mode. Once defined, the system mode control state machine moves to the required operating mode without further software interaction.

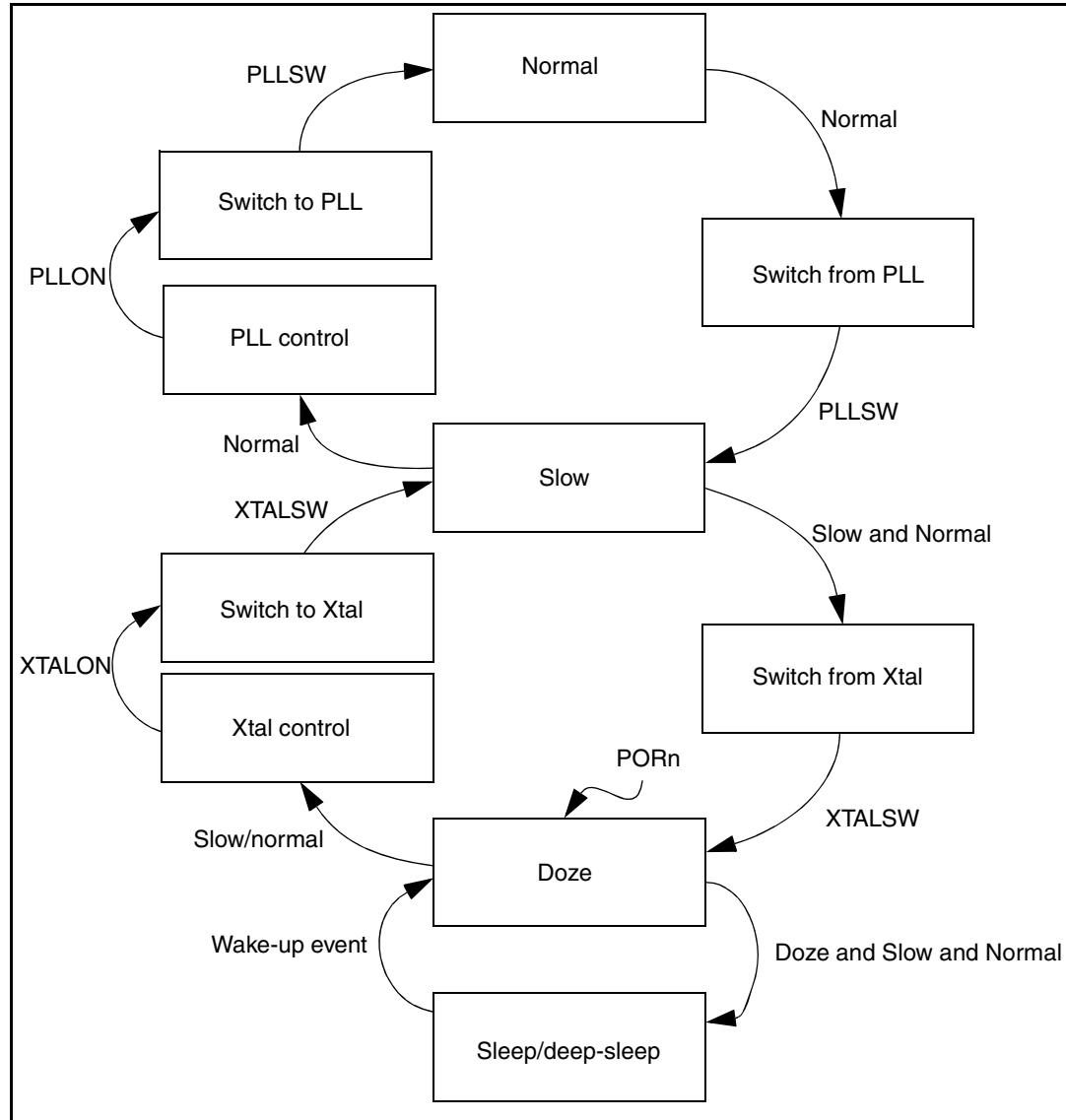
Table 9. System modes and clock sources

Mode	Clock sources	Comments
Normal	In normal mode, both of the system clocks (CLK and HCLK) and the system controller clock (SCLK) are driven from the output of PLL1.	
Slow	In slow mode, both of the system clocks (CLK and HCLK) and the system controller clock (SCLK) are driven from the 19.2 MHz input clock.	When in slow mode, if normal mode is required, the system moves into the PLL control transition state (see Section 5.2 on page 84).
Doze	In doze mode, both of the system clocks (CLK and HCLK) and the system controller clock (SCLK) are driven from the low frequency oscillator (32 768 Hz).	From doze mode, it is possible to move directly into sleep mode. If slow mode or normal mode is required, the system moves into the XTAL control transition state to initialize the 19.2 MHz crystal oscillator. The system mode control state machine enters doze mode following a power-on reset (PORn pin).
Sleep/ deep- sleep	In sleep/deep-sleep mode, both of the system clocks (HCLK and CLK) are disabled and the system controller clock (SCLK) is driven from the low frequency oscillator (32 768 Hz).	When a FIQ or an IRQ interrupt is activated through the vectored interrupt controller (VIC), the system moves into doze mode by default. The differentiation between sleep mode (logic power supply switched on) and deep-sleep mode (logic power supply internally switched off by embedded 1.2 V voltage switch) is performed by the power management unit (PMU).

5.2 System mode control state machine

The possible state transitions of the system mode control state machine are illustrated in [Figure 5](#).

Figure 5. System mode control state machine



- Note:**
- 1 If the power-on reset (PORn) pin is activated, the state machine and the required operating mode are set to doze.
 - 2 The system can be configured such that the mode-control settings are overridden when an interrupt is generated by the VIC.

The intermediate transition states in the above figure are described in [Table 10](#).

Table 10. Transition states

Transition state	Clock sources	Next state
Xtal control	This transition state is used to initialize the 19.2 MHz crystal oscillator. While in this state, both of the system clocks (CLK and HCLK) and the system controller clock (SCLK) are driven from the low frequency oscillator (32.768 kHz).	Once the crystal oscillator output is stable, the system moves into the switch to XTAL transition state.
Switch to Xtal	This transition state is used to initiate the switching of the system clock source from the low frequency oscillator to the crystal oscillator.	Once clock switching is complete, the system moves into slow mode.
Switch from Xtal	This transition state is entered when moving from slow mode to doze mode. It initiates the switching of the system clock source from the 19.2 MHz input clock to the low frequency oscillator (32.768 kHz).	Once clock switching is complete, the system moves into doze mode.
PLL control	This transition state is used to initialize PLL1. In this mode, both of the system clocks (CLK and HCLK) and the system controller clock (SCLK) are driven from the output of the 19.2 MHz input clock.	Once PLL1 has been initialized, the system moves into the switch to PLL transition state.
Switch to PLL	This transition state is used to initiate the switching of the system clock source from the 19.2 MHz input clock to the PLL1 output.	Once clock switching is complete, the system moves into normal mode.
Switch from PLL	This transition state is entered when moving from normal mode to slow mode. It initiates the switching of the system clock source from the PLL to the crystal oscillator output.	Once clock switching is complete, the system moves into slow mode.

5.3 Crystal oscillator and PLL control

The system control state machine can also be used to enable and disable the 19.2 MHz crystal oscillator and PLL1. The mappings of the operating modes to the crystal oscillator and PLL control signals are shown in [Table 11](#).

Table 11. Crystal and PLL control

Operating mode	Crystal enable (XTALEN)	Crystal switch (XTALRQSW)	PLL enable (PLLEN)	PLL switch (PLLRQSW)	Sleep mode
Normal	1	1	1	1	0
Switch to PLL	1	1	1	1	0
PLL control	1	1	1	0	0
Switch from PLL	1	1	1	0	0
Slow	1	1	0	0	0
Switch to XTAL	1	1	0	0	0
XTAL control	1	0	0	0	0
Switch from XTAL	1	0	0	0	0

Table 11. Crystal and PLL control

Operating mode	Crystal enable (XTALEN)	Crystal switch (XTALRQSW)	PLL enable (PLLEN)	PLL switch (PLLRQSW)	Sleep mode
Doze	0	0	0	0	0
Sleep	0	0	0	0	1

Control of the crystal and PLL1 can be overridden in software.

5.4 PLL1 frequency control

The clock generated by PLL1 (PLL1CLK) is derived from the MXTALI signal (the MXTALCLK clock). The frequency of PLL1CLK can be specified by means of two values:

- the feedback factor (PLL1NMUL),
- the post-divider (PLL1PDIV).

The PLLCLK frequency is then given by the formula:

$$\text{Freq (PLL1CLK)} = \text{Freq (MXTALI)} \times (\text{PLL1NMUL}+2) / 2^{\text{PLL1PDIV}}$$

Note: The PLL frequency control function is only reset when the power-on reset (PORn) input is active.

5.4.1 Restrictions on changing the PLL1 frequency

When changing the PLL1 output frequency while the system is in normal mode, the following procedure must be followed:

1. Request the system controller to enter slow mode.
2. Once slow mode has been entered, the PLL is disabled (PLL1EN is low). Its configuration can now be safely changed.
3. Once the change has been made, request the system controller to re-enter normal mode.

5.4.2 PLL unlocking

When PLL1 becomes unlocked, and the system is in normal operating mode, the system can be configured to:

- Remain in normal mode, and not generate an interrupt.
- Automatically enter slow mode (the system is clocked by the MXTAL clock), set the PLL1 interrupt status flag, and send an interrupt to the VIC. (PLL1 remains enabled.)

The software must then clear the PLL1 interrupt status flag. The system then reverts to *normal* mode when the PLL1 locks again.

5.4.3 Restrictions on the PLL1 output frequency

The VCO (voltage controlled oscillator) of PLL1 operates in the frequency range 600 MHz to 1280 MHz. The CPU system operates in the frequency range 0 to 264 MHz. PLL1NMUL and PLL1PDIV must be programmed such that the VCO output clock and the CPU clock

both operate in their respective frequency ranges. Thus, the following two restrictions must always be respected:

$$600 \text{ MHz} \leq \text{Freq (VCO1CLK)} = \text{Freq (MXTALCLK)} \times (\text{PLL1NMUL} + 2) \leq 1280 \text{ MHz}$$

$$\text{Normal: Freq (PLL1CLK)} = \text{Freq (CLK)} = \text{Freq (VCO1CLK)} / (2^{\text{PLL1PDIV}}) \leq 64 \text{ MHz}$$

$$\text{Overdrive: Freq (PLL1CLK)} = \text{Freq (CLK)} = \text{Freq (VCO1CLK)} / (2^{\text{PLL1PDIV}}) \leq 332 \text{ MHz}$$

The tables below give example frequencies corresponding to different values of PLL1NMUL and PLL1PDIV.

- *Table 12* assumes an MXTALCLK frequency of 19.2 MHz.
- *Table 13* assumes an MXTALCLK frequency of 13.0 MHz.

The cells with a grey background correspond to forbidden frequencies.

Table 12. Example of PLL1CLK frequencies (for MXTALCLK = 19.2 MHz)

PLL1CLK values	PLL1PDIV					
	0	1	2	3	4	5
63 = 0x3F	1248.0 MHz	624.0 MHz	312.0 MHz	156.0 MHz	78.0 MHz	39.0 MHz
62 = 0x3E	1228.8 MHz	614.4 MHz	307.2 MHz	153.6 MHz	76.8 MHz	38.4 MHz
61 = 0x3D	1209.6 MHz	604.8 MHz	302.4 MHz	151.2 MHz	75.6 MHz	37.8 MHz
60 = 0x3C	1190.4 MHz	595.2 MHz	297.6 MHz	148.8 MHz	74.4 MHz	37.2 MHz
59 = 0x3B	1171.2 MHz	585.6 MHz	292.8 MHz	146.4 MHz	73.2 MHz	36.6 MHz
58 = 0x3A	1152.0 MHz	576.0 MHz	288.0 MHz	144.0 MHz	72.0 MHz	36.0 MHz
57 = 0x39	1132.8 MHz	566.4 MHz	283.2 MHz	141.6 MHz	70.8 MHz	35.4 MHz
56 = 0x38	1113.6 MHz	556.8 MHz	278.4 MHz	139.2 MHz	69.6 MHz	34.8 MHz
55 = 0x37	1094.4 MHz	547.2 MHz	273.6 MHz	136.8 MHz	68.4 MHz	34.2 MHz
54 = 0x36	1075.2 MHz	537.6 MHz	268.8 MHz	134.4 MHz	67.2 MHz	33.6 MHz
53 = 0x35	1056.0 MHz	528.0 MHz	264.0 MHz	132.0 MHz	66.0 MHz	33.0 MHz
52 = 0x34	1036.8 MHz	518.4 MHz	259.2 MHz	129.6 MHz	64.8 MHz	32.4 MHz
51 = 0x33	1017.6 MHz	508.8 MHz	254.4 MHz	127.2 MHz	63.6 MHz	31.8 MHz
50 = 0x32	998.4 MHz	499.2 MHz	249.6 MHz	124.8 MHz	62.4 MHz	31.2 MHz
49 = 0x31	979.2 MHz	489.6 MHz	244.8 MHz	122.4 MHz	61.2 MHz	30.6 MHz
48 = 0x30	960.0 MHz	480.0 MHz	240.0 MHz	120.0 MHz	60.0 MHz	30.0 MHz
47 = 0x2F	940.8 MHz	470.4 MHz	235.2 MHz	117.6 MHz	58.8 MHz	29.4 MHz
46 = 0x2E	921.6 MHz	460.8 MHz	230.4 MHz	115.2 MHz	57.6 MHz	28.8 MHz
45 = 0x2D	902.4 MHz	451.2 MHz	225.6 MHz	112.8 MHz	56.4 MHz	28.2 MHz
44 = 0x2C	883.2 MHz	441.6 MHz	220.8 MHz	110.4 MHz	55.2 MHz	27.6 MHz
43 = 0x2B	864.0 MHz	432.0 MHz	216.0 MHz	108.0 MHz	54.0 MHz	27.0 MHz
42 = 0x2A	844.8 MHz	422.4 MHz	211.2 MHz	105.6 MHz	52.8 MHz	26.4 MHz
41 = 0x29	825.6 MHz	412.8 MHz	206.4 MHz	103.2 MHz	51.6 MHz	25.8 MHz

Table 12. Example of PLL1CLK frequencies (for MXTALCLK = 19.2 MHz)

PLL1CLK values	PLL1PDIV					
	0	1	2	3	4	5
40 = 0x28	806.4 MHz	403.2 MHz	201.6 MHz	100.8 MHz	50.4 MHz	25.2 MHz
39 = 0x27	787.2 MHz	393.6 MHz	196.8 MHz	98.4 MHz	49.2 MHz	24.6 MHz
38 = 0x26	768.0 MHz	384.0 MHz	192.0 MHz	96.0 MHz	48.0 MHz	24.0 MHz
37 = 0x25	748.8 MHz	374.4 MHz	187.2 MHz	93.6 MHz	46.8 MHz	23.4 MHz
36 = 0x24	729.6 MHz	364.8 MHz	182.4 MHz	91.2 MHz	45.6 MHz	22.8 MHz
35 = 0x23	710.4 MHz	355.2 MHz	177.6 MHz	88.8 MHz	44.4 MHz	22.2 MHz
34 = 0x22	691.2 MHz	345.6 MHz	172.8 MHz	86.4 MHz	43.2 MHz	21.6 MHz
33 = 0x21	672.0 MHz	336.0 MHz	168.0 MHz	84.0 MHz	42.0 MHz	21.0 MHz
32 = 0x20	652.8 MHz	326.4 MHz	163.2 MHz	81.6 MHz	40.8 MHz	20.4 MHz
31 = 0x1F	633.8 MHz	316.8 MHz	158.4 MHz	79.2 MHz	39.6 MHz	19.8 MHz
30 = 0x1E	614.4 MHz	307.2 MHz	153.6 MHz	76.8 MHz	38.4 MHz	19.2 MHz
29 to 0	VCO output frequency is too low: do not use these settings					

Table 13. Example of PLL1CLK frequencies (for MXTALCLK = 13.0 MHz)

PLL1CLK values	PLL1PDIV					
	0	1	2	3	4	5
63 = 0x3F	845.0 MHz	422.5 MHz	211.25 MHz	105.63 MHz	52.813 MHz	26.406 MHz
62 = 0x3E	832.0 MHz	416.0 MHz	208.00 MHz	104.00 MHz	52.000 MHz	26.000 MHz
61 = 0x3D	819.0 MHz	409.5 MHz	204.75 MHz	102.38 MHz	51.188 MHz	25.594 MHz
60 = 0x3C	806.0 MHz	403.0 MHz	201.50 MHz	100.75 MHz	50.375 MHz	25.188 MHz
59 = 0x3B	793.0 MHz	396.5 MHz	198.25 MHz	99.125 MHz	49.563 MHz	24.781 MHz
58 = 0x3A	780.0 MHz	390.0 MHz	195.00 MHz	97.500 MHz	48.750 MHz	24.375 MHz
57 = 0x39	767.0 MHz	383.5 MHz	191.75 MHz	95.875 MHz	47.938 MHz	23.969 MHz
56 = 0x38	754.0 MHz	377.0 MHz	188.50 MHz	94.250 MHz	47.125 MHz	23.563 MHz
55 = 0x37	741.0 MHz	370.5 MHz	185.25 MHz	92.625 MHz	46.313 MHz	23.156 MHz
54 = 0x36	728.0 MHz	364.0 MHz	182.00 MHz	91.000 MHz	45.500 MHz	22.750 MHz
53 = 0x35	715.0 MHz	357.5 MHz	178.75 MHz	89.375 MHz	44.688 MHz	22.344 MHz
52 = 0x34	702.0 MHz	351.0 MHz	175.50 MHz	87.750 MHz	43.875 MHz	21.938 MHz
51 = 0x33	689.0 MHz	344.5 MHz	172.25 MHz	86.125 MHz	43.063 MHz	21.531 MHz
50 = 0x32	676.0 MHz	338.0 MHz	169.00 MHz	84.500 MHz	42.250 MHz	21.125 MHz
49 = 0x31	663.0 MHz	331.5 MHz	165.75 MHz	82.875 MHz	41.438 MHz	20.719 MHz
48 = 0x30	650.0 MHz	325.0 MHz	162.50 MHz	81.250 MHz	40.625 MHz	20.313 MHz

Table 13. Example of PLL1CLK frequencies (for MXTALCLK = 13.0 MHz)

PLL1CLK values	PLL1PDIV					
	0	1	2	3	4	5
47 = 0x2F	637.0 MHz	318.5 MHz	159.25 MHz	79.625 MHz	39.813 MHz	19.906 MHz
46 = 0x2E	624.0 MHz	312.0 MHz	156.00 MHz	78.000 MHz	39.000 MHz	19.500 MHz
45 = 0x2D	611.0 MHz	305.5 MHz	152.75 MHz	76.375 MHz	38.188 MHz	19.094 MHz
44 to 0	VCO output frequency is too low: do not use these settings					

5.5 PLL2 frequency control

PLL2 delivers four clock signals used to drive the peripherals:

- CLK216 = 216 MHz or 208 MHz,
- CLK108 = CLK216 divided by 2 = 108 MHz or 104 MHz,
- CLK72 = 72 MHz or 78 MHz,
- CLK48 = 48 MHz.

These clocks are used by most of the device peripherals. Having a dedicated PLL for generating these peripheral clock frequencies allows the peripherals to continue running on their standard frequencies without having to reprogram their internal clock pre-scaler when the PLL1 output frequency is changed (or even if the clock is stopped).

In order to adapt to various MXTALI crystal or external signal frequencies, PLL2 can be programmed by defining the PLL feedback divider (PLL2NMUL) and the PLL post-dividers (PLL2PDIV). [Table 14](#) gives the correct PLL2 configuration to reach the correct peripheral frequencies for several standard MXTALI frequencies.

Table 14. PLL2 configurations for different MXTALI frequencies

MXTALI	PLL2NMUL	PLL2CLK	PLL2PDIV	CLK216	CLK108	CLK72	CLK48
13 MHz	0x2E	624 MHz	0	208 MHz	104 MHz	78 MHz	48 MHz
16 MHz	0x34	864 MHz	1	216 MHz	108 MHz	72 MHz	48 MHz
19.2 MHz	0x2B	864 MHz	1	216 MHz	108 MHz	72 MHz	48 MHz
24 MHz	0x22	864 MHz	1	216 MHz	108 MHz	72 MHz	48 MHz
26 MHz	0x16	624 MHz	0	208 MHz	104 MHz	78 MHz	48 MHz

The PLL2 frequency control function is only reset when the PORn input is active.

5.6 PLL1 and PLL2 interrupts

The PLL1and PLL2 interrupt status flags are combined with the PMU VDDOK and BATOK interrupt signals before being sent to the vectored interrupt controller (VIC) input line 16.

If a PLL unlock occurs when the PLL1 interrupt is enabled (the masked-interrupt status bit is set) and the system is in normal mode, an interrupt is sent to the VIC, and the system automatically enters slow mode. PLL1 remains enabled.

The software must clear the PLL1 interrupt status flag to clear interrupt on the VIC. Once this is cleared, the system goes back to normal mode when PLL1 locks again.

When the PLL2 interrupt is enabled, the PLL2 masked interrupt status bit is set when PLL2 unlocks. PLL2 remains enabled, and continues delivering peripheral clocks. The software must clear the PLL2 interrupt status bit to clear the interrupt signal to the VIC.

5.7 Interrupt response mode

The interrupt response mode allows interrupts to be responded to in the most appropriate way. For example, it enables the state machine to move from doze mode to normal mode following an interrupt.

The interrupt response function is configured by defining:

- if the interrupt response mode is enabled,
- the mode of operation that is required following an interrupt,
- the type of interrupt that is permitted in interrupt response mode,
- an interrupt clear mechanism.

It is not possible for the interrupt response mode to reduce the system's operating speed (for example, by changing mode from normal to slow).

To further improve the system's interrupt response time, the ARM instruction and data AHB priority may be increased through the AHB slave multiplexers when in interrupt response mode.

Following a power-on reset, the interrupt response mode is disabled.

5.8 Wait-for-interrupt control

When entering sleep mode, it is advisable to activate wait-for-interrupt on the ARM926EJ-S CPU system control coprocessor. This ensures that the CPU is in a low power state. Wait-for-interrupt must also be used whenever short pauses occur in processing requirements (the system mode control state machine can remain in the same state).

5.9 HCLK to CLK relationship

The system clocks, HCLK and CLK, are synchronous.

- In normal mode, HCLK can be configured as a divided down version of CLK. The possible division factors are 1, 2, 3 and 4.
- In modes other than normal, HCLK and CLK always have the same frequency.

Note: The duty cycle of HCLK is 50%, irrespective of the division factor used.

5.10 SDRAM refresh rates

The SDRAM refresh period is programmed into the SDRAM memory controller (SDMC) in HCLK tick units. This setting must be reprogrammed when the operating frequency of the subsystem changes (when the PLL frequency control is altered or if the system is switched between normal and slow operating modes).

When moving to:

- a higher HCLK frequency, the refresh period must be updated after the transition,
- a lower HCLK frequency, the refresh period must be updated (to a setting suitable for the target operating frequency) before the transition.

Before entering operating modes in which the HCLK frequency is very low or stalled (doze and sleep modes), the SDRAM must be put into self-refresh mode. In this mode, the SDRAM is not accessible.

Note: *The correct shutdown sequence must be applied to other peripherals before the system can be put into doze or sleep mode.*

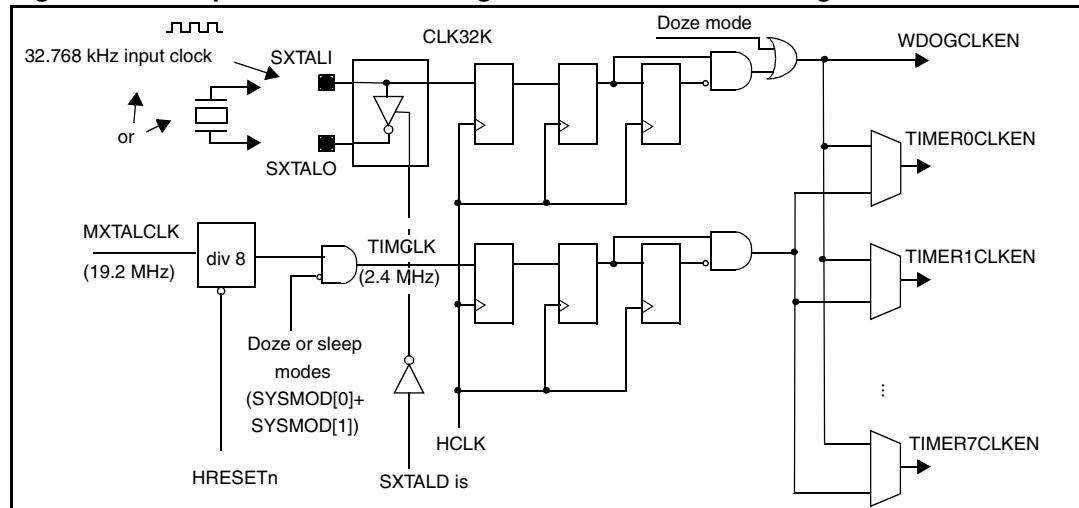
5.11 Watchdog and timer clock enable signals

The watchdog and timers have clock enable signals that are generated synchronously to HCLK by the system controller.

- Watchdog enable signal: WDOGCLKEN
The watchdog enable signal is generated from the CLK32K input (32.768 kHz oscillator output) and a single WDOGCLKEN pulse is generated on every rising edge of CLK32K.
- Timer clock enable signals: TIMER[7:0]CLKEN
The timer clock enable signal can be selectively generated from the CLK32K or TIMCLK (MXTALCLK divided by 8) clock inputs.

To allow the timer and watchdog enable signals to be generated at a constant rate independently of the system clock, the enable signals are derived from asynchronous, fixed-rate clock inputs. This is shown for WDOGCLKEN and TIMER[7:0]CLKEN in [Figure 6](#).

Figure 6. Frequencies for watchdog and timer clock enable signals



Since the watchdog and timers are clocked by HCLK, they are stopped in sleep mode.

Note the following points about the clocks and timers when the system is in doze mode.

- The HCLK is clocked by the 32.768 kHz oscillator output (CLK32K). This means that in doze mode WDOGCLKEN and TIMER[7:0]CLKEN run at 32.768 kHz (although TIMER[7:0]CLKEN must be configured to do so).
- The MXTALCLK signal can be stopped (default) or running (as configured).
- To prevent some timers from incrementing at an unpredictable rate, TIMCLK is stopped when doze mode is entered.

Note: Timers and watchdog are stalled when the ARM processor is in debug mode.

5.12 CLKOUT0/1 programmable clock signals

The SRC can deliver two programmable clock signals, CLKOUT0 and CLKOUT1, on the GPIO25/GPIO32/GPIO76 and GPIO55 pins respectively (when their alternate functions are enabled).

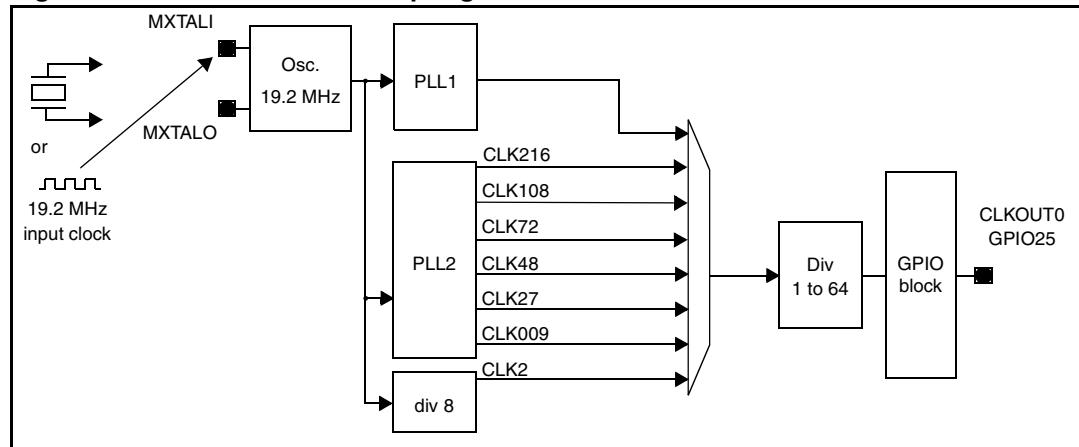
CLKOUT0

CLKOUT0 can deliver one of 8 internal PLL clock signals, divided by a programmable divisor. The clock sources are:

- Main clock signal input, MXTAL,
- PLL2 CLK216 clock output signal (216 MHz when MXTALI = 19.2 MHz, 208 MHz when MXTALI = 13 MHz),
- PLL2 CLK108 output (108 MHz if MXTALI = 19.2 MHz, 104 MHz if MXTALI = 13 MHz),
- PLL2 CLK72 output (72 MHz if MXTALI = 19.2 MHz, 78 MHz if MXTALI = 13 MHz),
- PLL2 CLK48 (48 MHz) output,
- PLL2 CLK27 clock output signal (27 MHz when MXTALI = 19.2 MHz, 26 MHz when MXTALI = 13 MHz),
- TIMCLK clock output signal ($MXTALI / 8 = 2.4$ MHz when MXTALI = 19.2 MHz, 1.625 MHz when MXTALI = 13 MHz),
- PLL2 CLK009 output signal ($CLK48 / 533 = 90.056$ kHz).

Figure 7 illustrates the generation of the CLKOUT0 signal, delivered on the GPIO25 pin.

Figure 7. CLKOUT0 clock output generation



When test mode is enabled (SCANMOD is high), CLKOUT0 always delivers TSTCLK divided by the programmed division factor, irrespective of the source clock that has been selected.

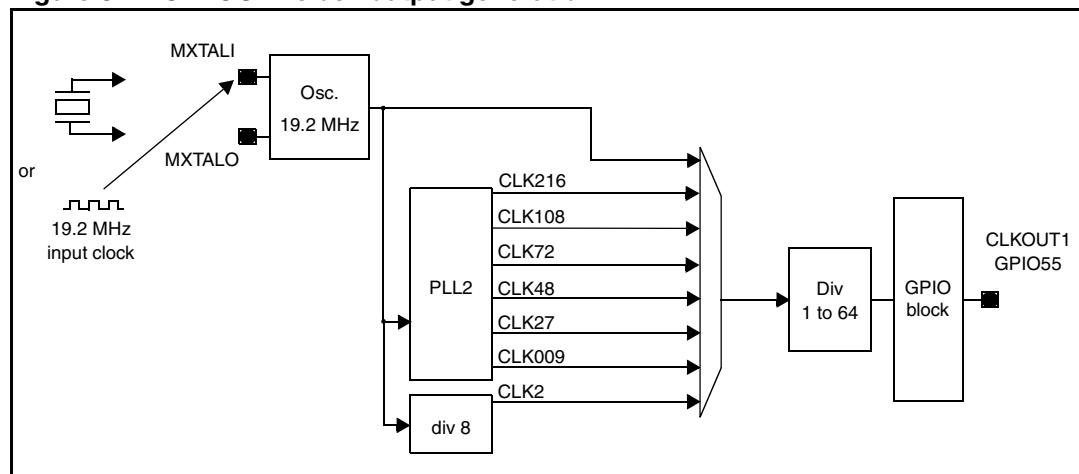
CLKOUT1

CLKOUT1 can deliver one of three internal PLL clock signals or the MXTAL clock signal, divided by a programmable divisor. The clock sources are:

- PLL1 clock output signal, PLL1CLK,
- PLL2 CLK216 clock output signal (216 MHz when MXTALI = 19.2 MHz, 208 MHz when MXTALI = 13 MHz),
- PLL2 CLK108 output (108 MHz if MXTALI = 19.2 MHz, 104 MHz if MXTALI = 13 MHz),
- PLL2 CLK72 output (72 MHz if MXTALI = 19.2 MHz, 78 MHz if MXTALI = 13 MHz),
- PLL2 CLK48 (48 MHz) output,
- PLL2 CLK27 clock output signal (27 MHz when MXTALI = 19.2 MHz, 26 MHz when MXTALI = 13 MHz),
- TIMCLK clock output signal (MXTALI /8 = 2.4 MHz when MXTALI = 19.2 MHz, 1.625 MHz when MXTALI = 13 MHz),
- PLL2 CLK009 output signal (CLK48 / 533 = 90.056 kHz).

Figure 8 illustrates the generation of the CLKOUT1 signal, delivered on the GPIO55 pin.

Figure 8. CLKOUT1 clock output generation



- Note:
- 1 *Changing the division factor or the selected clock source may produce a glitch on the delivered CLKOUT0 or CLKOUT1 clock signal.*
 - 2 *In DEEP-SLEEP mode, CLKOUT0 and CLKOUT1 are forced LOW whatever the setting of clock source selection, even if a signal is still present on MXTALI. In modes other than normal mode, PLL1 and PLL2 are off, thus CLKOUT0 and CLKOUT1 will not toggle if the clock source selected is delivered by these PLLs.*

5.13 Clocks and power supplies

This section summarizes the clocks and power supplies for the different operating modes of the system.

5.14 Clocks

Table 15. Clock summary for different operating modes

Mode	Clock	Status	Affected modules
Sleep	HCLK	Stopped.	AHB/APB clocks, VIC, timers, watchdog timer, DMA/FSMC/SDMC with enable
	CLK	Stopped.	ARM926EJ-S CPU, ETM9
	SCLK	Running at 32768 kHz.	System controller
	REFCLK	Running at 32768 kHz.	RTC
	CLK48	Stopped.	CLCD/I2Cs/SD-card/FIrDA /MSPs/SSP/UARTs/USB/RNG
	CLK72	Stopped.	CLCD/MDIF
Doze	HCLK	Running at 32.768 kHz.	AHB/APB clocks, VIC, timers, watchdog timer, DMA/FSMC/SDMC with enable
	CLK	Running at 32.768 kHz.	ARM926EJ-S core, ETM9
	SCLK	Running at 32.768 kHz.	System controller
	CLK32K	Running at 32.768 kHz.	RTC
	CLK48	Stopped (by default) or running at 48 MHz from PLL2 output if configured by software.	CLCD/I2Cs/SD-card/FIrDA/ MSPs/SSP/ UARTs/USB/RNG
	CLK72	Stopped (by default) or running at 72/78 MHz from PLL2 output if configured by software.	CLCD/MDIF
	CLK108	Stopped (by default) or running at 104/108 MHz from PLL2 output if configured by software.	
	CLK216	Stopped (by default) or running at 208/216 MHz from PLL2 output if configured by software.	

Table 15. Clock summary for different operating modes (continued)

Mode	Clock	Status	Affected modules
Slow	HCLK	Running at 19.2 MHz divided by programmed value (1, 2, 3 or 4).	AHB/APB clocks, VIC, timers, watchdog timer, DMA/FSMC/SDMC with enable
	CLK	Running at 19.2 MHz.	ARM926EJ-S core, ETM9
	SCLK	Running at 19.2 MHz.	System controller
	CLK32K	Running at 32.768 kHz.	RTC
	CLK48	Running at 48 MHz from PLL2 output.	CLCD/I2Cs/SD-card/ IrDA/MSPs/SSP/UARTs/USB/RNG
	CLK72	Running at 72/78 MHz from PLL2 output.	CLCD/MDIF
	CLK108	Running at 104/108 MHz from PLL2 output.	
	CLK216	Running at 208/216 MHz from PLL2 output.	
Normal	HCLK	Driven from PLL1 output divided by HCLK Div-Sel value (1, 2, 3 or 4).	AHB/APB clocks, VIC, timers, watchdog timer, DMA/FSMC/SDMC with enable
	CLK	Driven from PLL1 output.	ARM926EJ-S core, ETM9
	SCLK	Driven from PLL1 output.	System controller
	REFCLK	Running at 32768 kHz.	RTC
	CLK48	Running at 48 MHz from PLL2 output.	CLCD/I2Cs/SD-card/ IrDA/MSPs/SSP/UARTs/USB/RNG
	CLK72	Running at 72/78 MHz from PLL2 output.	CLCD/MDIF
	CLK108	Running at 104/108 MHz from PLL2 output.	
	CLK216	Running at 208/216 MHz from PLL2 output.	

5.15 Power supplies

Table 16. Power supply summary for different operating modes

Mode	Power	Voltage	Status	Affected modules
Doze	VDDIO	IO voltage	On	Pads
	VDD2 ⁽¹⁾	1.2 V	On	RTC, PWL, RTT, system and reset controller (SRC), power management unit (PMU), GPIOs
	VDD1 ⁽²⁾	1.2 V	On	ARM926 + all peripherals except RTC/VIC/system controller/GPIO

Table 16. Power supply summary for different operating modes

Mode	Power	Voltage	Status	Affected modules
Sleep/deep-sleep	VDDIO	IO voltage	On	PADs
	VDD2	1.2 V	On	RTC, PWL, RTT, system and reset controller (SRC), power management unit (PMU), GPIOs
	VDD1	1.2 V	On or off (default)	ARM926 + all peripherals except RTC/VIC/system controller(GPIO)
Slow	VDDIO	IO voltage	On	PADs
	VDD2	1.2 V	On	RTC, PWL, RTT, System and reset controller (SRC), power management unit (PMU), GPIOs,
	VDD1	1.2 V	On	ARM926EJ-S + all peripherals except RTC/VIC/system controller(GPIO)
Normal	VDDIO	IO voltage	On	PADs
	VDD2	1.2 V	On	RTC, PWL, RTT, system and reset controller (SRC), power management unit (PMU), GPIOs
	VDD1	1.2 V	On	ARM926 + all peripherals except RTC/VIC/system controller(GPIO)

1. VDD2 is the internal 1.2 V supply voltage domain that is not switched off by the 1.2 V embedded voltage switch.
2. VDD1 is the core logic 1.2 V supply voltage that is automatically switched off by the embedded 1.2 V voltage switch under the control of the power management unit (PMU).

5.16 Reset controller

The internal system reset function initializes the SRC into a defined default state and is invoked by asserting one of four reset sources. These are:

- power-on reset,
- software reset,
- watchdog reset.

5.16.1 Power-on reset

This reset is invoked by the active-low signal PORn and is intended to be used during power-up only. All DRAM and RTC contents are lost during such a power-on reset.

5.16.2 Software reset

This reset is invoked by writing software and performs a system reset.

5.16.3 Watchdog reset

This reset is invoked when the watchdog reset generation is enabled and the watchdog timer is not serviced by software. It performs a system reset.

On coming out of a reset state, the platform enters doze mode (see [Section 5.2 on page 84](#)).

The type of reset that caused the last reset condition can be determined in software.

[Table 17](#) shows the signals used by the SRC.

Table 17. System and reset controller signals

Signal	I/O	Description
MXTALI	I	Main crystal input or external oscillator input.
MXTALO	O	Main crystal oscillator output
SXTALI	I	32 kHz RTC crystal input
SXTALO	O	32 kHz RTC crystal output
PORn	I	System power-up reset. Glitches of less than 50 ns are filtered.
CLKOUT0	O	Programmable clock output 0 signal.
CLKOUT1	O	Clock output 1: programmable clock output 1 signal.
CLK32K	O	32 kHz raw clock output, the 32 kHz clock signal.
DBGCFG	I	MM-DSP debugger TAP source configuration:

6 Power management unit (PMU)

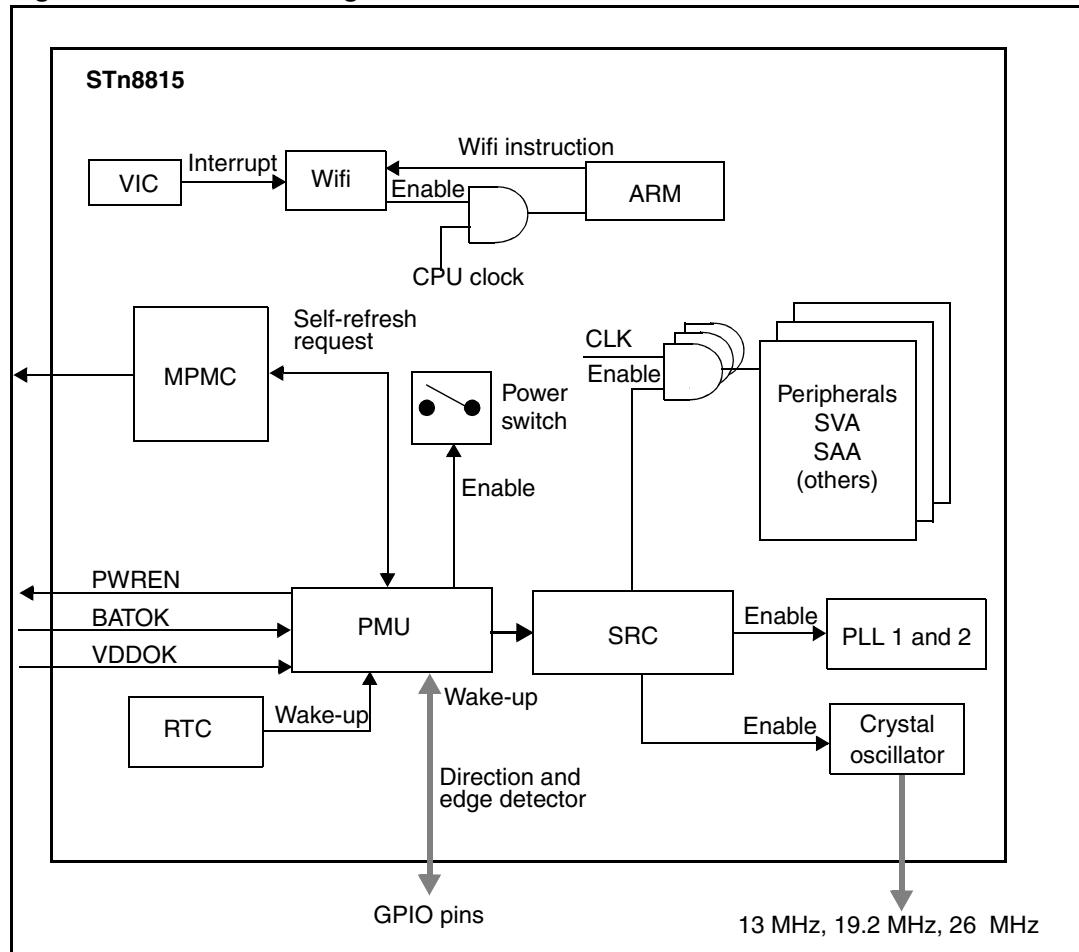
The PMU manages the power supply to the STn8815A12 package. The goal of the power-management strategy is to reduce to a minimum, the static and dynamic power consumption of the STn8815 device.

6.1 PMU features

The power management unit (PMU) controls the following tasks:

- transition from sleep to deep-sleep mode
- wake-up from deep-sleep mode
- embedded VDD12 voltage switch control
- external power-supply monitoring
- self-refresh mode entry (forced) of the SDRAM

Figure 9. PMU block diagram



6.2 Power-supply domains

The STn8815A12 has two separate 1.2 V supply domains, VDD1 and VDD2.

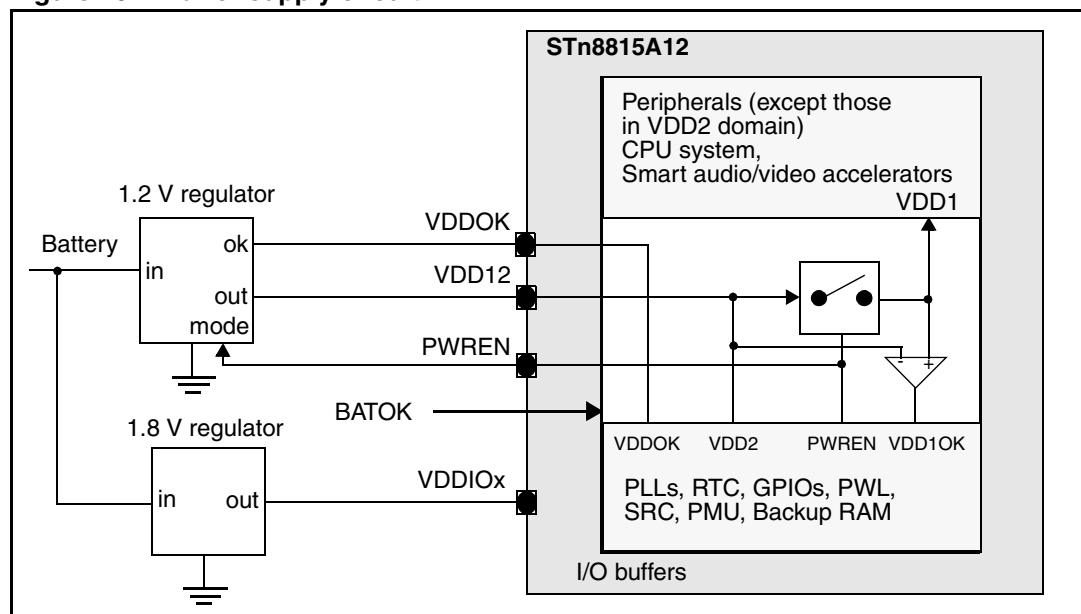
The PMU manages the external voltage regulator. It switches off the logic that is not needed in deep-sleep mode, while keeping the minimum necessary logic active to wake the system. The PMU also controls the embedded 1.2 V voltage switches that switch off the logic supply after the device has entered sleep mode. In deep-sleep mode, the chip logic gates and memories are no longer supplied, except the pins, GPIOs, RTC, system controller, power management unit and certain RAMs.

The VDD1 supply domain may be switched off in sleep mode. This reduces power consumption by avoiding current leakage from most of the device's transistors. The device has an embedded 1.2 V power switch so that it can be supplied by a single voltage regulator. This regulator always delivers the 1.2 V supply voltage, though it switches the delivered current according to the STn8815A12 state (deep-sleep or running), under the control of the PWREN signal.

Note: The VDD18 power pins must always remain supplied by an external 1.8 V voltage regulator.

Figure 10 shows the required external power-supply circuit.

Figure 10. Power supply circuit



The following blocks of the STn8815A12 remain powered under the VDD2 domain in deep-sleep mode:

- real-time clock (RTC)
- pulse width light modulator (PWL)
- GPIOs
- power management unit (PMU)
- system and reset controller (SRC)
- backup RAM (1 Kbyte)
- PLLs

The VDD1 domain supplies all other blocks:

- CPU system (including MMU and cache memories)
- all peripherals except those listed above
- smart audio and video accelerators (SAA and SVA)
- embedded memories (SRAM, ROM)

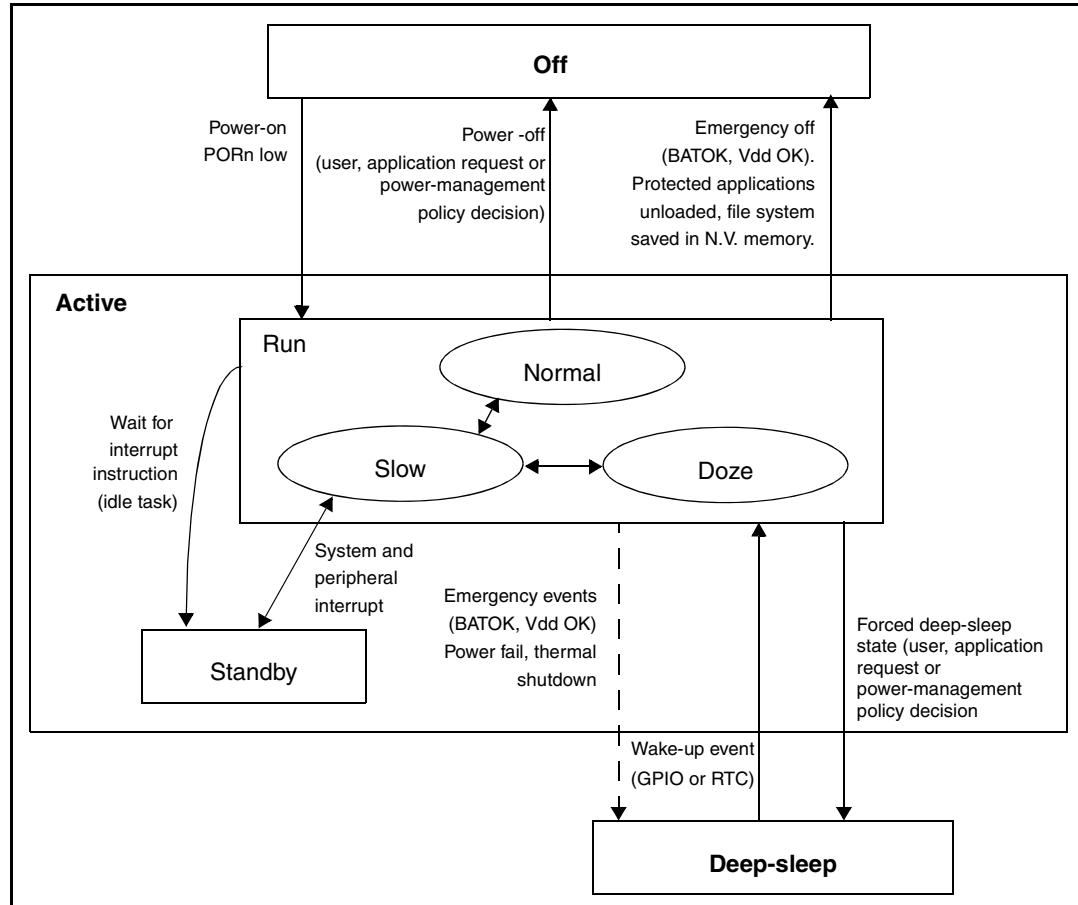
6.3 Power-state switching

The STn8815A12 power-management strategy is based on three system power states.

- **Off:** There is no system power, and the SDRAM data is lost.
- **Active:** The system is running in one of the following modes:
 - run (normal, slow or doze)
 - standby
- **Deep-sleep/sleep:** The lowest power state where the system context is saved in SDRAM.

Figure 11 shows the possible state transitions between these modes.

Figure 11. Power state diagram



6.3.1 Sleep / deep-sleep mode entry

Sleep mode may be entered in two ways:

- by the CPU, generally after a user request to stop the application processor, or when pin VDDOK or BATOK has generated an interrupt.
- by the PMU when BATOK or VDDOK are asserted low. In this case, the PMU forces the SDRAM to self-refresh.

The system controller state machine goes from the current mode (normal, slow or doze) to sleep mode. It can also switch off the PLL1 and the MXTAL oscillator. At this point:

1. The PMU forces the GPIOs to input with pull-up/down enabled, or (keeping the current direction/level), enables the clock for its GPIO edge detectors and resets the GPIO wake-up flag. For pins other than GPIOs, defined levels and directions are applied: see [Section 3 on page 30](#). In the case of a hardware request, the PMU sends a concurrent self-refresh request to the SDRAM controller.
2. After two clock cycles, the GPIO wake-up flag reset is released. This prevents an immediate wake-up due to GPIOs switching to their sleep level (pull-up and pull-down are less than 100 kohms, so with a 100 pF load, it takes less than 10 µs to reach the sleep level) and masks initialization of edge detectors.
3. As soon as the PMU receives the self-refresh acknowledgement from the SDRAM controller, clocks (except SCLK) are stopped and a power-down request is sent to the external regulator if deep-sleep mode is enabled. The whole system is then in sleep mode.
4. If deep-sleep mode is not enabled, the device remains in sleep mode. If deep-sleep mode is enabled, the PMU automatically moves the device to deep-sleep state by turning off the embedded voltage switch placed on VDD12. (See [Section 1](#) for description of the embedded voltage switch.)

To indicate during the wake-up phase, the reason for which the chip was put in sleep mode, the software, VDD failure and battery fault status flags are updated before entering sleep mode.

6.3.2 Sleep / deep-sleep mode

During sleep/deep-sleep mode, no external intervention except a reset (PORn signal going low), or a preprogrammed wake-up event (transition on an enabled GPIO pin, or an RTC alarm) may perturb this state.

6.3.3 Deep-sleep mode exit

In order to exit from sleep mode, the PMU must receive a valid wake-up event. This can be the alarm request coming from the RTC, or the activation of one of the enabled GPIOs. If the pin BATOK is high, the system is allowed to exit from deep-sleep mode. Otherwise, the wake-up demand is ignored and not memorized.

The PMU first requests the embedded voltage switch to turn on the VDD1 1.2 V supply (if deep-sleep was enabled before the sleep request). This is done by asserting PWREN high. Then the PMU waits until the voltage supply is present and stabilized from the external regulator (VDDOK high), or waits for Tempo to be completed if Tempo feature is enabled, and then it waits for the switched on voltage domain (using an embedded voltage comparator) to be stabilized. A warm reset is also sent in order to correctly initialize the modules which were not supplied during deep-sleep mode. In the meantime, clocks are restarted.

At this time, the system is able to move from the deep-sleep to doze state. This is possible by sending an interrupt to the system and reset controller (wake-up signal). Then, depending on whether interrupt mode is enabled, the system moves from doze to the state predefined in the system and reset controller (SRC).

Note that the SDRAM must also exit from self-refresh mode. In order to achieve this, the SDRAM controller must be reloaded by software, as it is not supplied during sleep mode. Then the PMU must release the GPIOs and SDRAM lines.

6.3.4 Sleep mode exit

In order to exit from sleep mode, the PMU must receive a valid wake-up event. This can be the alarm request coming from the RTC, or the activation of one of the enabled GPIOs (with minimum activation time of 31 µs). If the BATOK pin is high, the system is allowed to exit from sleep mode. Otherwise, the wake-up demand is ignored.

Clocks are restarted, and the system is able to move from sleep to doze state. This is set by sending a wake-up signal to the system and reset controller (SRC). Then, depending on whether interrupt mode is enabled or not, the system moves from doze to the state predefined in the SRC. Then the PMU must release the GPIOs and SDRAM lines.

6.3.5 Reviving DRAMs from self-refresh mode

Since the DRAMs are placed in self-refresh prior to the sleep mode shutdown, their contents are preserved during sleep. After exiting from deep-sleep mode, software must reconfigure the SDRAM memory controller (SDMC), and then take the SDRAMs out of self-refresh mode. Clearing the SDRAM controller output hold in the power management unit causes pins SDRRASn, SDRCASn, SDRCSn[1:0] and SDRDQML/U to be controlled by the SDMC, which after exit from deep-sleep, drives these pins to the inactive state (high) in preparation for a DRAM access.

In addition, bringing SDRAM out of self-refresh requires that the SDMC is re-initialized, and that it is transferred from a self-refresh and clock-stop state to an idle state by enabling the SDRAM clock.

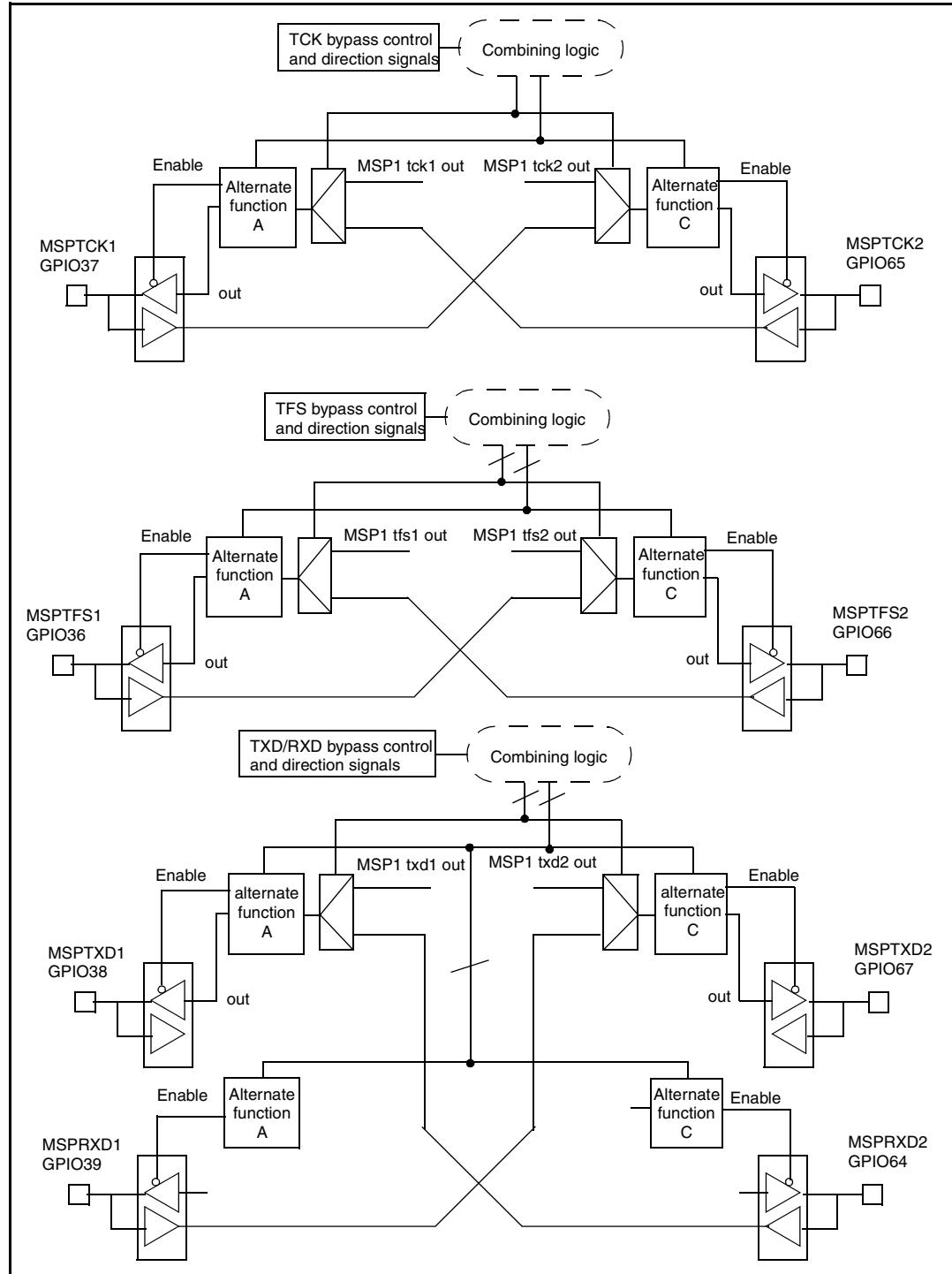
Note:

When PWREN goes from low to high (exit from deep-sleep), the reset signals of the blocks on the VDD1 supply domains become active, thus forcing the corresponding blocks to their default states (same default states as after an external power-on reset), and are released when VDDOK and VDD1OK (output of embedded voltage comparator) are both detected as high.

6.4 MSP bypass

STN8815 PMU supports bypass between MSP1 and MSP2 in deep-sleep mode, as shown in [Figure 12](#).

Figure 12. MSP bypass coupling



In MSP bypass mode, Rx and Tx pads have fixed directions, and are mapped as:

- Tx1 <-> Rx2
- Tx2 <-> Rx1

TFS and TCK pin directions are configurable in bypass mode to support coupling in all possible directions:

- TFS1 to TFS2 or TFS2 to TFS1
- TCK1 to TCK2 or TCK2 to TCK1

Note: The HiZ dynamic configuration of all these IOs is not supported in bypass, (in bypass mode, IOs cannot enter HiZ mode dynamically).

6.5 Tempo feature

Figure 13 shows the behavior of the tempo feature. VDDOK is tied high externally (on board), and WKUP_TEMPO is programmed to 4 (5 cycles tempo).

Figure 13. Tempo feature timing sequence

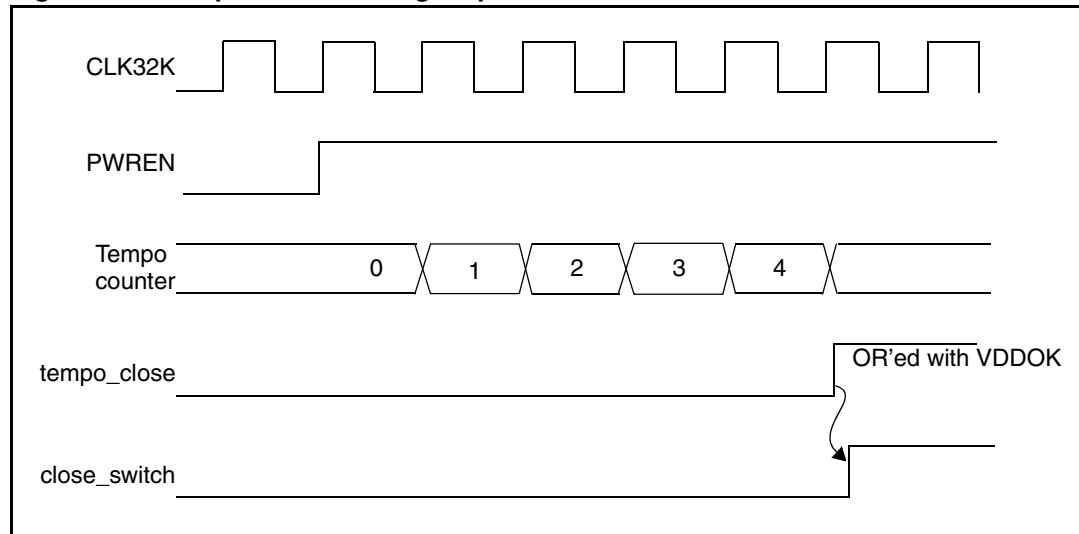
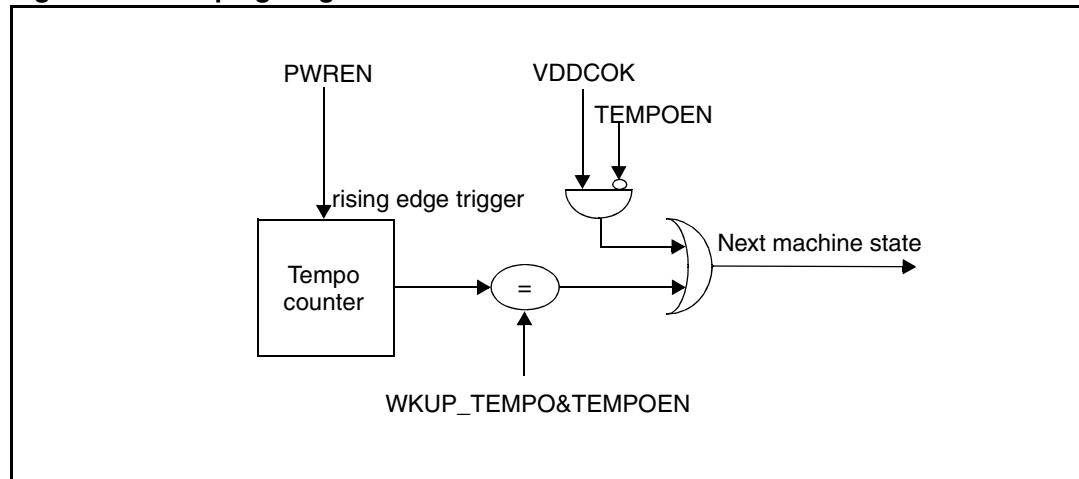


Figure 14. Tempo gating



6.6 Bypassing PLL2 and subsequent dividers

The internal BYPPLL2, BYPDIV1and BYPDIV2 flags allow the PLL2 clock and the dividers that generate the 48 and 72-78 MHz clock to be bypassed. The MXTAL clock (typically 19.2 MHz) is then sent to the peripherals usually clocked by the 48 MHz and 72 to 78 MHz clocks.

Note: *By default (after a PORn signal), all bypass control bits are set to 0, (bypass is not active).*

6.7 PMU signals

Table 18 summarizes the PMU signals.

Table 18. Power management unit signals

Signal	I/O	Description
BATOK	I	Battery OK signal.
VDDOK	I	VDD OK signal.
PWREN	O	Power-enable output:.

Table 19 summarizes the PMU power pins.

Table 19. Power pins

Signal	Type	Description
VDDIOA	PWR	Power supply pins for the IO buffers.
VDDIOB		
VDDIOC		
VDDIOD		
VDDIOE		
VDDIOF		
VDD12	PWR	Power supply pins for the internal logic (1.2 V).
VDDA	PWR	PLL analog supply, 1.8 V +/-5% with noise < 50 mV amplitude.
GNDA		PLL analog ground: must follow VSS voltage (+/-0.1 V).
VOTP		2.5 V +/- 10% voltage supply for OTP polarization (no current).
RCOMPSSL	Ref	Pad compensation for SubLVDS.
RCOMP		Pad compensation.
DEC1	-	Decoupling of internal 1.2 V logic supply.
DEC2	-	
DEC3	-	
DEC4	-	

7 ARM926EJ processor

The CPU of the STn8815A12 platform is an ARM926EJ® reduced instruction set computer (RISC) processor.

This chapter outlines the ARM926EJ processor. A complete description of the ARM926EJ core can be found in the ARM literature listed below (available at www.arm.com).

- *ARM Architecture Reference Manual* (ARM DDI 0100)
- *ARM9EJS Technical Reference Manual* (ARM DDI 0199)
- *ARM926EJS Technical Reference Manual* (ARM DDI 0198)

The ARM926EJ is a 32-bit processor core that supports 32-bit ARM and 16-bit Thumb instruction sets (architecture version 5TEJ), allowing a balance between high performance and high code density. The processor includes features for the efficient execution of Java byte codes, providing Java performance similar to JIT, but without the associated code overhead. It also supports the ARM debug architecture and includes logic to assist in both hardware and software debug.

7.1 Features

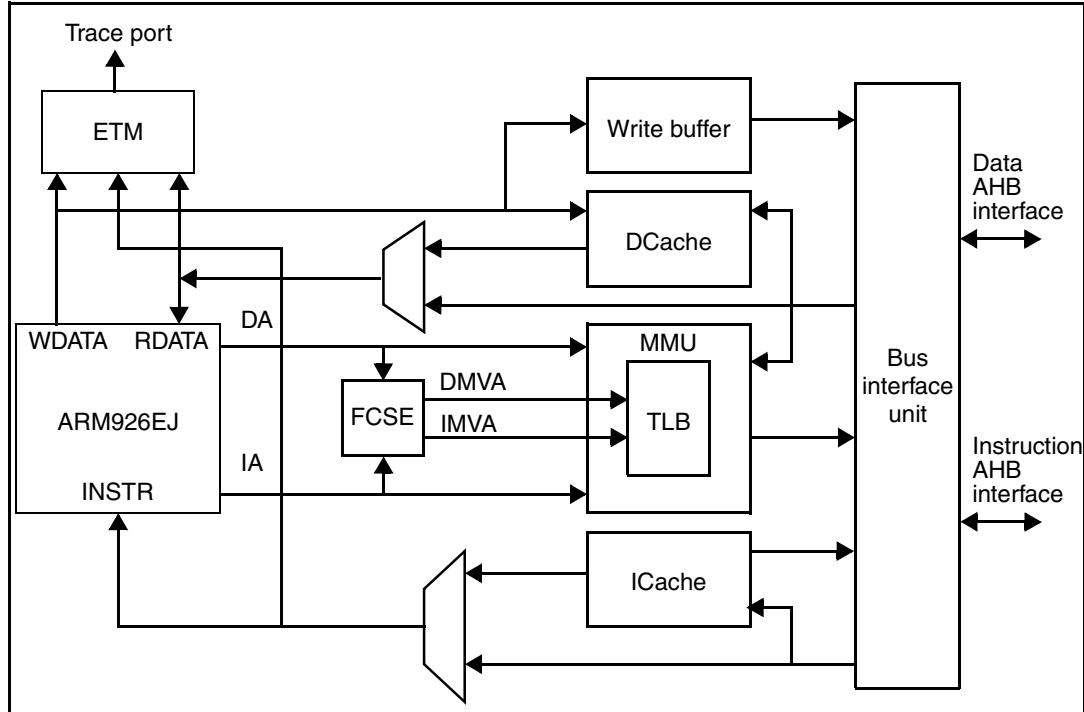
- ARM9EJ integer core
- A 16 Kbyte instruction cache (ICache), four-way set associative
- 16 Kbyte data cache (DCache), four-way set associative
- 16 word write buffer (WB)
- Data and program memory management units (MMUs) with table look-aside buffers
- Coprocessor 15 (CP15) used for programming caches, MMU and protection modules
- 32-bit AHB master bus interface for instruction fetches
- 32-bit AHB master bus interface for data reads and writes
- Embedded trace module (ETM) for real-time CPU activity tracing and debugging

Note: The STn8815A12 uses the ARM9EJ core in little-endian mode only.

7.2 Architecture

Figure 15 illustrates the architecture of the ARM926EJ processor.

Figure 15. CPU subsystem architecture



7.3 Addresses

The ARM926EJ system uses three types of address.

- Virtual address (VA), issued by the ARM9EJ core.
- Modified virtual address (MVA). The VA is translated to the MVA, using the FCSE PID value. This modified address is seen by the caches and MMU.
- Physical address (PA). The MVA is translated by the MMU to produce the PA. This address is used by the AMBA bus interface for external accesses.

7.4 Instruction cache

The instruction cache (ICache) is virtual index and virtual tag, addressed using the MVA. This avoids cache cleaning or invalidation on context switches.

7.4.1 ICache features

- 16 Kbyte, four-way set associative, with a cache line length of eight words per line (32 bytes per line)
- Supports allocate on read-miss, and performs critical-word first cache refilling. It is always reloaded one line at a time
- Disabled and flushed upon reset
- Can be enabled or disabled independently of the DCache, and of the MMU
- Pseudo-random or round-robin replacement can be selected in software
- Cache lockdown features allow control over which cache ways are used for allocation on a line-fill, providing a mechanism for both lockdown and control of cache pollution
- Disabling the ICache does not invalidate its contents

7.4.2 ICache operation

For best performance, the ICache should be enabled as soon as possible after reset. The ICache is enabled or disabled in software.

When the ICache is enabled, it is searched whenever the processor requests an instruction.

When both the ICache and the MMU are enabled:

- instruction fetches are cachable or non-cachable, as configured in software
- protection checks are performed
- the VA is translated to an MVA and this is remapped to a PA, according to the page entry

When the ICache is enabled and the MMU is disabled:

- instruction fetches are cachable
- no protection checks are performed
- addresses are flat mapped, that is, VA = MVA = PA

7.4.3 Caching behavior

When the ICache is enabled (and the instruction is cachable, as configured in software):

- if the instruction is in the cache, it is returned to the core, whether or not the MMU is enabled
- if the instruction is not in the cache, a line is fetched and written to the cache

When the ICache is disabled, all instructions are fetched from external memory (via the AHB bus).

The caching behavior of the ICache depends on the page mode, described in [Table 20](#).

Table 20. ICache caching behavior

Page mode	ARM926EJ behavior
Non-cachable	ICache disabled: read from external memory.
Cachable	Cache hit: read from ICache. Cache miss: line-fill from external memory.

7.4.4 ICache validity

The flush ICache instruction is fetched at cycle 0, for example, but is not executed until cycle 4 (the ARM9EJ core uses a 5-stage opcode pipe). Thus, four additional opcodes are potentially fetched from the ICache before the flush ICache opcode is executed. Once executed, the entire ICache is invalidated before the next opcode executes. Typically, to avoid confusion, the ICache is flushed by four non-opcodes following the CP15 (system control coprocessor) instruction.

The ICache content is not flushed when the ICache is disabled. Its contents remain valid and are accessible again when the ICache is re-enabled.

7.5 Data cache

The data cache (DCache) is virtual index and virtual tag, addressed using the MVA. This avoids cache cleaning or invalidation on context switches.

7.5.1 DCache features

- 16 Kbytes, four-way set associative, with a cache line length of eight words per line (32 bytes per line).
- Supports allocate on read-miss. The DCache performs critical-word first cache refilling. The DCache is always reloaded one line at a time.
- Pseudo-random or round-robin replacement can be selected in software.
- Cache lockdown features allow control over which cache ways are used for allocation on a line-fill, providing a mechanism for both lockdown and control of cache pollution.
- Can be enabled or disabled independently of the ICache, and of the MMU.
- On reset, it is disabled and flushed, and entries are invalidated. The DCache is then not accessible for data.

7.5.2 DCache operation

For the data cache to be useful, both the DCache and the MMU must be enabled. The DCache and MMU are enabled or disabled in software.

When both DCache and the MMU are enabled:

- DCache is searched whenever the processor requests a data load or store
- protection checks are performed
- the VA is translated to an MVA and this is remapped to a PA, according to the page entry

When the DCache is enabled and the MMU is disabled:

- data accesses are non-cachable and non-bufferable
- no protection checks are performed
- addresses are flat mapped, that is, VA = MVA = PA

When the DCache is disabled, all data accesses are to or from external memory.

7.5.3 Caching and buffering behavior

When both DCache and the MMU are enabled (and the data is cachable or bufferable, as configured in software):

- On data load:
 - If the data is in the cache, it is returned to the core.
 - If the data is not in the cache, a line is fetched and written to the cache.
- On data store:
 - If the data is in the cache, the cache is updated. In write-through mode, data is also pushed on to the write buffer.
 - If the data is not in the cache, the cache is not updated. The data is pushed on to the write buffer.

The caching and buffering behavior of the DCache depends on the page mode (see [Table 21](#)).

Table 21. DCache caching and buffering behavior

Page mode	ARM926EJ behavior
Non-cachable, Non-bufferable	DCache disabled. Read from external memory. Write as a non-buffered store(s) to external memory. DCache is not updated.
Non-cachable, bufferable	DCache disabled. Read from external memory. Write as a buffered store(s) to external memory. DCache is not updated.
Write-through	DCache enabled. Read hit: read from DCache. Read miss: line-fill. Write hit: write to DCache, and buffered store to external memory. Write miss: buffered store to external memory.
Write-back	DCache enabled. Read hit: read from DCache. Read miss: line-fill. Write hit: write to DCache only. Write miss: buffered store to external memory.

7.5.4 Validity

The entire DCache can be invalidated with a single flush DCache instruction. The DCache is flushed upon reset.

If the DCache is disabled, its content is preserved, and is accessible again when the DCache is re-enabled; but the content may be incoherent.

7.6 Write buffer

The main write buffer comprises a 16-word data buffer and a 4-address buffer, and is used for all:

- writes to a non-cachable, bufferable region, as well as to a write-through region,
- write misses to a write-back region.

A separate buffer is incorporated in the DCache for holding write-back data for cache line evictions or the cleaning of dirty cache lines. The DCache write-back buffer has 8 data word entries and a single address entry.

The MCR drain write buffer allows both write buffers to be drained under software control.

The MCR wait for interrupt causes both write buffers to be drained and the ARM926EJ processor to be put into a low-power state until an interrupt occurs.

7.7 Memory management unit (MMU)

The ARM926EJ memory management unit (MMU) provides virtual memory features required by systems operating on platforms such as Symbian™ OS, Windows CE® and Linux.

A single set of two-level page tables stored in main memory is used to control the address translation, permission checks, and memory region attributes for both data and instruction accesses. The MMU uses a single unified translation look-aside buffer (TLB) to cache the information held in the page tables. TLB entries can be locked down to ensure that a memory access to a given region never incurs the penalty of a page table walk.

7.7.1 MMU features

- 16 domains for access protection scheme
- Mapping sizes of:
 - 1 Mbyte (sections)
 - 64 Kbytes (large pages)
 - 4 Kbytes (small pages)
 - 1 Kbyte (tiny pages)
- Ability to separately specify access permissions for large pages and small pages for each quarter of the page (subpage permissions)
- Hardware page table walks
- Invalidation of entire TLB
- Invalidation of TLB entry, selected by MVA
- Lockdown of TLB entries

The MMU is enabled or disabled in software.

Note: *If the MMU is enabled, then disabled, and subsequently re-enabled, the contents of the TLB are preserved. If these are now invalid, then the TLB must be invalidated before re-enabling the MMU.*

7.7.2 Access permissions and domains

For large and small pages, access permissions are defined for each subpage (1 Kbyte for small pages, 16 Kbytes for large pages). Sections and tiny pages have a single set of access permissions.

All regions of memory have an associated domain. A domain is the primary access control mechanism for a region of memory. It defines the conditions necessary for an access to proceed. The domain determines:

- if access permissions are used to qualify the access
- if the access is unconditionally allowed to proceed
- if the access is unconditionally aborted

In the latter two cases, the access permission attributes are ignored.

There are 16 domains.

Note: Since load and store multiple instructions can cross a page boundary, the permission access is checked for each sequential address.

7.7.3 Translated entries

The translation look-aside buffer (TLB) caches translated entries. During CPU memory accesses, the TLB provides protection information to the access control logic.

When the TLB contains a translated entry for the MVA, the access control logic determines the action to be taken.

- If access is permitted and an off-chip access is required, the MMU outputs the appropriate physical address corresponding to the MVA.
- If access is permitted and an off-chip access is not required, the cache or TCM services the access.
- If access is not permitted, the MMU signals to the CPU core to abort.

If the TLB misses (it does not contain an entry for the MVA), the translation table walk hardware is invoked to retrieve the translation information from a translation table in physical memory. When retrieved, the translation information is written into the TLB, possibly overwriting an existing value.

At reset, the MMU is turned off, no address mapping occurs, and all regions are marked as non-cachable and non-bufferable.

7.7.4 Address translation

The virtual address (VA) generated by the CPU core is converted to a modified virtual address (MVA) by the fast context switch extension (FCSE). The MMU translates MVAs into physical addresses to access external memory, and also performs access permission checking.

The MMU table walking hardware is used to add entries to the TLB. The translation information, that comprises both the address translation data and the access permission data, resides in a translation table located in physical memory. The MMU provides the logic for automatically traversing this translation table and loading entries into the TLB.

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

There are three sizes of page-mapped accesses and one size of section-mapped access.
There are page-mapped accesses for:

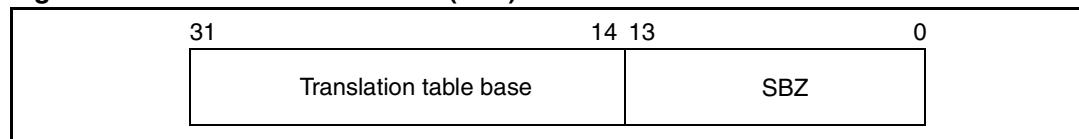
- large pages
- small pages
- tiny pages

The translation process always begins in the same way, with a level 1 fetch. A section-mapped access requires only a level 1 fetch, but a page-mapped access requires an additional level 2 fetch.

Translation table

The hardware translation process is initiated when the TLB does not contain a translation for the requested MVA. A translation table exists in physical memory that contains section or page descriptors, or both. The translation table has up to 4096×32 -bit entries, each describing 1 Mbyte of virtual memory. This allows up to 4 Gbytes of virtual memory to be addressed. The translation table is pointed to by a 32-bit value referred to as the translation table base (TTB). In fact, the TTB is stored in bits [31:14] of this value, bits [13:0] being unpredictable. This is illustrated in [Figure 16](#).

Figure 16. Translation table base (TTB)

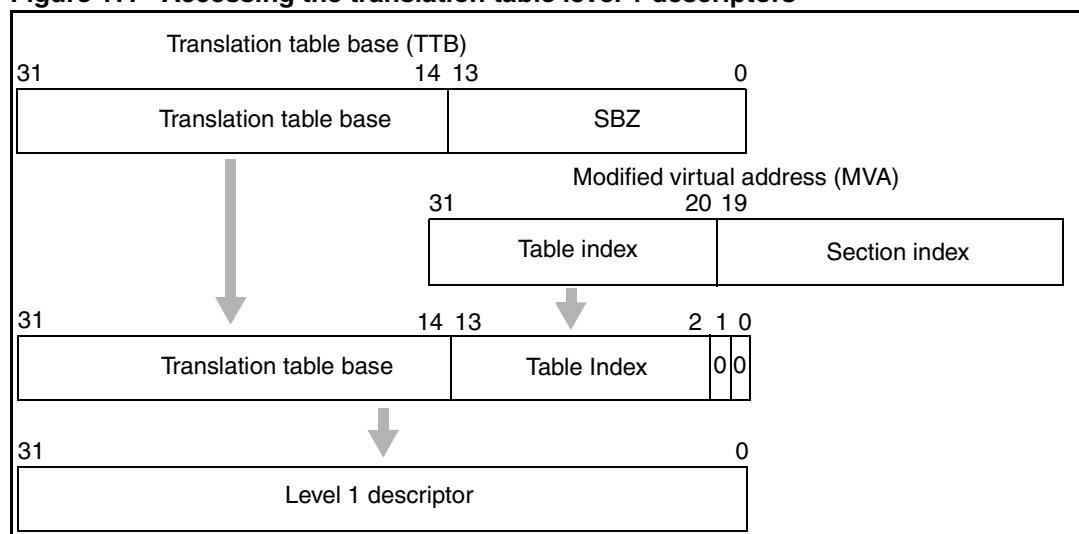


The translation table must then reside on a 16 Kbyte boundary.

Level 1 fetch

In order to access the translation table level 1 descriptors, bits [31:14] of the TTB are concatenated with bits [31:20] of the virtual address (VA) to produce a 30-bit address. This address selects a 4-byte translation table entry, which is a level 1 descriptor for either a section or a page table: see [Figure 17](#).

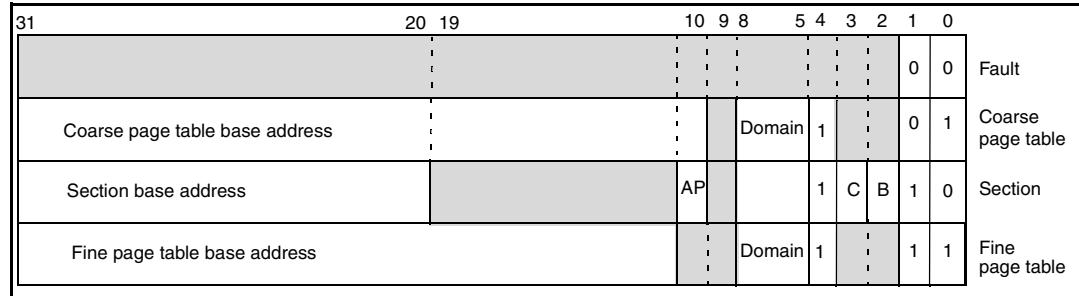
Figure 17. Accessing the translation table level 1 descriptors



Level 1 descriptor

The level 1 descriptor returned is either a coarse or fine page table descriptor, or a section descriptor. [Figure 18](#) shows the format of level 1 descriptors.

Figure 18. Level 1 descriptors



A section descriptor provides the base address of a 1-Mbyte block of memory.

The page table descriptors provide the base address of a page table that contains level 2 descriptors. There are two sizes of page table.

- Coarse page tables have 256 entries, splitting the 1 Mbyte that the table describes into 4-Kbyte blocks.
- Fine page tables have 1024 entries, splitting the 1 Mbyte that the table describes into 1-Kbyte blocks.

Level 1 descriptor bit assignments are shown in [Table 22](#).

Table 22. Level 1 descriptor bits

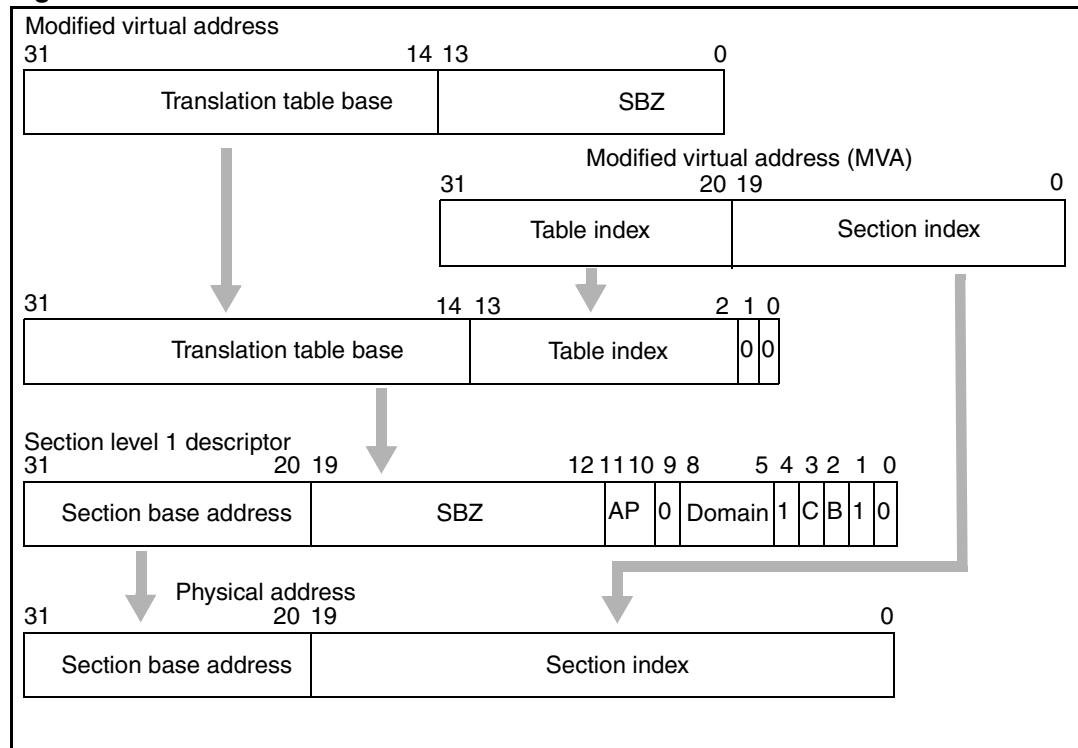
Bits			Description
Section	Coarse	Fine	
[31:20]	[31:10]	[31:12]	These bits form the corresponding bits of the physical address.
[19:12]	-	-	Should be zero.
[11:10]	-	-	Access permission (AP) bits.
9	9	[11:9]	Should be zero.
[8:5]	[8:5]	[8:5]	Domain control bits: specify one of the 16 possible domains that contain the primary access controls.
4	4	4	Should be 1.
[3:2]	-	-	Bits C and B indicates whether the area of memory mapped by this page is treated as write-back cachable, write-through cachable, non-cached buffered or non-cached non-buffered.
-	[3:2]	[3:2]	Should be zero.
[1:0]	[1:0]	[1:0]	These bits indicate the type of descriptor: 00: Invalid descriptor; generates a section translation fault. 01: Coarse page table descriptor. 10: Section descriptor. 11: Fine page descriptor.

Note: If a coarse or fine page table descriptor is returned from the level 1 fetch, a level 2 fetch is initiated.

Translating section references

Figure 19 illustrates the complete section translation sequence. The access permissions contained in the level 1 descriptor must be checked before the physical address is put on the address bus.

Figure 19. Section translation



Level 2 descriptor

If the level 1 fetch returns either a coarse page table descriptor or a fine page table descriptor, this provides the base address of the page table to be used. The page table is then accessed and a level 2 descriptor is returned. *Figure 20* shows the format of level 2 descriptors.

Figure 20. Level 2 descriptors

	0	1	2	3	4	5	6	7	8	9	10	11	12	15	16	31
Fault	0	0														
Large page	1	0	B	C	ap0	ap1	ap2	ap3							Large page base address	
Small page	0	1	B	C	ap0	ap1	ap2	ap3							Small page base address	
Tiny page	1	1	B	C	ap										Tiny page base address	

A level 2 descriptor defines a tiny, small or large descriptor, or is invalid.

- A large page descriptor provides the base address of a 64 Kbyte block of memory.
- A small page descriptor provides the base address of a 4 Kbyte block of memory.
- A tiny page descriptor provides the base address of a 1 Kbyte block of memory.

Coarse page tables provide base addresses for either small or large pages.

- Large page descriptors must be repeated in 16 consecutive entries.
- Small page descriptors must be repeated in each consecutive entry.

Fine page tables provide base addresses for large, small or tiny pages.

- Large page descriptors must be repeated in 64 consecutive entries.
- Small page descriptors must be repeated in 4 consecutive entries.
- Tiny page descriptors must be repeated in each consecutive entry.

Level 2 descriptor bit assignments are shown in *Table 23*.

Table 23. Level 2 descriptor bits

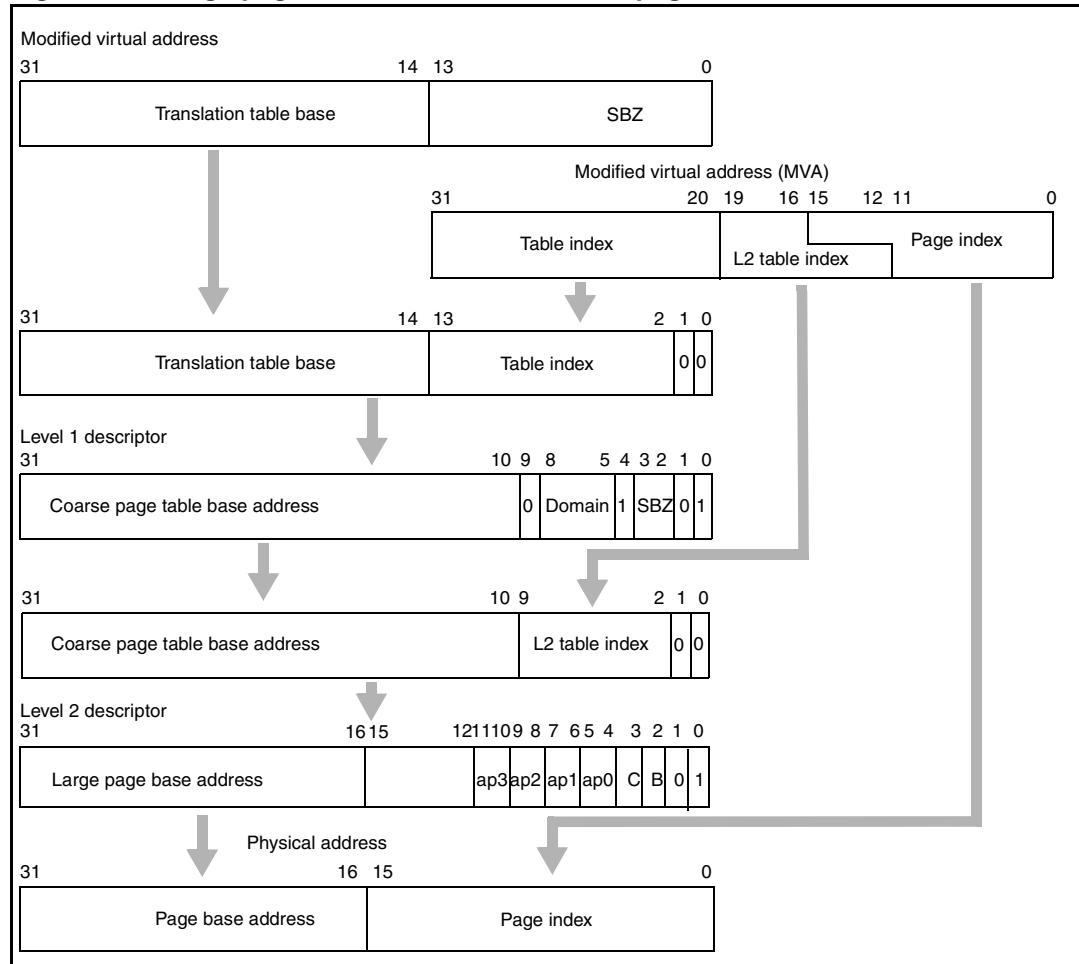
Bits			Description
Large	Small	Tiny	
[31:16]	[31:12]	[31:10]	These bits form the corresponding bits of the physical address.
[15:12]	-	[9:6]	Should be zero.
[11:4]	[11:4]	[5:4]	Access permission (AP) bits.
[3:2]	[3:2]	[3:2]	Bits C and B indicates whether the area of memory mapped by this page is treated as write-back cachable, write-through cachable, non-cached buffered or non-cached non-buffered.
[1:0]	[1:0]	[1:0]	These bits indicate the descriptor: 00: Invalid descriptor; generates a page translation fault. 01: Large page; 64 Kbyte page. 10: Small page; 4 Kbyte page. 11: Tiny page; 1 Kbyte page.

Note: *Tiny pages do not support subpage permissions and therefore only have one set of access permission bits.*

Translating large page references

Figure 21 illustrates the complete translation sequence for a 64 Kbyte large page. As the upper four bits of the page index and the lower four bits of the coarse page table index overlap, each coarse page table entry for a large page descriptor must be duplicated 16 times (in consecutive memory locations). If the large page is included in a fine page table, the large page descriptor must be duplicated 64 times in consecutive memory locations. If the large page is included in a fine page table, the large page descriptor must be duplicated 64 times.

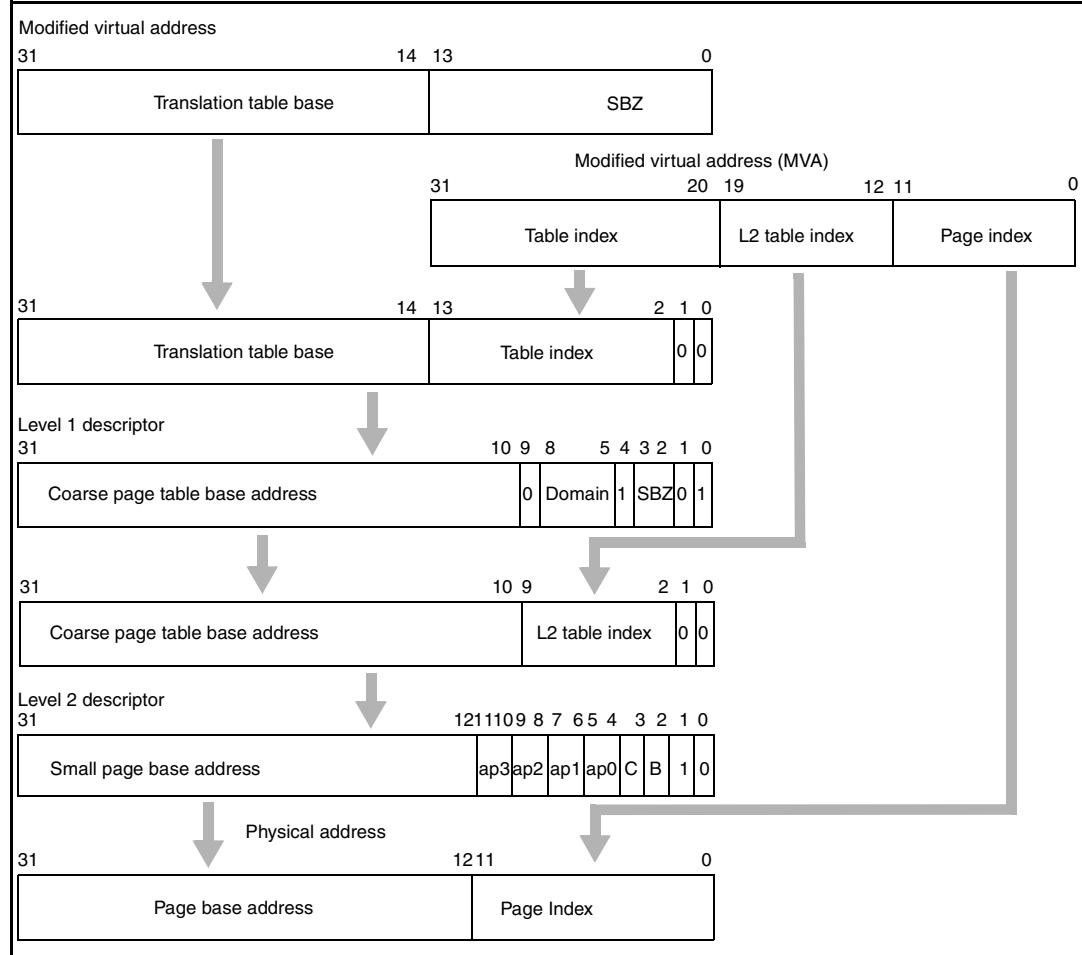
Figure 21. Large page translation from a coarse page table



Translating small page references

Figure 22 illustrates the complete translation sequence for a 4 Kbyte small page. If a small page descriptor is included in a fine page table, the upper two bits of the index of the page overlap the lower two bits of the index of the fine page table. In other words, if the small page is included in a fine page table, the small page descriptor must be duplicated 4 times in consecutive entries.

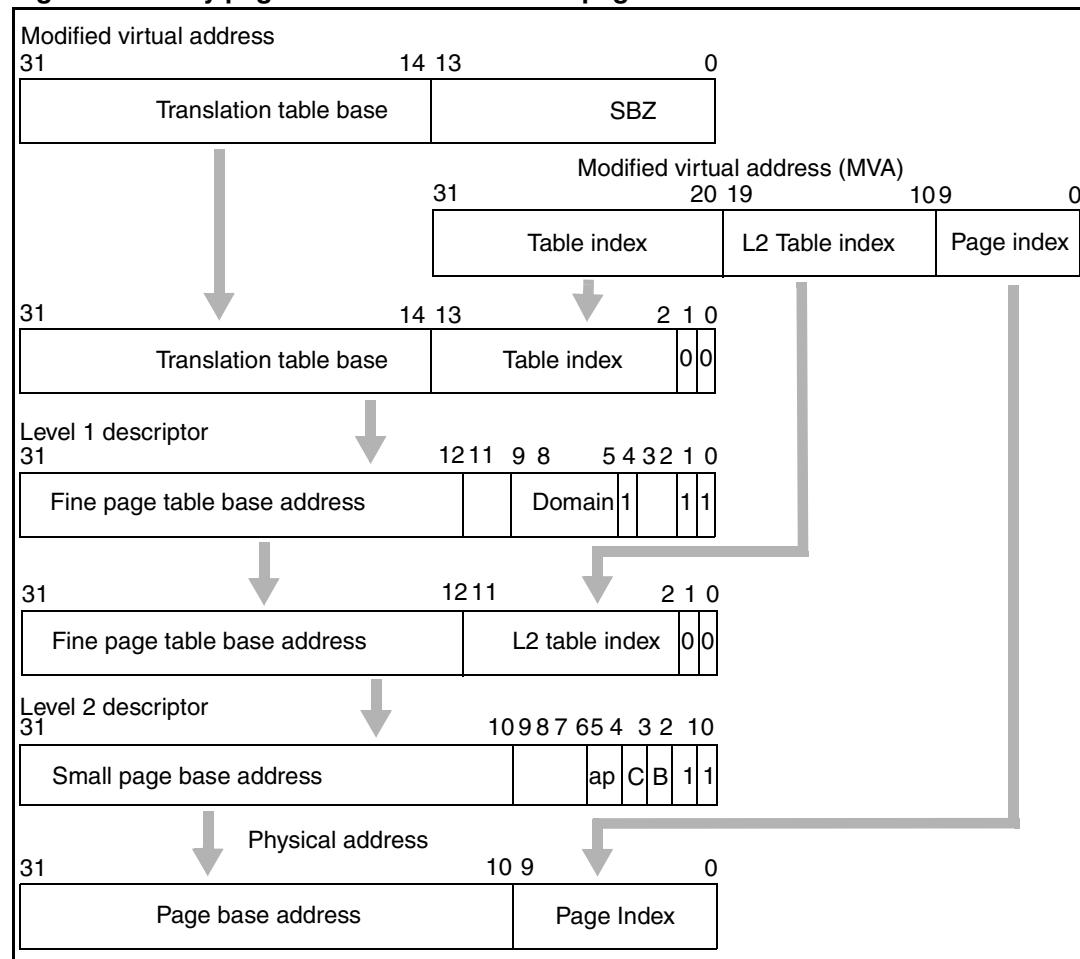
Figure 22. Small page translation from a coarse page table



Translating tiny page references

Figure 23 illustrates the complete translation sequence for a 64 Kbyte large page. As the upper four bits of the page index and the lower four bits of the coarse page table index overlap, each coarse page table entry for a large page descriptor must be duplicated 16 times (in consecutive memory locations). If the large page is included in a fine page table, the large page descriptor must be duplicated 64 times.

Figure 23. Tiny page translation from a fine page table



Note:

The domain specified in the level 1 descriptor and access permissions specified in the level 1 descriptor together determine whether the access has permission to proceed.

Subpages

Access permissions can be defined for subpages of small and large pages. If, during a page walk, a small or large page has a non-identical subpage permissions, only the subpage being accessed is written into the TLB. For example, a 16 Kbyte (large page) subpage entry is written into the TLB if the subpage permissions differ, and a 64-Kbyte entry is put in the TLB if the subpage permissions are identical.

When using subpage permissions, and the page entry has to be invalidated, the four subpages must be invalidated separately.

7.8 MMU faults and CPU aborts

The MMU generates an abort on the following fault types:

- alignment (data accesses only)
- translation
- domain
- permission

In addition, an external abort can be raised by the external system. This can happen only for access types that have the core synchronized to the external system:

- page walks
- non-cached reads
- non-buffered writes
- non-cached read-lock-write sequence (SWP instruction)

Alignment fault checking is not affected by whether or not the MMU is enabled. Translation, domain and permission faults are only generated when the MMU is enabled.

The access control mechanisms of the MMU detect the conditions that produce these faults. If a fault is detected as a result of a memory access, the MMU aborts the access and signals the fault condition to the CPU core. The MMU retains status and address information about faults generated by the data accesses.

The MMU also retains status information about faults generated by instruction fetches.

An access violation for a given memory access inhibits any corresponding external access to the AHB interface, with an abort returned to the CPU core.

7.8.1 Fault status and fault address

On a data abort, the MMU retains an encoded 4-bit value, the fault status, along with the 4-bit encoded domain number, and similarly on a prefetch abort. In addition, the MVA associated with the data abort is stored. If an access violation simultaneously generates more than one source of abort, they are encoded according to the priorities given in [Table 24 on page 121](#).

Fault status

[Table 24](#) shows the various access permissions and controls supported by the data MMU, and how these are interpreted to generate faults.

Table 24. Priority encoding of fault status

Priority	Source	Size	Domain
Highest	Alignment	-	Invalid
	External abort on translation	First level Second level	Invalid Valid
	Translation	Section Page	Invalid Valid
	Domain	Section Page	Valid Valid
	Permission	Section Page	Valid Valid
Lowest	External abort	Section Page	Valid Valid

Note: Invalid values can occur in the status encoding for domain faults. This happens when the fault is raised before a valid domain field has been read from a page table description. Aborts masked by a higher priority abort can be regenerated by fixing the cause of the higher priority abort, and repeating the access. Alignment faults are not possible for instruction fetches.

Fault address

For load and store instructions that can involve the transfer of more than one word (LDM/STM, LDRD, STRD and STC/LDC), the fault address stored depends on the type of access, and for external aborts, on whether or not the access crosses a 1 Kbyte boundary. [Table 25](#) shows the fault addresses for multi-word transfers.

Table 25. Fault addresses for multi-word transfers

Source	Fault address
Alignment	MVA of first aborted address in transfer.
External abort on translation	MVA of first aborted address in transfer.
Translation	MVA of first aborted address in transfer.
Domain	MVA of first aborted address in transfer.
Permission	MVA of first aborted address in transfer.
External abort or non-cached reads or non-buffered writes	MVA of last address before 1 Kbyte boundary if any word of the transfer before 1 Kbyte boundary is externally aborted. MVA of last address in transfer if the first externally aborted word is after 1 Kbyte boundary.

Compatibility issue

To allow code to be easily ported to future ARM architectures, it is recommended that no reliance is made upon the external abort behavior.

The instruction FSR (fault status register) is intended for debugging purposes only. Code that is intended to be ported to other ARM architectures must not use the instruction FSR.

7.8.2 Domain access control

MMU accesses are primarily controlled through the use of domains. There are 16 domains and each has a 2-bit field to define access to it. The 2 bits are referred to as the domain access (DX) bits.

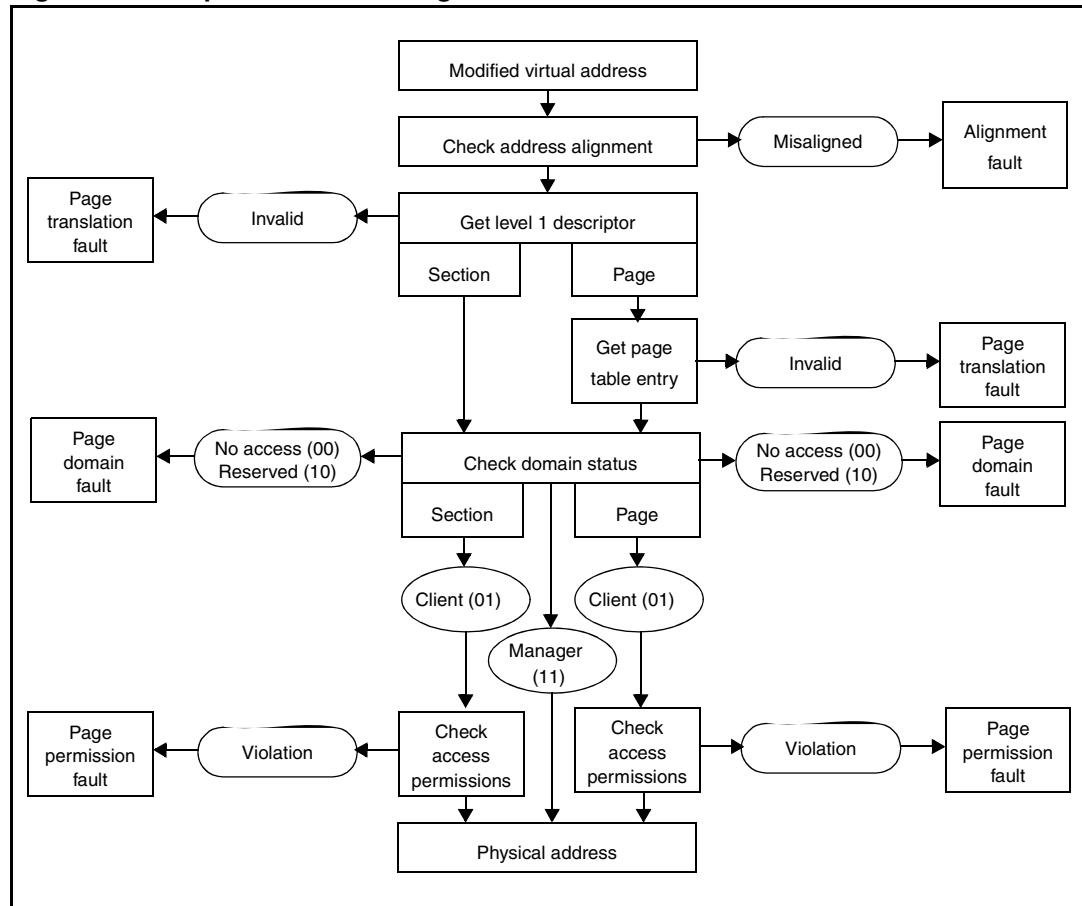
Two types of user are supported: clients and managers.

Note: When the MMU is turned off, the physical address is output directly and no memory access permission checks are performed.

7.8.3 Fault checking sequence

The sequences that the MMU uses to check for access faults are different for sections and pages. The sequences for both types of access are shown in [Figure 24](#).

Figure 24. Sequence for checking faults



The conditions that generate each of the faults are:

- alignment faults
- translation faults
- domain faults
- permission faults

These are described in the subsections that follow.

Alignment faults

If alignment fault checking is enabled, the MMU generates an alignment fault:

- on any data word access, if the address is not word aligned
- on any half-word access, if the address is not half-word aligned, irrespective of whether the MMU is enabled

An alignment fault is not generated on any instruction fetch or any byte access.

Note:

If an access generates an alignment fault, the access sequence aborts without reference to other permission checks.

Translation faults

There are two types of translation fault:

- Section: A section translation fault is generated if the level 1 descriptor is marked as invalid. This happens if bits [1:0] of the descriptor are both 0.
- Page: A page translation fault is generated if the level 1 descriptor is marked as invalid. This happens if bits [1:0] of the descriptor are both 0.

Domain faults

There are two types of domain fault:

- Section: The level 1 descriptor holds the 4-bit domain field, which selects one of the sixteen 2-bit domain access control fields, described in [Section 7.8.2 on page 122](#). The two bits for the specified domain are then checked for access permissions. The domain is checked when the level 1 descriptor is returned.
- Page: The level 1 descriptor holds the 4-bit domain field, which selects one of the sixteen 2-bit domain access control fields, described in [Section 7.8.2 on page 122](#). The two bits of the specified domain are then checked for access permissions. The domain is checked when the level 1 descriptor is returned.

If the specified access is either no access (00) or reserved (10) then either a section domain fault or page domain fault occurs.

Permission faults

If the 2-bit domain field returns 01 (client) then access permissions are checked as follows:

- Section: If the level 1 descriptor defines a section-mapped access, the 2-bit access permission (AP) field of the descriptor defines whether or not the access is allowed. If the access is not allowed, a section permission fault is generated.
 - Large page or small page: If the level 1 descriptor defines a page-mapped access and the level 2 descriptor is for a large or small page, four access permission fields (AP3 to AP0) are specified, each corresponding to one quarter of the page.
 - For small pages, AP3 is selected by the top 1 Kbyte of the page and AP0 is selected by the bottom 1 Kbyte of the page.
 - For large pages, AP3 is selected by the top 16 Kbytes of the page and AP0 is selected by the bottom 16 Kbytes of the page.
- If the access is not allowed, a page permission fault is generated.
- Tiny page: If the level 1 descriptor defines a page-mapped access, and the level 2 descriptor is for a tiny page, the AP bits of the level 1 descriptor define whether or not the access is allowed in the same way as for a section. The fault generated is a page permission fault.

7.8.4 External aborts

In addition to the MMU generated aborts, external aborts can be generated for certain types of access that involve transfers over the AHB bus. These can be used to flag errors on external memory accesses. However, not all accesses can be aborted in this way.

The following accesses can be externally aborted:

- page walks
- non-cached reads
- non-buffered writes
- non-cached read-lock-write (SWP) sequences

For a read-lock-write (SWP) sequence, if the read externally aborts, the write is always attempted.

A swap to a non-cachable, bufferable region is forced to have precisely the same behavior as a swap to a non-cachable, non-bufferable region. This means that the write part of a swap to a non-cachable, bufferable region can be externally aborted.

7.8.5 TLB structure

The MMU contains a single unified TLB (translation look-aside buffer) used for both data accesses and instruction fetches. The TLB is divided into two parts:

- Lockdown part: This is an 8-entry, fully-associative part used exclusively for holding locked down TLB entries.
- Set associative part: The set associative part is for all other entries.

Whether an entry is placed in the set-associative or lockdown part of the TLB is dependent on the state of the TLB lockdown feature, when the entry is written into the TLB.

When an entry has been written into the lockdown part of the TLB, it can only be removed by being overwritten explicitly, or by an MVA-based TLB invalidate operation, where the MVA matches the locked down entry.

The structure of the set-associative part of the TLB does not form part of the programmer's model for the ARM926EJS processor. No assumptions must be made about the structure, replacement algorithm or persistence of entries in the set-associative part. Specifically, when using the set associative part, the following points must be taken into account:

- Any entry written into the set associative part of the TLB can be removed at any time.
- The set associative part of the TLB must be considered as a temporary cache of translation/page table information.
- No reliance must be placed on an entry either residing or not residing in the set associative TLB, unless that entry already exists in the lockdown TLB.
- The set associative part of the TLB can contain entries that are defined in the page tables but do not correspond to address values that have been accessed after the TLB invalidation.
- The set associative part of the TLB must be considered as a cache of the underlying page table, where memory coherency must be maintained at all times.
- If a level 1 descriptor is modified in main memory then to guarantee coherency, either an invalidate TLB or invalidate TLB by entry operation must be used to remove any cached copies of the level 1 descriptor. This is required regardless of the type of level 1 descriptor (section, level 2 page table reference, or fault).

- If any of the subpage permissions for a given page are different then each of the subpages are treated separately. To invalidate all the entries associated with a page that has subpage permissions, four MVA-based invalidate operations are required, one for each subpage.

7.9 System control processor (CP15)

The system control processor (CP15) is used to configure and control the ARM926EJ processor. The caches, memory management units (MMUs) and other system options are configured in software.

7.10 Embedded trace module (ETM)

The STn8815A12 platform has an embedded trace module (ETM) to provide instruction and data trace capabilities for the ARM926EJ processor. The device is the ARM ETM9 and in the Medium+ configuration uses either an 8-bit data output or a 4-bit demultiplexed data output.

- The instruction trace shows the instruction flow of the CPU.
- The data trace shows the data access results after the CPU executes load and store operations.

7.10.1 ETM features

- Instruction/data trace
- 8-bit trace packet width
- 45-byte trace packet capture FIFO
- Eight pairs of address comparators for trace trigger
- Height comparators for trace trigger
- Four 16-bit counters
- 3-state sequencer (state machine)

7.10.2 ETM interface

The ETM can work either in normal mode or in demultiplexed mode. These modes are described in the subsections below.

Normal mode

The ETM logical signal interface requires 13 trace interface pins.

The ETM trace interface signals and the STn8815A12 pins on which they are delivered in normal mode are detailed in [Table 26](#).

Table 26. ETM signals and STn8815A12 pins (ETM normal mode)

ETM signals	Pins	Description
TRACEPKTA[0:7]	ETMPKTA[0:3] ETMPKTA[4:7] ETMPKTB[0:3]	The TRACEPKTA signals comprise the 8-bit data trace packets (packaged address and data information).
PIPESTAT[0:2]	ETMPSTA[0:2]	The PIPESTAT signals are used to output the CPU pipeline at the CPU execute stage on every TRACECLK and are used by the software to reconstruct the compressed trace output.
TRACESYNCA	ETMSYNCA	The TRACESYNCA signal is used to indicate when the first of multiple packets are to be output on the TRACEPKT bus.
TRACECLK	ETMCLK	The TRACECLK signal operates at one of two frequencies: – The same frequency as the CPU. – The CPU frequency divided by two (half-rate clocking). When this rate is selected, the trace port analyzer (TPA) samples the trace data signals on both the rising and the falling edges of TRACECLK.

Demultiplexed mode

This mode is recommended for high-speed systems in which the switching frequency is too high for an external trace probe.

The demultiplexed ETM outputs need the 13 trace interface pins listed above for normal mode, plus some GPIOs that are automatically configured to ETM demultiplexed output signals. The port width is reduced to 4-bit trace data in this mode.

The ETM trace interface signals and the STn8815A12 pins on which they are delivered in demultiplexed mode are detailed in [Table 27](#).

TRACECLK, delivered on ETMCLK pin, operates at one of two frequencies:

- half the CPU frequency
- one-quarter of the CPU frequency.

Table 27. ETM signals and STn8815A12 pins (ETM demultiplexed mode)

ETM signals	Pins
TRACEPKTA[0:3]	ETMPKTA[0:3]
TRACEPKTB[0:3]	ETMPKTA[4:7]B[0:3]
PIPESTAT[0:2]	ETMPSTA[0:2]
PIPESTATB[0:2]	ETMPSTB[2:0]
TRACESYNCA	ETMSYNCA
TRACESYNCB	ETMSYNCB
TRACECLK	ETMCLK

7.10.3 Additional reference material

Further information on the ETM can be found in:

ETM9 (Rev 2a) Technical Reference Manual (ARM DDI 0157E)

Embedded Trace Macrocell Specification (ARM IHI 0014H)

Documentation is also available directly from Advanced RISC Machines at www.arm.com.

8 Level 2 Cache Controller (L2CC)

The L2CC is an eight-way unified, 128 KBytes, physically addressed (indexed), physically tagged cache. The L2CC does not maintain coherency among caches (of Smart Accelerators). The L2CC can be locked on a way basis.

8.1 L2CC features

The L2CC is an eight-way unified, physically addressed (indexed), physically tagged cache. The L2CC does not maintain coherency among caches. The L2CC can be locked on a way basis.

The L2CC features are:

- Level 2 Cache size is 128 KBytes.
- Fixed line length of 32 bytes, eight words.
- Physically addressed and physically tagged.
- Lockdown format C supported, with separate way locking mechanisms for data and instructions.
- Eight-way associativity which can be direct mapped, depending on the use of lockdown registers.
- Data RAM is byte-writable.
- Support for these cache modes:
 - Write-Through, read allocate
 - Write-Back, read allocate
 - Write allocate override option to always have allocation on write misses in the L2CC.
 - Performs critical word first refilling, with the option of refilling starting with word 0. The same option is used to transform nonbufferable wrap write bursts and non cachable wrap read bursts into linear accesses.
- Pseudo-random victim selection policy - can be made deterministic with use of lockdown registers.

8.1.1 L2CC buffers

The L2CC has the following buffers:

- Two linefill buffers, LFB:
These buffers capture linefill data from main memory, waiting for a complete line before writing to L2 memory. This makes the L2CC non-blocking for requests from the other slave ports
- Two line read buffers, LRB:
These buffers hold a line from the L2CC, for subsequent requests that hit on the line.
- One eviction buffer, EB:
This holds an evicted line from the L2CC, to be written back to main memory.
- One write buffer, WB:
This holds buffered writes before their draining to main memory, as well as to the L2CC.

The write buffer is made of two slots, each with a 256-bit data line and one address per slot. It enables multiple writes to the same line to be merged.

- One write-allocate buffer, WB:
If the write buffer line is not full, this buffer merges data from the write buffer and missing data, from master port 1 before requesting an allocation to the cache.

L2CC write buffer WB

The write buffer has two slots that each contain one 256-bit data line and its associated address. The write buffer has merging capabilities, so that successive writes to the same line address are merged in the same buffer slot. Two buffered write accesses to the same address cause the first one to be overridden if the write buffer has not been drained in between the writes.

Merging capability means that lines are not treated as soon as they contain data. Write buffer draining policy is:

- the write buffer is drained at each non cachable read occurrence
- the write buffer is drained at each non bufferable write occurrence
- if the two slots of the write buffer contain data, the least recently accessed is drained
- if a hazard is detected with one write buffer slot, it is drained to resolve the hazard.

Each slot contains a byte-valid field that allows the control logic to know the data line fill level. If a drained slot is drained while its data line is not full, the write buffer must request correct transactions to the master port in the form of bursts or linear access requests.

For write allocate transactions, the write buffer transfers information for one of its slots to a write-allocate buffer described in L2CC write-allocate buffer.

L2CC write-allocate buffer WB

Allocation to the cache in case of write-allocate transaction is not performed directly by the write buffer. In cases of a write miss, when a write buffer slot is drained while its L2 memory attributes correspond to a write-allocate region (or write allocate behavior is

forced by the corresponding control bit in the Auxiliary Configuration Register), the data, address, and byte-valid information is sent to the write allocate buffer.

Based on byte-valid information, the write allocate buffer requests, through the MI port to L3, the data required to fill its data line if not full:

- if the line is full, no request is made.
- if one to eight sequential bytes within the same double word boundary are missing, one SINGLE 64-bit transaction is performed on L3
- in all other cases, a ‘full line fill’ request is performed.

Data from the write buffer and MI master port are then merged in the write-allocate buffer that can then request an allocation to the cache.

If the AHB Master port M1 receives an ERROR response during a read transfer for the write-allocate buffer, the allocation from the write-allocate buffer to the cache is not performed.

L2CC eviction buffer EB

The eviction buffer is incorporated for holding Write-Back data for L2CC line evictions or cleaning of dirty cache lines. It holds two halves of an L2CC line (32 bytes in total) and two address entries.

8.1.2 L2CC replacement strategy

The L2CC uses a pseudo-random replacement strategy. When used in combination with the lockdown Format C (see next section), a deterministic replacement strategy can be achieved.

The pseudo-random replacement strategy fills empty, unlocked ways first. If a line is completely full, the victim is chosen as the next unlocked way.

If a deterministic replacement strategy is required, the lockdown registers are used to prevent ways from being allocated. With the L2 size of 128KB (each way is 16KB), if the user wants a piece of code to reside in two ways (32KB space) with a deterministic replacement strategy, ways 1 to 7 must be locked before the code is filled into the L2CC. The first 16KB of code is allocated like this into way 0 only. Then, way 0 must be locked and way 1 unlocked so that the second half of the code is allocated in way one.

There are two lockdown registers, (one for data and one for instructions, so the user can also separate data and instructions into separate ways of the L2CC, if required.

8.1.3 Uses of lockdown format C

Lockdown format C, used in the ARM926EJ-S caches, provides a method to restrict the replacement algorithm used on cache line fills and to only use selected cache ways within a set. Using this method, code can be fetched or loaded into the L2CC and protected from being evicted.

- Use lockdown format C to restrict the available ways for cache filling to n-ways, where n is less than the total number of ways, for example by only enabling filling to way 0.
- Use lockdown format C to fetch code into the cache:
- executing the routine for the first time. The L2CC_LCKWI (Lockdown I-Side) register is used,
- loading the code into the cache, using a non cachable load routine. The L2CC_LCKWD (Lockdown D-Side) register is used.
- Use lockdown format C to load data into the cache by:
- loading data into the cache for the first time. The L2CC_LCKWD (Lockdown D-Side) register is used,
- loading data into the cache, using a non cachable load routine. The L2CC_LCKWD (Lockdown D-Side) register is used.
- Write to the lockdown register to prevent filling to way 0, but enable filling to ways 1 to 7. The code and data is now protected in the cache and is not evicted on linefill.

Preventing or reducing cache pollution

There can be benefits in having a critical piece of software or data being cached in the L2CC memories, with the insurance that it cannot be polluted or evicted, due to a new allocation. Using Lockdown format C provides a simple method to load data into the L2CC using the linefill mechanism, then locking the cache memory to prevent eviction, and finally

using the L2CC cache maintenance operations to efficiently clean the data to the main memory system. To do that:

1. Use lockdown format C (I and/or D Lockdown) to restrict the permitted ways for cache filling to n-ways, where n is less than the total numbers of ways (8 is this implementation). For example, only permit filling to way 0.
2. Cache code or data into the cache:
 - Fetch code into the cache by executing the routine for the first time,
 - Load data into the cache by:
 - loading data for the first time,
 - cache the data in L2CC by executing the read loop being cached at L1 only.
3. Write to the lockdown register, I and/or D lockdown, to prevent allocation to way 0, but enable allocation to ways 1 to 7. The code/data is now protected in the cache and cannot be evicted on a linefill.

8.1.4 External abort support for System (L3) memory

The L2CC gives limited support for external aborts, because of restrictions of the AHB protocol (the AHB protocol does not provide a method for passing back an ERROR response that is not combined with its original transaction).

If an ERROR response is received by the L2CC on an AHB master port:

- If this request was a non cachable read, or non bufferable write (using ARM v5 nomenclature), then the associated slave port has been waited until the response was received from the main memory. The ERROR response is passed through to L1 on the appropriate word.
- If the request was a cachable read which missed at L2, then the ERROR response will be passed through to L1 on the appropriate word, and the line is not marked as valid in the L2.
- If the request was a bufferable write, then the slave port completes its transaction before the ERROR response is received from main memory. The ARB protocol does not permit this response to be passed back to L1. In this case, the L2CC produces a pulse on the Event bus. See Appendix C *Event Monitor* for details. If there is an event monitor, it can then produce an interrupt to the L1 core.

This way, all system (L3) external aborts can be detected at level 1.

8.2 Using L2CC with ARM926EJ cores

The way ARM926EJ cores drive the **HPROT** signals on the ARB imposes some restrictions when using L2CC with ARM926EJ cores.

When using the L2CC with ARM926EJ cores:

- Before disabling the L1 cache, first clean and invalidate, and disable the L2CC cache.
- All page tables must reside in a WT memory region.
- ARM926EJ does not distinguish between WT and WB accesses at L2. By default, L2CC treats all cachable accesses by ARM926EJ as WB. To ensure memory consistency when WT memory accesses are used, L2CC is treating all cachable memory accesses as WT. This is done in hardware by implementing a piece of logic between the ARM926EJ core and L2CC to detect HPROT = WB and map it to HPROT = WT at L2.

Here is the code of this piece of hardware implemented:

```
if (HPROT[3:2] == 2'b11)
    HPROTSn[3:2] = 2'b10; // WB mapped to WT
else
    HPROTSn[3:2] = HPROT[3:2]; // WT, NCBN, NCB unaltered
```

8.3 L2CC Event Monitor (L2EM)

The L2CC event monitor enables monitoring of L2CC events and errors in the RAMs that make up the L2CC. This information can be used to tune the overall system performance.

To optimize the performance of a system, the L2CC event monitor block provides four event counters, EMN0-EMN3. They can be used to count the instances of four different events selected from a list of possible L2CC events, or internal events such as overflows or clock edges. Each counter is a 32-bit counter and has its own interrupt generation logic. By combining different statistics, you can obtain a variety of performances.

The L2CC Event Monitor has the following features:

- four 32-bit read-only event counters, EMC0-EMC3
- ability to track the following events
 - CPU instruction read request to L2CC cache
 - CPU instruction read hit in L2CC cache
 - CPU data read request to L2CC cache
 - CPU data read hit in L2CC cache
 - CPU data write request to L2CC cache, not Write-Through)
 - CPU data write request to L2CC cache, Write-Through)
 - CPU data write hit in L2CC cache
 - L2CC buffered write abort
 - L2CC cache half-line eviction, two events for one full-line eviction
 - allocation to theL2CC caused by write transaction, write-allocate
 - error on data RAM read access
 - error on data RAM write access
 - error on tag RAM read access
 - error on tag RAM write access
 - parity error on data RAM read access
 - parity error on tag RAM read transaction.

9 Hardware semaphore unit (HSEM)

The hardware semaphore unit (HSEM) provides 16 hardware semaphores, which are an efficient way of locking using a single write access. This avoids the need for a read-modify-write bus transfer, which is not supported by all STn8815A12 units.

The hardware semaphores can be used by the ARM926 CPU, the smart audio accelerator (SAA), and the smart video accelerator (SVA). The unit is situated on the AMBA peripheral bus (APB), and can be accessed via the DMA controller.

Each hardware semaphore works as follows.

- The semaphore is not locked when its value is zero (the default after a reset). The semaphore can be locked if the current value of the semaphore is zero, and a non-zero value is written by a master. Each master must have a unique (or unique set of) non-zero 4-bit value(s) to be used as a code for locking the semaphore.
- After writing to a semaphore, a master must read the semaphore. If the semaphore value does not match the master's unique code, it means that the semaphore has been locked by another master.
- The semaphore can be unlocked only by the master that locked the semaphore. To unlock the semaphore, the master writes a zero.
- When unlocking a semaphore, the master can trigger an interrupt to one or more masters in the same write operation.

The HSEM has no external signals.

10 LCD controller (LCDC)

The STn8815A12 platform features an LCD controller (LCDC) which can drive a range of different types of LCD panels.

Note: *The LCDC shares its I/O pins with the master display interface (MDIF); only one of these interfaces can be selected and active at any one time.*

The DIF implementation of the LCDC on the STn8815A12 platform has the following specification.

- The LCDC is clocked by the CLK48 (48 MHz) or CLK72 (72 or 78 MHz) clock delivered by the PLL2 device (CLCDCLK signal). The 48 MHz clock consumes least power.
- The DIF interface signals are not directly connected to the I/O pins, but are routed via a multiplexing logic where they are multiplexed with the master display interface (MDIF) pin interface signals.
- The CLCD[23:16] signals are alternate B functions of GPIO[39:32].

10.1 Features

- Dual 16-deep, programmable, 32-bit wide FIFOs for buffering incoming display data
- Single- and dual-panel monochrome super twisted nematic (STN) displays with 4 or 8-bit interfaces
- Single- and dual-panel color STN displays with 8-bit interfaces
- Active matrix thin film transistor (TFT) color displays
- Advanced TFT (AD-TFT) and high-reflectivity TFT (HR-TFT) color displays
- Resolution programmable up to 1024 lines of 1024 pixels
- 1-, 2- or 4-bpp (bits-per-pixel) palettized displays for monochrome STN
- 1-, 2-, 4- or 8-bpp palettized color displays for color panels
- 12-bpp (4:4:4), 15(+l)-bpp (l:5:5:5) or 16-bpp (5:6:5) true-color for color panels
- 24-bpp packed and non-packed true-color (non-palettized) for color TFT and HR-TFT panels
- Programmable timing for different display panels
- 256-entry, 16-bit palette RAM, physically arranged as 128 x 32-bit RAM
- Frame, line, and pixel clock signal generation
- AC bias signal for STN, data enable signal for TFT and HR-TFT panels
- Patented grayscale algorithm
- True-color for 6/5/4-bit/color component panels with optional frame modulation
- Color enhancement (16- to 18-bpp conversion) for addressing 18-bit (RGB 666) TFT
- Little and big-endian, as well as WinCE formats
- Interrupt and synchro generation events

10.2 Overview

The LCDC translates pixel-coded data into the required LCD formats and timings.

10.2.1 LCD panel resolution

The LCDC can be programmed to support a wide range of panel resolutions:

- From 1 to 1024 lines per panel, in 1-line steps
- From 16 to 1024 pixels per line, in 16-pixel steps

10.2.2 Types of LCD panel supported

The LCDC supports the following types of LCD panels:

- HR-TFT panels with a 12-, 16-, 18- or 24-bit bus interface,
- TFT panels with a 12-, 16-, 18- or 24-bit bus interface,
- Single-panel monochrome STN panels with a 4- or 8-bit bus interface,
- Dual-panel monochrome STN panels with a 4- or 8-bit bus interface per panel,
- Single-panel color STN panels with an 8-bit bus interface,
- Dual-panel color STN panels with an 8-bit bus interface per panel.

10.2.3 Number of colors supported

TFT panels

TFT panels, including AD-TFT and HR-TFT panels, support one or more of the following color modes.

- 1 bpp, palettized, 2 colors selected from available colors
- 2 bpp, palettized, 4 colors selected from available colors
- 4 bpp, palettized, 16 colors selected from available colors
- 8 bpp, palettized, 256 colors selected from available colors
- 12 bpp, direct 4:4:4 RGB or BGR, with 4 bpp not used
- 15 bpp + Intensity, direct I:5:5:5 RGB or BGR, with intensity bit I. This pixel is still output, and can be used as a bright bit to connect to the least significant bit (LSB) of the R, G, and B components of a 6:6:6 TFT panel
- 16 bpp, direct 5:6:5 RGB or BGR providing 65536 colors
- 24 bpp, direct 8:8:8 RGB or BGR, providing over 16 million colors. The input format can be:
 - Packed: a 32-bit word comprises four bytes of data, allowing three words to contain four 24-bit pixels
 - Non-packed: a 32-bit word contains the three bytes that represent a pixel, and the remaining byte is unused

Color STN panels

Color STN panels are driven by a grayscale algorithm patented by ARM Limited. These panels support one or more of the following color modes:

- 1 bpp, palettized, 2 colors selected from 3375 (15 x 15 x 15)
- 2 bpp, palettized, 4 colors selected from 3375
- 4 bpp, palettized, 16 colors selected from 3375
- 8 bpp, palettized, 256 colors selected from 3375
- 12 bpp, direct 4:4:4 RGB or BGR, with 4 bpp not being used (3375 colors)
- 15 bpp, direct I:5:5:5 RGB or BGR (only 4 bits per component are used, 3375 colors)
- 16 bpp, direct 5:6:5 RGB or BGR (only 4 bits per component are used, 3375 colors)

Monochrome STN panels

Monochrome STN panels support one or more of the following modes:

- 1 bpp, palettized, 2 gray scales selected from 15
- 2 bpp, palettized, 4 gray scales selected from 15
- 4 bpp, palettized, 16 gray scales selected from 15

More than 4 bpp can be programmed for monochrome panels, but since the maximum number of gray scales supported on the display is 15, such modes are redundant.

10.2.4 Requirements for different display types

STN displays

These require algorithmic pixel pattern generation to provide pseudo gray scaling on monochrome displays or color creation on color displays.

AD-TFT and HR-TFT displays

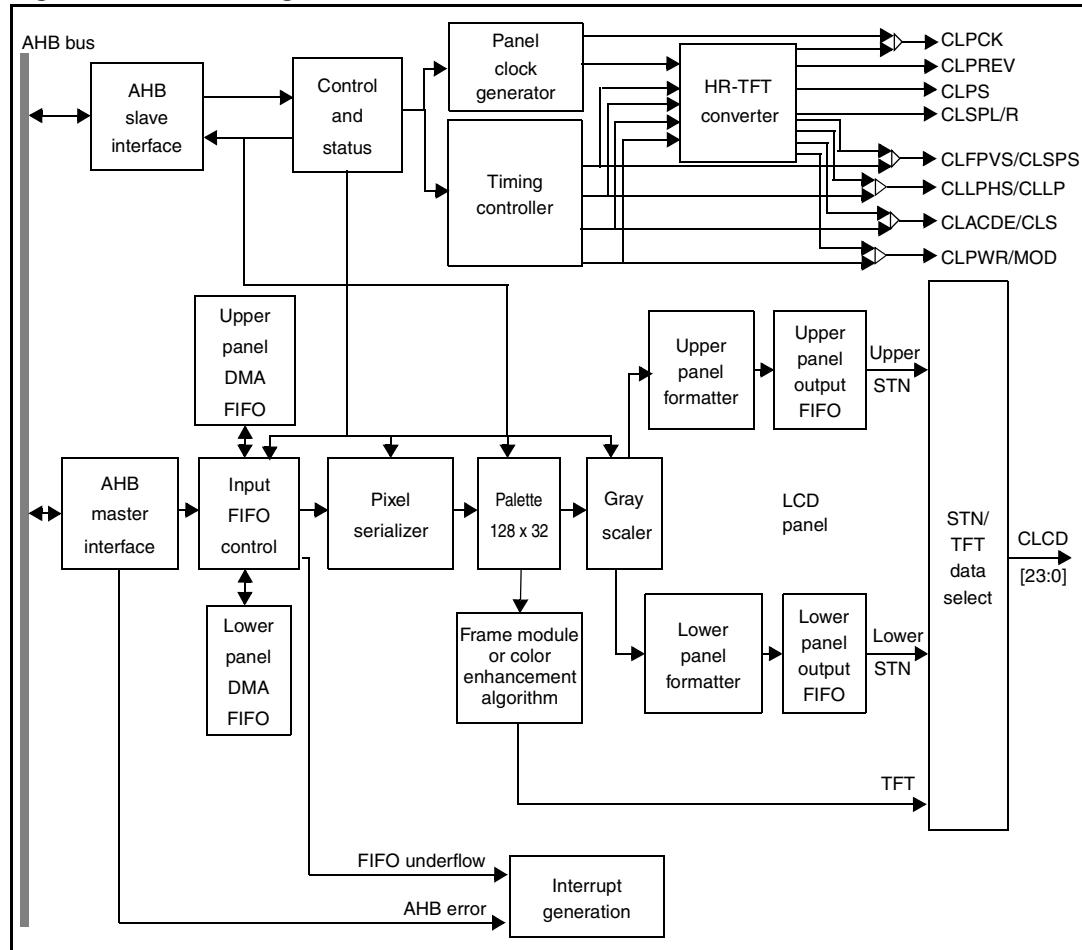
AD-TFT and HR-TFT timings are based on TFT waveforms. The LCDC must be programmed so that consistent TFT waveforms are generated internally (CLPCK, ENAB, syncs, and so on) before being transformed into HR-TFT waveforms, which require additional signals and timings.

TFT displays

These require the digital color value of each pixel to be applied to the display data inputs.

Figure 25 on page 138 illustrates the functional design of the LCDC.

Figure 25. Block diagram of LCDC



10.2.5 HR-TFT signals

HR-TFT panels have the following set of control signals.

- CLPCK. This is similar to the TFT clock signal but is active only when data is valid. Unlike the STN and TFT clock, CLPCK is not a continuous square waveform.
- CLLP. This is similar to the horizontal synchronization signal CLLPHS.
- CLSPS. This is similar to the vertical synchronization signal CLFPVS.
- CLS. This is the clock signal for the panel row gate drivers. The repeat rate is similar to the horizontal synchronization, but it has different timings.
- CLPS. This is the source driver control signal. The waveform shape is the same as CLS but its polarity is reversed.
- CLSPL. This is the source driver start signal. Its repeat rate is identical to the horizontal synchronization. It occurs just after the LP pulse to indicate the first valid data on the line. See note.
- CLSPR. This signal has the same function as SPL. See note.
- LBR. See note below.
- MOD. This signal controls the power down sequence of the HR-TFT panel and is identical to CLPWR.
- U/L. This is the vertical scan direction signal. It has a static value and as a consequence is managed at board level, and not by the LCDC.
- CLPREV. This is the polarity reversal signal. It is intended for input to a gray scaler chip associated with the panel. It works as an AC bias signal, switching at an horizontal line rate.

Note: *CSPL and CSPR are mutually exclusive signals. In conjunction with the LBR static level, they are used by the panel to decide the horizontal scanning direction. Consequently, the LCDC provides only one single CSPL/CSPR signal. LBR is managed on-board.*

10.2.6 Interrupts

The following individually maskable, active-high interrupts are generated by the LCDC. They are also output as a combined single interrupt CLCDINTR which is asserted if any of the individual interrupts are enabled and asserted.

Provision of individual outputs as well as a combined interrupt allows the use of either a global interrupt service routine, or modular device drivers to handle interrupts.

CLCDMIBEINTR

This is the master bus error interrupt. It is asserted when an ERROR response is received by the master interface during a transaction with a slave. When such an error is encountered, the master interface enters an error state and remains in this state until clearance of the error has been signaled. On completion of the relevant interrupt service routine, the interrupt must be cleared; this moves the master interface from the error state to the start of frame state, which allows a new frame of data to be initiated.

CLCDVCOMPINTR

This is the vertical compare interrupt. It is asserted when one of four vertical display regions is reached. The interrupt can be configured to be asserted at the start of any one of the following:

- vertical synchronization,
- back porch,
- active video,
- front porch.

CLCDLNUINTR

This is the panel next base address interrupt. It is asserted when the LCDC is ready to receive new frame base addresses, if required.

CLCDFUFINTR

This is the FIFO underflow interrupt. It is asserted when data is requested from an empty DMA FIFO. Internally, individual upper and lower panel DMA FIFO underflow interrupt signals are generated and CLCDFUFINTR is the single combined version of these.

10.3 Functional description

10.3.1 Data transfer

Pixel data is passed to the display as follows.

1. Packets of pixel coded data are fed, via the AMBA AHB interface, to two independent, programmable, 32-bit wide, DMA FIFOs which act as input data flow buffers.
2. The buffered pixel coded data is unpacked via a pixel serializer.
Depending on the panel type and mode, the unpacked data can represent:
 - an actual true display gray- or color value,
 - an address to a gray- or color value in 256 x 16-bit wide palette RAM.
3. The way the data is then passed to the display depends on the panel type, as follows.
 - For STN displays, either an addressed palette value or the true value is passed to the gray scaler hardware. The gray scaler then sequences the addressed pixels over a programmed number of frames to provide the effective display appearance.
 - For TFT displays, either an addressed palette value or the true color value is passed directly to the output display drivers, bypassing the gray scaler.

10.3.2 LCD power-up and power-down sequence

The LCDC is powered up as follows:

1. The VDD voltage is simultaneously applied to the STn8815A12 and the panel display logic. Signals CLLPHS/CLLP, CLPCK, CLFPVS/CLSPS, CLACDE/CLS and CLCD[23:0] are held low (inactive).
2. When the VDD has stabilized, signals CLLPHS/CLLP, CLPCK, CLFPVS/CLSPS and CLACDE/CLS, are made active. The CLCD[23:0] signals remain low (inactive).
3. When the signals in the previous step have stabilized, the contrast voltage (which is not controlled or supplied by the LCDC) is applied appropriately.
4. If required, a software timer provides the minimum display-specific delay between application of the control signals, and the application of power to the panel display. On completion of the software timer, power is applied to the panel, the CLPWR/MOD signal is set high, and the CLCD[23:0] signals are made active. The CLPWR/MOD signal should be used to gate the power to the LCD panel. HR-TFT panels must be connected to the MOD input.

The power down sequence is the reverse of the above, and must be strictly followed.

10.3.3 Dual DMA FIFOs and associated control logic

The pixel data accessed from memory is buffered by two DMA FIFOs which can be independently controlled to cover single- and dual-panel LCDs. Each FIFO is 16 words deep by 32 bits wide. In single panel mode, the two FIFOs can be cascaded to form, in effect, a 32-word deep FIFO. The input ports of the FIFOs are connected to the AMBA AHB interface, and the output port feeds the pixel serializer.

Synchronization logic is used to transfer the pixel data from the AMBA AHB HCLK clock domain to the CLCDCLK clock domain, the DMA FIFOs being clocked by HCLK.

The watermark level of each FIFO is set such that the FIFO requests data when at least four locations become available.

An interrupt signal (CLCDFUFINTR) is asserted if an attempt is made to read either of the two FIFOs when they are empty (that is, when an underflow condition has occurred).

10.3.4 Pixel serializer

This device reads the 32-bit wide panel data from the output port of the DMA FIFO, and extracts 24-, 16-, 8-, 4-, 2- or 1-bpp data, according to the current mode of operation. The LCDC supports big-endian, little-endian, and WinCE data formats. In dual-panel mode, data is alternately read from the upper and lower DMA FIFOs. Depending on the mode of operation, the extracted data may be used to point to a color/grayscale value in the palette RAM or may actually be a true-color value that can be directly applied to a panel input.

The tables below show the structure of the data in each DMA FIFO word, according to the endianness and bpp combinations. For each of the three supported data formats, the required data for each display pixel must be extracted from the data word.

The terminology used in the tables is:

- little-endian byte, little-endian pixel (LBLP) order,
- big-endian byte, big-endian pixel (BBBP) order,
- little-endian byte, big-endian pixel (LBBP) order: this is the WinCE format.

Little-endian byte, little-endian pixel order (LBLP)**Table 28.** LBLP, DMA FIFO output bits [31:16]

Bits per pixel	DMA FIFO output bits															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16
2	p15		p14		p13		p12		p11		p10		p9		p8	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p7				p6				p5				p4			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 non-packed	p0															
	nu	nu	nu	nu	nu	nu	nu	nu	23	22	21	20	19	18	17	16
24 packed case 1	p1								p0							
	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16
24 packed case 2	p2															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 3	p3															
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8

Table 29. LBLP, DMA FIFO output bits [15:0]

Bits per pixel	DMA FIFO output bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
2	p7		p6		p5		p4		p3		p2		p1		p0	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p2				p2				p1				p0			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 non-packed	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 1	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 2	p1															
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
24 packed case 3	p3								p2							
	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16

Big-endian byte, big-endian pixel order (BBBP)**Table 30. BBBP, DMA FIFO output bits [31:16]**

Bits per pixel	DMA FIFO output bits															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
2	p0		p1		p2		p3		p4		p5		p6		p7	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p0				p1				p2				p3			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p0								p1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 non-packed	p0															
	nu	nu	nu	nu	nu	nu	nu	nu	23	22	21	20	19	18	17	16
24 packed case 1	p0															
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
24 packed case 2	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 3	p2								p3							
	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16

Table 31. BBBP, DMA FIFO output bits [15:0]

Bits per pixel	DMA FIFO output bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31
2	p8		p9		p10		p11		p12		p13		p14		p15	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p4				p5				p6				p7			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p2								p3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 non-packed	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 1	p0								p1							
	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16
24 packed case 2	p2															
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
24 packed case 3	p3															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Little-endian byte, big-endian pixel order (LBBP)**Table 32. LBBP, DMA FIFO output bits [31:16]**

Bits per pixel	DMA FIFO output bits															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p24	p25	p26	p27	p28	p29	p30	p31	p16	p17	p18	p19	p20	p21	p22	p23
2	p12		p13		p14		p15		p8		p9		p10		p11	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p6				p7				p4				p5			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 non-packed	p0															
	nu	nu	nu	nu	nu	nu	nu	nu	23	22	21	20	19	18	17	16
24 packed case 1	p1								p0							
	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16
24 packed case 2	p2															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 3	p3															
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8

Table 33. LBBP, DMA FIFO output bits [15:0]

Bits per pixel	DMA FIFO output bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p8	p9	p10	p11	p12	p13	p14	p15	p0	p1	p2	p3	p4	p5	p6	p7
2	p4		p5		p6		p7		p0		p1		p2		p3	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p2				p3				p0				p1			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 non-packed	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 1	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24 packed case 2	p1															
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
24 packed case 3	p3								p2							
	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16

10.3.5 Palette RAM

The palette is 256 x 16-bit, dual-port RAM, physically structured as 128 x 32-bit. This allows two entries to be written into the palette from a single-word write access.

The least significant bit (LSB) of the serialized pixel data is used to select between the upper and lower halves of the palette RAM. The byte ordering mode determines the half that is selected:

- LSB is little-endian: upper half selected.
- LSB is big-endian: lower half selected.

WinCE byte ordering is little-endian, so the upper half is selected.

The palette RAM has independent controls and addresses for each port.

- Port 1 is a read/write port, connected to the AMBA AHB slave interface. The palette entries can be written and verified through this port.
- Port 2 is a read-only port, connected to the unpacker and gray scaler.

Table 34 shows the bit representation of each word in the palette.

Table 34. Palette data storage

Bits	Name	Description	
31:27	R[4:0]	Red palette data	Upper-half
26:21	G[5:0]	Green palette data	
20:16	B[4:0]	Blue palette data	
15:11	R[4:0]	Red palette data	Lower-half
10:5	G[5:0]	Green palette data	
4:0	B[4:0]	Blue palette data	

Note:

In monochrome STN mode, only the blue palette field bits B[4:1] are used.

In color STN mode, the green bits G[5:2] and red bits R[4:1] are also used.

The red and blue pixel data can be swapped to support the BGR data format.

In true-color modes (12, 15, and 16 bpp for color STN and TFT panels, and 24 bpp for TFT panels), the palette is bypassed.

10.3.6 Gray scaler

An ARM patented gray scaler drives monochrome and color STN panels.

- For monochrome displays, this provides 15 gray scales.
- For STN color displays, the three color components (red, green and blue) are gray scaled simultaneously which results in 3375 (15 x 15 x 15) colors being available.

The gray scaler transforms each 4-bit gray value into a pixel on-off sequence over several frames to give the representation of gray scales and color, relying to a certain extent on the display characteristics. *Table 35* shows the resulting intensities.

Table 35. Grayscale intensities from 4-bit palette

4-bit palette value	Duty cycle ⁽¹⁾	Resulting intensity
0000	0/90	0% (black)
0001	10/90	11.1%
0010	18/90	20.0%
0011	24/90	26.7%
0100	30/90	33.3%
0101	36/90	40.0%
0110	40/90	44.4%
0111	45/90	50.0%
1000	45/90	50.0%
1001	50/90	55.6%
1010	54/90	60.0%
1011	60/90	66.6%
1100	66/90	73.3%
1101	72/90	80.0%
1110	80/90	88.9%
1111	90/90	100.0% (white)

1. Duty cycle = (pixel on / (pixel on + pixel off))

In palettized mode, the gray scaler input bits are delivered by the palette:

- For monochrome STN: bits [4:1] and [20:17] of the palette RAM (corresponding to the blue palette B[4:1]).
- For color STN: bits ([4:1], [10:7], [15:12]) and ([20:17], [26:22], [31:28]) of the palette RAM (corresponding to the palettes blue B[4:1], green G[5:2] and red R[4:1]).

In true-color (non-palettized) mode, the gray scaler inputs bits as a 16-bit data frame delivered by the FIFO, reformatted as follows:

- For 12-bpp 4:4:4 data, all blue, red and green bits (R[3:0], G[3:0], B[3:0]) are sent to the gray scaler.
- For 15-bpp 1:5:5:5 data, only the upper four bits of the blue, red and green fields (R[4:1], G[4:1], B[4:1]) are sent to the gray scaler.
- For 16-bpp 5:6:5 data, only the upper four bits of the blue, red and green fields (R[4:1], G[5:2], B[4:1]) are sent to the gray scaler.

The following tables show which bits of the incoming data frame are used by the gray scaler.

Table 36. 12-bpp (4:4:4) data usage for gray scaler

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data frame from FIFO	unused			R3	R2	R1	R0	G3	G2	G1	G0	B3	B2	B1	B0	
Gray scaler	nu			Red				Green				Blue				

Table 37. 15-bpp (I:5:5:5) data usage for gray scaler

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data frame from FIFO	I	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
Gray scaler	nu	Red				nu	Green				nu	Blue				nu

Table 38. 16-bpp (5:6:5) data usage for gray scaler

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data frame from FIFO	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
Gray scaler	Red				nu	Green				nu	nu	Blue				nu

10.3.7 True color support for 12-, 15-, 16- and 18-bit panels

Often, TFT panels have only a 12-, 15-, 16- or 18-bit interface, but images are higher resolution. The LCDC can reformat a higher resolution image to a lower interface width, either by discarding the lower bits of the pixel components, or by using frame modulation to improve the image quality.

Frame modulation

Frame modulation can be used in non-palettized modes to emulate 24-bpp (R8:G8:B8) true color support on 12-bit (R3:G3:B3), 15-bit (I:R5:G5:B5), 16-bit (R5:G6:B6) or 18-bit (R6:G6:B6) panels.

Frame modulation usually applies to a source frame of 24-bpp (as described in the tables below), but it can be used for source frames in 6:6:6, 5:6:5 or I:5:5:5 formats (for addressing 5- or 4-bit panels).

Like the grey scaler, frame modulation transforms the least significant bits of a pixel color component (R, G or B) into an intensity, which is added to the most significant bits. In other words, the color value sent to the LCD panel changes from frame to frame, based on the values of the less significant bits. This takes advantage of the low-pass filtering effect of the panel to modulate the color value and change its effective rendering.

The following tables illustrate frame modulation when the number of LSBs taken into account is one, two, three or four.

- Note:
- 1 *The software must ensure the consistency of frame modulation.*
 - 2 *Frame modulation can be reset in software, which can avoid artefacts when changing the image on display.*

Table 39. 1-bit frame modulation on a color component

Color's less significant bits	Intensity increase	Bits sent to panel for 2 consecutive frames (x=MSBs from FIFO) (1) (2) (3)
0	0%	x, x
1	50%	x+1, x

1. The color value is sent to the panel on a frame sync basis.
2. The '+1' operation means an increase in intensity. The real arithmetic operation performed on the bit field is endianness dependent. The increment is performed with a saturation effect (see following note).
3. If the remaining MSBs from the FIFO already define the maximum color intensity, then no frame modulation is done.

Table 40. 2-bit frame modulation on a color component

Color's 2 less significant bits	Intensity increase	Bits sent to panel for 4 consecutive frames (x=MSBs from FIFO) ^{(1) (2) (3)}
00	0%	x, x, x, x
01	25%	x+1, x, x, x
10	50%	x+1, x, x+1, x
11	75%	x+1, x+1, x+1, x

1. The color value is sent to the panel on a frame sync basis.
2. The ' +1' operation means an increase in intensity. The real arithmetic operation performed on the bit field is endianness dependent. The increment is performed with a saturation effect (see following note).
3. If the remaining MSBs from the FIFO already define the maximum color intensity, then no frame modulation is done.

Table 41. 3-bit frame modulation on a color component

Color's 3 less significant bits	Intensity increase	Bits sent to panel for 8 consecutive frames (x=MSBs from FIFO) ^{(1) (2) (3)}
000	0%	x, x, x, x, x, x, x, x
001	12.5%	x+1, x, x, x, x, x, x, x
010	25%	x+1, x, x, x, x+1, x, x, x
011	37.5%	x+1, x, x, x+1, x, x, x+1, x
100	50%	x+1, x, x+1, x, x+1, x, x+1, x
101	62.5%	x+1, x+1, x, x+1, x+1, x, x+1, x
110	75%	x+1, x+1, x+1, x, x+1, x+1, x+1, x
111	87.5%	x+1, x+1, x+1, x+1, x+1, x+1, x+1, x

1. The color value is sent to the panel on a frame sync basis.
2. The ' +1' operation means an increase in intensity. The real arithmetic operation performed on the bit field is endianness dependent. The increment is performed with a saturation effect (see following note).
3. If the remaining MSBs from the FIFO already define the maximum color intensity, then no frame modulation is done.

Table 42. 4-bit frame modulation on a color component

Color's 4 LSB	Intensity increase	Bits sent to panel for 16 consecutive frames (x=MSBs from FIFO) ^{(1) (2)} ⁽³⁾
0000	0%	x, x
0001	6.25%	x+1, x, x
0010	12.5%	x+1, x, x, x, x, x, x, x, x+1, x, x, x, x, x, x, x
0011	18.75%	x+1, x, x, x, x+1, x, x, x, x+1, x, x, x, x, x, x
0100	25%	x+1, x, x, x, x+1, x, x, x, x+1, x, x, x, x+1, x, x, x
0101	31.25%	x+1, x, x+1, x, x+1, x, x, x+1, x, x, x+1, x, x, x
0110	37.5%	x+1, x, x+1, x, x+1, x, x+1, x, x+1, x, x, x, x+1, x, x, x
0111	43.75%	x+1, x, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x, x, x
1000	50%	x+1, x, x+1, x
1001	56.25%	x+1, x+1, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x
1010	62.5%	x+1, x+1, x+1, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x
1011	68.75%	x+1, x+1, x+1, x+1, x+1, x+1, x, x+1, x, x+1, x, x+1, x, x+1, x
1100	75%	x+1, x+1, x+1, x+1, x+1, x+1, x+1, x+1, x, x+1, x, x+1, x, x+1, x
1101	81.25%	x+1, x, x+1, x, x+1, x
1110	87.5%	x+1, x, x+1, x
1111	93.75%	x+1, x+1

1. 5-bit color value is sent to the panel on a frame sync basis.
2. The '+1' operation means an increase in intensity. The real arithmetic operation performed on the bit field is endianness dependent. The increment is performed with a saturation effect (see following note).
3. If the remaining 5 bits from the FIFO already define the maximum color intensity, then no frame modulation is done.

Effect of frame modulation on different BPP and panel configurations

Table 43 shows which type of modulation is used for each combination of bpp and panel.

Table 43. Frame modulation

Bpp	Panel configuration				
	8:8:8	6:6:6	5:6:5	1:5:5:5	4:4:4
24	n/a	2-bit FRM	R, B: 3-bit FRM G: 2-bit FRM	3-bit FRM	4-bit FRM
18	n/a	n/a	R, B: 1-bit FRM G: n/a	1-bit FRM	2-bit FRM
16	n/a	n/a	n/a	R, B: n/a G: 1-bit FRM	R, B: 1-bit FRM G: 2-bit FRM
15	n/a	n/a	n/a	n/a	1-bit FRM
12	n/a	n/a	n/a	n/a	n/a

10.3.8 Color enhancement from 16-bpp to 18-bpp (for 18-bit TFT panels)

Due to frame-buffer size reduction and memory transfer bandwidth limitations, frame buffers sometimes only use 16-bpp resolution but the TFT-LCD panel has an 18-bit interface.

16-bpp (RGB 5:6:5) to 18-bpp (RGB 6:6:6) color conversion is performed on the fly by an automatic color enhancement conversion algorithm (patent pending). This displays the image with maximum color resolution.

Note: *Maximum panel horizontal resolution is limited to 800 pixels per line when this color enhancement algorithm (16-bpp to 18-bpp) feature is enabled.*

10.3.9 Upper and lower panel formatters

Each formatter consists of three 3-bit (red, green and blue) shift-left registers. Red, green and blue pixel bit values from the gray scaler are concurrently shifted into the relevant registers. When enough data is available, a byte is constructed by multiplexing the registered data to the correct bit position to satisfy the RGB data pattern of the panel. The byte is transferred to the 3-byte output FIFO, which can store eight color pixels.

10.3.10 Panel clock generator

The panel clock is produced by the panel clock generator, and is a divided version of CLCDCLK. It can be programmed in the range CLCDCLK (divider bypassed) or CLCDCLK/2 to CLCDCLK/1025 to match the data rate of the LCD panel.

10.3.11 Timing controller

The primary function of the timing controller is to generate the horizontal and vertical timing panel signals. It also provides the panel bias/enable signal. These timings are all programmable via the AMBA AHB slave interface.

10.3.12 Vertical compare event

The vertical compare event is asserted when one of four vertical display regions is reached. The event can be made to occur at the start of:

- vertical synchronization,
- vertical back porch,
- vertical active video,
- vertical front porch.

The event is then used by the interrupt logic block:

- to set the interrupt CLCDVCOMPINTR. Once set, this interrupt must be cleared by software: see [Section 10.2.6 on page 139](#).
- to toggle the intra-chip signal CLCDVCOMPSYNC. This signal is used to synchronize the video refresh performed by the smart video accelerator (SVA) on panel refresh. CLCDVCOMPSYNC is then toggled each time the vertical compare event occurs. This signal is synchronous with the CLCDCLK clock.

10.4 Data bus usage for STN panels

Table 44 shows which CLCD[23:0] pins supply the pixel data to the STN panel.

Table 44. CLCD STN panel signal connection

Pin	Color STN		4-bit monochrome STN		8-bit monochrome STN	
	Single panel	Dual panel	Single panel	Dual panel	Single panel	Dual panel
CLCD23						
CLCD22						
CLCD21						
CLCD20						
CLCD19						
CLCD18						
CLCD17						
CLCD16						
CLCD15		CLSTN0				MLSTN0
CLCD14		CLSTN1				MLSTN1
CLCD13		CLSTN2				MLSTN2
CLCD12		CLSTN3				MLSTN3
CLCD11		CLSTN4		MLSTN0		MLSTN4
CLCD10		CLSTN5		MLSTN1		MLSTN5
CLCD9		CLSTN6		MLSTN2		MLSTN6
CLCD8		CLSTN7		MLSTN3		MLSTN7
CLCD7	CUSTN0	CUSTN0			MUSTN0	MUSTN0
CLCD6	CUSTN1	CUSTN1			MUSTN1	MUSTN1
CLCD5	CUSTN2	CUSTN2			MUSTN2	MUSTN2
CLCD4	CUSTN3	CUSTN3			MUSTN3	MUSTN3
CLCD3	CUSTN4	CUSTN4	MUSTN0	MUSTN0	MUSTN4	MUSTN4
CLCD2	CUSTN5	CUSTN5	MUSTN1	MUSTN1	MUSTN5	MUSTN5
CLCD1	CUSTN6	CUSTN6	MUSTN2	MUSTN2	MUSTN6	MUSTN6
CLCD0	CUSTN7	CUSTN7	MUSTN3	MUSTN3	MUSTN7	MUSTN7

Note: CUSTNx: color upper panel STN, dual and/or single panel bit x.

CLSTNx: color lower panel STN, dual panel bit x.

MUSTNx: monochrome upper panel STN, dual and/or single panel bit x.

MLSTNx: monochrome lower panel STN, dual panel bit x.

10.5 Data bus usage for TFT panels

For TFT panels, the LCDC supports palettized graphics depths of 2, 4 and 8 bpp (256 color palette), and true-color graphics depths of 12, 15 (+intensity), 16 and 24 bpp.

For 24 bpp depth, pixels can be packed or non-packed.

- When non-packed, only the 24 least significant bits of the 32-bit memory word are used.
- When packed, the remaining 8 bits are used to store the first part of the next pixel.

10.5.1 Data bus usage for different graphics modes

The tables below detail the data bus usage for the different graphics modes: 12 bpp (4:4:4), 15+Intensity bpp (I:5:5:5), 16 bpp (5:6:5) and 24 bpp (8:8:8).

Note: In these modes, the CLCD[23:18] signals are low when delivered off-chip.

12-bpp (4:4:4) mode

[Table 45](#) shows the data bus usage for 12 bpp (4:4:4) mode. Note that for some bus widths the data is routed to the pins CLCD[17:0] by duplicating some of the input bits, as shown.

Table 45. 12-bpp (4:4:4) data bus usage

Pins CLCDx	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory half-word	Unused												R3	R2	R1	R0	G3	G2	G1	G0	B3	B2	B1	B0
12-bit bus input bits	0	0	0	0	0	0	0	0	0	0	0	0	R3	R2	R1	R0	G3	G2	G1	G0	B3	B2	B1	B0
16-bit bus input bits	0	0	0	0	0	0	0	R3	R2	R1	R0	R3	G3	G2	G1	G0	G3	G2	B3	B2	B1	B0	B3	
18-bit bus input bits	0	0	0	0	0	0	R3	R2	R1	R0	R3	R2	G3	G2	G1	G0	G3	G2	B3	B2	B1	B0	B3	
24-bit bus input bits	R3	R2	R1	R0	R3	R2	R1	R0	G3	G2	G1	G0	G3	G2	G1	G0	B3	B2	B1	B0	B3	B2	B1	B0

15+Intensity-bpp (I:5:5:5) mode

Table 46 shows the data bus usage for 15+Intensity bpp (I:5:5:5) mode. Note that for some bus widths the data is routed to the pins CLCD[17:0] by duplicating some of the input bits, as shown.

Table 46. 15-bpp (I:5:5:5) data bus usage

Pins CLCDx	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Memory half-word	Unused								I	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	
12-bit bus input bits	0	0	0	0	0	0	0	0	0	0	0	0	R4	R3	R2	R1	G4	G3	G2	G1	B4	B3	B2	B1	B1
16-bit bus input bits	0	0	0	0	0	0	0	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	I	B4	B3	B2	B1	B0		
18-bit bus input bits	0	0	0	0	0	0	R4	R3	R2	R1	R0	I	G4	G3	G2	G1	G0	I	B4	B3	B2	B1	B0	I	
24-bit bus input bits	R4	R3	R2	R1	R0	R4	R3	I	G4	G3	G2	G1	G0	G4	G3	I	B4	B3	B2	B1	B0	B4	B3	I	

16-bpp (5:6:5) mode

Table 47 shows the data bus usage for 16 bpp (5:6:5) mode. Note that for some bus widths the data is routed to the pins CLCD[17:0] by duplicating some of the input bits, as shown.

Table 47. 16-bpp (5:6:5) data bus usage

Pins CLCDx	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Memory half-word	Unused								R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	
12-bit bus input bits	0	0	0	0	0	0	0	0	0	0	0	0	R4	R3	R2	R1	G5	G4	G3	G2	B4	B3	B2	B1	B1
16-bit bus input bits	0	0	0	0	0	0	0	0	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	
18-bit bus input bits	0	0	0	0	0	0	R4	R3	R2	R1	R0	R4	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	B4	
24-bit bus input bits	R4	R3	R2	R1	R0	R4	R3	R2	G5	G4	G3	G2	G1	G0	G5	G4	B4	B3	B2	B1	B0	B4	B3	B2	

24-bpp (8:8:8) mode

Table 48 shows the data bus usage for 24 bpp (8:8:8) mode. This applies to both packed and non-packed data.

Table 48. 24-bpp (8:8:8) data bus usage

Pins CLCDx	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory half-word	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
12-bit bus input bits	0	0	0	0	0	0	0	0	0	0	0	R7	R6	R5	R4	G7	G6	G5	G4	B7	B6	B5	B4	
16-bit bus input bits	0	0	0	0	0	0	0	R7	R6	R5	R4	R3	G7	G6	G5	G4	G3	G2	B7	B6	B5	B4	B3	
18-bit bus input bits	0	0	0	0	0	0	R7	R6	R5	R4	R3	R2	G7	G6	G5	G4	G3	G2	B7	B6	B5	B4	B3	
24-bit bus input bits	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

10.5.2 TFT panel connections for different data bus widths

The tables below detail how the LCDC pins are connected to a TFT panel for different data bus widths: 12-, 16-, 18- and 24-bit.

Note that connecting a 15-, 18- or 24-bit panel requires some duplicated connections; this is preferable to hard wiring the extra panel bits to a fixed level (this connection scheme improves the dynamic range and removes color errors).

12-bit wide bus

When the selected LCDC bus width is 12 bits, [Table 49](#) shows how the CLCD[15:0] pins must be connected to the TFT panel for different color depths (12- and 15-bit panel).

Table 49. TFT panel connection when LCDC bus width is 12 bits

Pin	TFT 24-bit panel	TFT 18-bit panel	TFT 15-bit panel	TFT 12-bit panel
CLCD15	Not connected	Not connected	Not connected	Not connected
CLCD14	Not connected	Not connected	Not connected	Not connected
CLCD13	Not connected	Not connected	Not connected	Not connected
CLCD12	Not connected	Not connected	Not connected	Not connected
CLCD11	red7, red3	red5, red1	red4, red0	red3
CLCD10	red6, red2	red4, red0	red3	red2
CLCD9	red5, red1	red3	red2	red1
CLCD8	red4, red0	red2	red1	red0
CLCD7	green7, green3	green5, green1	green4, green0	green3
CLCD6	green6, green2	green4, green0	green3	green2
CLCD5	green5, green1	green3	green2	green1
CLCD4	green4, green0	green2	green1	green0
CLCD3	blue7, blue3	blue5, blue1	blue4, blue0	blue3
CLCD2	blue6, blue2	blue4, blue0	blue3	blue2
CLCD1	blue5, blue1	blue3	blue2	blue1
CLCD0	blue4, blue0	blue2	blue1	blue0

16-bit wide bus

When the selected LCDC bus width is 16 bits, [Table 50](#) shows how the CLCD[15:0] pins have to be connected to the TFT panel for different color depths (15-, 18- and 24-bit panel).

Table 50. TFT panel connection when LCDC bus width is 16 bits

Pin	TFT 24-bit panel	TFT 18-bit panel	TFT 15-bit panel
CLCD15	red7, red2	red5, red0	red4
CLCD14	red6, red1	red4	red3
CLCD13	red5, red0	red3	red2
CLCD12	red4	red2	red1
CLCD11	red3	red1	red0
CLCD10	green7, green1	green5	green4
CLCD9	green6, green0	green4	green3
CLCD8	green5	green3	green2
CLCD7	green4	green2	green1
CLCD6	green3	green1	green0
CLCD5	green2	green0	Not connected
CLCD4	blue7, blue2	blue5, blue0	blue4
CLCD3	blue6, blue1	blue4	blue3
CLCD2	blue5, blue0	blue3	blue2
CLCD1	blue4	blue2	blue1
CLCD0	blue3	blue1	blue0

18-bit wide bus

When the selected LCDC bus width is 18 bits, [Table 51](#) shows how the CLCD[17:0] pins have to be connected to the TFT panel for different color depths (18- and 24-bit panel).

Table 51. TFT panel connection when LCDC bus width is 18 bits

Pin	TFT 24-bit panel	TFT 18-bit panel
CLCD17	red7, red1	red5
CLCD16	red6, red0	red4
CLCD15	red5	red3
CLCD14	red4	red2
CLCD13	red3	red1
CLCD12	red2	red0
CLCD11	green7, green1	green5
CLCD10	green6, green0	green4
CLCD9	green5	green3
CLCD8	green4	green2
CLCD7	green3	green1
CLCD6	green2	green0
CLCD5	blue7, blue1	blue5
CLCD4	blue6, blue0	blue4
CLCD3	blue5	blue3
CLCD2	blue4	blue2
CLCD1	blue3	blue1
CLCD0	blue2	blue0

24-bit wide bus

When the selected LCDC bus width is 24 bits, [Table 52](#) shows which CLCD[23:0] pins have to be connected to the 24-bit TFT panel.

Table 52. TFT panel connection when LCDC bus width is 24 bits

Pin	TFT 24-bit panel
CLCD23	red7
CLCD22	red6
CLCD21	red5
CLCD20	red4
CLCD19	red3
CLCD18	red2
CLCD17	red1
CLCD16	red0
CLCD15	green7
CLCD14	green6
CLCD13	green5
CLCD12	green4
CLCD11	green3
CLCD10	green2
CLCD9	green1
CLCD8	green0
CLCD7	blue7
CLCD6	blue6
CLCD5	blue5
CLCD4	blue4
CLCD3	blue3
CLCD2	blue2
CLCD1	blue1
CLCD0	blue0

10.6 Timing waveforms

This section provides timing diagrams for STN, TFT and HR-TFT panels.

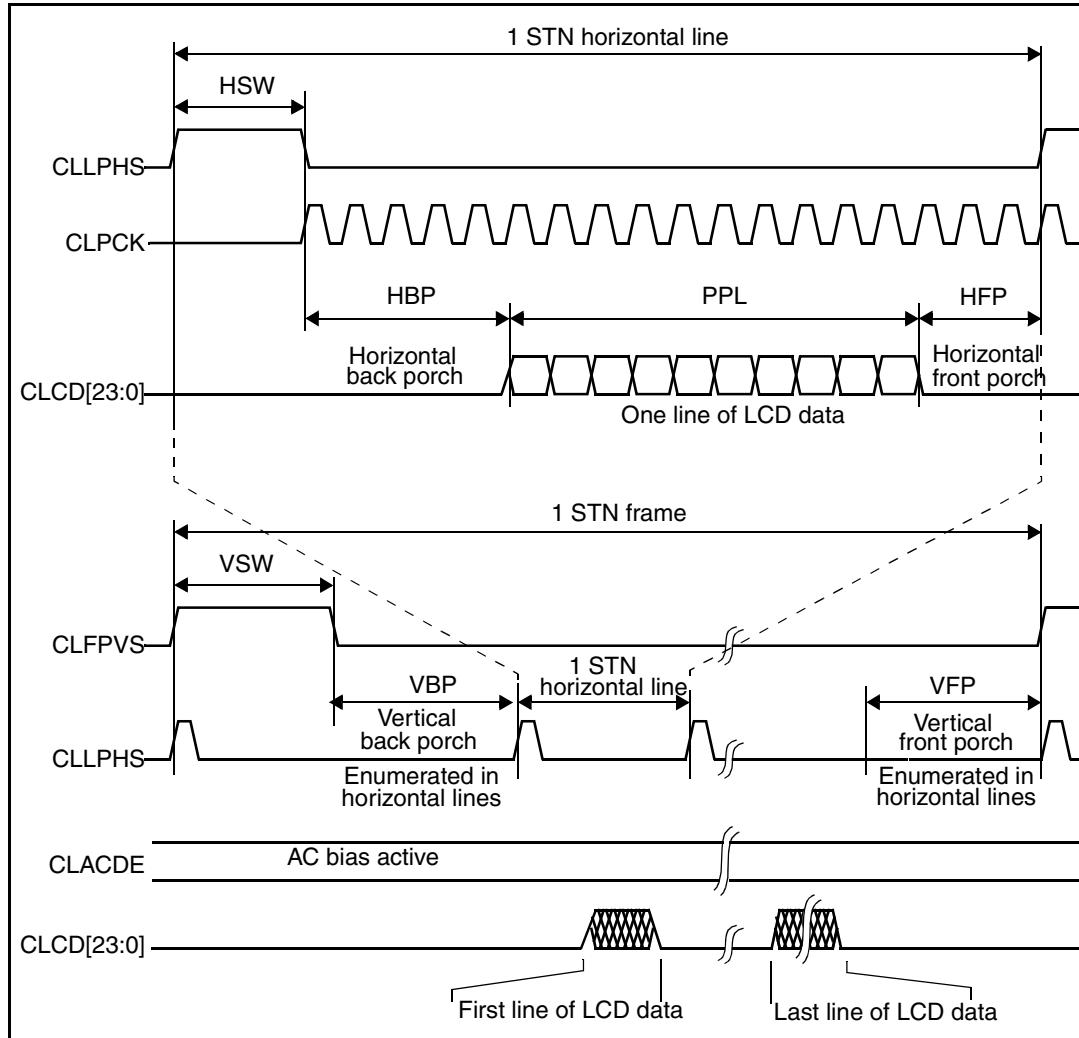
The following terms are used in these diagrams:

- **HBP:** The number of panel clock (CLPCK) periods inserted between the last line clock synchronization pulse and the start of valid data (on CLCD[23:0]) of the current line.
- **HFP:** The number of panel clock (CLPCK) periods inserted between the end of valid data (on CLCD[23:0]) of the current line and the next line clock synchronization pulse.
- **HSW:** The width of the CLLPHS/LP pulse (line clock in passive mode, or the horizontal synchronization pulse in active mode), expressed in CLPCK periods.
- **PPL:** The number of pixels in each line or row of the panel. PPL is a 16-bit value that represents between 16 and 1024 pixels per line. It is used to count the number of pixel clocks that occurs before the HFP is applied.
- **VBP:** The number of inactive lines inserted at the beginning of each frame, after the vertical synchronization period (falling edge of CLFPVS/rising edge of SPS). Its units are specified in horizontal line clock cycles. VBP generates from 0 to 255 line clock cycles.
In STN modes, to avoid reduced contrast, VBP should be 0.
- **VFP:** The number of line clock cycles inserted at the end of each frame. After the count has elapsed, the vertical synchronization signal is asserted (TFT and HR-TFT), or extra line clocks are inserted as specified by the VSW (STN). VFP generates from 0 to 255 line clock cycles.
- **VSW:** The width of the vertical synchronization pulse, in line clock cycles. For STN panels, this value must be small (zero, for example): as the value gets higher, the contrast gets worse.
- **LPP:** The number of active lines per panel.
- **REVDEL:** the delay in CLPCK periods of the polarity reversal signal from the rising edge of CLLPHS.
- **CLSDEL:** CLSDEL is the delay in CLPCK periods of the CLS and PS signals from the rising edge of CLLPHS. CLSDEL2 is the delay in CLPCK periods from the rising edge of the SPL/SPR signal to the falling edge of the CLS signal.
- **LPDEL:** The delay in CLPCK period of the line pulse LP signal from the rising edge of CLLPHS.
- **SPLDEL:** The delay in CLPCK period of the SPL signal during vertical front and back porch. This must be greater than or equal to HSW+HBP.

10.6.1 STN timings

Figure 26 presents typical horizontal and vertical timing waveforms for STN panels.

Figure 26. STN horizontal and vertical timings



The active data lines depend on whether the STN mode is 4- or 8-bit, color or monochrome.

10.6.2 Horizontal timing restrictions for STN panels

The DMA requests new data at the start of a horizontal display line. Some time must be allowed for the DMA transfer and for the data to propagate down the FIFO path in the LCD interface. The data path latency forces some restrictions on the usable minimum values for horizontal porch width in STN mode. The minimum values are:

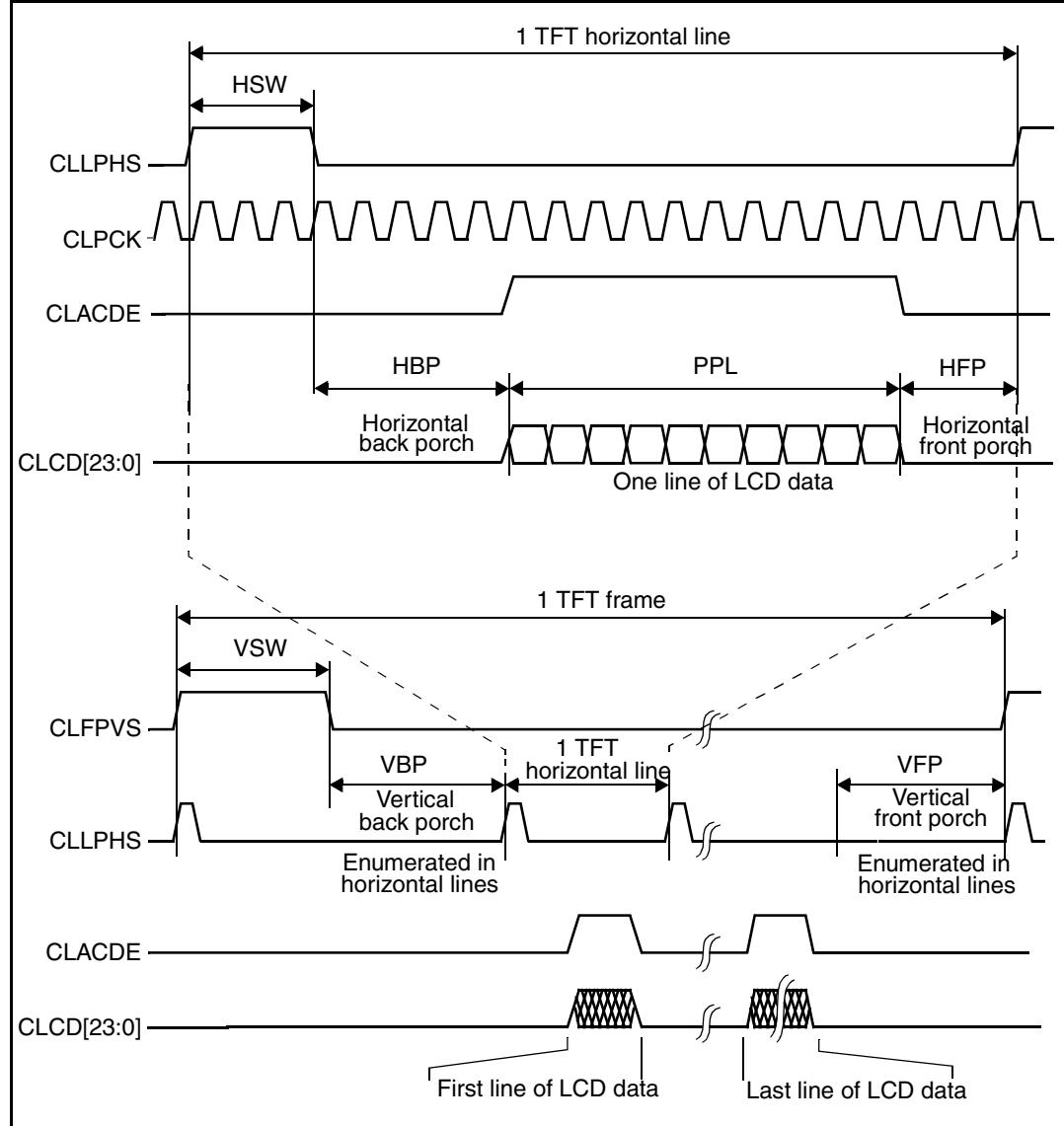
- Single panel mode: HSW = 3, HBP = 5, HFP = 5.
- Dual panel mode: HSW = 3, HBP = 5, HFP = 5.

CLPCK is stopped during CLLPHS.

10.6.3 TFT timings

Figure 27 presents typical horizontal and vertical timing waveforms for TFT panels.

Figure 27. TFT horizontal and vertical timings



10.6.4 HR-TFT timings

Figure 28 and *Figure 29* present typical horizontal and vertical timing waveforms for HR-TFT panels.

Figure 28. HR-TFT horizontal timings

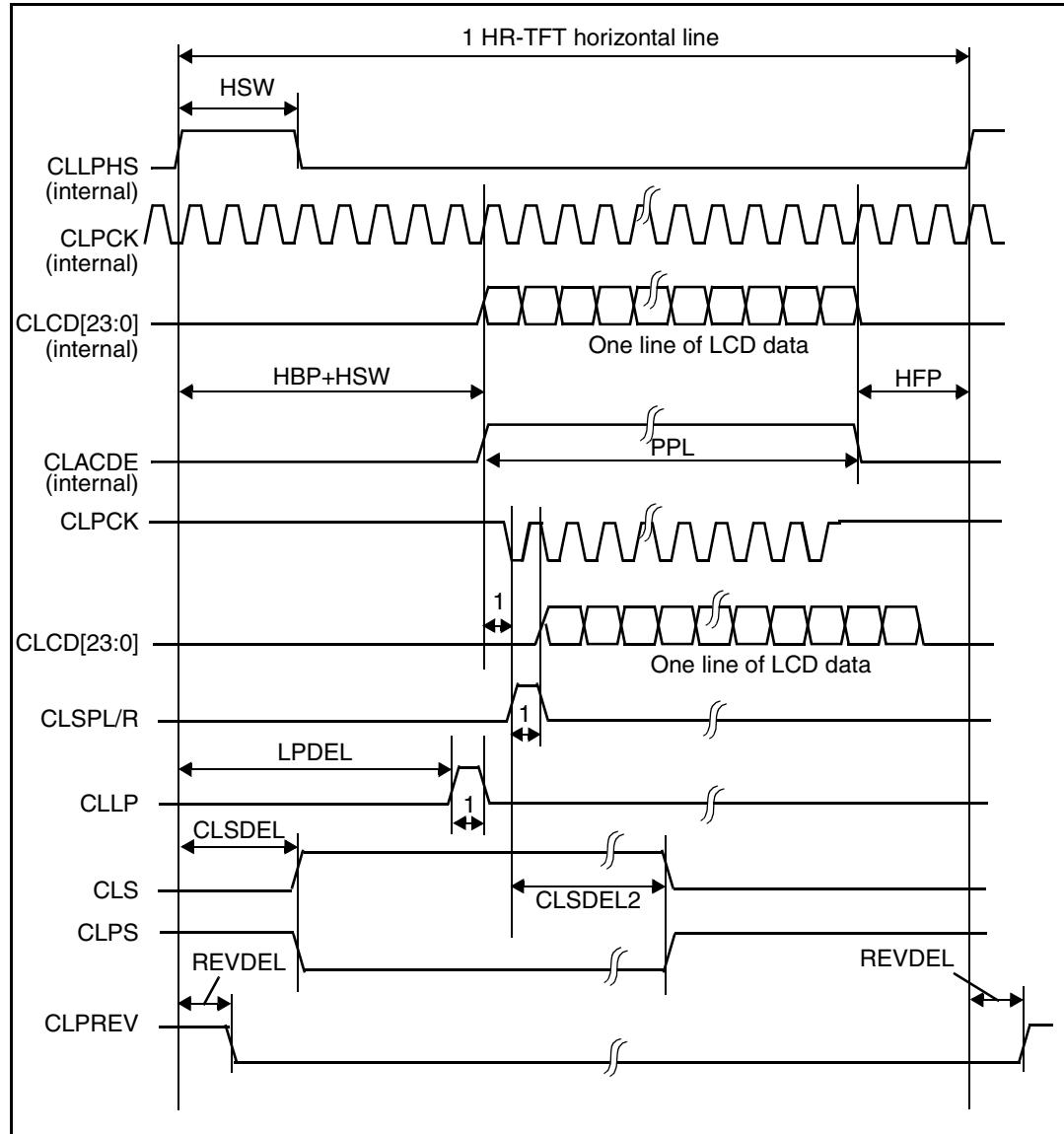
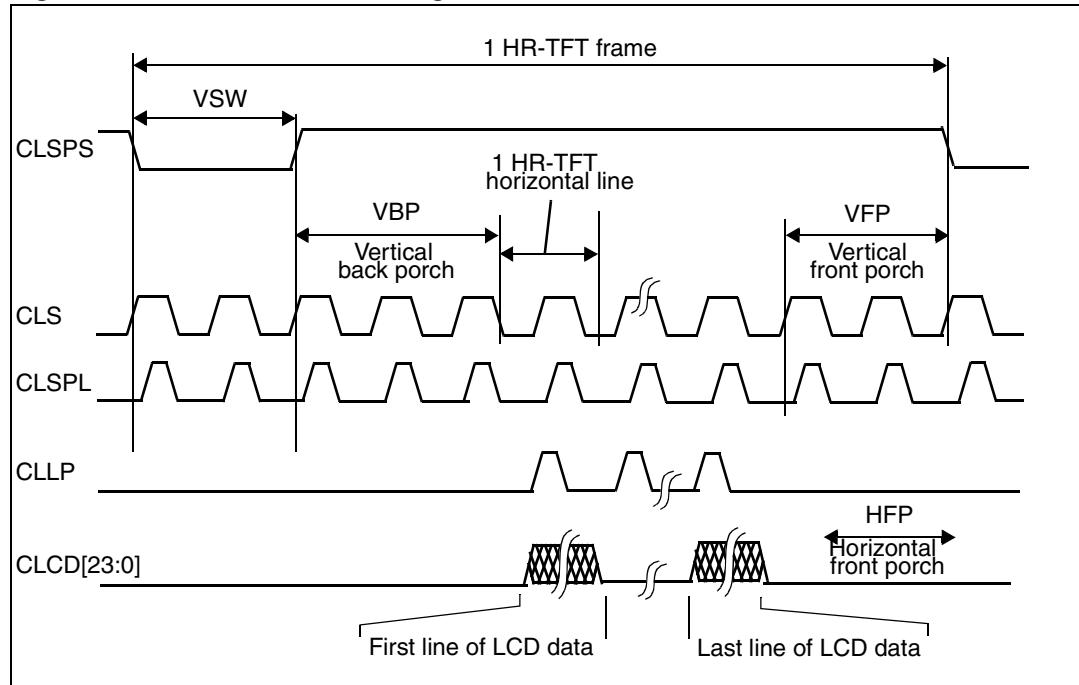


Figure 29. HR-TFT vertical timings

Some HR-TFT signals are multiplexed in the following signals:

- CLS/CLACDE
- CLLP/CLLPHS
- CLSPS/CLFPVS
- MOD/CLPWR

CL SPL/R and CL PREV are delivered on alternate functions of GPIOs.

10.7 LCDC signals

Table 53 summarizes the signals delivered by the LCDC. The pins are shared with signals used by the master display interface (MDIF), and many pins are alternate functions of GPIO pins.

Table 53. LCDC signals

Signal	I/O	Description
CLACDE/CLS	O	STN (CLACDE): ac bias drive. TFT (CLACDE): data enable. HR-TFT (CLS): gate driver clock.
CLFPVS/CLSPS	O	STN (CLFPVS): frame pulse. TFT (CLFPVS) and HR-TFT (CLSPS): vertical synchronization pulse.
CLLPHS/CLLP	O	STN (CLLPHS) and HR-TFT (CLLP): line synchronization pulse. TFT (CLLPHS): horizontal synchronization pulse.
CLPCK	O	Panel clock.
CLPS	O	HR-TFT: PS signal.
CLPWR/MOD	O	Panel power enable. For HR-TFT panels, this should be connected to the MOD signal.
CLCD23	I	B: Panel data, upper bits for 24-bit TFT display panel.
CLCD22	O	
CLCD21	O	
CLCD20	O	
CLSPL/R	O	A: HR-TFT: line start pulse left/right.
CLCD19	O	B: CLCD19.
CLPREV	O	A: HR-TFT: polarity reversal.
CLCD18	O	B: CLCD18.
CLCD17	O	A: CLCD upper bits for 18-bit TFT display panel.
CLCD16	O	
CLCD[15:0]	IO	Panel output data bits.

11 Multi-timer units (MTUs)

The STn8815A12 platform has two identical multi-timer units (MTUs), providing a total of eight timers. This section describes the functionality and signals of one unit (four timers), but this information is equally applicable to both units:

- MTU0 (timers 1 to 3)
- MTU1 (timers 4 to 7)

11.1 Overview

The MTU provides access to four interrupt generating, programmable, 32-bit, free-running decrementing counters (FRCs). The FRCs have their own clock input, allowing the counters to run from a much slower clock than the system clock.

The FRC is the part of the timer that performs the counting. There are four FRC blocks in the MTU, allowing four counts to be performed in parallel. The 32-bit counter in the FRC is split into two 16-bit counters.

Each of the four timers within the MTU is clocked by HCLK. The timer module enable signals, TIMCLKEN x ($x = 0$ to 7), are generated by the system controller. The enable pulses can be generated at the frequency of either the platform's TIMCLK (MXTAL frequency divided by 8; that is, 2.4 MHz with a 19.2 MHz crystal) or CLK32K (32.768 kHz) inputs.

The combined interrupt, INTCTC, is connected to the vectored interrupt controller (VIC). This signal is the logical OR of the individual interrupts (INTCT x) from the timers.

Note:

The timers are stalled when the ARM processor is in debug mode.

11.2 Interrupts

An interrupt is generated when the full 32-bit counter reaches zero.

The pre-scaled timer clock is fed to the least significant bit of the counter, with the carry bits of the least significant 16-bit counter being passed to the most significant counter. The interrupt is generated from the most significant carry bit, which indicates that the whole counter has reached zero.

The interrupts from the counters can be individually masked. Both the raw interrupt status (prior to masking) and the final interrupt status (after masking) can be read.

The interrupts from the individual counters (after masking) are logically ORed into a combined interrupt, INTCTC, which is routed to the vectored interrupt controller (VIC).

11.3 Functional description

The operation of each timer is identical. The timer is loaded with a starting count value, and once enabled, counts down to zero. When zero is reached, an interrupt can be generated.

The counter can operate in one of three modes:

- **Free-running mode:** In free-running mode, the counter always counts down from the maximum value that it can hold: 0xFFFF for a 16-bit counter or 0xFFFF FFFF for a 32-bit counter.

bit counter. On reaching zero, the timer wraps around, and starts to decrement from the maximum value again. This is the default mode of the timer.

- **Periodic mode:** In periodic mode, on reaching zero the timer generates an interrupt, reloads the starting count value, and continues to decrement. In this mode, the counter effectively generates a periodic interrupt.
- **One-shot mode:** In one-shot mode, the counter stops upon reaching zero until one-shot mode is deselected, or a new starting count value is loaded. An interrupt is also generated on reaching zero.

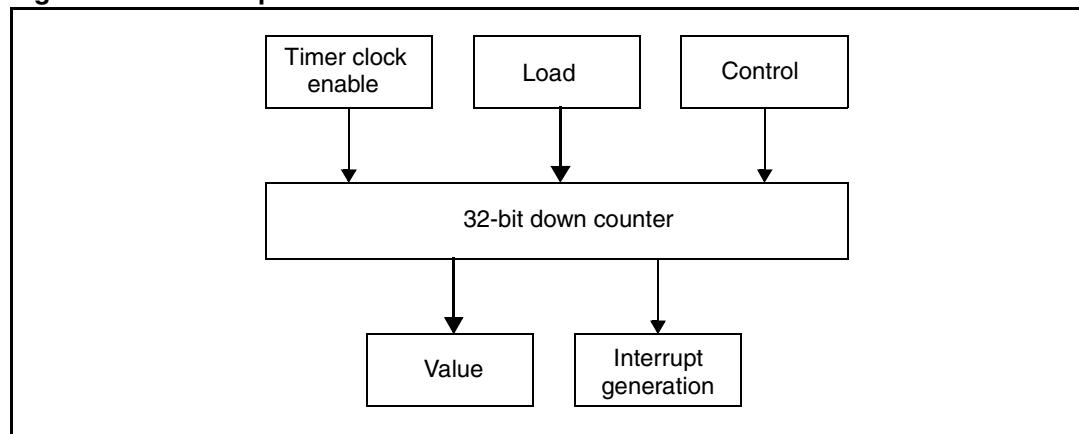
Once the counter is running, loading a new starting count value causes the counter to immediately re-start at the new value. A background load value can alternatively be specified, which has no effect on the current count, but on reaching zero the counter re-starts from the new value (if in periodic mode).

At reset, the:

- counter is disabled
- interrupt is cleared
- starting count value is set to zero
- mode is set to free-running
- pre-scale values are set for a clock dividing factor of 1

Figure 30 shows a block diagram of the timer operation.

Figure 30. Timer operation

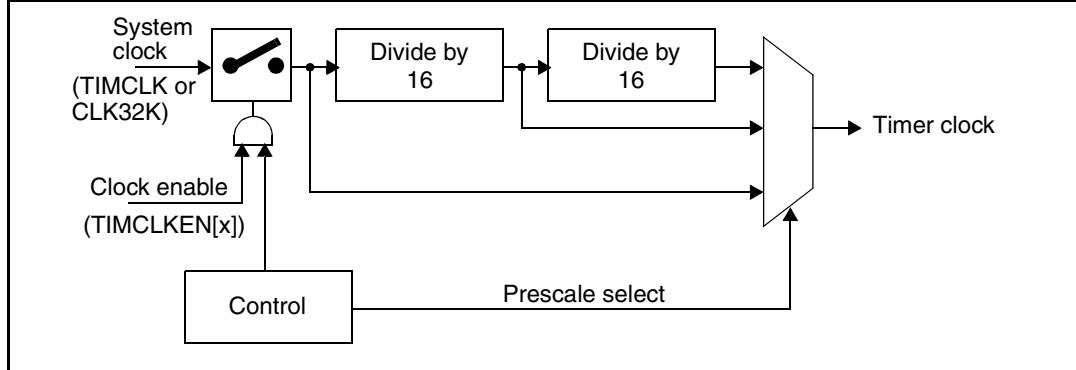


11.3.1 Timer clock

The timer clock is generated by a pre-scale unit, allowing a clock to be generated from one of the following:

- system clock (directly)
- system clock divided by 16 (achieved using a 4-bit prescaler)
- system clock divided by 256 (achieved using two 4-bit prescalers)

Figure 31 shows how the timer clock frequency is selected in the pre-scale unit. This allows the timer to be clocked at different frequencies.

Figure 31. Clock pre-scaling

11.3.2 Timer modes

The timer modes of operation are described in [Figure 54](#).

Table 54. Timer modes

Mode	Description
Free-running	The counter wraps after reaching its zero value, and continues to count down from the maximum value. This is the default mode.
Periodic	The counter generates an interrupt at a constant interval, reloading the original value after wrapping past zero.
One-shot	The counter generates an interrupt once. When the counter reaches zero, it halts until it is reprogrammed.

11.3.3 FRC

The prescaled timer clock is fed into the LSB of the counter, with the carry bits of the least significant 16-bit counter being passed to the most-significant counter. The interrupt is generated from the most-significant carry bit, which indicates that the whole counter has reached zero.

11.4 MTU signals

[Table 55](#) describes the interrupt signals of the MTU.

Table 55. Timer interrupt signals

Signal	Description
INTCTx	Timer x interrupts ($x = 0$ to 3): active-high interrupt signal. This signal indicates that the counter of timer x has been decremented to zero.
INTCTC	Combined timer interrupt: active-high interrupt signal to the VIC. This signal indicates an interrupt has been generated by one of the timer counters having been decremented to zero, and is the logical OR of INTCTx ($x = 0$ to 3).

Note: INTCTx are not directly connected to the vectored interrupt controller (VIC). Only the logical OR of INTCT[x] ($x = 0$ to 3) is connected to the VIC.

12 Watchdog timer (WDT)

The watchdog timer (WDT) provides a way of recovering from software crashes.

12.1 Overview

The watchdog clock generates a regular interrupt (WDOGINT) with a period that is programmable. It does this by counting down from a programmed value and then generating the interrupt. The watchdog monitors the interrupt and asserts a reset signal (WDOGRES) if the interrupt remains unserviced for the entire programmed period.

The watchdog unit can be enabled or disabled, as required. In the platform's different power modes, the WDT behaves as follows.

- In normal and slow modes, the WDT counts down at a fixed frequency of 32.768 kHz.
- In doze and sleep modes, the WDT is in a stopped state.

- Note:*
- 1 *The watchdog unit is stalled when the ARM processor is in debug mode.*
 - 2 *Write access to the watchdog unit can be disabled. This provides protection against software that might otherwise disable the watchdog functionality during debug.*

The watchdog unit interfaces with the platform as follows.

- The watchdog unit's WDOGCLK input clock is derived from the platform's HCLK.
- The unit's enable signal, WDOGCLKEN, is generated by the system controller. The enable pulses are generated at the CLK32K frequency (32.768 kHz).
- The unit's interrupt output WDOGINT is connected to VIC interrupt source input 0.

12.2 Functional description

The watchdog timer is a 32-bit down counter that divides the clock input to produce an interrupt. The divide ratio is fully programmable and controls the interrupt interval, which can be calculated using the following formula:

$$\text{Interrupt interval} = (\text{Programmed value} + 1) / (\text{Clock frequency in Hz})$$

The default programmed value is 4294967295. With a 32.768 kHz clock frequency, an interrupt is then generated every ~131072 seconds. This corresponds to the maximum programmable value; the valid values are from 1 to 4,294,967,295.

Table 56 shows example programmed values and the corresponding interrupt intervals when using a 32.768 kHz watchdog clock.

Table 56. Example interrupt intervals using a 32.768 kHz clock

Programmed value	Interrupt interval (ms)
4294967295	131 072 000
65535	2000
32767	1000
4095	125
127	3.90625
63	1.953125
1	0.0610

The watchdog interrupt generation and reset can be enabled or disabled. When the interrupt generation is disabled, the watchdog counter is also stopped. When the interrupt is enabled, the counter starts from the programmed value, and not from the last count value.

12.3 WDT signals

The watchdog timer delivers one interrupt signal to the vectored interrupt controller (VIC) and one reset signal to the power management unit (PMU). [Table 57](#) contains a list of interrupt and reset signals delivered by the WDT.

Table 57. WDT interrupt signals

Signal	Direction	Description
WDOGINT	Output	Watchdog timer interrupt: Active-high interrupt signal to the interrupt controller module. This signal indicates an interrupt has been generated by counter 1 having been decremented to zero.
WDOGRES	Output	Watchdog timer reset: Active-high reset signal. This signal becomes active if the interrupt remains unserviced for the entire programmed period.

13 Real-time clock (RTC)

The real-time clock unit (RTC) can be used to provide a basic alarm function or long time-base counter which counts in one second intervals, and provides a general-purpose real-time reference for the platform.

The RTC is a free running counter and starts incrementing the count value once the power-on reset (PORn) has been de-asserted. The RTC is not reset by any other reset.

The device can be programmed with a 32-bit match value that is compared with the RTC counter. If the values match, the wake-up signal (RTCWKUP) is set. An interrupt is also generated and routed to the vectored interrupt controller (VIC).

Note: *The value of the counter is unaffected by transitions into and out of sleep mode.*

13.1 Functional description

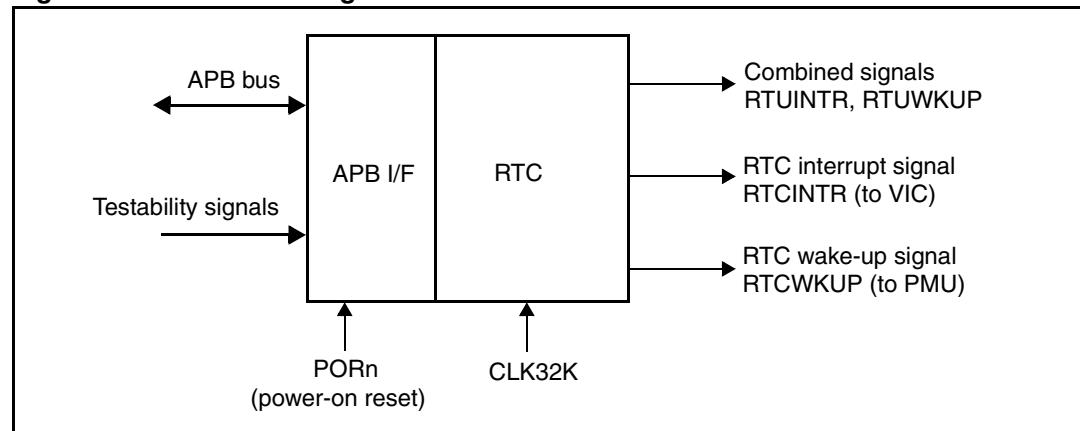
The RTC counter is incremented on the rising edges of a 1 Hz clock signal, CLK1HZ (internally, the counter is clocked by the CLK32K signal; the CLK1HZ is used as a counting enable signal, and is active for one CLK32K period).

The CLK1HZ signal is generated by dividing the slow clock source (32.768 kHz crystal or signal on the SXTALI pin). The divider logic for generating this CLK1HZ signal is programmable and allows the counter to be trimmed to adjust for inherent inaccuracies in the crystal frequency. The trimming mechanism allows the RTC time to be adjusted to an accuracy of ± 5 seconds per month.

Counting in one second intervals is achieved by using the 32.768 kHz clock input to the RTC, and dividing by an integer and fractional divider to produce a precise 1 Hz reference clock.

The RTC includes a 32-bit free running counter, a 32-bit match register and a 32-bit comparator. CPU data reads and writes are done through the AMBA peripheral bus (APB) interface.

Figure 32. RTC block diagram



13.2 RTC signals

The RTC provides one interrupt signal, RTCINTR, and one wake-up signal, RTCWKUP.

13.2.1 RTC interrupt

The RTC provides a single interrupt signal, RTCINTR. This interrupt is generated when the 32-bit counter reaches the programmed match value.

This single interrupt line is combined (ORed) with the real-time timer (RTT) interrupt signal, RTTINTR, before reaching the vectored interrupt controller (VIC). For the efficient determination and management of the source of the interrupt between the RTC and RTT, the two blocks share interrupt registers. The interrupts from the two sources can be masked.

13.2.2 RTC wake-up signal

When the platform is in deep-sleep mode, the RTC continues counting (when enabled). On reaching the programmed match value, it sends the wake-up signal RTCWKUP to the power management unit (PMU) to restore the core supply. This signal also goes to the system and reset controller (SRC) to restore the clocks. The wake-up event can be masked.

The signal RTCWKUP is equivalent to the RTCINTR signal, but is not re-synchronized with the PCLK clock (as this clock does not operate in deep-sleep mode).

The CLK1HZ signal that drives the RTC is generated by dividing the incoming 32.768 kHz clock signal delivered by an external crystal or oscillator. The inherent inaccuracies of the crystal may cause the time-base to be inaccurate. A slight adjustment of the desired clock period is then required.

The application processor allows the CLK1HZ time-base to be adjusted (trimmed) to an error of less than 1 part per million, equivalent to an error of less than 1 second per year. CLK1HZ is reset to a frequency of approximately 1 Hz each time the power-on reset signal (PORn) is asserted.

To trim the clock, the RTC divides the incoming 32.768 kHz clock signal by an integer value and then fractionally adjusts it by periodically deleting clock pulses from the original stream that drives this integer divider. This is described further below.

13.3 Oscillator frequency calibration

To determine how the CLK1HZ should be adjusted, the output frequency at the CLK32K pin must first be measured using an accurate frequency counter (provided that the alternate function of the associated GPIO is enabled).

13.3.1 Trimming calculations

Once the true frequency of the oscillator is known, it must be divided to achieve the desired CLK1HZ clock frequency. The divisor value is split into integer and fractional parts, which are used as follows:

Integer part

The integer part of the value (minus one) is compared against a 16-bit counter clocked by the incoming 32.768 kHz (approximately) clock signal. When the two values are equal, the counter resets and generates a pulse which constitutes the raw CLK1HZ signal period.

Fractional part

The fractional part is adjusted by periodically deleting clock pulses from the incoming 32.768 kHz clock stream driving the integer counter. The trim interval period is hardwired to be $(2^{10} - 1)$ periods of the 1 Hz clock. Thus, if the CLK1HZ clock is programmed to be 1 Hz, the trim interval is approximately 17 minutes. The number of clock pulses deleted (the trim delete value) is a 10-bit programmable counter allowing from 0 to $(2^{10} - 1)$ 32 kHz clock pulses to be deleted from the input clock stream once per trim interval.

In summary, every $(2^{10} - 1)$ CLK1HZ periods, the integer counter stops clocking for a period equal to the fractional error that has accumulated. If this fractional error is programmed to be zero then no trim operations occur and the RTC is clocked with the raw 32 kHz clock. The relationship between the CLK1HZ signal frequency and the nominal 32 kHz clock (f_1 and f_{32K} , respectively) is expressed by the following formula:

$$f_1 = \frac{(2^{10} - 1) \times (CKDIV + 1) - CKDEL}{(2^{10} - 1) \times (CKDIV + 1)} \times \frac{f_{32k}}{(CKDIV + 1)}$$

f_1 : CLK1HZ signal frequency,

f_{32k} : Theoretical incoming clock frequency - Measured incoming clock frequency,

CKDEL: Trim delete value (number of clock pulses to delete per trim period),

CKDIV: Integer part of clock divider.

13.3.2 Examples

Two examples of trimming are provided below.

Trim example 1: Measured value has no fractional component

In this example, the desired CLK1HZ signal frequency is 1 Hz. The oscillator output is measured as 36045.000 Hz. This output is exactly 3277 cycles above the nominal frequency of the crystal (32.768 kHz) and has no fractional component. As such, only the integer trim function is needed, no fractional trim is required.

Accordingly, CLKDIV is equal to 36045 and CKDEL is zero to disable fractional trimming. This trim exercise leaves an error of zero in trimming.

Trim example 2: Measured value has a fractional component

This example is more common in that the measured frequency of the oscillator has a fractional component. Again, the desired CLK1HZ signal output frequency is 1 Hz. If the oscillator output is measured as 32.76896 kHz, an integer trim is necessary so that the average number of cycles counted before generating one 1 Hz clock pulse is 32 768.96. In this case, CKDIV is equal to 32 768.

Since the actual clock frequency is 0.96 cycles per second faster than the integer value, the CLK1HZ signal generated by just the integer trimming is slightly faster than needed and must be slowed down. Accordingly, on average 0.96 cycles per second must be deleted to bring the CLK1HZ output frequency down to the proper value. Since the trimming procedure is performed every 1023 ($2^{10} - 1$) seconds, the trim must be set to delete $0.96 \times 1023 (= 982.08)$ clock pulses every 1023 seconds. Thus, CKDEL is equal to 982.

The fractional component 0.08 cannot be trimmed out and constitutes the error in trimming. This trim setting leaves an error of 0.08 cycles per 1023 seconds. The error calculation yields (in parts per million or ppm):

$$\text{Error} = \frac{0.08\text{cycles}}{1023\text{seconds}} \times \frac{1\text{second}}{32768\text{cycles}} = 0.002\text{ppm}$$

14 Real-time timer (RTT)

The real-time timer (RTT) is a 32-bit free-running counter, clocked by the 32 768 Hz clock signal (from the SXTAL1 pin). It can be used to provide an interrupt at regular time intervals (for OS scheduling purposes, for example), in sleep, doze, slow and normal modes. It can also be used to provide a wake-up event to exit the device from deep-sleep mode.

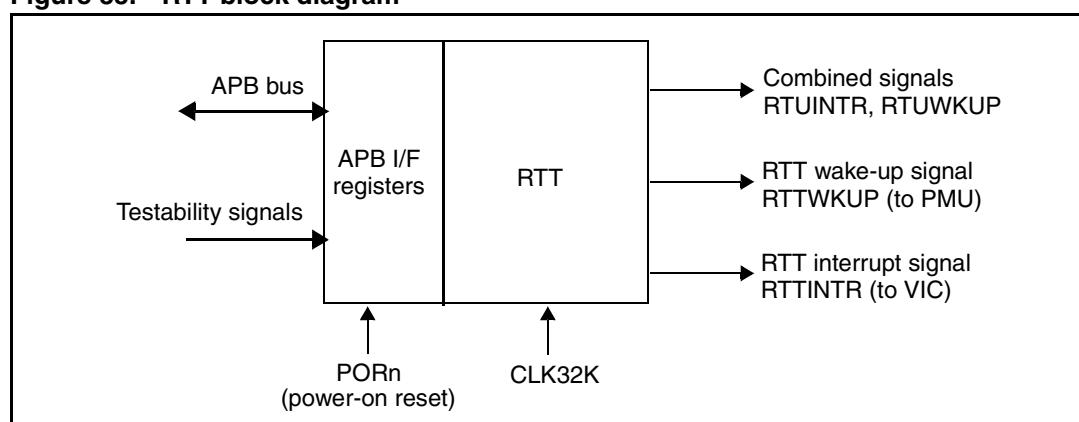
14.1 Features

- 32-bit down-counter
- interrupt/wake-up generation when timer reaches zero
- start, auto-restart (on reaching zero) and stop
- on-the-fly register read and write access
- interrupt signal combined with the interrupt of the real-time clock (RTC) and sharing the same interrupt registers (for efficient determination of the source of the interrupt)
- minimum period of 1/32 768 s

14.2 Functional description

The RTT counts down from a programmed number of cycles of a real-time clock input (the 32.768 kHz clock) and when it reaches zero, generates an interrupt or wake-up signal. On reaching zero, it can stop or, if the auto-restart mode is enabled, can automatically start counting down again from the programmed value.

Figure 33. RTT block diagram



14.2.1 RTT modes

The RTT can operate in one of two modes:

- **Periodic mode:** The counter generates an interrupt repeatedly at a constant interval, reloading the programmed counter value after wrapping past zero.
- **One-shot mode:** The counter generates an interrupt only once. When the counter reaches zero, it halts until it is restarted manually.

The timer must first be programmed with a counter value from which it counts down. This value cannot be changed while the timer is running; the timer must first be stopped. The RTT can be stopped by software or, in one-shot mode, stops automatically when the counter reaches zero.

14.2.2 RTT signals

The RTT has an interrupt signal and a wake-up signal.

RTT interrupt

This single interrupt line, RTTINTR, is combined (ORed) with the real-time clock (RTC) interrupt signal, RTCINTR, before reaching the vectored interrupt controller (VIC). The interrupts from the two sources can be masked.

RTT wake-up signal

When the platform is in sleep or deep-sleep mode, the RTT continues counting down (when enabled). On reaching zero:

- in deep-sleep mode, it sends the signal RTTWKUP to the power management unit (PMU) to restore the core supply,
- in both sleep and deep-sleep mode, the signal RTTWKUP also goes to the system and reset controller (SRC) to restore the clocks.

The signal RTTWKUP can be masked.

The signal RTTWKUP is equivalent to the RTTINTR signal, but is not re-synchronized with the PCLK clock (as this clock does not operate in deep-sleep mode).

15 SDRAM memory controller (SDMC)

The device has one SDRAM memory controller (SDMC) that can support up to 2 chip-selects of simple data rate or double data rate synchronous RAM memories (SDR/DDR-SDRAM).

15.1 Features

- Supports dynamic memory devices, including SDRAM, low power SDRAM, Micron SyncFlash™ and mobile DDR-SDRAM
- Low transaction latency
- Read and write buffers to reduce latency and to improve performance
- The priority of the ARM AHB interface may be dynamically increased to reduce latency for important software routines
- Supports 16-bit wide data bus
- Two chip selects
- Power saving modes dynamically control SDRAM clock and clock enable signals
- Supports dynamic memory self-refresh
- Supports 2K, 4K and 8K row address dynamic memory parts; that is, typical 512-, 256-128-, 64- and 16-Mbit parts, with 8 or 16 DQ bits per chip select
- Preserves dynamic memory contents over a soft reset

15.2 Functional description

The external memories connected on the SDRAMC can be accessed by:

- ARM926EJ CPU Instruction fetch bus (attached on AHB bus 5)
- ARM926EJ CPU data bus (attached on AHB bus 4)
- system DMA0 and DMA1 controllers, and the smart graphic accelerator (attached on AHB bus 3)
- smart video accelerator data and cache refill buses (attached on AHB bus 2)
- the LCD and MDIF bus (attached on AHB 1)
- the smart audio accelerator buses (attached on AHB bus 0)

When an IRQ or FIQ interrupt occurs, the The ARM926EJ CPU instruction fetch bus (attached on AHB bus 5), and the ARM926EJ CPU data bus (attached on AHB bus 4) can have their priority increased, as if their timeout counters reached zero. This feature can be enabled/disabled by software.

15.2.1 AHB slave memory interfaces

The AHB slave memory interfaces enable devices to access external memories. The memory interfaces are prioritized, with interface 0 having the highest priority. Having more than one memory interface enables high-bandwidth peripherals direct access to the SDMC, without data having to pass over the main system bus.

Memory transaction endianness

The endianness of data transfers to and from external memories is selectable.

Note: The SDMC must be idle before endianness is changed.

Memory transaction size and burst length

Memory transactions can be 8 or 16 bits wide. For 16-bit wide chip selects, SDRAM bursts of two are used; for 8-bit wide chip selects, SDRAM bursts of four are used.

Write protected memory areas

Write transactions to write-protected memory areas generate an error response on HRESP and the transfer is terminated.

Arbiter

The arbiter arbitrates between the AHB slave memory interfaces. The following arbitration priority is used:

1. auto-refresh.
2. highest priority AHB port. This is a port which has timed out. If more than one port has timed out, the port with the lowest port number is selected, that is, port 0 is selected over port 1.
3. port already open to an SDRAM memory row. If more than one port is open, the port with the lowest port number is selected.
4. memory requests to the lowest port number.

AHB interface 0 has the highest access priority, and AHB interface 3 has the lowest priority.

Timeout counters

Each AHB memory port has a programmable timeout counter that:

- enables the port to be programmed with a determined latency, and
- enables the bandwidth that a port consumes to be indirectly defined

When a memory request is made to the port, the value of the counter is loaded. For each cycle where the port is not serviced, the timeout counter counts down. When the counter reaches zero, the priority of the port is increased.

Locked accesses

Locked accesses on the AHB bus are transactions where HMASTLOCK is high.

When the SDMC performs a locked access, it does not re-arbitrate to another AHB port until the lock transfer is complete.

Data buffers

The SDMC contains six 32-bit word buffers, one associated with each AHB interface. These buffers are used as a combination read-and-write buffer. Use of the buffers:

- maximizes memory performance for 8- and 16-bit reads and writes, since 32 bits are transferred at a time. During buffered reads, if the data for the next 8- or 16-bit read is already in the buffer, the data is read directly from the buffer.
- reduces transaction latency.
- reduces power consumption.

The SDMC re-arbitrates to an open page transfer during a buffered transaction.

Each AHB port buffer can be enabled or disabled independently.

15.2.2 Low power operation

In many systems, the contents of the memory must be maintained in low-power modes. The SDMC provides two features to enable this:

- dynamic refresh over soft reset,
- dynamic self-refresh.

Dynamic self-refresh mode can be entered automatically using the power management unit (PMU). On battery fail detection (signal BATOK going low):

1. The PMU asserts the signal SDMCSREFREQ.
2. The SDMC closes any open memory banks and puts the external memory into self-refresh mode.
3. The SDMC asserts the signal SDMCSREFACK to indicate to the PMU that self-refresh mode is entered.

The system must ensure that the memory subsystem is idle before asserting SDMCSREFREQ. Any memory transfers requested while the SDMC is in self-refresh mode are rejected, and an error response is generated (HRESP = ERROR).

De-asserting SDMCSREFREQ returns the memory to normal operation. For refresh requirements, see the data sheet for the memory device.

Low-power SDRAM deep sleep mode

The SDMC supports JEDEC low-power SDRAM deep-sleep mode. The device is powered down and no longer refreshed. All data in the memory is lost.

Low power SDRAM partial array refresh

The SDMC supports JEDEC low-power SDRAM partial array refresh. Partial array refresh can be programmed by initializing the memory device appropriately. When the memory device is put into self-refresh mode, only the banks specified are refreshed. The banks that are not refreshed lose their data.

15.2.3 Memory chip-selects

One or two independently configurable memory chip selects are supported for SDRAM devices. [Table 58](#) gives the memory mapping for the STn8815A12 device.

Table 58. SDMC memory map

Address	Comment
0x1011 FFFF	SDRAM control configuration
0x1011 0000	
0x0FFF FFFF	SDRAM chip select 1
0x0800 0000	
0x07FF FFFF	SDRAM chip select 0
0x0000 0000	

15.3 Supported memory devices

The STn8815A12 supports 16-bit SDRs/DDRs. Refer to ST document number NM569 *STn8815 list of compatible memories*, latest version.

Two mapping modes are supported:

- 16-bit external bus high-performance mapping (row-bank-column)
- 16-bit external bus low-power SDRAM mapping (bank-row-column)

15.4 Address mapping

The tables in this section show how the AHB bus addresses are mapped to the external memory address bus SDRAD[14:0] for various memory configurations and bus widths.

In the tables, BA, BA0, and BA1 indicate the bank address signals. AP indicates the auto precharge signal (normally address bit 10).

Separate tables are provided for the two different address mapping schemes, row-bank-column (RBC), or bank-row-column (BRC), for 16-bit wide data buses.

- SDRAM (RBC): these address mappings are used for 16-bit data bus chip select with SDR-SDRAM memory devices
- SDRAM (BRC): these address mappings are used for 16-bit data bus chip select with mobile/low-power SDR-SDRAM and DDR-SDRAM, or Micron SyncFlash memory devices

Note:

For SDRAD address bus bit 0, the AHB address to column address is forced low by the SDMC. The SDRAM controller always transfers 32 bits of data at a time by performing a burst transfer of length 2 from/to the memory.

16-bit wide data bus 16-Mbit SDRAM (1 Mbit x 16, RBC)

Table 59 shows the SDMC address bus output and its connection to the 16-Mbit SDRAM (1 Mbit x 16, pin 14 used as bank select).

Table 59. Address mapping for 16-Mbit SDRAM (1 Mbit x 16, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA	9	9
13	-	-	-
12	-	-	-
11	-	-	-
10	10/AP	20	AP
9	9	19	-
8	8	18	-
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	See note on page 180 .

16-bit wide data bus 16-Mbit SDRAM (2 Mbit x 8, RBC)

Table 60 shows the SDMC address bus output and its connection to the 16-Mbit SDRAM (2 Mbit x 8, pin 13 used as bank select).

Table 60. Address mapping for 16-Mbit SDRAM (2 Mbit x 8, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA	-	-
13	-	10	10
12	-	-	-
11	-	-	-
10	10/AP	11	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	See note on page 180

16-bit wide data bus 64-Mbit SDRAM (4 Mbit x 16, RBC)

Table 61 shows the SDMC address bus output and its connection to the 64-Mbit SDRAM (2 Mbit x 32, pin 13 and 14 used as bank selects).

Table 61. Address mapping for 64-Mbit SDRAM (4 Mbit x 16, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	9	9
13	BA0	10	10
12	-	-	-
11	11	22	-
10	10/AP	21	AP
9	9	20	-
8	8	19	-
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	See note on page 180

16-bit wide data bus 64-Mbit SDRAM (8 Mbit x 8, RBC)

Table 62 shows the SDMC address bus output and its connection to the 64-Mbit SDRAM (8 Mbit x 8, pin 13 and 14 used as bank selects).

Table 62. Address mapping for 64-Mbit SDRAM (8 Mbit x 8, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	10	10
12	-	-	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	See note on page 180

16-bit wide data bus 128-Mbit SDRAM (8 Mbit x 16, RBC)

Table 63 shows the SDMC address bus output and its connection to the 128-Mbit SDRAM (8 Mbit x 16, pin 13 and 14 used as bank selects).

Table 63. Address mapping for 128-Mbit SDRAM (8 Mbit x 16, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	10	10
12	-	-	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	See note on page 180

16-bit wide data bus 128-Mbit SDRAM (16 Mbit x 8, RBC)

Table 64 shows the SDMC address bus output and its connection to the 128-Mbit SDRAM (16 Mbit x 8, pin 13 and 14 used as bank selects).

Table 64. Address mapping for 128-Mbit SDRAM (16 Mbit x 8, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	12	12
12	12	-	-
11	11	24	-
10	10/AP	23	AP
9	9	22	10
8	8	21	9
7	7	20	8
6	6	19	7
5	5	18	6
4	4	17	5
3	3	16	4
2	2	15	3
1	1	14	2
0	0	13	See note on page 180

16-bit wide data bus 256-Mbit SDRAM (16 Mbit x 16, RBC)

Table 65 shows the SDMC address bus output and its connection to the 256-Mbit SDRAM (16 Mbit x 16, pin 13 and 14 used as bank selects).

Table 65. Address mapping for 256-Mbit SDRAM (16 Mbit x 16, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	10	10
12	12	24	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	See note on page 180

16-bit wide data bus 256-Mbit SDRAM (32 Mbit x 8, RBC)

Table 66 shows the SDMC address bus output and its connection to the 256-Mbit SDRAM (32 Mbit x 8, pin 13 and 14 used as bank selects).

Table 66. Address mapping for 256-Mbit SDRAM (32 Mbit x 8, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	12	12
12	12	25	-
11	11	24	-
10	10/AP	23	AP
9	9	22	10
8	8	21	9
7	7	20	8
6	6	19	7
5	5	18	6
4	4	17	5
3	3	16	4
2	2	15	3
1	1	14	2
0	0	13	See note on page 180

16-bit wide data bus 512-Mbit SDRAM (32 Mbit x 16, RBC)

Table 67 shows the SDMC address bus output and its connection to the 512-Mbit SDRAM (32 Mbit x 16, pin 13 and 14 used as bank selects).

Table 67. Address mapping for 512-Mbit SDRAM (32 Mbit x 16, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	12	12
12	12	25	-
11	11	24	-
10	10/AP	23	AP
9	9	22	10
8	8	21	9
7	7	20	8
6	6	19	7
5	5	18	6
4	4	17	5
3	3	16	4
2	2	15	3
1	1	14	2
0	0	13	See note on page 180

16-bit wide data bus 512-Mbit SDRAM (64 Mbit x 8, RBC)

Table 68 shows the SDMC address bus output and its connection to the 512-Mbit SDRAM (64 Mbit x 8, pin 13 and 14 used as bank selects).

Table 68. Address mapping for 512-Mbit SDRAM (64 Mbit x 8, RBC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	13	13
13	BA0	12	12
12	12	26	-
11	11	25	11
10	10/AP	24	AP
9	9	23	10
8	8	22	9
7	7	21	8
6	6	20	7
5	5	19	6
4	4	18	5
3	3	17	4
2	2	16	3
1	1	15	2
0	0	14	See note on page 180

16-bit wide data bus 16-Mbit SDRAM (1 Mbit x 16, BRC)

Table 69 shows the SDMC address bus output and its connection to the 16-Mbit SDRAM (1 Mbit x 16, pin 13 used as bank select).

Table 69. Address mapping for 16-Mbit SDRAM (1 Mbit x 16, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	-	-	-
13	BA	20	20
12	-	-	-
11	-	-	-
10	10/AP	19	AP
9	9	18	-
8	8	17	-
7	7	16	8
6	6	15	7
5	5	14	6
4	4	13	5
3	3	12	4
2	2	11	3
1	1	10	2
0	0	9	See note on page 180

16-bit wide data bus 16-Mbit SDRAM (2 Mbit x 8, BRC)

Table 70 shows the SDMC address bus output and its connection to the 16-Mbit SDRAM (2 Mbit x 8, pin 14 used as bank select).

Table 70. Address mapping for 16-Mbit SDRAM (2 Mbit x 8, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA	21	21
13	-	-	-
12	-	-	-
11	-	-	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	See note on page 180

16-bit wide data bus 64-Mbit SDRAM (4 Mbit x 16, BRC)

Table 71 shows the SDMC address bus output and its connection to the 64-Mbit SDRAM (4 Mbit x 16, pin 13 and 14 used as bank selects).

Table 71. Address mapping for 64-Mbit SDRAM (4 Mbit x 16, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	21	21
13	BA0	22	22
12	-	-	-
11	11	20	-
10	10/AP	19	AP
9	9	18	-
8	8	17	-
7	7	16	8
6	6	15	7
5	5	14	6
4	4	13	5
3	3	12	4
2	2	11	3
1	1	10	2
0	0	9	See note on page 180

16-bit wide data bus 64-Mbit SDRAM (8 Mbit x 8, BRC)

Table 72 shows the SDMC address bus output and its connection to the 64-Mbit SDRAM (8 Mbit x 8, pin 13 and 14 used as bank selects).

Table 72. Address mapping for 64-Mbit SDRAM (8 Mbit x 8, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	23	23
13	BA0	22	22
12	-	-	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	See note on page 180

16-bit wide data bus 128-Mbit SDRAM (8 Mbit x 16, BRC)

Table 73 shows the SDMC address bus output and its connection to the 128-Mbit SDRAM (8 Mbit x 16, pin 13 and 14 used as bank selects).

Table 73. Address mapping for 128-Mbit SDRAM (8 Mbit x 16, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	23	23
13	BA0	22	22
12	-	-	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	See note on page 180

16-bit wide data bus 128-Mbit SDRAM (16 Mbit x 8, BRC)

Table 74 shows the SDMC address bus output and its connection to the 128-Mbit SDRAM (16 Mbit x 8, pin 13 and 14 used as bank selects).

Table 74. Address mapping for 128-Mbit SDRAM (16 Mbit x 8, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	23	23
13	BA0	24	24
12	-	-	-
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	See note on page 180

16-bit wide data bus 256-Mbit SDRAM (16 Mbit x 16, BRC)

Table 75 shows the SDMC address bus output and its connection to the 256-Mbit SDRAM (16 Mbit x 16, pin 13 and 14 used as bank selects).

Table 75. Address mapping for 256-Mbit SDRAM (16 Mbit x 16, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	23	23
13	BA0	24	24
12	12	22	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	See note on page 180

16-bit wide data bus 256-Mbit SDRAM (32 Mbit x 8, BRC)

Table 76 shows the SDMC address bus output and its connection to the 256-Mbit SDRAM (32 Mbit x 8, pin 13 and 14 used as bank selects).

Table 76. Address mapping for 256-Mbit SDRAM (32 Mbit x 8, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	25	25
13	BA0	24	24
12	12	23	-
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	See note on page 180

16-bit wide data bus 512-Mbit SDRAM (32 Mbit x 16, BRC)

Table 77 shows the SDMC address bus output and its connection to the 512-Mbit SDRAM (32 Mbit x 16, pin 13 and 14 used as bank selects).

Table 77. Address mapping for 512-Mbit SDRAM (32 Mbit x 16, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	25	25
13	BA0	24	24
12	12	23	-
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	See note on page 180

16-bit wide data bus 512-Mbit SDRAM (64 Mbit x 8, BRC)

Table 78 shows the SDMC address bus output and its connection to the 512-Mbit SDRAM (64 Mbit x 8, pin 13 and 14 used as bank selects).

Table 78. Address mapping for 512-Mbit SDRAM (64 Mbit x 8, BRC)

SDRAD address bus	Device address/bank connections	AHB address to row address	AHB address to column address
14	BA1	26	26
13	BA0	25	25
12	12	24	-
11	11	23	11
10	10/AP	22	AP
9	9	21	10
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	See note on page 180

15.5 Commands

The SDMC supports the following SDRAM memory commands. The commands listed in [Table 79](#) are generated automatically. The commands listed in [Table 80](#) must be programmed.

Table 79. Generated SDRAM memory commands

Mnemonic	Operation
ACT	Opens an SDRAM row
REF	CAS before RAS style refresh
SREF	Self-refresh
PRE	Precharge: closes a bank
RD	Read from an open row, row left open
WR	Write to an open row, row left open
RDA	Read followed by precharge
WRA	Write followed by precharge

Table 80. Programmed SDRAM memory commands

Mnemonic	Operation
MRS	Mode register set: programs SDRAM mode register
NOP	No operation: used during the SDRAM initialization
PALL	Precharge all: used during the SDRAM initialization
DSM	Deep sleep mode: for low-power SDRAM

15.6 SDMC signals

This section describes the pin interface signals for the SDMC.

Table 81. SDMC signals

Signal	I/O	Description
SDRCKP	O	SDRAM clock, positive (for SDR-SDRAM and DDR-SDRAM)
SDRCKN	O	SDRAM clock, inverted (for DDR_SDRAM)
SDRFBCK	I	Feed-back clocks. Used for SDR-SDRAM devices.
SDRDQSU	IO	Data strobe to/from DDR-SDRAMs upper byte
SDRDQSL	IO	Data strobe to/from DDR-SDRAMs lower byte
SDRCS1n, SDRCS0n	O	SDRAM chip selects
SDRCKE0	O	SDRAM clock enable for chip-select 0
SDRCKE1	O	SDRAM clock enable for chip-select 1
SDRDQMU	O	Data mask output to SDRAMs upper byte
SDRDQML	O	Data mask output to SDRAMs lower byte
SDRRASn	O	Row address strobe
SDRCASn	O	Column address strobe
SDRWRn	O	Write enable.
SDRAD[14:0]	O	Address output.
SDRDQ[15:0]	IO	Data to/from memory

16 Flexible static memory controller (FSMC)

The flexible static memory controller (FSMC) interfaces with several types of external storage devices, such as SRAMs, NOR and NAND Flash memories, PC-Cards, ATA, CompactFlash and CF+ devices.

16.1 Features

- Interfaces static memory-mapped devices including RAM, ROM, synchronous burst Flash, 16-bit PC card PCMCIA/CompactFlash/CF+ interfaces.
- 16-bit PC card and ATA 8- and 16-bit wide data transfer are supported. AHB transactions with 32-bit wide data are translated to consecutive 16- or 8-bit accesses.
- AHB burst transfers handled so as to reduce access time to the external devices.
- Independent configuration for each memory chip-select.
- Programmable timings to support a wide range of devices:
 - programmable wait states (up to 31),
 - programmable bus turnaround cycles (up to 15),
 - programmable output enable and write enable delays (up to 15).
- Write enable and byte lane select outputs for use with 16- and 8-bit SRAM devices.
- Independent chip select control for each memory chip-select.
- External asynchronous wait control.
- Configurable size at reset for boot memory chip-select using external control pins.

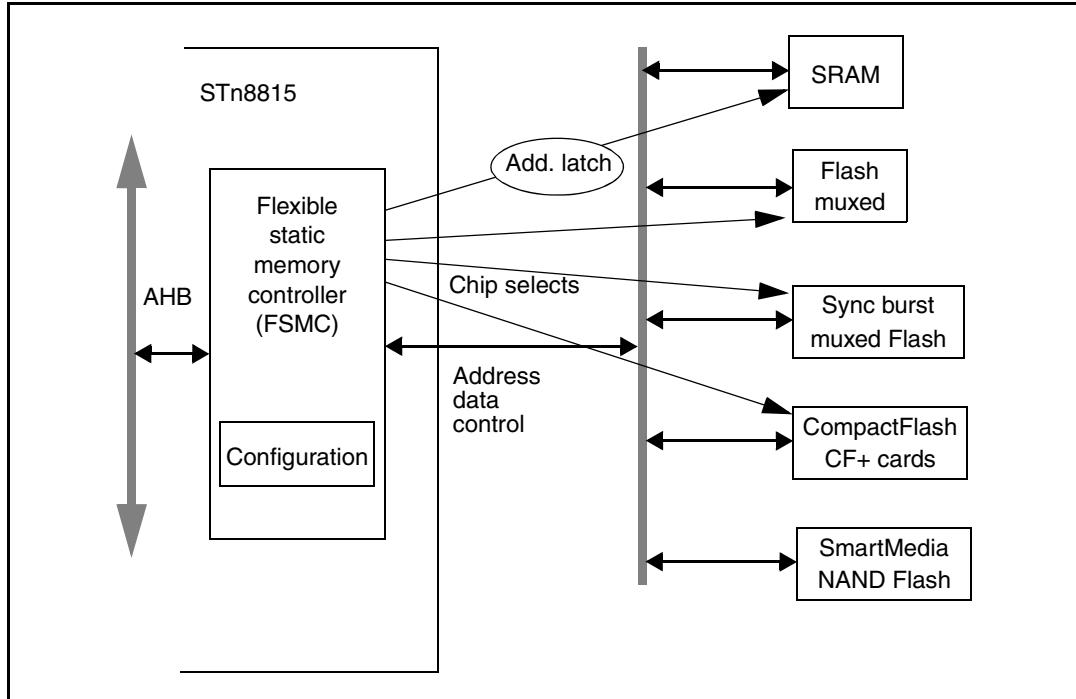
16.2 Overview

The FSMC controls up to three chip-selects of SRAM/NOR Flash, plus two PC cards/ATA devices (PCMCIA or CompactFlash cards) or NAND Flash (including SmartMedia) devices.

In the STn8815A12, some signals are not delivered on pins.

- CompactFlash or CF+ slots or NAND Flash devices are directly supported.
- PCMCIA slots need address lines [25] which are not delivered on pins, but can be provided by a GPIO managed by software.
- The SMAD bus delivers:
 - address lines [24:16] when accessing SRAM/NOR Flash devices,
 - address lines [8:0] when accessing CompactFlash/CF+/NAND Flash devices.
- Write protect (SMFWPn) and Flash reset (SMFRSTn) signals are only delivered for NOR Flash chip-select 0.

The type of device, associated timings and other external device characteristics are usually set at boot time, and do not change until the next reset or power up. However, it is possible to change the settings at any moment. The controller may also be booted directly from an external Flash memory after power-on reset.

Figure 34. FSMC block diagram

16.3 SRAM/NOR Flash controller

16.3.1 Supported memory types

This controller generates the appropriate signal timings to drive ST 1.8 V multiplexed burst NOR Flash memories.

If the multiplexed address bus is externally demultiplexed, or when the SMAD[15:0] address lines are delivered on an alternate function of GPIO pins, the controller can drive:

- asynchronous SRAM and ROM devices (8- or 16-bit data bus width),
- non-multiplexed ST 1.8 V NOR Flash devices in conventional or burst mode.

The STn8815A12 supports 16-bit NAND Flash devices. Refer to ST document number NM569 *STn8815 list of compatible memories*, latest version.

16.3.2 SRAM/NOR Flash controller timing parameters

In SRAM/ROM and asynchronous Flash modes, the FSMC signals are referenced to the internal AHB clock (HCLK). The timings of the signals are, for each chip-select, highly programmable with up to four parameters:

- Address setup duration (ADDSET),
- Address hold duration (ADDHLD), for multiplexed NOR Flash access,
- Data read/write strobe duration (DATAST),
- Bus turn around period (BUSTURN): refer to [Section 16.3.5: Bus turn around on page 212](#).

The following figures show the meaning of the chip-select timing parameters.

Figure 35. SRAM read access timing

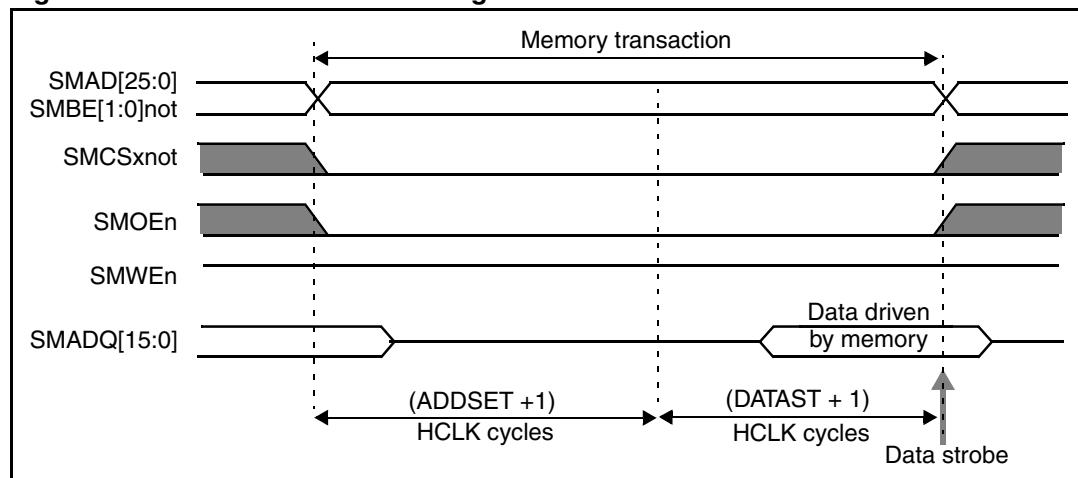
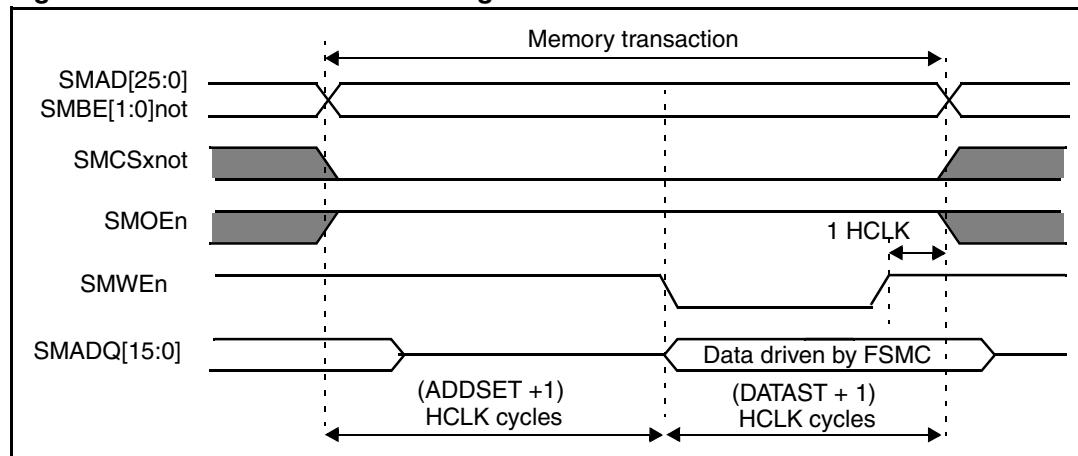
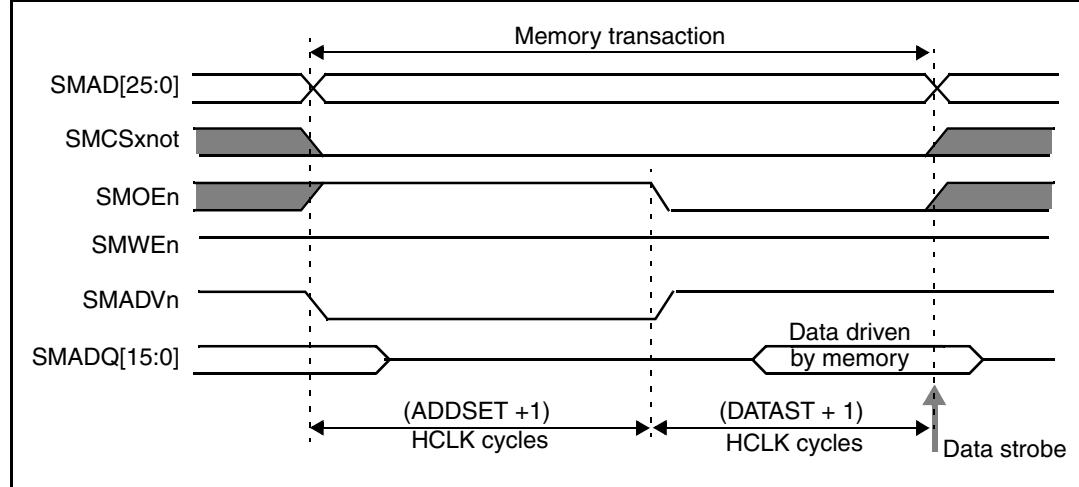


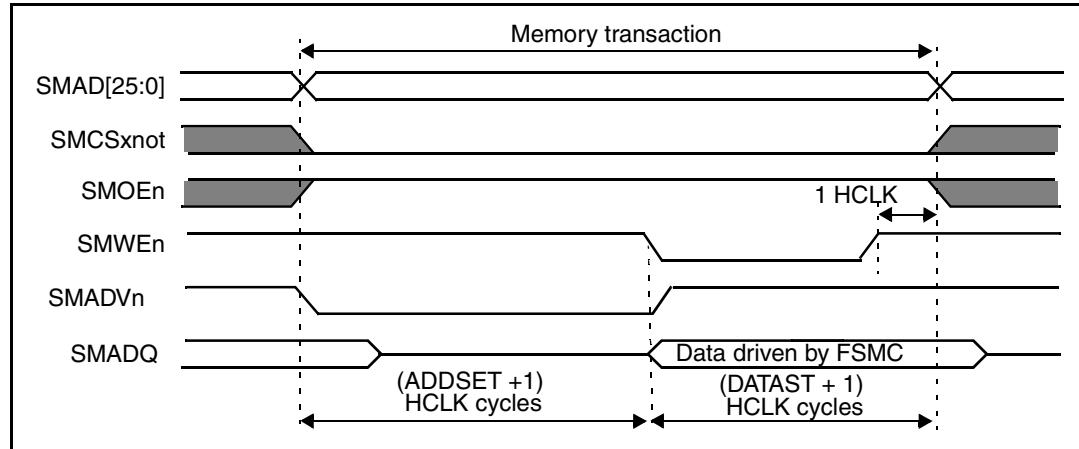
Figure 36. SRAM write access timing



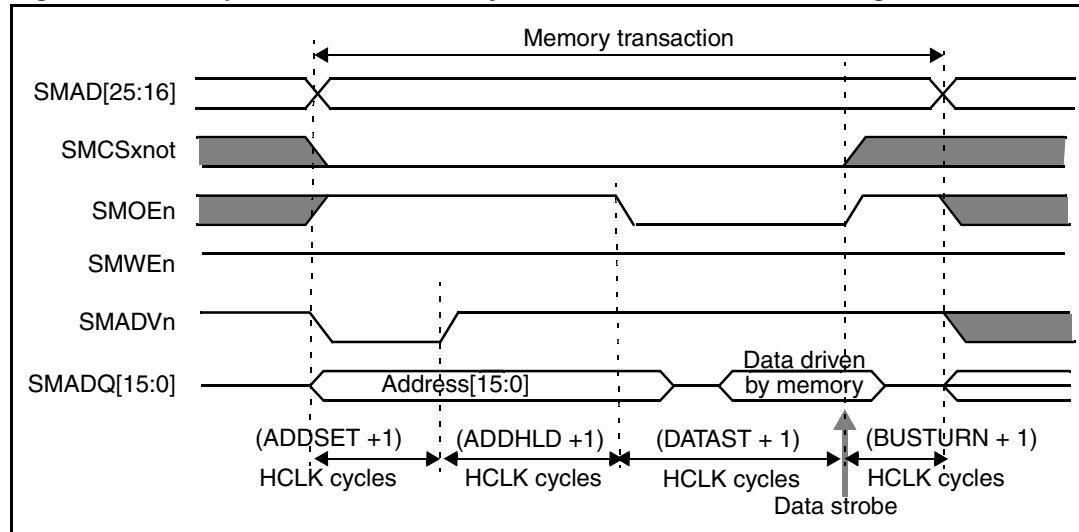
The one HCLK cycle at the end of the write transaction helps to guarantee the address and data hold time after SMWEn rising edge. Due to the presence of this one HCLK cycle, the DATAST value must be greater than zero.

Figure 37. Non-multiplexed NOR Flash asynchronous read access timing

If, after the read access, the memory is deselected, the controller enters the bus turn around phase (refer to [Section 16.3.5: Bus turn around on page 212](#)).

Figure 38. Non-multiplexed NOR Flash asynchronous write access timing

The HCLK cycle at the end of the write transaction helps to guarantee the address and data hold time after the SMWEn rising edge. Due to the presence of this HCLK cycle, DATAST must be greater than zero.

Figure 39. Multiplexed NOR Flash asynchronous read access timing

In this mode, the bus turn around period is always executed, even if the next access is for the same chip-select.

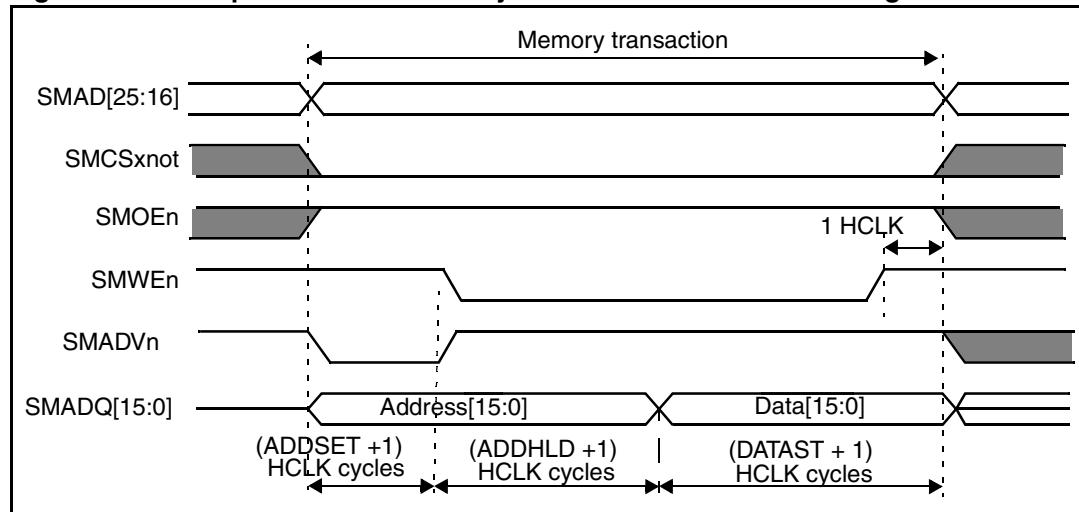
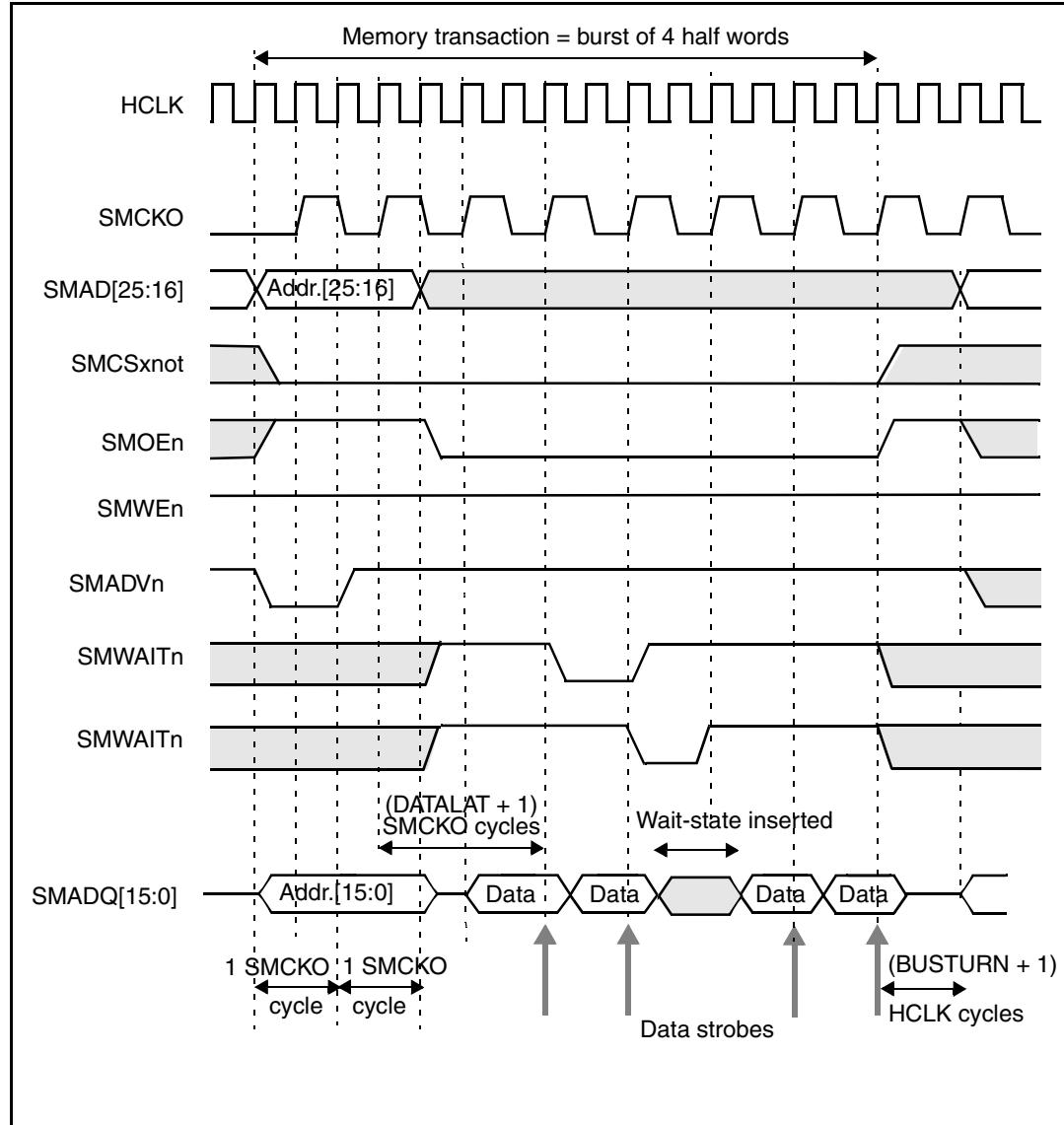
Figure 40. Multiplexed NOR Flash asynchronous write access timing

Figure 41. (Non)-multiplexed NOR Flash synchronous read access timing



All output signals change on the rising edge of HCLK. If SMCKO is a submultiple of HCLK (as shown in [Figure 41](#), where SMCKO = HCLK/2), some output signals change on the falling edge of HCLK, still remaining synchronous with the rising edge of HCLK.

At the end of the burst read transaction, the controller enters the bus turn around phase (refer to [Section 16.3.5: Bus turn around on page 212](#)).

DATALAT is the number of SMCKO (not HCLK) cycles to wait before strobing the data. The latency must correspond to the one specified in the NOR Flash configuration register (for an example, see the datasheet for the M58MR032 NOR Flash from ST-ERICSSON). As the Flash counts all the clock cycles except the one during SMADVn low, the exact relationship between the two latencies is:

$$\text{Nor_Flash_latency} = \text{DATALAT} + 2$$

Note: *The HCLK cycle at the end of a write cycle is meant to ensure address data hold after the rising edge of the signal SMWEn. This is always a minimum of 0 ns, so cannot be negative. Therefore, DATAST must be greater than 0x1.*

16.3.3 Wait management by the SRAM/NOR Flash controller

Signal SMWAITn is used by Flash memory when synchronous burst is active. As all memory wait# signals are active low and in high impedance when memory devices are not selected, they can be externally wired together to a pull up to drive SMWAITn.

The controller does not take into account the value of SMWAITn when driving a SRAM, a NOR-Flash in asynchronous mode, or with the first data from NOR Flash in synchronous burst. If SMWAITn is sensed active, then wait states are inserted until SMWAITn is sensed inactive. When SMWAITn is inactive, the data is considered valid immediately.

During wait-state insertion via the SMWAITn signal, the controller continues to send the clock to the memory, keeping the chip-select and output enable valid, and does not consider the data valid.

Wait polarity

The active level of SMWAITn is programmable independently for each chip-select.

Wait timing configuration

There are two timing configurations for NOR Flash SMWAITn signals in burst:

- Flash asserts SMWAITn one data cycle before the wait-state (default after reset),
- Flash asserts SMWAITn during the wait-state.

These two NOR Flash wait state configurations are supported by the FSMC independently for each chip-select.

Figure 42. Wait timing 1 for NOR Flash synchronous burst read

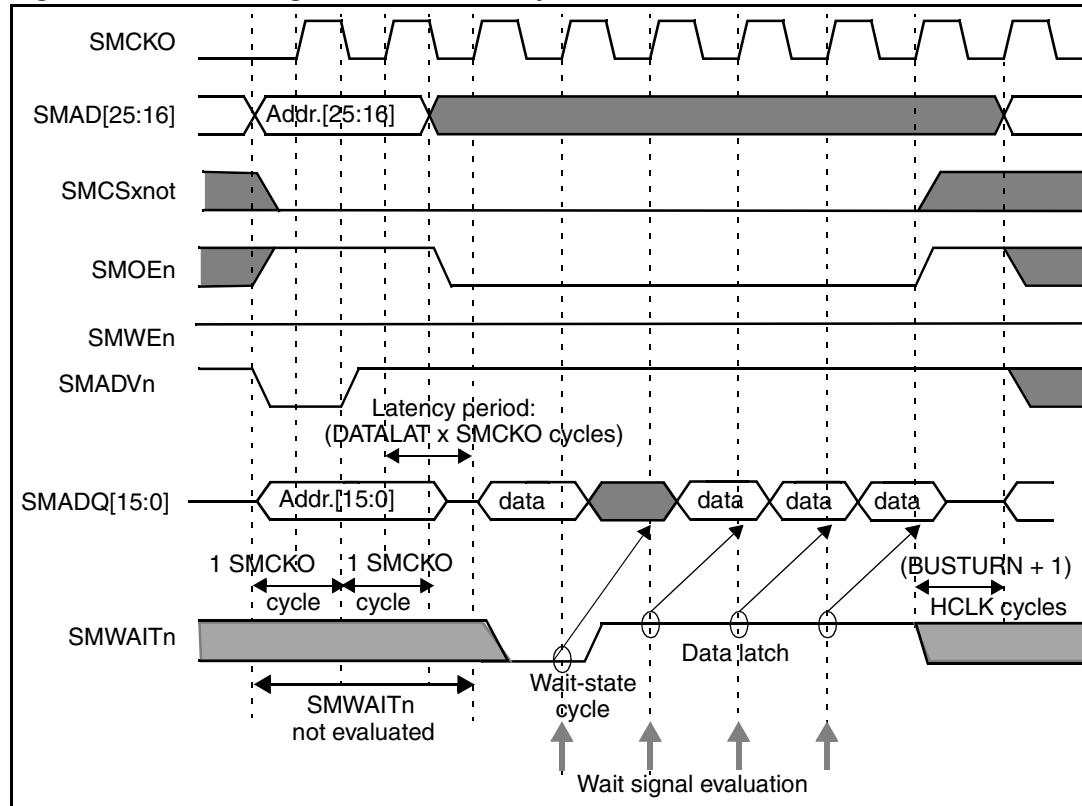
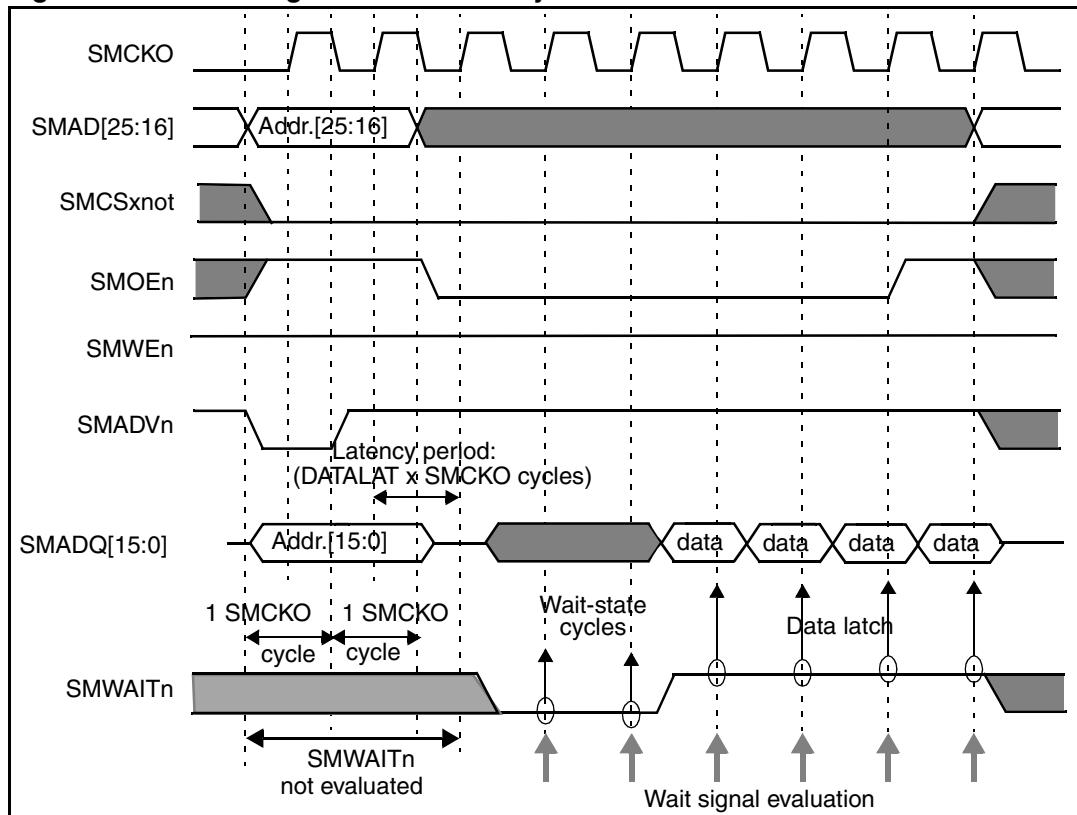


Figure 43. Wait timing 2 for NOR Flash synchronous burst read

16.3.4 Burst wrap support

The AHB interface processes the AHB wrap requests in order to control the SRAM/NOR Flash memory driver or the NAND/ATA driver accordingly.

The AMBA 2.0 protocol states that a master can generate burst wrap requests of three types: WRAP4, WRAP8 and WRAP16. The three modes can be of size word (4 bytes), half word (2 bytes) or byte: the wrap number specifies the beats of the burst transfer, so that the total number of bytes transferred in a burst (wrapped or not) is the burst beats times the size.

[Table 82](#) shows an example of addresses (HADDR) in a WRAP8 transfer.

Table 82. HADDR[4:0] values for a burst WRAP8 transaction

Word transfer	Half word transfer	Byte transfer
... 010 00	... X010 0	... XX010
... 011 00	... X011 0	... XX011
... 100 00	... X100 0	... XX100
... 101 00	... X101 0	... XX101
... 110 00	... X110 0	... XX110
... 111 00	... X111 0	... XX111
... 000 00	... X000 0	... XX000
... 001 00	... X001 0	... XX001

A WRAP8 can start at any point in the address sequence. When it reaches 111 in the address, it returns to 000 and completes the burst. Note that the actual HADDR is aligned taking into account the transfer size.

A memory directly supports the wrap sequence if it reads the address every access. Therefore an SRAM and NOR Flash with burst mode disabled work correctly when an AHB burst wrap transaction of any length and size is performed.

For a NOR Flash with burst enabled, the controller issues the address on SMAD and SMADQ once only, at the beginning of the burst, and then issues just the clock. This means that if the Flash memory does not support wrap, the controller can manage the wrapped access from AHB by splitting the access into two sequential accesses.

Some burst mode NOR Flash memories support linear burst with wrap around, in which a fixed number of words are read from consecutive addresses modulo N (N is typically 8, 16 or 32, and is programmable in the Flash configuration register). In this case, if all the AHB burst wrap transactions received on the SRAM/NOR Flash controller are aligned with the NOR Flash burst wrap capability, the SRAM/NOR Flash controller can avoid splitting the AHB burst wrap into two consecutive linear bursts when the address is wrapping. This improves system performance by reducing the number of HCLK clock cycles required to perform the complete burst transaction.

If the AHB master (or masters) can generate wrap transactions with changing size, then wrap support must be disabled, and the Flash memory must only work in linear access mode. In that case, the controller always splits the wrap transactions into linear accesses.

16.3.5 Bus turn around

Each time the controller ends a memory read, a single access, or the last of a burst, a data bus turn around condition occurs. That is, the controller inserts ($BUSTURN + 1$) HCLK clock cycles between completion of the current read transaction (the rising edge of SMOEn) and the next transaction.

In particular, if after a memory of any type is read, and a multiplexed NOR Flash memory is accessed, a data bus turn around condition occurs. This is to avoid electrical bus conflict, as in the current transaction (memory read) the data bus is driven by the memory, and has then to be driven by the controller in the next transaction (delivering the address of the new transaction).

To turn from one transaction to another, when there is a memory chip select change, the controller takes at least 3 HCLK cycles if the two AHB transactions are side by side. During these 3 HCLK cycles, the controller does not drive the data bus, and disables all chip-selects and output enable.

16.4 PC card/CompactFlash/NAND Flash controller

This controls the AHB interface to the external PC card, CompactFlash, CF+ or ATA device. One or two slots can be controlled. This controller also supports NAND Flash and SmartMedia devices, with error correction code computation. It is referred to here as the NAND Flash memory controller.

The NAND Flash memory map is divided into 8 consecutive 64-Mbyte areas, that is, four 64-Mbyte partitions per slot.

Table 83. CompactFlash/NAND Flash memory map

Address	Memory space
0x5FFF FFFF	Socket 1 I/O space
0x5C00 0000	
0x5BFF FFFF	Socket 1 attribute memory space
0x5800 0000	
0x57FF FFFF	Reserved
0x5400 0000	
0x53FF FFFF	Socket 1 common memory space
0x5000 0000	
0x4FFF FFFF	Socket 0 I/O space
0x4C00 0000	
0x4BFF FFFF	Socket 0 attribute memory space
0x4800 0000	
0x47FF FFFF	Reserved
0x4400 0000	
0x43FF FFFF	Socket 0 common memory space
0x4000 0000	

Each partition has distinct programmable timing characteristics for read/write accesses, and different behavior, as described below.

- Common memory space access. This can be 8 or 16 bits wide, according to the type of AHB transaction:
 - HSIZE is 0: 8-bit access,
 - HSIZE is 1: 16-bit access,
 - HSIZE is 2: 32-bit access. For a PC-card, these are decomposed into two 16-bit accesses.
 - Transfers larger than 32 bits are not generated by the ARM core, and are not supported by this interface.
- Attribute memory space access. This is 8 bits wide. Every supported AHB transaction is decomposed into a sequence of 8-bit transactions.
- I/O space access. This can be 8 or 16 bits wide, depending on the value of SMPIOIS16n. The controller starts the transaction as if 16-bit transfer is supported. 16-bit PC cards must assert SMPIOIS16n within 35ns, otherwise the controller assumes an 8-bit PC card and continues the transaction accordingly.

Attribute memory read and write functions are performed in the same way as common-memory mode. The only difference is that pin SMPREGn is held low during the transaction, as the data width in such transaction is always 8 bits (SMPCE2n is high and SMPCE1n is low).

During any partition access, SMAD[25:0], SMPREGn, SMPDIRn, and SMPSxnot switch at the same time.

SMPCE2n is used to indicate that the upper half of the data bus (SMADQ[15:8]) is used for the transfer, and SMPCE1n to indicate that the lower half of data bus (SMADQ[7:0]) is used. Both are asserted (low) for 16-bit accesses.

SMPIORn and SMPIOwn are used to indicate I/O space read and write accesses respectively, while SMOEn and SMWEn are used to indicate common memory and attribute space read and write accesses respectively.

SMPREGn is used to indicate read and write accesses to attribute memory space.

Table 84. NAND Flash controller signal activity

Access	Control signal activity								Data bus (SMADQ)	
Type	SMPCE2n	SMPCE1n	SMAD[0]	SMOEn	SMWEn	SMPIORn	SMPIOwn	SMPREGn	[15:8]	[7:0]
Common memory space										
Read	0	0	0	0	1	1	1	1	odd byte	even byte
	1	0	0	0	1	1	1	1	not used	even byte
	1	0	1	0	1	1	1	1	not used	odd byte
Write	0	0	0	1	0	1	1	1	odd byte	even byte
	1	0	0	1	0	1	1	1	not used	even byte
	1	0	1	1	0	1	1	1	not used	even byte
Attribute memory space										
Read	1	0	0	0	1	1	1	0	not used	even byte
	1	0	1	0	1	1	1	0	not used	even byte
Write	1	0	0	1	0	1	1	0	not used	even byte
	1	0	0	1	0	1	1	0	not used	even byte
16-bit I/O space (SMPIOIS16n = 0)										
Read	0	0	0	1	1	0	1	0	odd byte	even byte
Write	0	0	0	1	1	1	0	0	odd byte	even byte
8-bit I/O space (SMPIOIS16n = 1)										
Read	1	0	0	1	1	0	1	0	not used	even byte
	1	0	1	1	1	0	1	0	not used	odd byte
Write	1	0	0	1	1	1	0	0	not used	even byte
	1	0	1	1	1	1	0	0	not used	odd byte

16.4.1 NAND Flash controller connection to PC card

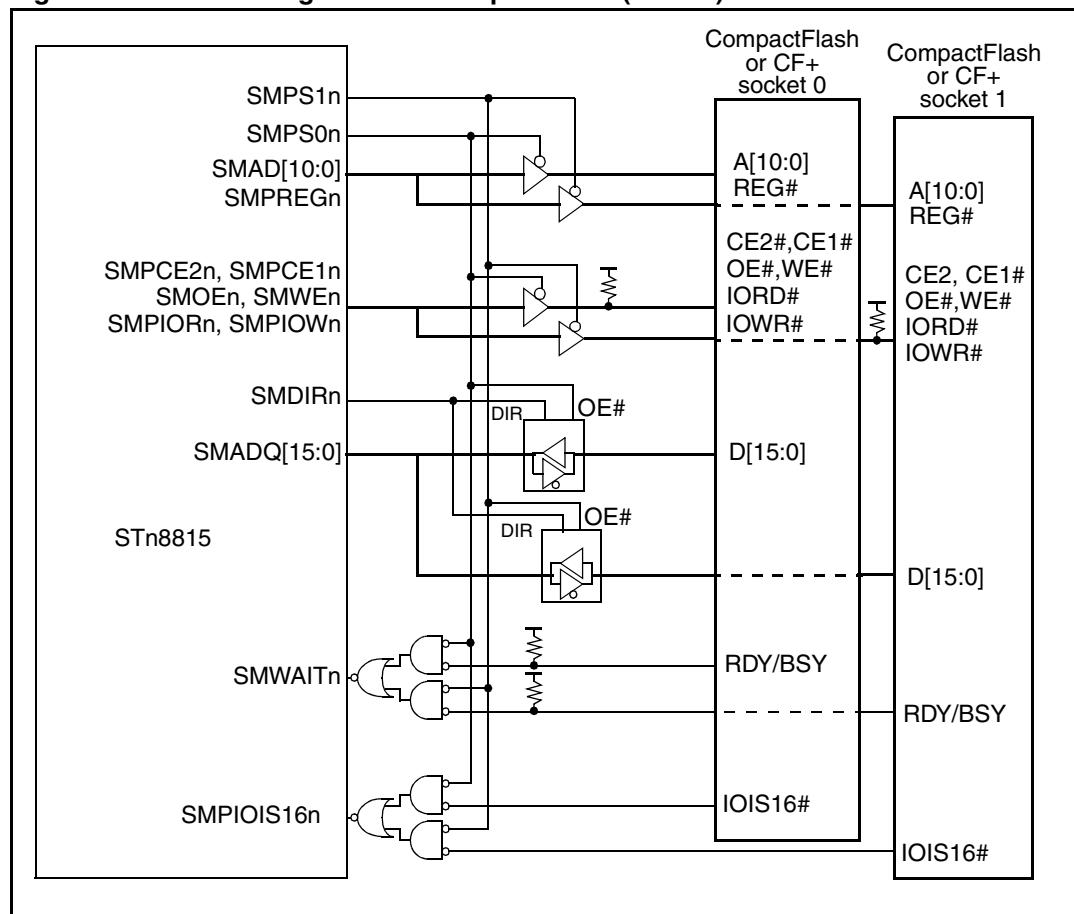
The FSMC requires some external glue logic between the NAND Flash controller and the PC card socket or ATA device.

Level shifting buffers are mandatory, even if dual-voltage (for 3.3 V and 5 V devices) is not supported, because the STn8815A12 has 1.8 V input and output pads only.

Figure 44 shows the general solution for a two-socket configuration, which supports hot insertion/extraction. The pull-up resistors shown are requested as specified in the *PC-Card Standard - Volume 2 - Electrical Specifications*. Low-power systems must remove power from the pull-ups during sleep periods to avoid unnecessary power-consumption.

Note that some CompactFlash or CF+ cards are not directly supported by the FSMC, and thus some GPIOs must be used. For example, R/B (in PC card memory mode) or R/I REQ (in PC card I/O mode) must be connected to a GPIO correctly programmed to trigger an interrupt on edge or level, according to card configuration; -CD1, -CD2 card detect pins can be connected to a single GPIO per slot.

Figure 44. External logic for a 2 CompactFlash (or CF+) socket



16.4.2 NAND Flash controller timing parameters

Each CompactFlash/CF+/NAND Flash chip-select can be configured using three timing parameters that define the number of HCLK cycles for the three phases of any

CompactFlash/CF+ or NAND Flash access, plus one parameter that defines the timing to start driving the data bus in the case of a write.

Figure 45 shows the timing definitions for common-memory accesses, *Figure 46* shows them for attribute-memory accesses, and *Figure 47* shows them for I/O space accesses.

Figure 45. NAND Flash controller timing for common memory access

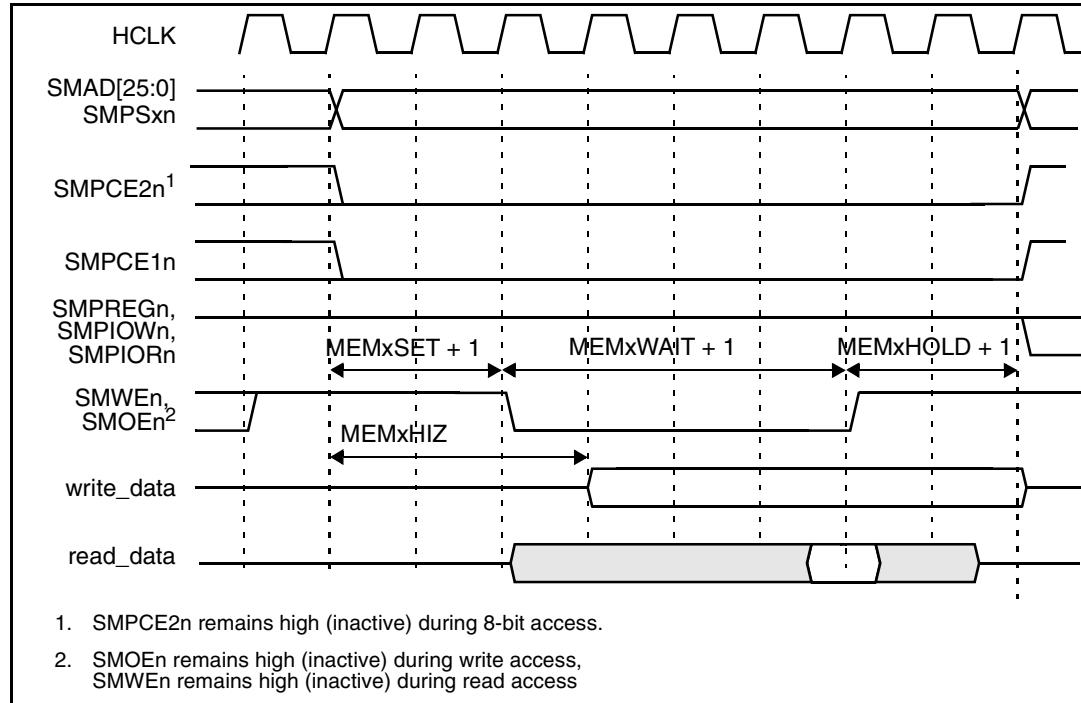


Figure 46. NAND Flash controller timing for attribute memory access

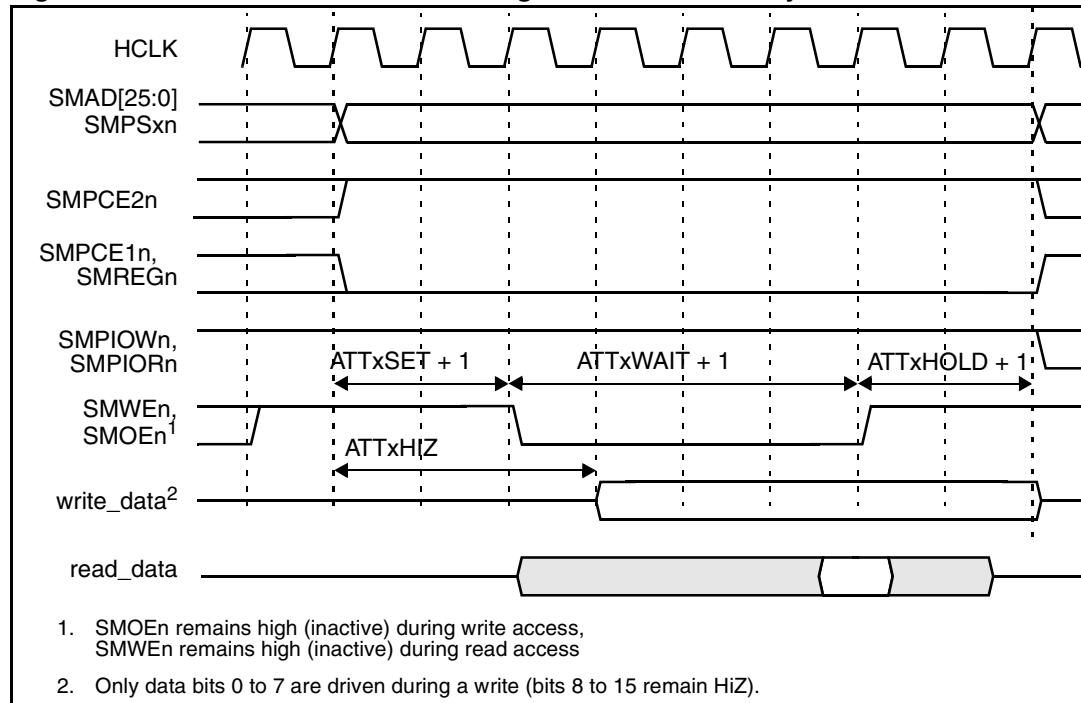
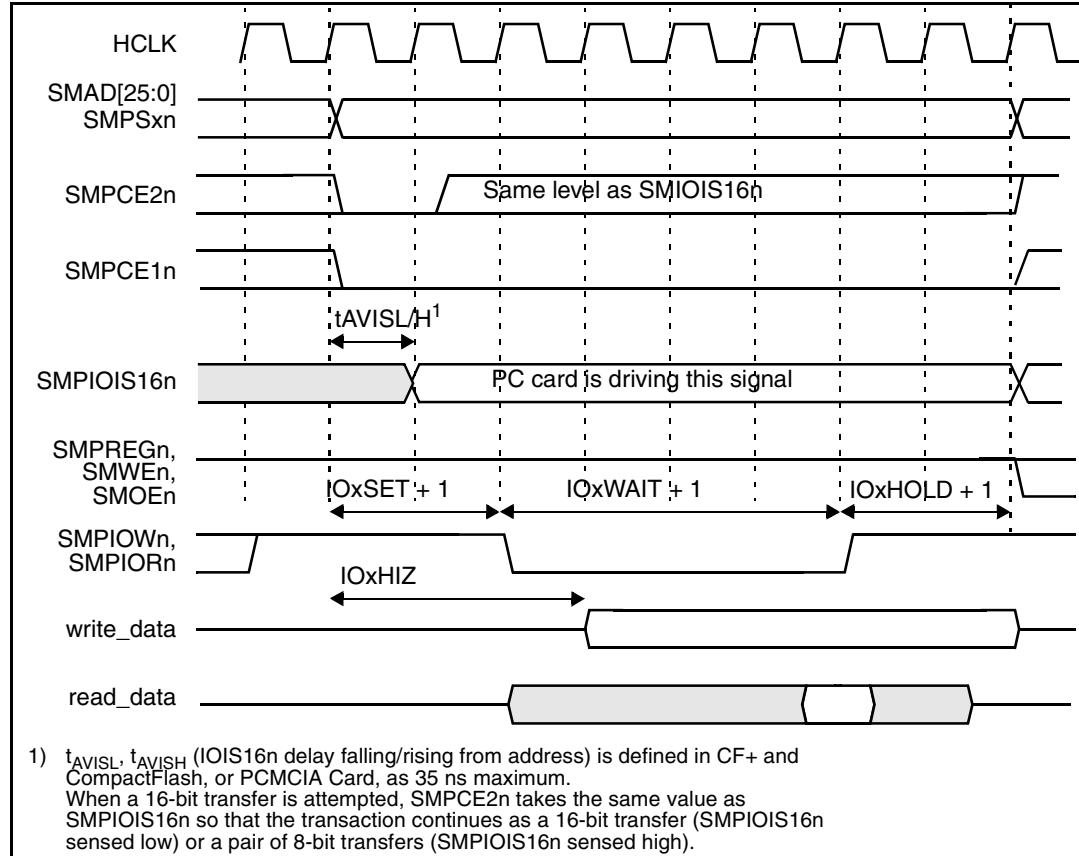


Figure 47. NAND Flash controller timing I/O space access

16.4.3 Wait management by NAND Flash controller

The SMWAITn input can be used to extend PC card and NAND Flash read and write access, independently for each socket or memory chip-select.

For PC cards, signal SMWAITn is driven by the selected card to delay (lengthen) the middle phase, that is, the period during which WE#, OE#, IORD# or IOWD# are low. When SMWAITn is low, the controller continues count down of the internal counters until the middle phase count is completed. After that, if SMWAITn is still low, the controller waits until it goes high and then continues the normal cycle. If at the end of the middle phase, SMWAITn has already returned high, the controller continues without further delay.

For NAND Flash devices, the signal SMWAITn is driven by the selected NAND Flash after an address write cycle or a data read cycle to say it is not ready to accept the next write or read cycle. Note that some NAND Flash devices require that the CE signal is kept low after the last address write cycle in read page operations, until the R/B pin goes from low to high. When SMWAITn is low, the controller holds its state until SMWAITn goes high.

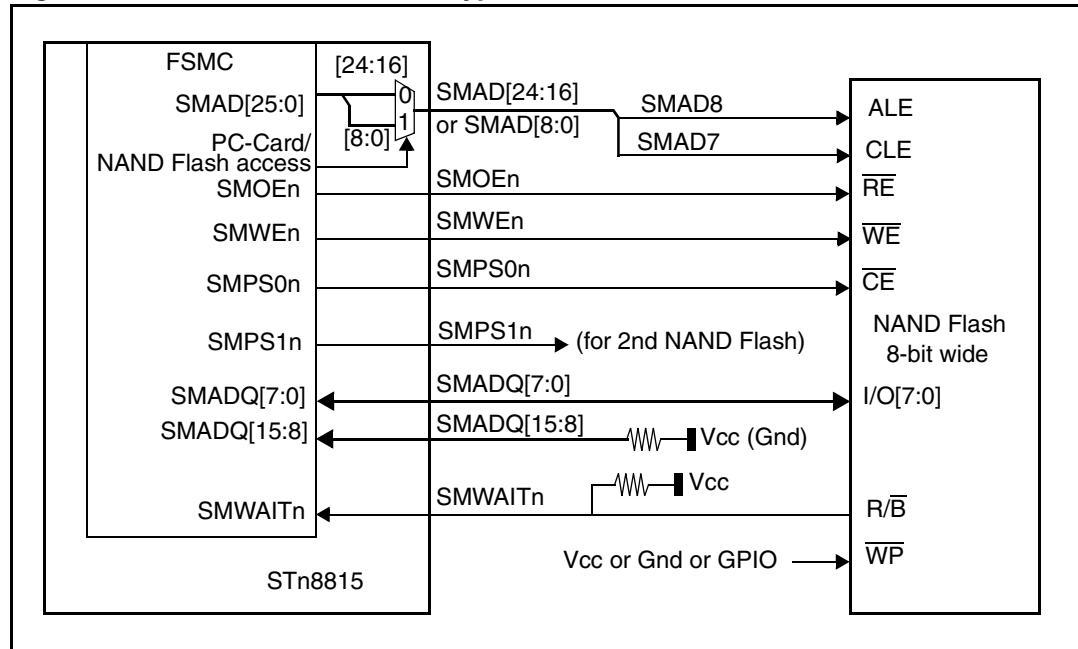
16.4.4 NAND Flash support

The FSMC NAND Flash controller can control one or two 8-bit or 16-bit wide NAND Flash devices without external glue logic. NAND Flash and multiplexed NOR Flash can be simultaneously attached to the FSMC.

The FSMC performs the error code computation (ECC) in hardware, but not the correction. When reading a page from the NAND Flash, the correction must be done through software, according to the validity of the block (refer to [Error correction code on page 221](#)).

When the NAND Flash controller chip-select is programmed to support NAND Flash, the data bus width is programmed in the controller, and is the same for common-memory and attribute-memory space accesses. Note that I/O space must not be used for accessing NAND Flash.

Figure 48. 8-bit wide NAND Flash typical connection to FSMC



NAND Flash operations

As shown in [Figure 48](#), the command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash are driven by address signals of the FSMC controller. This means that to send a command or an address to the NAND Flash, the CPU has to perform a write in its memory space.

A typical page read operation from the NAND Flash selected by signal SMPSxnot ($x = 0$ or 1) is as follows.

1. The corresponding chip-select is configured and enabled according the characteristics of the NAND Flash.
2. The CPU performs a byte write in the common memory space at an address where SMAD[8:7] is 01 (for example, address 0x4000 0080 for the STn8815A12), with a data byte equal to one Flash command byte (for example, 0x00 for Samsung NAND Flash devices). The CLE input of NAND Flash is active during the write strobe (low pulse on \overline{WE}), thus the written byte is interpreted as a command by the NAND Flash. Once the

command is latched by the NAND Flash, it does not need to be written for the following page read operations.

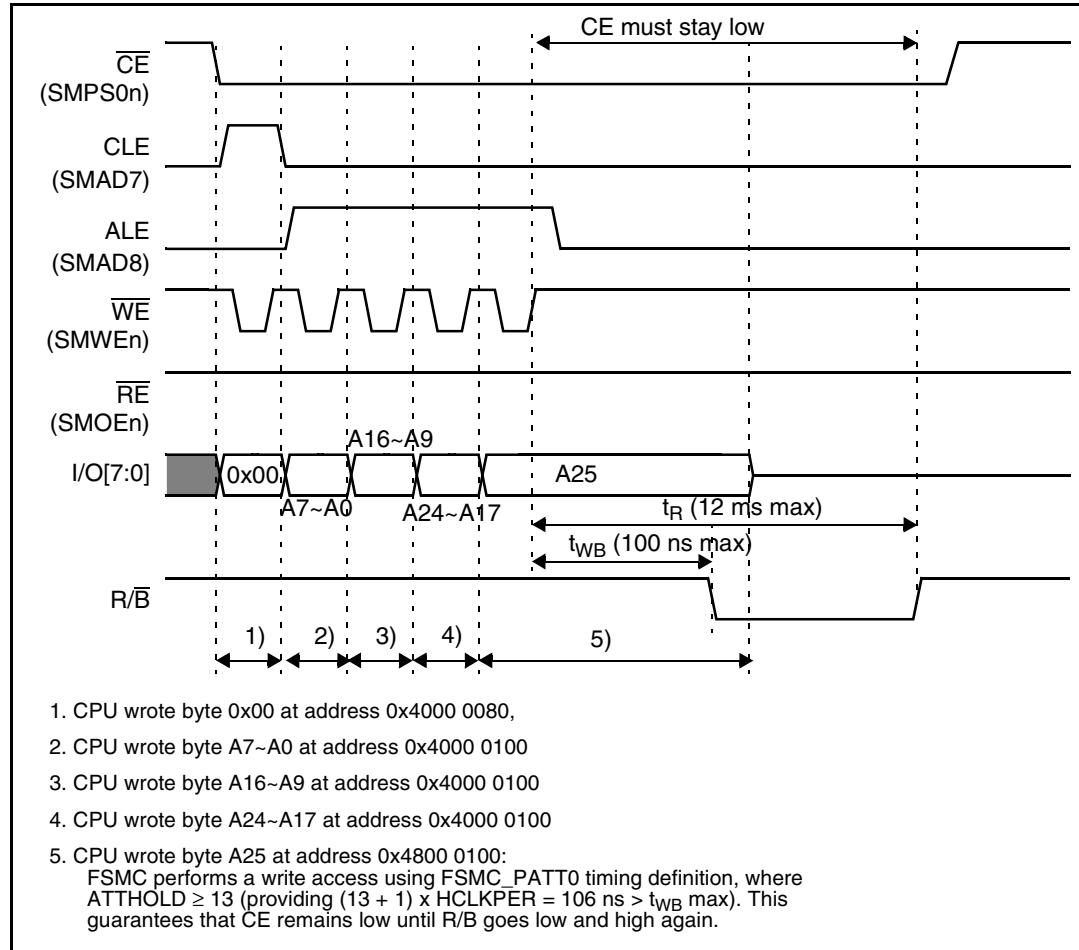
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices) (STARTAD[7:0], then STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] for 64M x 8 Bit NAND Flash) in the common memory or attribute space at an address where SMAD[8:7] = 10b (for example, address 0x4000 0100 or 0x4800 0100 for the STn8815A12). The ALE input of the NAND Flash is active during the write strobe (low pulse on WE), thus the written bytes are interpreted as the start address for read operations. Using the control memory space (address 0x4800 0100) allows a different timing configuration of the FSMC to be used. This can be used to implement the pre-wait functionality needed by some NAND Flash (see detail in [NAND Flash ready/busy management on page 220](#)).
4. The controller waits for Flash ready (R/B pin high) to become high before starting a new access (to the same chip-select or another one). While waiting, the controller maintains CE active (low).
5. The CPU can then perform byte read operations in the common memory space at an address where SMAD[8:7] is 00 (for example, address 0x4000 0000 for the STn8815A12) to read byte per byte the NAND Flash page (data field + spare field).
6. The next NAND Flash page can be read without a CPU command or address write, by repeating step 5; or a new random address can be accessed by restarting at step 3, or a new command can be sent to the NAND Flash by restarting at step 2.

Note: *Some glitches can appear on line addresses when the chip select is inactive. These have no impact on the Flash memory.*

NAND Flash ready/busy management

Some NAND Flash devices require the \overline{CE} signal to stay low after the last address input and during busy state (R/B pin low) for correct operation, as shown in [Figure 49](#).

Figure 49. 8-bit wide NAND Flash timing for a typical connection to FSMC



When this function is needed, it can be guaranteed by programming the MEMHOLD value to meet the t_{WB} timing. Any CPU read or write access to the NAND Flash then has a hold delay of (MEMHOLD + 1) HCLK cycles inserted from WR rising edge to next access.

To overcome this timing penalty, the attribute memory space can be used. Then, the CPU must use common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash, where the CPU must write to the attribute memory space.

In any case, and with every access, R/B low holds the controller with \overline{CE} low.

Error correction code

The FSMC NAND Flash controller includes two error correction code (ECC) computation hardware blocks, one per chip-select, that can reduce host CPU workload when software is processing the error correction code.

The ECC can perform 1-bit error correction and 2-bit error detection functions per 256 or 512 bytes read or written from/to NAND Flash. The two ECC modules monitor the NAND Flash data bus and read/write signals (\overline{RE} , \overline{WE}) each time the NAND Flash chip-select is active.

- When the access to NAND Flash occurs at address 0x4000 0000 for NAND Flash chip-select 0 (or at address 0x5000 000 for chip-select 1), data present on bus SMADQ[15:0] is latched on the rising edge of signals \overline{WE} (SMWEn) or \overline{RE} (SMOEn). The SMAD[7:0] value is entered in the ECC computation unit 0 (or 1). If the data bus width is 16 bits, then one AHB clock cycle later, the SMAD[15:8] value is entered in the ECC computation unit.
- When access to the NAND Flash occurs at any other address, the ECC logic is not used. Thus, write operations for defining command or address to NAND Flash are not taken into account for ECC computation.

After the desired number of bytes have been read from or written to NAND Flash memory chip-select x by the host CPU, the computed value is retrieved.

256 byte mode

When this mode is selected, parity computation takes place only for 256 bytes. ECC logic is activated twice to read/write one page in size of 512 bytes conforming to the standard format of SmartMedia and to obtain the ECC result. The first 256 bytes are processed, and the 3 parity bytes are read, then ECC computation is reset and the other 256 bytes are processed. As only one result is stored, the first result is lost if it is not read before the next 256 bytes are processed.

512 byte mode

When this mode is selected, parity computation takes place for 512 bytes. As only the result of one 512-byte processing is stored, the first result is lost if it is not read before the next 512 bytes are processed.

Customer mode

To increase error coverage, a reduced ECC page size can be used (128, 64, and so on). To use this mode, the result is read after the required number of bytes: for example, for a 128 x 16 page size, processing is stopped at half-word 128. Then the result is read and the ECC reset ready for the next page.

16.5 FSMC address mapping

The AHB address (HADDR) is a 32-bit address, so the master on the bus can address up to a 4-Gbyte address space. Some of the most significant bits are used by the AHB decoder to determine the value of the various HSEL signals. Three of them are connected to this peripheral.

- HSELregs: the HSEL used for configuration. Only HADDR[10:2] are used.
- HSELmem: the HSEL used to address the SRAMs and Flash memories
- HSELPC-Card: HSEL used to address PC cards.

The following tables are used to select the external devices.

Table 85. External static-memory address mapping

HADDR[27:26]	Name of the region
00	Static memory chip-select 0
01	Static memory chip-select 1
10	Static memory chip-select 2
11	Static memory chip-select 3

Table 86. PC card address mapping

HADDR[29:26]	Name of the region
0000	PC card 0, common memory space
0001	Not used
0010	PC card 0, attribute memory space
0011	PC card 0, I/O space
0100	PC card 1, common memory space
0101	Not used
0110	PC card 1, attribute memory space
0111	PC card 1, I/O space
1XXX	Reserved

Each region is 64 Mbytes. If the physical memory is smaller, the region is partially unused. Each region can be enabled and disabled, if there is no need to use all of them.

The controller allows the AHB master to access the external device disregarding its actual data size. For instance, if the external device has a 16-bit data bus and the AHB master is performing a 32-bit data access, the controller accesses the external peripheral twice, with consecutive addresses, to gather the 32-bit data for the master.

16.6 FSMC signals

Table 87 lists and details the signals of the FSMC.

Note: *In the STn8815A12, some signals of the SRAM/NOR Flash controller are multiplexed with signals of the CompactFlash/NAND Flash controller. Refer to [Section 3.1: Signal definitions and assignments](#) for details the pin multiplexing.*

Table 87. FSMC signals

Signal	I/O	Description
SMCKO	O	Clock for synchronous burst Flash. Toggle at the same or sub multiple frequency as HCLK, only during an access to NOR-Flash when the burst mode is enabled.
SMAD[25:0]	O	Address bus, bit 25 to 0.
SMAD[24:16]	O	Address bus, bits 24 to 16.
SMAD[15:0]	O	Address bus, bits 15 to 0.
SMA/DQ[15:0]	Bidir	16-bit parallel multiplexed address or data bidirectional bus.
SMADVn	O	NOR-Flash Address Valid line, active LOW. This signal is only active in multiplexed address/data bus during address phase.
SMCS0n	O	SRAM/NOR-Flash chip-select 0 chip select line.
SMCS1n	O	SRAM/NOR-Flash chip-select 1 chip select line.
SMCS2n	O	SRAM/NOR-Flash chip-select 2 chip select line.
SMPS0n	O	PC card/NAND-Flash slot 0 select line.
SMPS1n	O	PC card/NAND-Flash slot 1 select line.
SMWAITn	I	SRAM/NOR-Flash/PC card/NAND-Flash wait input (active low).
SMOEn	O	SRAM/NOR-Flash output enable line (active low). PC card common and attribute memory output enable line (active low). NAND-Flash read enable (active low).
SMWEn	O	SRAM/NOR-Flash write enable (active low). PC card common and attribute memory write enable (active low). NAND-Flash write enable (active low).
SMFWP0n	O	NOR-Flash write protect (active low) for NOR-Flash chip-select.
SMFWP1n	O	NOR-Flash write protect (active low) for NOR-Flash chip-select.
SMFWP2n	O	NOR-Flash write protect line, active low, for NOR-Flash chip-select 2.
SMFRST0n	O	NOR-Flash reset line, active low, for NOR-Flash chip-select 0.
SMFRST1n	O	NOR-Flash reset line, active low, for NOR-Flash chip-select 1.
SMFRST2n	O	NOR-Flash reset line, active low, for NOR-Flash chip-select 2.
SMPREGn	O	PC card attribute memory space selection line -REG (active low).
SMPIORn	O	PC card I/O space read enable line -IORD (active low).
SMPIOWn	O	PC card I/O space write enable line -IOWR (active low).
SMPCE1n	O	PC card low-byte lane enable line -CE1 (active low).
SMPCE2n	O	PC card high-byte lane enable -CE2 (active low).

Table 87. FSMC signals

Signal	I/O	Description
SMDIRn	O	Data-bus direction control signal (low for a write, high for a read).
SMPRST0n	O	PC card/CF/CF+ reset (active low) for slot 0.
SMPRST1n	O	PC card/CF/CF+ reset (active low) for slot 1.
SMPCD0n	I	PC card/CF/CF+ card detect (active low) for slot 0.
SMPCD1n	I	PC card/CF/CF+ card detect (active low) for slot 1.
SMPIOIS16n	I	PC card I/O operation bus width. When the PC-Card (CF, CF+, PCMCIA) is configured for I/O operation, a low level on this pin indicates that a 16-bit or odd byte only operation can be performed at the addressed port.

17 DMA controllers (DMAC)

The STn8815A12 platform provides two controllers for direct memory access (DMA) transfers. These DMA controllers are referred to as DMA0 and DMA1. They allow on-chip data transfers with minimum CPU intervention.

DMA0 and DMA1 are both 8-channel, dual-AHB master port, DMA controllers. Together, they are functionally equivalent to a 16-channel, quad-AHB master port, DMA controller. This arrangement can double the transfer rate compared with a 16-channel, dual-AHB master port, device.

17.1 Features

- 8 DMA channels: these allow eight data streams running in parallel. Each channel can support a unidirectional transfer.
- 32 DMA request lines.
- Single DMA and burst DMA request signals: each peripheral connected to the DMA controller can assert either a single-word DMA request or a burst (multi-word) DMA request. The DMA burst size is configurable.
- Full range of data transfer routes: memory-to-memory, memory-to-peripheral, peripheral-to-memory and peripheral-to-peripheral transfers are supported.
- Scatter and gather DMAs: these are supported through the use of linked lists.
- Hardware DMA channel priority: each DMA channel has a specific hardware priority. DMA channel 0 has the highest priority and channel 7 has the lowest priority. If requests from two channels become active at the same time, the channel with the highest priority is serviced first.
- Incrementing or non-incrementing addressing for source and destination.
- Programmable DMA burst size: the DMA burst size can be programmed for more efficient data transfers. Usually the burst size is set to half the size of the FIFO in the peripheral.
- Internal 4-word FIFO for each channel.
- 8-, 16- and 32-bit wide transactions.
- Big-endian and little-endian modes: the DMA controller defaults to little-endian mode on reset.
- Separate and combined DMA error and DMA count interrupt requests.
- Interrupt masking: the DMA error and DMA count interrupt requests can be masked.
- Raw interrupt status: the DMA error and DMA count raw interrupt status can be read prior to masking.
- Two 32-bit AHB bus masters for transferring data: these interfaces are used to transfer data when a DMA request goes active.

17.2 Overview

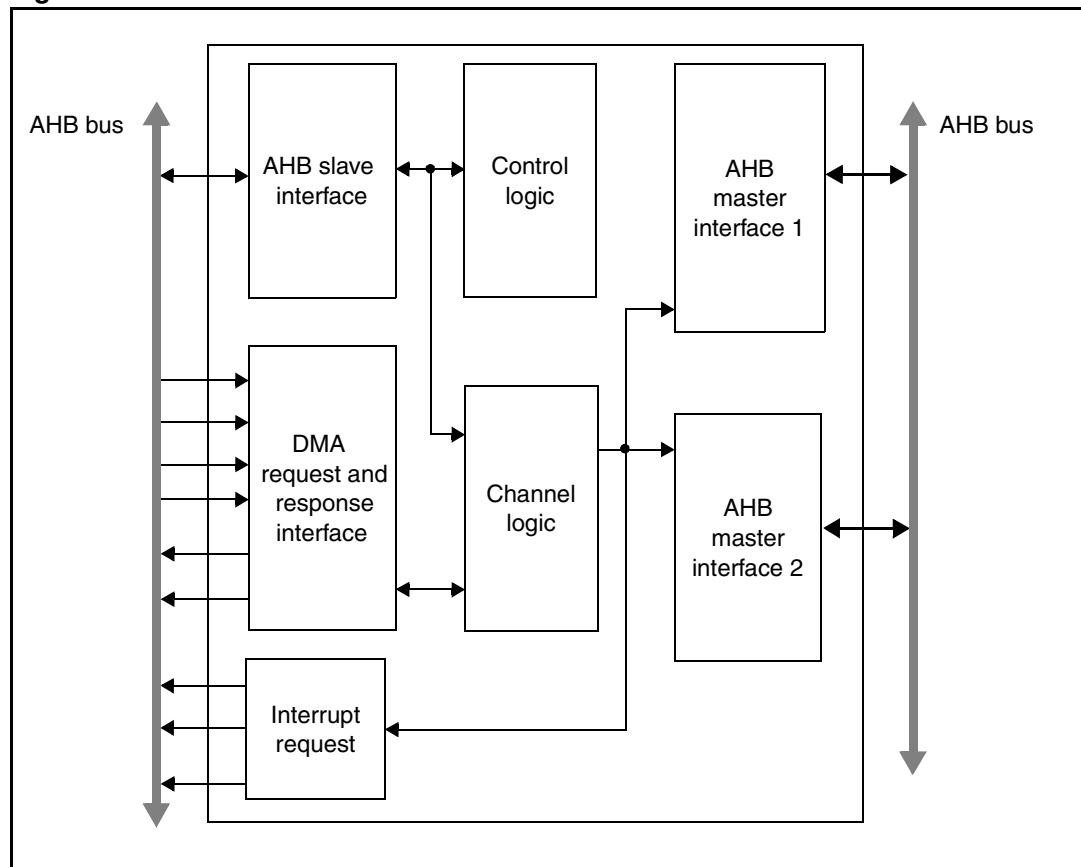
The DMA controller allows peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral and memory-to-memory transactions.

Each DMA stream provides unidirectional serial DMA transfers for a single source and destination. Therefore, a bidirectional port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral, and can be accessed through the same AHB master, or one area by each master.

Each stream is supported by a dedicated hardware channel, including source and destination controllers, and a FIFO. This allows for better latency than a DMA controller with only a single hardware channel shared between several DMA streams.

Figure 50 illustrates the architecture of the DMA controller, and the elements are described below.

Figure 50. DMA controller architecture



The DMA controller contains two full AHB masters. This allows, for example, the DMA controller to transfer data directly from the memory connected to AHB port 1 to any AHB peripheral connected to AHB port 2. It also allows transactions between the DMA controller and any APB peripheral to occur independently of those on the AHB bus 1.

All transactions on the AHB slave bus are 32 bits wide. This eliminates endian issues when programming the DMA controller.

Interrupts

The DMA controller can generate individual, maskable, active-high interrupts. A combined interrupt output is also generated as an OR function of the individual interrupts. Only the combined interrupt is connected to the platform's vectored interrupt controller (VIC).

Interrupt requests can be generated when an AHB error is encountered, or at the end of a transfer (terminal count) after all the data corresponding to the current LLI has been transferred to the destination.

17.3 Functional description

This section describes a single DMA controller, but the information is equally applicable to both DMA0 and DMA1. The differences between the two are as follows.

- DMA1 has higher priority than DMA0.
- DMA0 can perform transfers to and from APB-bus peripherals, while DMA1 can perform transfers to and from AHB-bus peripherals. However, DMA1 can also perform peripheral-to-peripheral transfers between most APB-bus peripherals and AHB-bus peripherals.

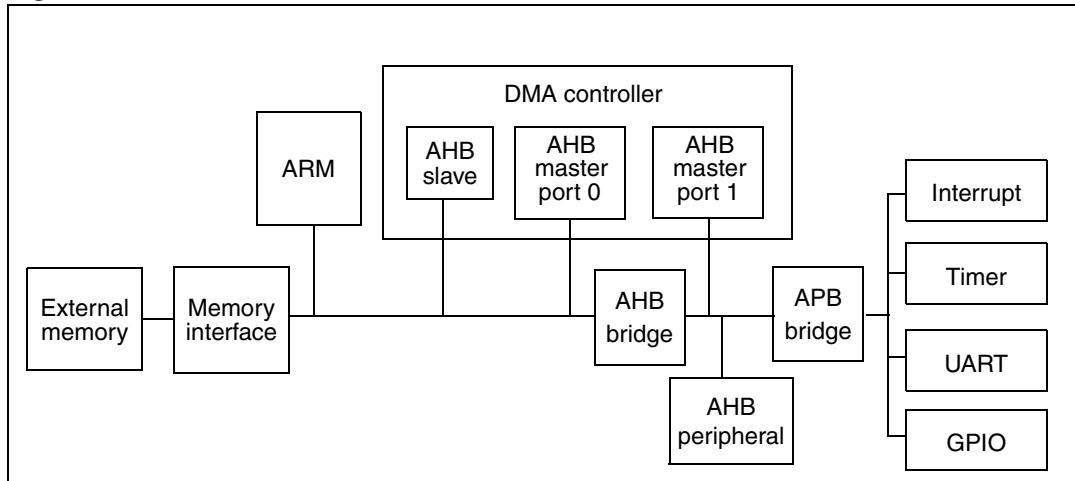
17.3.1 AHB master interfaces

The DMA controller contains two full AHB masters. This allows, for example, the DMA controller to transfer data directly from the memory connected to AHB master port0 to any AHB peripheral connected to AHB master port1. It also allows transactions between the DMA controller and any APB peripheral to occur independently of transactions on the AHB bus attached to master port0.

The two AHB masters are each capable of dealing with all types of AHB transactions, including:

- Split, retry, and error responses from slaves. If a peripheral performs a split or retry, the DMA controller stalls, and waits until the transaction completes.
- Locked transfers for source and destination of each stream.
- Setting of protection bits for transfers on each stream.

All AHB signals are connected as defined in the AHB specification. If the two AHB masters are required to be synchronous, they must use the same HCLK. Support for asynchronous AHB buses is not defined within the DMA controller, and must be implemented using wrappers, if required. The two AHB masters are shown in [Figure 51](#) (AHB port 1 and AHB port 2).

Figure 51. Dual AHB masters

Bus and transfer widths

The two AHB masters are connected to 32-bit wide buses. Source and destination transfers can be of differing widths; byte (8-bit data), half-word (16-bit data) or word (32-bit data). The DMA controller packs or unpacks data as appropriate.

Endianness

The DMA controller can cope with both little-endian and big-endian addressing. The endianness of each AHB master can be set individually.

Error conditions

An error during a DMA transfer is flagged directly by the peripheral by asserting an error response on the AHB bus during the transfer. The DMA controller automatically disables the DMA stream after the current transfer has completed and can optionally generate an error interrupt to the CPU. This error interrupt can be masked.

Protection control

The following protection features are provided:

- **User or privileged:** This can be used to protect certain peripherals or memory spaces from user mode transactions. For a linked list item (LLI), this is set to privileged.
- **Bufferable or non-bufferable:** This can be used to indicate to an AMBA bridge that the write can complete in zero wait-states on the source bus (that is, without waiting for it to arbitrate for the destination bus and for the slave to accept the data). For an LLI, this is set to non-bufferable.
- **Cacheable or non-cacheable:** This can be used by an AMBA bridge so that on the first read of a burst of eight, the whole burst of eight reads can be transferred on the destination bus rather than having to pass the transactions through one at a time. For an LLI, this is set to cacheable.

Locked transfers

The DMA controller supports the lock mode in which a DMA transfer is indivisible. In this mode, no other bus master can be given access to the bus until the current transfer has finished.

17.3.2 DMA request priority

DMA channel priority is fixed, with DMA channel 0 having the highest priority and DMA channel 7 having the lowest priority.

If the DMA controller is transferring data for a lower priority channel and then a higher priority channel becomes active, it completes the number of transfers specified to the master interface by the lower priority channel before switching over to the data transfer for the higher priority channel. In the worst case, this is as large as one quad-word.

The two lowest priority channels (6 and 7) in the DMA controller are designed so that they cannot saturate the AHB bus. If one of these lowest priority channels becomes active, the DMA controller relinquishes control of the bus (for a bus cycle) after four transfers of the programmed size, irrespective of the size of the transfer. This allows other AHB masters to access the bus.

It is recommended that memory-to-memory transactions use one of these low priority channels. Otherwise, other (lower priority) AHB masters are prevented from accessing the bus during memory-to-memory transfers.

17.3.3 Channel hardware

Each stream is supported by a dedicated hardware channel, including source and destination controllers, and a FIFO. This simplifies the control logic, and allows for better latency than a DMA controller with only a single hardware channel shared between several DMA streams.

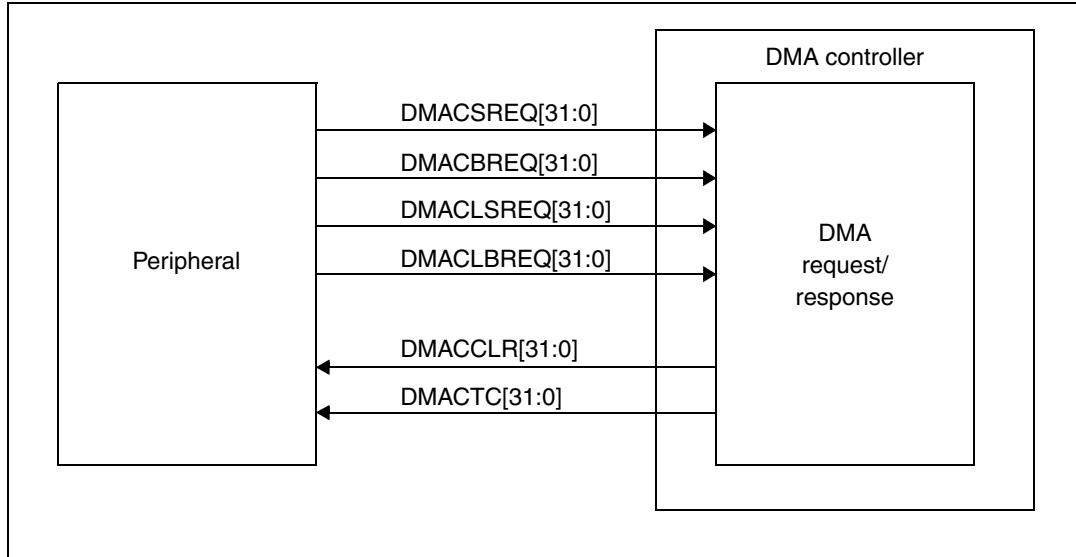
17.3.4 System considerations

Reducing the number of transactions that occur on the buses reduces the latency on the buses, improves system performance and reduces power consumption. Therefore, the following design considerations have been built into the platform for optimization purposes.

- All memory transactions are normally 32 bits wide to improve bus efficiency.
- Peripherals with a natural word size less than 32 bits contain byte or half-word packing hardware so that all transactions can be 32 bits wide.
- Slow peripherals that normally use wait-states, contain FIFOs so that data can be transferred at full speed using burst transfers.

17.4 DMA request and response connectivity

The diagram shows how the DMA request and response signals can be connected to a peripheral. However some peripherals do not make use of all these signals. Output signals that are not required can be left unconnected, but input signals that are not required must be tied low.

Figure 52. DMA request/response connectivity

The DMA request signals are used by peripherals to request a data transfer.

The DMA request signals indicate:

- whether a single word or a burst (multi-word) transfer of data is required
- whether the transfer is the last in the data packet

The DMA request signals that the DMA can receive from a peripheral are as follows.

- DMACBREQ[31:0]: burst transfer request signal. This causes a programmed burst number of words to be transferred, to or from the peripheral.
- DMACSREQ[31:0]: single transfer request signal. This causes a single word to be transferred, to or from the peripheral.
- DMACLBREQ[31:0]: last burst transfer request signal. This signal is only needed when the peripheral is the flow controller.
- DMACLSREQ[31:0]: last single transfer request signal. This signal is only needed when the peripheral is the flow controller.

If a peripheral transfers only bursts of data, it is not necessary to connect the single transfer request signals. If a peripheral transfers only single words of data, it is not necessary to connect the burst request signals.

17.5 Peripheral request lines

The assignments of the DMA request lines to peripherals are given in the tables below.

- [Table 88](#) gives the DMA0 request line assignments to APB peripherals.
- [Table 89 on page 232](#) gives the DMA1 request line assignments to APB and AHB peripherals.

Table 88. DMA0 request line assignments

Request line	Associated peripheral	Request line	Associated peripheral
15	UART0 Tx	31	MSP1 Tx
14	UART0 Rx	30	MSP1 Rx
13	SSP Tx	29	
12	SSP Rx	28	
11	MSP0 Tx	27	
10	MSP0 Rx	26	
9	FIrDA Tx/Rx	25	
8	Not used	24	
7	Smart audio accelerator, channel 7	23	MSP2 Tx
6	Smart audio accelerator, channel 6	22	MSP2 Rx
5	Smart audio accelerator, channel 5	21	
4	Smart audio accelerator, channel 4	20	
3	Smart audio accelerator, channel 3	19	
2	Smart audio accelerator, channel 2	18	
1	Smart audio accelerator, channel 1	17	
0	Smart audio accelerator, channel 0	16	

Table 89. DMA1 request line assignments

Request line	Associated peripheral	Request line	Associated peripheral
15	UART0 Tx	31	UART2 Tx
14	UART0 Rx	30	UART2 Rx
13	SSP Tx	29	USB-OTG, channel 5
12	SSP Rx	28	USB-OTG, channel 4
11	MSP0 Tx	27	USB-OTG, channel 3
10	MSP0 Rx	26	USB-OTG, channel 2
9	IrDA Tx/Rx	25	USB-OTG, channel 1
8	Not used	24	USB-OTG, channel 0
7	Smart audio accelerator, channel 7	23	UART1 Tx
6	Smart audio accelerator, channel 6	22	UART1 Rx
5	Smart audio accelerator, channel 5	21	SD/MM-card Tx/Rx
4	Smart audio accelerator, channel 4		
3	Smart audio accelerator, channel 3	19	I2C0 Tx/Rx
2	Smart audio accelerator, channel 2	18	I2C1 Tx/Rx
1	Smart audio accelerator, channel 1		
0	Smart audio accelerator, channel 0		

18 General-purpose inputs/outputs (GPIOs)

There are 124 general-purpose input/output (GPIO) pins, in four 32-GPIO blocks. This section describes the features of one of the 32-GPIO blocks.

Each GPIO provides 32 programmable inputs and outputs that can be controlled in two modes.

- Software mode, through an APB bus interface
- Alternate mode, in which the GPIO becomes an input or output line for one to three on-chip peripherals

Note:

- 1 *For GPIO block 3 (pins GPIO96 to GPIO128), the 4 upper bits are not delivered on pins.*
- 2 *For GPIO3 block (GPIO[123:96]), the reset values of the alternate function selection registers select the alternate function A by default after power-on reset, and pull resistor are disabled by default after reset.*

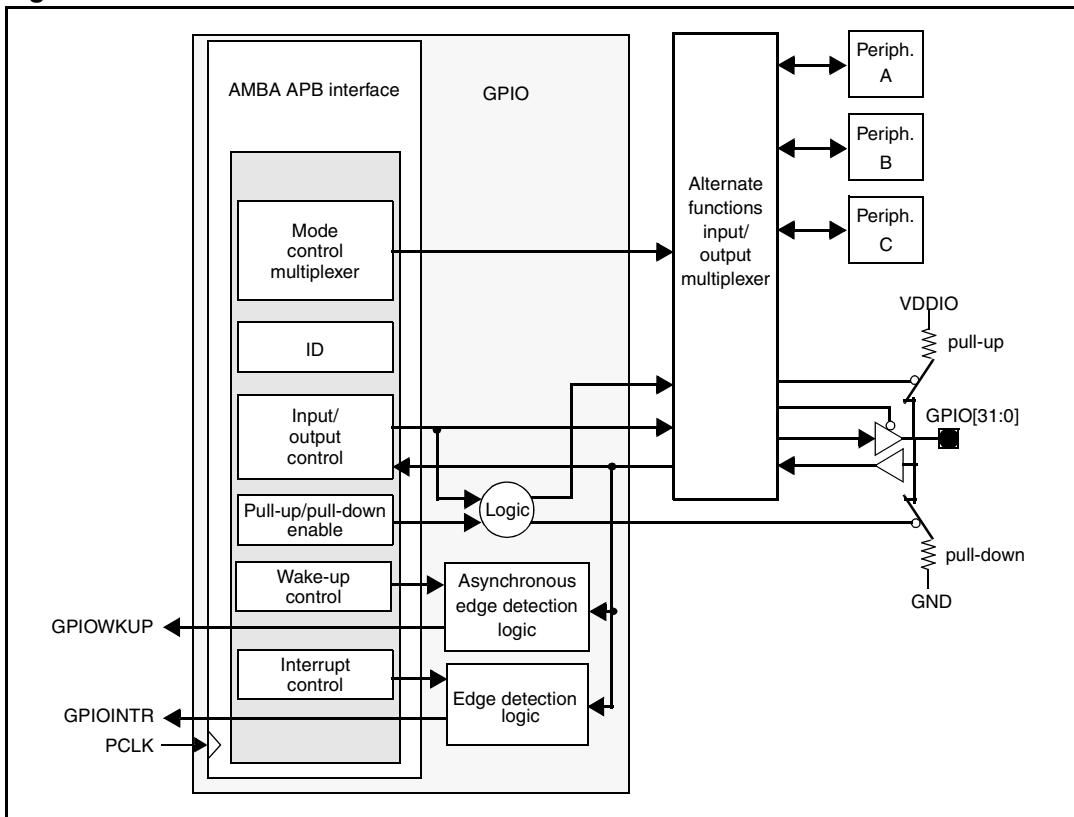
18.1 Features

- An interrupt interface is provided to configure any number of pins as interrupt sources
- A wake-up feature is used to wake up the platform from sleep mode

18.2 Functional description

Figure 53 shows the architecture of a GPIO block.

Figure 53. GPIO architecture



Each GPIO has one to three alternate functions, used for peripheral inputs or outputs. GPIO alternate functions are described in [Section 3.3 on page 69](#). If an alternate function is not programmed, selecting it puts the corresponding pin into HiZ.

Interrupts

Interrupt generation can be based on the transitional value of a pin. The source and edge properties of each interrupt can be configured.

When one or more GPIO lines cause an interrupt, a single interrupt output, **GPIOINTR**, is sent to the vectored interrupt controller (VIC). Software must clear the interrupt to enable any further interrupts.

Interrupt detection logic works with the PCLK clock. The GPIO signals are synchronized to PCLK, except when the GPIO is configured as an output: in that case, the data is already synchronous to PCLK, and the synchronization stage is by-passed.

18.3 GPIO states after reset

At power-on reset (PORn):

- pins are inputs
- blocks 0 to 2 (GPIO[0:95]) are configured as inputs (under software control), with pull resistors enabled
- block 3 (GPIO[96:123]) is in alternate function A mode, with pull resistors disabled
- interrupts are disabled

18.3.1 Operation of input/output pins

Data and direction for the pins can be selected in software. For reads, a pin can be configured as an input or output. Writes affect only the pins that are configured as outputs.

18.3.2 Wake-up

This feature is independent of the interrupt enable/disable configuration.

One or more GPIO pins can be used to wake the platform from sleep mode. The rising or falling edge that triggers the wake-up can be configured in software. The pin or pins that requested the wake-up can be determined by software.

When one or more configured GPIO pins receive a wake-up event, a single wake-up output, GPIOWKUP, is sent to the power management unit (PMU).

Note: *The wake-up event must last at least four 32 kHz clock cycles to be captured.*

18.3.3 Alternate function control

At any one time, one of the three alternate functions associated with a GPIO line can be enabled. In alternate function mode, the direction and level of each GPIO pin is controlled by the associated peripheral. Data and control transfers over the GPIO pin are then driven or read by the peripheral or by internal logic.

18.4 GPIO signals

Table 90 shows the signals for the GPIO.

Table 90. GPIO signals

Signal	I/O	Description
GPIO[123:0]	I/O	General purpose input/output pin.

Table 6 on page 69 summarizes the alternate functions for each pin. If an alternate function is not available for a pin, selecting it puts the pin in HiZ.

19 Asynchronous serial ports

The STn8815A12 platform has three asynchronous serial ports (UARTs) that perform many of the functions of the industry-standard 16C650 universal asynchronous receiver/transmitter unit.

19.1 Features

- full duplex operation using simultaneous, independent, transmission and reception
- programmable baud rates up to UARTCLK/16 (3.0 Mbit/s with UARTCLK at 48 MHz) or up to UARTCLK/8 (6 Mbit/s with UARTCLK at 48 MHz), with a fractional baud-rate generator
- 5, 6, 7 or 8-bit data
- even, odd, stick or no-parity bit generation and detection
- 1 or 2 stop bit generation
- automatic extraction of UART baud rate setting, character size (7- or 8-bit), parity configuration and number of stop bits
- support for the modem control functions
 - CTS and RTS (all UARTs)
 - DCD, DSR, RTS, DTS and RI (UART0 only)
- support of software flow control using programmable Xon/Xoff characters
- false start bit detection
- line break generation and detection
- separate 8-bit wide, 64-deep transmit FIFO and 12-bit wide, 64-deep receive FIFO
- programmable FIFO disabling for 1-byte depth data path
- support for direct memory access (DMA) transfers

19.2 Overview

This section describes the functions and signals of one UART unit. The information is applicable to all UARTs, the only differences between the units being the signals that the UART generates:

- UART0 provides all modem control signals.
- UART1 and UART2 provide only transmit/receive signals and hardware flow control signals (RTS/CTS).

All UARTs perform:

- serial-to-parallel conversion on data asynchronously received from a peripheral device on the UART receive pins (URXD[0:2]),
- parallel-to-serial conversion on data written by the CPU for transmission on the UART transmit pins (UTXD[0:2]).

The transmit and receive paths are buffered with internal FIFOs, allowing up to 64 data bytes for transmission and 64 data bytes with 4-bit status (break, frame, parity and overrun) for reception. FIFOs may be burst-loaded or emptied by the system processor or by DMA, with 1 to 16 words per transfer.

These UARTs differ from the industry-standard 16C650 on some minor points, as follows.

- The receive FIFO trigger levels are provided.
- The internal register map address space and the bit functions of each register are different.
- The deltas of the modem status signals are not available.
- 1.5 stop bits is not supported.
- The independent receive clock feature is not supported.

19.3 Functional description

19.3.1 Data transmission

Data is transmitted via a 64-byte FIFO.

The data for transmission is written into the transmit FIFO. If the UART is enabled, a data frame is transmitted with the specified parameters. Data continues to be transmitted until there is no data left in the transmit FIFO.

A busy flag is set as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted while data is being transmitted. The flag is cleared only when the transmit FIFO is empty and the last character has been transmitted from the shifter, including the stop bits. The busy flag can be set even when the UART is not enabled.

For each sample of data, three readings are taken and the most common value is kept. The middle sampling point is defined and one sample is taken on either side.

19.3.2 Data reception

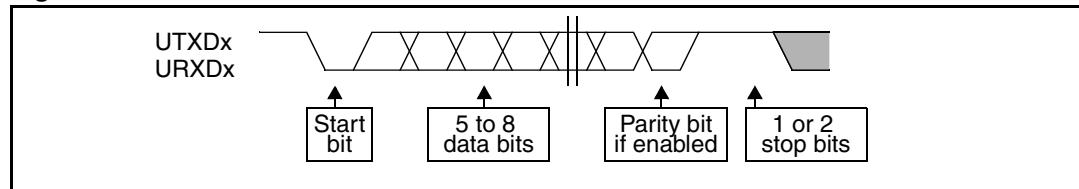
Data is received via a 64-byte FIFO that has an extra four bits per character for status information.

When the receiver is idle (the receive pin is continuously high, in the marking state) and a low is detected on the data input (indicating that a start bit has been received), the receive counter begins running with the clock configured for Baud16 (see [Section 19.3.4 on page 238](#)). Data is sampled on the eighth cycle of the counter.

The start bit is valid if the receive pin is still low on the eighth cycle of Baud16, otherwise a false start bit is detected and is ignored. If the start bit is valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. If parity mode has been enabled, the parity bit is then checked. Finally, a valid stop bit is confirmed if the receive pin is high, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, along with any error bits associated with that word (see [Table 91 on page 238](#)).

The UART character frame is shown in [Figure 54](#).

Figure 54. UART character frame



19.3.3 Transmit and receive FIFOs

Data received or transmitted is stored in two 64-byte FIFOs. The receive FIFO has an extra four bits per character for status information, as follows:

- Error bits: bits 8 to 10,
- Overrun indicator: bit 11.

Table 91 shows the bit functions of the receive FIFO.

Table 91. Receive FIFO bit functions

FIFO bits	Function
11	Overrun indicator
10	Break error
9	Parity error
8	Framing error
[7:0]	Received data

Error bits

Three error bits are stored in bits [10:8] of the receive FIFO and are associated with a particular character.

Overrun bit

The overrun bit is not associated with the character in the receive FIFO. The overrun error is set when the FIFO is full and the next character has been completely received in the shifter. The data in the shifter is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO, and another character is received, the state of the overrun bit is copied into the receive FIFO along with the received character. The overrun state is then cleared.

Disabling the FIFOs

The FIFOs can be disabled. If they are disabled, the transmit and receive sides of the UART have 1-byte holding registers (the bottom entry of each FIFO). The overrun bit is set when a word has been received and the previous one has not yet been read. In this implementation, the FIFOs are not physically disabled, but the flags are set to give the effect of a 1-byte register.

19.3.4 Baud rate divisor

The baud rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. This is used by the baud rate generator to determine the bit period. The fractional baud rate divider allows any clock to act as the UART clock (UARTCLK), while still allowing all the standard baud rates to be generated.

Depending on whether the factor 16 or 8 is selected, the relationship between the baud rate divisor (BRD) and UARTCLK is:

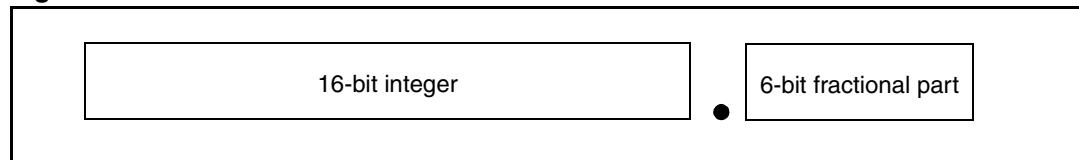
$$\text{BRD} = \text{UARTCLK} / (16 \times \text{Baud rate}) = \text{BRD}_I + \text{BRD}_F$$

or

$$\text{BRD} = \text{UARTCLK} / (8 \times \text{Baud rate}) = \text{BRD}_I + \text{BRD}_F$$

where BRD_I is the integer part and BRD_F is the fractional part, separated by a decimal point, as shown in [Figure 55](#).

Figure 55. Baud rate divisor



To calculate the fractional 6-bit number (DIVFRAC):

- take the fractional part of the required baud rate divisor and multiply it by 64,
- add 0.5 to account for rounding errors.

That is:

$$\text{DIVFRAC} = \text{integer}(\text{BRD}_F * 64 + 0.5)$$

The maximum deviation error is $1/64 * 100 = 1.56\%$. This occurs when $\text{DIVFRAC} = 1$, and the error is cumulative over 64 clock ticks.

Baudl6 is an internal clock enable signal. It is generated using a stream of pulses, of width UARTCLK cycle, with an average frequency of 16 or 8 times the desired baud rate. This signal is then divided by 16 or 8 to give the transmit clock. A baud rate divisor with a low value gives a short bit period and a baud rate divisor with a high value gives a long bit period.

The tables below show some typical bit rates and their corresponding divisors, given a UART clock frequency of 48 MHz.

- [Table 92](#) corresponds to a factor of 16.
- [Table 93](#) corresponds to a factor of 8.

Table 92. Typical baud rates and integer/fractional divisors (factor 16)

Required bit rate (bit/s)	Programmed divisor		Generated bit rate (bit/s)	Error (%)
	Integer part	Fractional part		
3 000 000	1 (0x0001)	0 (0x00)	3 000 000	0.000
1 843 200	1 (0x0001)	41 (0x29)	1 828 571	0.794
921 600	3 (0x0003)	17 (0x11)	918 660	0.319
460 800	6 (0x0006)	33 (0x21)	460 432	0.080
230 400	13 (0x000D)	2 (0x02)	230 215	0.080
115 200	26 (0x001A)	3 (0x03)	115 177	0.020
38 400	78 (0x004E)	8 (0x08)	38 400	0.000
14 400	208 (0x00D0)	22 (0x16)	14 399	0.005
9 600	312 (0x0138)	32 (0x20)	9 600	0.000
2 400	1250 (0x04E2)	0 (0x00)	2 400	0.000
1 200	2500 (0x09C4)	0 (0x00)	1 200	0.000
110	27272 (0x6A88)	47 (0x2F)	110	0.000

Table 93. Typical baud rates and integer/fractional divisors (factor 8)

Required bit rate (bit/s)	Programmed divisor		Generated bit rate (bit/s)	Error (%)
	Integer part	Fractional part		
6 000 000	1 (0x0001)	0 (0x00)	6 000 000	0.000
5 529 600	1 (0x0001)	6 (0x06)	5 485 714	0.794
3 686 400	1 (0x0001)	41 (0x29)	3 657 142	0.794
1 843 200	3 (0x0003)	17 (0x11)	1 837 320	0.319
921 600	6 (0x0006)	33 (0x21)	920 863	0.080
460 800	13 (0x000D)	2 (0x02)	460 431	0.080
230 400	26 (0x001A)	3 (0x03)	230 353	0.020
115 200	52 (0x0034)	6 (0x06)	115 177	0.020
38 400	156 (0x009C)	16 (0x10)	38 400	0.000
14 400	416 (0x01A0)	43 (0x2B)	14 400	0.001
9 600	625 (0x0271)	0 (0x00)	9 600	0.000
2 400	2500 (0x09C4)	0 (0x00)	2 400	0.000
1 200	5000 (0x1388)	0 (0x00)	1 200	0.000
110	54545 (0xD511)	30 (0x1E)	110	0.000

19.3.5 System and diagnostic loopback testing

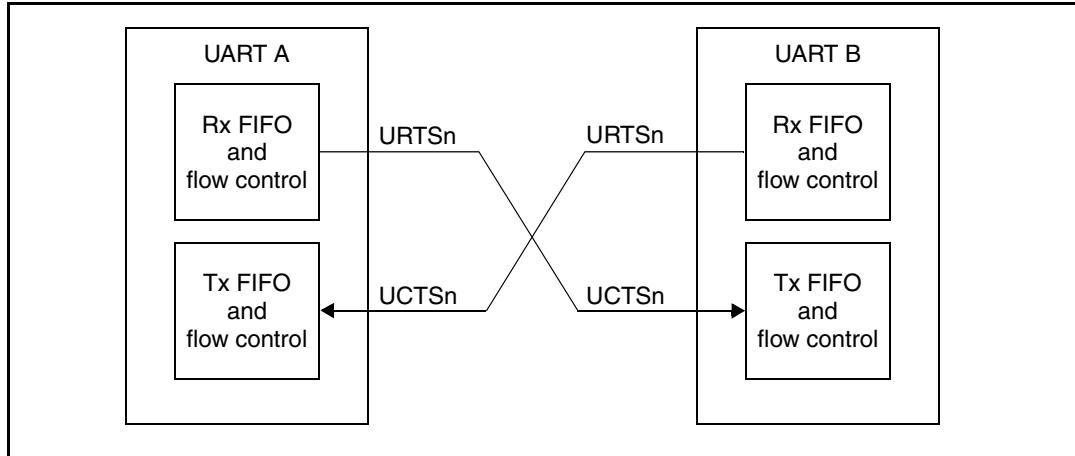
Loopback testing can be performed for UART data by configuring loopback mode. In loopback mode, the UART receive pin (URXD[0:2]) is internally connected through a multiplexer to the UART transmit pin (UTXD[0:2]).

19.3.6 Hardware flow control

Hardware and software flow control cannot be enabled simultaneously.

The serial data flow can be controlled using the URTSn output signals, and the UCTS_n input signals. See [Figure 56](#). RTS and CTS can be enabled simultaneously and independently.

Figure 56. Hardware flow control between two similar devices



Request to send (RTS) flow control

When RTS flow control is enabled, the URTSn signal is asserted until the receive FIFO is filled to the watermark level. Then the URTSn signal is de-asserted, and no more data is expected after the current character has been transmitted. When data has been read from the receive FIFO, the URTSn signal is reasserted, so that the FIFO is filled to the watermark again.

If RTS flow control is disabled, and the UART is still enabled, data is received until the receive FIFO is full or no more data is transmitted to it.

Clear to send (CTS) flow control

When CTS flow control is enabled, the transmitter checks the UCTS_n signal before transmitting each byte. If the signal is asserted, it transmits the byte, otherwise it does not.

The data continues to be transmitted while the UCTS_n signal is asserted and the transmit FIFO is not empty. If the transmit FIFO is empty and the UCTS_n signal is asserted, no data is transmitted.

If CTS flow control is enabled, and the UCTS_n signal is de-asserted, transmission of the current character is completed before stopping.

If CTS flow control is disabled, and the UART is enabled, data continues to be transmitted until the transmit FIFO is empty.

19.3.7 Software flow control

Hardware and software flow control cannot be enabled simultaneously.

Software receive flow control

If software flow control is enabled, and data is received:

- the receiver compares the incoming data with the programmed Xoff value(s).
- if the received characters match the Xoff value(s), transmission stops as soon as the current character is transmitted.
- the Xoff interrupt is set. If the corresponding mask is set, the UART interrupt pin is asserted.
- the receiver monitors incoming characters for a match with the programmed Xon value(s). Once a match is found, the Xoff interrupt is disabled. With appropriate programming, any incoming character can be accepted as a valid Xon condition.
- transmission then resumes. The received character is written into the receive FIFO.

The received Xon/Xoff characters are not written into the received FIFO, except when special character detection is enabled.

The specified status (parity, framing and break error) Xon/Xoff characters are accepted as valid matches, even if they are not valid status characters.

Software transmit flow control

If software flow control is enabled, and data is to be transmitted:

- when the receive FIFO passes its trigger level, the transmitter inserts an Xoff character.
- the UART sets a stop flag to signal that remote transfer is stopped.
- when the receive FIFO falls below the trigger level, an Xon character is automatically inserted in the transmission stream, and the UART clears the stop flag.

If software flow control is turned off after an Xoff character has been transmitted, an Xon character is automatically inserted in the transmission stream, and the UART clears the stop flag.

Xon/Xoff characters are transmitted using the standard protocol, as programmed.

Special character detection

When special character detection is enabled:

- software flow control is turned off.
- the receiver compares received characters with the Xoff2 value.
- when a match is found, the Xoff interrupt is set. If the corresponding mask is set, the UART interrupt pin is asserted. Transmission is not stopped.
- the special character is written into the receive FIFO.
- the Xoff interrupt must be cleared by the program.

The specified status (parity, framing and break error) Xon/Xoff characters are accepted as valid matches, even if they are not valid status characters.

19.3.8 Automatic format extractor (autobaud)

The autobaud feature extracts the UART settings from an input sequence. The result can be used to configure the UART automatically. For automatic configuration, autobaud must be enabled while the UART receive line is idle.

To detect the UART settings, autobaud looks for the two-character sequence *AT*, *at*, *A*/ or *a*/ appearing on the receive ports. The sequence can specify both format (character size, parity configuration and number of stop bits) and baud rate; or just baud rate. Incorrect character sequences are rejected.

The UART is disabled during autobaud detection and the *AT* sequence is not stored.

An invalid initialization sequence can result in an incorrect measurement of baud rate, which in turn leads to the sequence being incorrectly decoded. The probability of an invalid sequence being detected as a valid sequence is small. It is possible that, after a missed sequence, if the next sequence is sent too soon, it may be judged as invalid even if it is valid. For example, a character of value 0x0 with even parity and 8-bit character length yields a temporary baud period measurement of 10 times the correct period and therefore, the actual character ends long before the receive logic finishes sampling the received input. This sequence is rejected, but until the receive logic finishes sampling at the wrong rate, any new data is read incorrectly.

Baud period detection

The baud period is measured by detecting the width of the first logic zero pulse that is seen after the process starts. This is used to decode the sequence that follows. If the sequence is rejected, a new measurement is made and the process re-starts, but the expected number of bits are sampled first.

For autobaud to complete successfully, only the character bits are required to be received, and not the extra bits that determine the format. Therefore, a sequence can give a valid baud rate measurement while giving ambiguous format information.

The supported baud rates for the autobaud operation are:

- 19 200,
- 38 400,
- 57 600,
- 76 800,
- 115 200.

The measured baud rate must be between $23*16$ and $281*16$ UARTCLK cycles. Any measurements out of this range results in an error. Within this range, six intervals are defined for which a divisor factor is associated (an integer and fractional part). [Table 94](#) shows all divisors in decimal. The values shown are the defaults, and can be redefined as required.

Table 94. Interval definition and divisor value

Constant name	Number of UARTCLK cycles	Equivalent baud rate	Supported baud rate	Divisor factor	
				Integer	Fractional
RANGE_A	234*16	12820			
	to	to	19 200	9	49
RANGE_B	117*16	25641			
	to	to	38 400	4	57
RANGE_C	65*16	46153			
	to	to	57 600	3	17
RANGE_D	45*16	66666			
	to	to	76 800	2	29
RANGE_E	32*16	93750			
	to	to	115 200	1	41
AUTOBAUDUR	23*16	130 435			

The nominal values are calculated using the formula given in [Section 19.3.4 on page 238](#). When using the double baud rate feature, the supported baud rates are effectively twice the above. In other words, the divisor factors remain the same whether 16 or 8 is used as the clock divisor.

These values are read as soon as the start bit duration has been measured, which improves the chances of correctly identifying the configuration sequence.

Autobaud can be configured to keep searching for a valid sequence, or to terminate if either the baud rate or the received characters are invalid. Flags are set for an invalid baud rate or invalid character sequence. These can be used to generate interrupts, if required.

Measurements are made in UARTCLK cycles, but the factor values returned are formatted correctly to be used as the integer and fractional parts for generating Baud16 for the UART.

AUTOBAUDUR is used to set the maximum baud period. Any data-low pulse that does not last longer than AUTOBAUDUR is ignored, rather than flagging an error.

Unless autobaud is set to automatically restart, the sequence completes following the detection of a data-low pulse lasting longer than AUTOBAUDUR. Otherwise, it continues until a valid configuration sequence is detected. At this point, an UARTABDONE interrupt is generated.

Format detection

When a valid baud measurement has been made, two characters-worth of data is sampled and analyzed.

- The logic value in the 8th to 12th bit positions (bits [7:11]) after the first start bit, and the 8th to 10th bit positions (bits [7:9]) after the second start bit, are stored and used for calculating the data format.
- If the string *A/* (or *a/*) is received, or the *AT* (or *at*) command is received with more than two stop bits, an interrupt is generated (if interrupts are enabled), but no format changes are made.
- If the string *AT* (or *at*) is received, and the parity and stop bits fit into a recognizable category, an interrupt can be generated, and the format made available.
- If the logic detects a bad character sequence or baud error, an interrupt can be generated. The process can be configured to start again or terminate.

19.3.9 DMA interface

The UART provides an interface to connect to the DMA controller. The DMA interface includes the following signals:

Transmit DMA signals

- UARTRXDMASREQ: single character DMA transfer request, asserted by the UART. This signal is asserted when there is at least one empty location in the transmit FIFO (when enabled or disabled, as this acts like a one-word deep FIFO).
- UARTRXDMABREQ: burst DMA transfer request, asserted by the UART. This signal is asserted when the transmit FIFO contains less characters than the programmed watermark level.
- UARTRXDMACLR: DMA request clear, asserted by the DMA controller to clear the transmit request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

Receive DMA signals

- UARTRXDMASREQ: single character DMA transfer request, asserted by the UART. This signal is asserted when the receive FIFO contains at least one character.
- UARTRXDMABREQ: burst DMA transfer request, asserted by the UART. This signal is asserted when the receive FIFO is enabled and contains more characters than the programmed watermark level.
- UARTRXDMACLR: DMA request clear, asserted by the DMA controller to clear the receive DMA request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

The burst transfer and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there is more data than the watermark level in the receive FIFO, the burst transfer request and the single transfer request are asserted. When the amount of data left in the receive FIFO is less than the watermark level, the single request only is asserted.

This is useful for situations where the number of words left to be received in the stream is less than a burst. For example, if 19 characters have to be received and the watermark level is programmed to be four, the DMA controller then transfers four bursts of four characters and three single transfers to complete the stream.

Note: For the remaining three characters, the UART cannot assert the burst request.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is de-asserted, a request signal can become active again, depending on the conditions described above. All request signals are de-asserted if the UART is disabled or DMA transfers are disabled.

When the UART is in the FIFO disabled mode, only the DMA single transfer mode can operate, since only one character can be transferred to or from the FIFOs at any one time. UARTRXDMASREQ and UARTRXDMABREQ are the only request signals that can be asserted. When the UART is in the FIFO enabled mode, data transfers can be made either as single or burst transfers, depending on the programmed watermark level and the amount of data in the FIFO. [Table 95](#) shows the trigger points for UARTRXDMABREQ and UARTRXDMASREQ depending on the watermark level (for both the transmit and receive FIFOs).

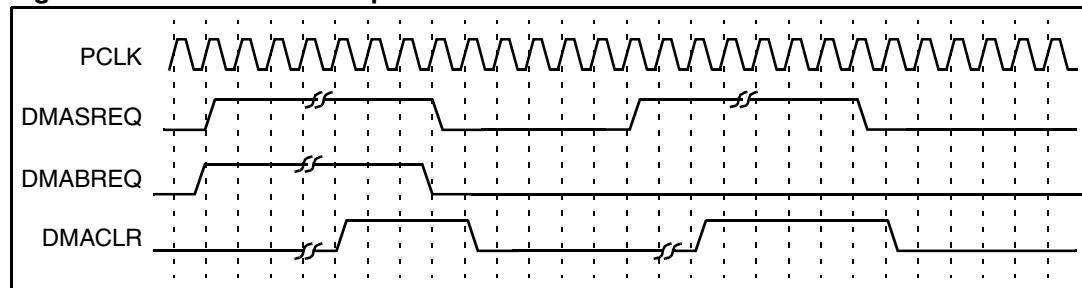
Table 95. UART DMA trigger points for the transmit and receive FIFOs

Watermark level	Burst length	
	Transmit (number of empty locations)	Receive (number of filled locations)
1/64	1	1
1/32	2	2
1/16	4	4
1/8	8	8
1/4	16	16
1/2	32	32
3/4	48	48

In addition, if the receive error interrupt, UARTEINTR, is asserted, then the DMA receive request outputs, UARTRXDMASREQ and UARTRXDMABREQ, are masked out. They remain inactive until UARTEINTR is cleared. The DMA transmit request outputs are unaffected.

[Figure 57](#) shows the timing diagram for a single transfer request and a burst transfer request, with the appropriate DMA clear signal. The signals are all synchronous with the PCLK clock. For clarity, the figure assumes that there is no synchronization of the request signals in the DMA controller.

Figure 57. DMA transfer request waveforms



19.4 UART signals

19.4.1 Interface signals

UART0 supports data terminal equipment (DTE) and data communication equipment (DCE) modes of operation. The naming convention for the UART modem signals assumes the UART is in DTE mode. In DCE mode, some signal functions are swapped.

UART[1:2] are identical to UART0 except that they provide fewer signals.

Table 96. UART pin summary

Signal	Type	Description
UCTS0n	I	UART0 clear to send modem status (active low)
UCTS1n	I	UART1 clear to send modem status (active low)
UCTS2n	I	UART2 Clear To send modem status (active low)
UDCD0n	I	UART0 data carrier detect modem status (active low)
UDSR0n	I	UART0 data set ready modem status (active low)
UDTR0n	O	UART0 data terminal ready modem status (active low)
URI0n	I	UART0 ring Indicator modem status (active low)
URTS0n	O	UART0 request To send modem status (active low)
URTS1n	O	UART1 request To send modem status (active low)
URTS2n	O	UART2 request To send modem status (active low)
URXD0	I	UART0 received serial data
URXD1	I	UART1 received serial data
URXD2	I	UART2 received serial data
UTXD0	O	UART0 transmitted serial data
UTXD1	O	UART1 transmitted serial data
UTXD2	O	UART2 transmitted serial data

19.4.2 Interrupt signals

There are eight interrupt sources generated by the UART. These eight sources are combined into a single interrupt signal, **UART[0:2]INTR**, which is the only interrupt signal that is passed to the vectored interrupt controller (VIC). This combined interrupt is asserted if any of the individual interrupts are enabled and asserted.

Xoff/special character interrupt

The Xoff/special character interrupt is asserted when an Xoff condition or special character is detected.

Modem status interrupt (**UART[0:2]MSINTR**)

The modem status interrupt is asserted if any of the modem status lines (UCTSn, UDCCDn, UDSRn and URInot) change. The interrupt can be cleared by software.

Receive interrupt (UART[0:2]RXINTR)

The receive interrupt is asserted high when one of the following conditions occurs.

- The FIFOs are enabled and the number of characters received reaches the programmed watermark level. The interrupt can be cleared by reading data from the receive FIFO until it falls below the watermark level, or by software.
- The FIFOs are disabled (have a depth of one location) and there is data present in the single receive location. The interrupt can be cleared by performing a single read, or by software.

Transmit interrupt (UART[0:2]TXINTR)

The transmit interrupt is asserted high when one of the following conditions occurs.

- The FIFOs are enabled and the number of characters in the transmit FIFO is less than the programmed watermark level. The interrupt can be cleared by performing writes to the transmit FIFO until the number of characters it holds rises above the watermark level, or by software.
- The FIFOs are disabled (have a depth of one location) and there is no data present in the single transmit location. The interrupt can be cleared by performing a single write to the transmit FIFO, or by software.

Note:

The transmit interrupt is based on a transition through a level, rather than on the level itself. When the interrupt and the UART are enabled before any data is written to the transmit FIFO, the interrupt is not set. The interrupt is only set once written data leaves the single location of the transmit FIFO and it becomes empty.

Receive timeout interrupt (UART[0:2]TINTR)

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. This mechanism ensures that the user is aware that data is still present in the receive FIFO and requires servicing. The interrupt can be cleared when the FIFO becomes empty through reading all the data, or by software.

Receive error interrupt (UART[0:2]EINTR)

The receive error interrupt is asserted when a receive error occurs. The interrupt can be caused by an error condition in framing, parity, break or overrun. The interrupt can be cleared by software.

Autobaud done interrupt (UART[0:2]ABDONEINTR)

The autobaud done interrupt is asserted if either:

- The autobaud process has completed successfully.
- Autobaud is configured to make only one attempt at detection, and the attempt failed.

The interrupt can be cleared by software.

Autobaud error interrupt (UART[0:2]ABERRINTR)

The autobaud error interrupt is asserted when the autobaud process has decoded an invalid command sequence, or when the measured baud rate is out of range. The interrupt can be cleared by software.

20 Scroll key and keypad encoder (SKE)

A scroll key consists of a cylindrical rotor with metal strips and a pair of sensors that deliver two binary signals. When the scroll key is rotated, the signals change in quadrature (one at a time).

Scroll keys are also called rotary encoders, jog-dials, or thumbwheels.

20.1 Features

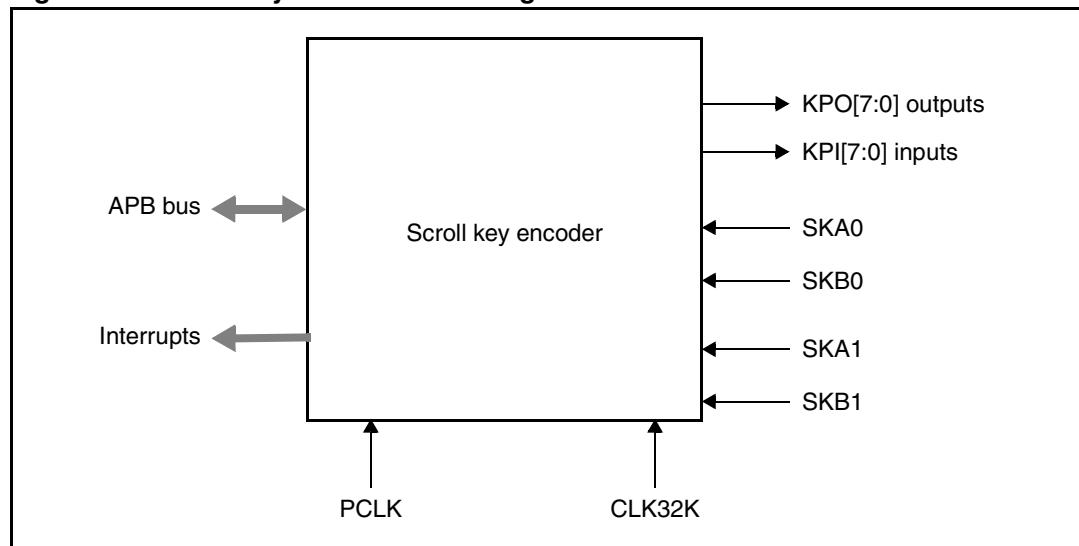
- supports one or two scroll keys
- supports one keypad of up to 64 keys, with manual or automatic scanning
- communicates via the 32-bit width AMBA peripheral bus (APB)
- uses the Vectored Interrupt Controller (VIC)

20.2 Overview

The SKE tracks the rotation of scroll keys by incrementing or decrementing an 8-bit counter, which has overflow and underflow flags.

For keypads, the SKE detects key presses and provides key data.

Figure 58. Scroll key encoder block diagram



For scroll keys, the SKE:

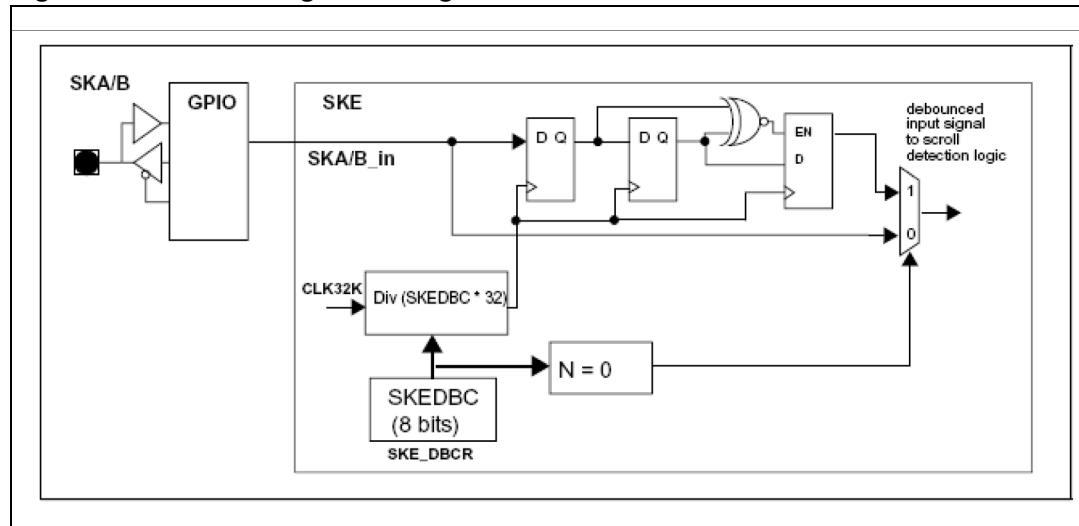
- debounces signals, using a debounce interval programmable from 1 to 255 ms,
- interprets the debounced signals to determine the scroll direction,
- on each internal CLK32K rising edge, increments, decrements or leaves the count value unchanged.

20.2.1 Scroll key debounce interval

The signals from the scroll key can take several microseconds to become stable. To avoid faulty scroll detection, the signals can be filtered for a period known as the debounce interval. Each signal must remain stable during the debounce interval.

The SKE is clocked by the 32.768 kHz clock (CLK32K). This clock is divided by 32 and then by the programmed debounce interval. This gives a debounce interval from about 1 ms to about 250 ms (in steps of $32/32\ 768 = 0.976625$ ms). If the debounce interval is set to 0, no debouncing occurs. Due to the filtering sequence, the active input level must be present for at least 3 debounce periods to be detected.

Figure 59. Debouncing block diagram



20.2.2 Scroll key operation

The scroll key signals are delivered by four general purpose input/output (GPIO) pins in alternate function mode.

Each scroll key has two signals, called SKA and SKB. The two scroll keys are called 0 and 1. So there are four signals in all: SKA0, SKA1; SKB0 and SKB1. In this document, SKAx means signal A on both scroll keys, and SKBx means signal B on both scroll keys.

[Table 97](#) shows the effect of changes in the scroll key output.

Table 97. Count value according to scroll key output

SKAx output	SKBx output	Effect on SKE count
Low to high	Low	Decremented (-1)
High to low	Low	Incremented (+1)
Low to high	High	Incremented (+1)
High to low	High	Decremented (-1)
Low	Low to high	Incremented (+1)
Low	High to low	Decremented (-1)
High	Low to high	Decremented (-1)

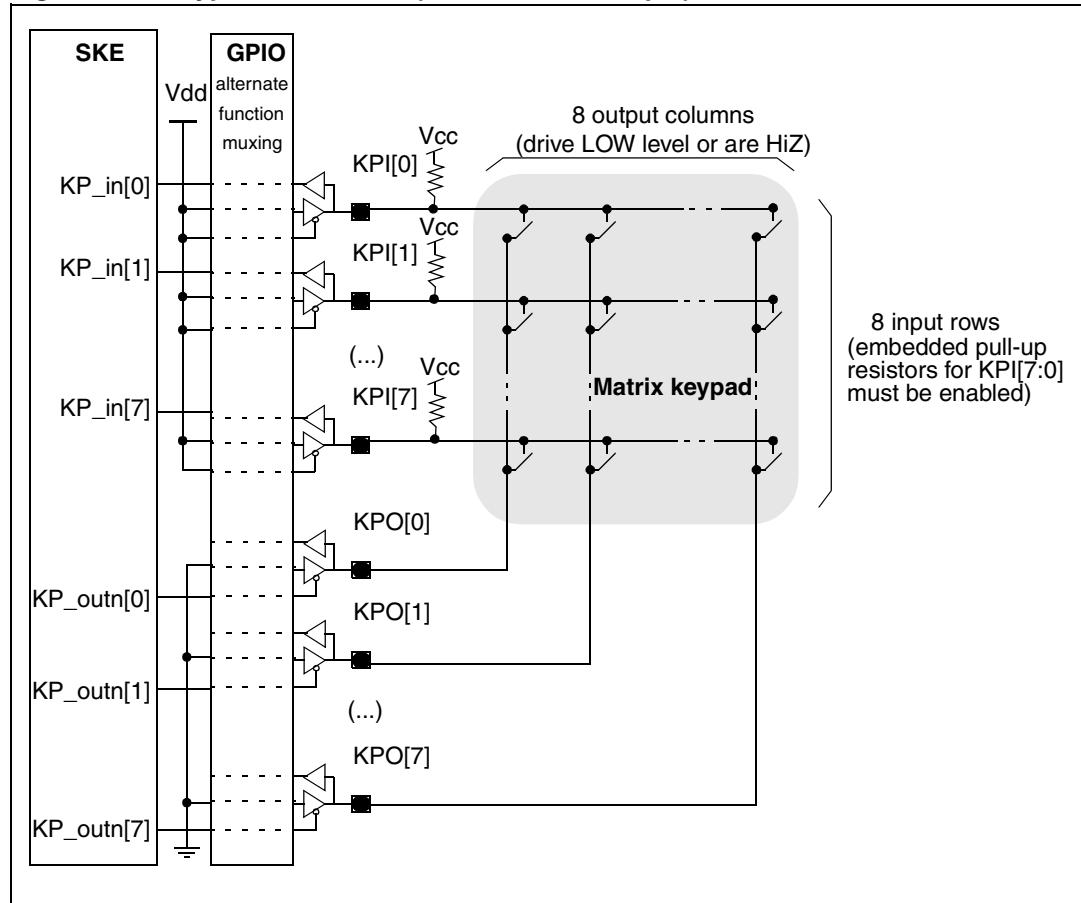
Table 97. Count value according to scroll key output (continued)

SKAx output	SKBx output	Effect on SKE count
High	High to low	Incremented (+1)
All other states		No effect

20.3 Keypad encoder

Figure 60 shows a typical connection for an 8x8 keypad. The number of rows or columns can take any value from one to eight, so all N[1:8] by M[1:8] keypads are supported.

A key pressed for at least the debouncing period triggers an interrupt.

Figure 60. Keypad connection (8 x 8 matrix example)

20.3.1 Keypad debounce interval

A key press is detected when a low level on one or more of KPI[7:0] remains stable during the debouncing period. Debouncing is done in a way similar to the scroll-key encoder. See [Section 20.2.1 on page 250](#).

20.3.2 Keypad manual scan

Manual scanning can be done in software, by disabling autoscan.

The scanning software must program KPO[7:0] to drive a low level, so that the keypad interrupt is raised when a key is pressed. The keypress is detected by a low to high transition of the ORed KPI[7:0] inputs.

The keypad interrupt must be serviced by putting all KPO outputs but one in HiZ as follows:

1. Read the levels on the KPI inputs.
2. Invert the levels. The non-pressed keys in the column appear as a row high, because of the pull-up resistors associated with the pin.
3. Wait one debounce interval.
4. Read the KPI levels again.
5. If the KPI levels from the two reads are equal, then the keys have debounced; if they are not equal, go to step 3.
6. Scan the matrix by setting the correct sequence of levels on the column output pins (KPO[7:0]), and reading the levels on the row input pins (KPI[7:0]).

To scan a matrix of less than 8 columns or rows, just limit the scanning sequence to the required number of KPO pins, and mask the GPIO_DAT reads for the associated KPI pins.

Table 98 shows a typical keypad scanning sequence.

Table 98. Keypad software scanning sequence

Column pin	Idle state	Scanning sequence								Idle state
		1	2	3	4	5	6	7	8	
KPO[0]	Low	Low	HiZ	Low						
KPO[1]	Low	HiZ	Low	HiZ	HiZ	HiZ	HiZ	HiZ	HiZ	Low
KPO[2]	Low	HiZ	HiZ	Low	HiZ	HiZ	HiZ	HiZ	HiZ	Low
KPO[3]	Low	HiZ	HiZ	HiZ	Low	HiZ	HiZ	HiZ	HiZ	Low
KPO[4]	Low	HiZ	HiZ	HiZ	HiZ	Low	HiZ	HiZ	HiZ	Low
KPO[5]	Low	HiZ	HiZ	HiZ	HiZ	HiZ	Low	HiZ	HiZ	Low
KPO[6]	Low	HiZ	HiZ	HiZ	HiZ	HiZ	HiZ	Low	HiZ	Low
KPO[7]	Low	HiZ	HiZ	HiZ	HiZ	HiZ	HiZ	HiZ	Low	Low

20.3.3 Keypad automatic scan

When automatic scan is enabled, the following process is executed automatically:

1. Wait until a key press is detected (until a debounced keypress signal becomes high).
2. Set a flag to indicate that scanning is in progress.
3. Drive low one column at a time, to perform the same scanning sequence described for software scanning mode. For each step, store the row readings. Each step lasts one 32.768 kHz clock period (about 30 µs). The number of steps (equivalent to the number of columns), is programmable from 1 to 8. If a row signal is not used, then disabling the alternate function KPI pin for that row forces a low level on the SKE input.
4. Wait one debouncing interval.
5. Repeat step 3.
6. If the keys have debounced, and if it is new keypress, then raise the KPINTR interrupt.
7. Clear the flag set in step 2. The keypress data is now available for reading.
8. If multiple keypress detection is disabled, wait until all pressed keys are released (that is, until a debounced keypress signal becomes low); or wait one debouncing period. These alternatives are selected in software.
9. Restart the scan process.

20.4 SKE signals

20.4.1 Interface signals

The scroll-key encoder signals are described in [Table 99](#).

Table 99. SKE signals

Signal	I/O	Description
KPO[7:0]	O	Keypad column outputs (HiZ or driven low by SKE).
KPI[7:0]	I	Keypad column inputs.
SKA0	I	Scroll key 0 quadrature digital signals.
SKB0	I	
SKA1	I	Scroll key 1 quadrature digital signals.
SKB1	I	

20.4.2 Interrupts

The SKE generates one interrupt for each scroll key:

- Scroll key 0 generates SKEINTR0.
- Scroll key 1 generates SKEINTR1.

These two interrupts are set when the corresponding scroll key count value is incremented or decremented. They are then ORed to generate the combined interrupt signal SKEINTR. SKEINTR is then output to the vectored interrupt controller (VIC).

The SKE generates one interrupt for a keypad, KPINTR. This is raised when a key is pressed and output to the vectored interrupt controller (VIC).

21 USB On-The-Go interface (USB-OTG)

This chapter details the main USB-OTG interface of the STn8815. The implementation is compliant with the USB 2.0 standard for high-speed and full-speed functions and the On-The-Go supplement to the USB 2.0 specification. Up to 16 bidirectional endpoints are supported.

21.1 Introduction

The OTG controller is a single-core design providing:

- The function controller of a high/full-speed USB peripheral.
- A dual-role USB controller for point-to-point OTG communications with another USB function (which can be either high-speed, full-speed or low-speed).
- The host controller for a multi-point USB system connected to a hub. The device in which the OTG controller is used can switch between roles as required.
- Split-transaction support (full-or low-speed devices can be used with a USB 2.0 hub).
- 16 IN and 16 OUT unidirectional endpoints can be configured as 16 bidirectional fully flexible endpoints (control endpoint 0).

Each endpoint can handle any type of transaction (bulk, isochronous, interrupt, control). The endpoints can also be allocated ‘on the fly’ to different target device functions. This maximizes the number of devices that can be supported simultaneously. Every active endpoint must be associated with a FIFO of non-zero size.

The OTG controller RAM interface connects to a single synchronous single-port RAM block that provides all the endpoint FIFOs. The FIFO for endpoint 0 must be 64 bytes deep to buffer one maximum-sized packet. The RAM interface is configurable with regard to the other endpoint FIFOs, which may range from 8 to 8 192 bytes and buffer 1 or 2 packets. Separate FIFOs can be associated with each endpoint. Alternatively a Tx and an Rx endpoint with the same endpoint number can be configured to use the same FIFO. This can, for example, reduce the RAM-block size needed, (note that the endpoints cannot be active simultaneously). A 32-bit synchronous AHB bus interface, running at a wide range of bus speeds is included with support for multi-layer AHB-bus operations. DMA access to endpoint FIFOs is possible.

From the USB physical interface side, the OTG controller provides a UTMI+ Level 3-compatible interface (with ULPI wrapper) for connection to an on-board OTG (USB) high/full-speed transceiver. The 8-bit (60 MHz) ULPI interface is designed to connect to an OTG high-speed transceiver, the full-speed transceiver bridge is included to connect to legacy OTG full-speed transceivers (in full-speed mode only).

The OTG controller provides all the encoding, decoding and checking needed to send and receive USB packets. The CPU is only interrupted when endpoint data has been successfully transferred. When acting as the host, the OTG controller maintains a frame counter and automatically schedules SOF, isochronous, interrupt and bulk transfers. Session-request, and host-negotiation protocols are fully supported for use in point-to-point communications. Several test modes are provided (primarily the four test modes for high-speed operation described in the USB 2.0 specification). Options that force full-speed, high-speed or host mode are available. The latter mode is useful when debugging PHY problems in hardware.

21.2 USB-OTG features

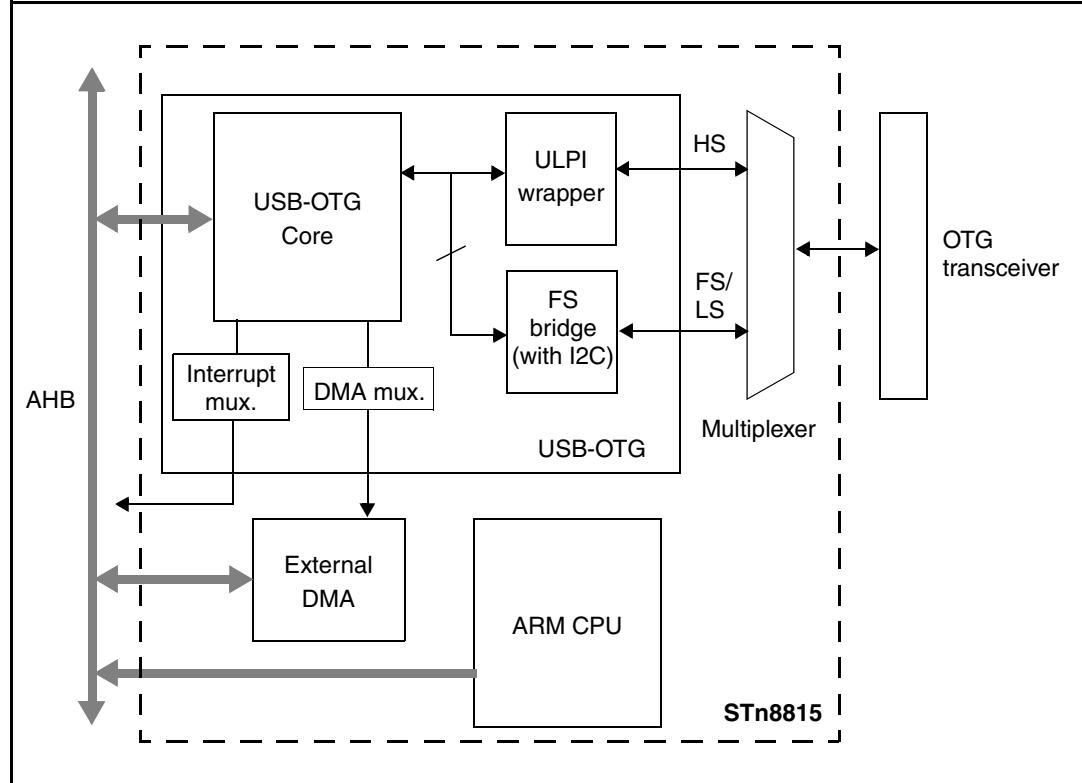
The OTG controller features are summarized below:

- Operates either as the function controller of a high (and full)- speed USB peripheral, or the host/peripheral in point-to-point or multi-point communications with other USB devices.
- Complies with the USB 2.0 standard for high-speed (480 Mbit/s) functions and with the On-The-Go supplement to the USB 2.0 specification.
- Supports OTG communications with one or more high, full or low-speed device
- Session request protocol (SRP) and host negotiation protocol (HNP)
- Suspend and resume signaling
- UTMI+ Level 3 transceiver interface support with 12-pin ULPI link wrapper.
- ULPI DDR (double data rate) not supported, only ULPI SDR supported.
- Full-speed USB 1.1 PHY Interface fully compatible with OTG transceiver specification v0.92a (for full/ low-speed operation only).
- Soft connect/disconnect.
- Sixteen IN and 16 OUT endpoints
- Dynamic allocation of endpoints, to maximize number of devices.
- Dynamic FIFO sizing
- DMA access to FIFOs
- High-level AMBA AHB-compatible CPU interface (works with a wide range of AHB bus speeds)
- Supports multi-layer operations on the AHB bus
- Performs all transaction scheduling in hardware

21.3 Architecture

The OTG controller connects the STn8815A12 to the outside world over the USB (OTG) bus. This section describes the connections of the OTG controller to other system components and the outside world. A block diagram of the OTG controller connections is shown in *Figure 61*.

Figure 61. OTG subsystems



The following system elements are connected to the OTG controller:

- CPU,
- DMA controller,
- OTG transceiver (outside STn8815A12),
- system memory (not shown).

21.3.1 CPU and DMA

Application software running on the ARM CPU controls all functional aspects of the OTG controller. The application software performs the initial and runtime configuration and power management operations of the OTG controller (suspend/resume operations and clock control). It also services interrupts and manages transfer of data packets to and from the OTG controller. Data packets to be sent and incoming packets from the USB are stored in system memory. The software handles the data transfer either by accessing the OTG FIFO directly, or by programming the system DMA controller. All packet processing operations are executed automatically by the OTG controller, once packets are in the FIFO. The CPU then deals only with transfers to the FIFO. However, packet scheduling can be done in software if required by the application.

21.3.2 OTG transceiver

The OTG transceiver converts data streams to and from an OTG controller into USB electrical signals. It is an analog component, usually located on the application board. All control of the transceiver is done automatically by the OTG controller. However, in rare cases (for example suspend), a CPU interrupt may occur.

Various OTG transceivers may use different interfaces, high-speed transceivers use ULPI 12- or 20-pin interfaces while full-speed devices connect via an FS OTG transceiver interface compatible with OTG transceiver specification v0.92. To maintain compatibility with different transceivers, the STn8815A12 has two transceiver interfaces, a 12 or 8-bit ULPI for high-speed operation and a FS OTG transceiver interface for full-speed devices. Only one transceiver interface can be activated at a time. Activation of the correct interface is handled by software. Note that the choice of interface depends solely on the transceiver on the board. The transceiver interface should not be switched just to change the speed of communication (that is, high-speed transceivers still use the same ULPI interface when working in full-speed mode).

21.4 Functional description

21.4.1 OTG controller modes

The OTG controller has two principal modes of operation:

- Peripheral mode. The OTG controller encodes, decodes, checks and directs all USB packets sent and received. IN transactions are handled through the device Tx FIFOs, OUT transactions are handled through the Rx FIFOs. control, bulk, isochronous and interrupt transactions are supported.
- Host mode. The OTG controller core behavior depends on whether it is linked up for point-to-point communications with another USB device, or whether it is attached to a hub. When attached to another USB device, the OTG controller offers the range of capabilities needed to host point-to-point communications with this USB device.

When attached to a hub, the OTG controller provides the facilities required to host a number of devices simultaneously. When operating in host mode for point-to-point communications with a single other USB device (which can be high, full or low-speed), the OTG controller supports control, bulk, isochronous or interrupt transactions.

IN transactions are handled through the Rx FIFOs, and OUT transactions are handled through the Tx FIFOs.

As well as encoding, decoding and checking the USB packets sent and received, the OTG controller automatically schedules isochronous endpoints and interrupt endpoints to perform one transaction every n frames or micro-frames, or up to three transactions if the high-bandwidth option is selected. n represents the polling interval programmed for the endpoint.

The remaining bus bandwidth is shared equally between the control and bulk endpoints. When attached to a hub, the OTG controller continues to offer the above facilities but it further needs to be programmed with:

- The function address of the target device.
- The operating speed of the target device (so that the appropriate speed conversion can be carried out).

If the target device is a full or low-speed and is accessed through a high-speed hub, the endpoint must also be programmed with the function address and port number of the hub.

To maximize the number of devices that can be supported simultaneously in this mode, the OTG controller allows endpoints to be dynamically reprogrammed on a per-transaction or per-transfer basis. This means, for example, that the endpoint 0 controller to be used to access endpoint 0 of any connected device. Another possibility is to set up a Tx and Rx endpoint pair to handle the control protocol required to support a device's endpoint 0. The device might be required to power the VBus to 5 V as the 'A' device of the connection (source of power and default host), or as the 'B' device (default peripheral) in order to wake the 'A' device by charging VBus to 2 V. Outputs from the OTG controller indicate when these charging options are required.

Mode initialization

Whether the OTG controller initially operates in host mode or in peripheral mode depends on whether it is being used in an 'A' device or a 'B' device, which in turn depends on whether the IDDIG input is low or high. When operating as an A device, the OTG controller is initially configured to operate in host mode. When operating as a 'B' device, it is initially configured to operate in peripheral mode. However, the CPU can request that the 'B' device becomes the host the next time there is no activity on the USB.

The procedures for session request and for transferring host/peripheral roles between the devices at either end of the connection are described in [Section 21.8: OTG session request \(SRP\) on page 269](#), and [Section 21.8.3: Host negotiation \(HNP\) on page 271](#). The transfers that are made are all subject to the standard USB data transfer protocols.

21.4.2 Clocks and resets

The OTG controller uses the system and transceiver clocks.

- System clock:
This 48 MHz clock is taken from the AHB bus interface. This avoids any synchronization logic between the OTG controller and the AHB allowing single cycle access to internal data and FIFO.
- Transceiver clock: This 60 MHz clock is provided by the OTG transceiver for USB data synchronization. Its parameters are described in the UTMI+ and ULPI specifications.

21.4.3 OTG transceiver interface

The OTG transceiver interface includes high-speed and full-speed blocks. To connect with the high-speed transceiver the 8-bit, 12-pin ULPI interface is used. To connect to the full-speed transceivers, the OTG Full-speed PHY interface (compliant with OTG Transceiver Specification v0.92) is used.

For configuration details for these interfaces, refer to [Section 21.5](#).

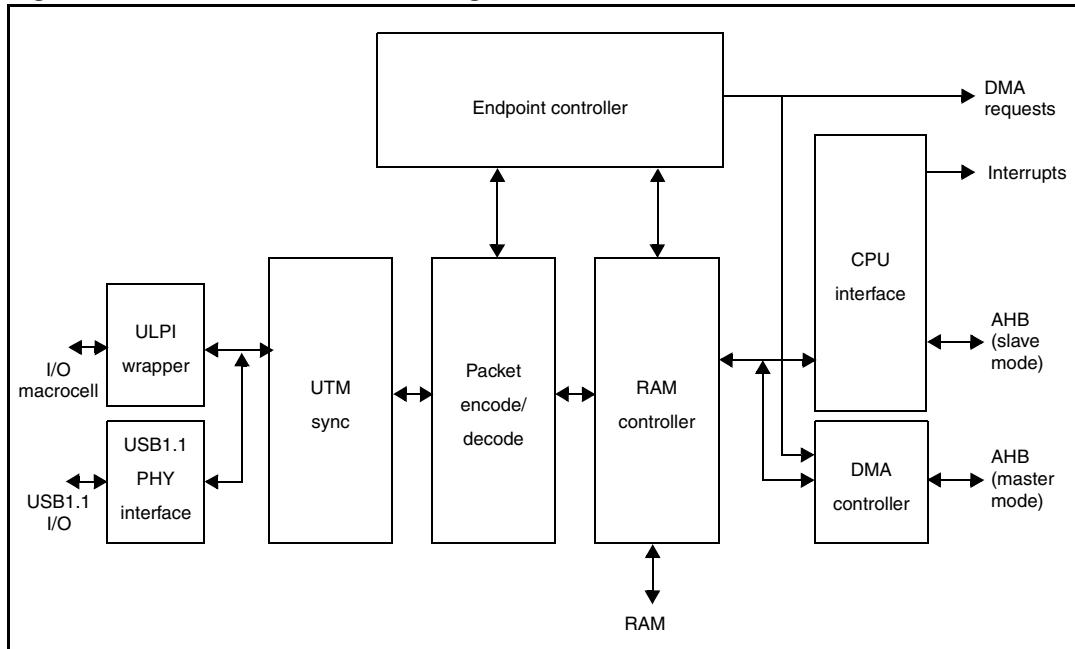
21.4.4 AHB bridge

The AHB bridge translates the AHB transactions from the high-speed system bus to the low-speed USB AHB. This translation reduces the overall power consumption. The bridge requires no programming.

21.4.5 OTG controller system

The block diagram of OTG controller is shown [Figure 62](#). The main blocks shown are UTM synchronization block, endpoint control block, CPU Interface, Packet Encode/ Decode block and RAM Controller.

Figure 62. OTG controller block diagram



21.4.6 UTM synchronization block

The role of the UTM Synchronization block is to re-synchronize between the transceiver macrocell 60 MHz clock domain and the dual role controller's system clock CLK, which drives the remainder of the OTG controller up to and including the CPU interface. This allows the rest of the OTG controller to run at the CPU bus speed without requiring any further synchronization. This block also performs high-speed detection handshaking and handles HNP and SRP in point-to-point communications with another USB OTG device. Since the 8-bit (60 MHz transceiver clock) ULPI interface is used, the block first converts the data to 16-bit allowing the OTG controller to be driven by a system clock running at a little over 30 MHz.

21.4.7 Packet encoding and decoding

The packet encode/decode block generates headers for packets to be transmitted and decodes the headers on received packets. It also generates the CRC for packets to be transmitted and checks the CRC on received packets.

21.4.8 Endpoint controller

Two controller state machines are used: one for control transfers over Endpoint 0, and one for bulk/Interrupt/Isochronous transactions over Endpoints 1 to 15.

21.4.9 CPU Interface

The CPU interface allows access to the control / status values and the FIFOs for each endpoint. It also generates interrupts to the CPU when packets are successfully transmitted or received, and when the OTG controller enters Suspend mode or resumes from Suspend mode. The interface provided by the OTG controller is a 32-bit synchronous interface that follows the design specified for interfaces to an AMBA AHB bus. Interface to other bus standards may be achieved through the addition of an appropriate wrapper to the OTG controller.

21.4.10 RAM controller

The RAM controller provides an interface to a block of synchronous single-port RAM, which is used to buffer packets between the CPU and USB. It takes the FIFO pointers from the endpoint controllers, converts them to address pointers within the RAM block and generates RAM access control signals.

21.4.11 Bit/byte ordering block

Internally, the OTG controller is little-endian. the OTG controller can, however, be configured to reverse the byte-ordering for data transferred over its CPU interface for operation within big-endian systems.

21.5 OTG transceiver interfaces

This section provides details on the functionality of the OTG transceiver interface. The ULPI wrapper translates UTMI signals into ULPI interface (used by most HS OTG transceivers), the I2C bridge allows connection to legacy FS OTG transceivers (for system view refer to [Figure 61](#)). The transceiver interfaces connect to outside world via GPIO block alternate functions.

They share the same pins therefore only one transceiver interface can be activated at a time. Different transceivers may be installed on the board but in this case all the multiplexing logic should be located on board as well. Switching on-the-fly between high- and Full-speed interfaces is **not** supported as this would require resetting the OTG controller and repeating all the configuration procedures. The selection of the interface depends **only** on the transceiver being used, high-speed transceivers are capable of working in full- (and low-)speed mode via the same ULPI interface, therefore no switching is necessary.

21.6 Operation in peripheral mode

This section describes how the OTG controller handles USB operations when working in peripheral mode.

21.6.1 USB reset

When a reset condition is detected on the USB, the OTG controller performs the following actions:

- sets OTG_FAddr to 0,
- sets OTG_INDX to 0,
- flushes all endpoint FIFOs,
- clears all control and status values,
- enables all interrupts, except Suspend,
- generates a reset interrupt

21.6.2 Peripheral mode IN transactions

When the OTG controller operates in peripheral mode, data for IN transactions is handled through the Tx FIFOs. The sizes of the Tx FIFOs for endpoints 1 to 15 are determined by configuration constants in the core configuration file or, where dynamic FIFO sizing is selected, through the internal controller settings. The maximum size of data packet that may be placed in any Tx-endpoint FIFO for transmission is programmable for each endpoint.

maximum payload = number of transactions/micro frame

Except where dynamic FIFO sizing is used, double packet buffering is enabled when the maximum packet size is set to less than, or equal to, half the FIFO size for IN transactions. Single packet buffering is enabled when the maximum packet size is greater than half the FIFO size. When dynamic FIFO sizing is selected, the use of single or double packet buffering is part of the specification for the endpoint).

When double packet buffering is enabled, two data packets can be buffered in the FIFO. When single packet buffering is enabled, only one packet can be buffered even if the packet is less than half the FIFO size.

Note: *The maximum packet size set for any endpoint must not exceed the FIFO size. You should also note that the Tx maximum packet size value should not be written if there is data in the FIFO, as unexpected results may occur.*

Single packet buffering

If the size of the Tx endpoint FIFO is less than twice the maximum packet size for this endpoint (as programmed in the OTG controller), only one packet can be buffered in the FIFO and

single-packet buffering is enabled. As each packet to be sent is loaded into the Tx FIFO, the controller *Tx ready* flag must be set. If auto-setting is enabled in the controller, the ready flag is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the ready bit must be set through the CPU.

When the ready bit is set, either by the CPU or automatically, the *FIFO not empty* flag in the OTG controller is also set and the packet is ready to be sent. When the packet has been sent successfully, both the *Tx packet ready* and *FIFO not empty* flags are cleared and the

appropriate Tx-endpoint interrupt is generated (if enabled). The next packet can then be loaded into the FIFO.

Double packet buffering

If the size of the Tx-endpoint FIFO is at least twice the maximum packet size for this endpoint (as programmed internally in the OTG controller), two packets can be buffered in the FIFO and double packet buffering is enabled. As each packet to be sent is loaded into the Tx FIFO, the controller *Tx packet ready* flag must be set. If the *auto set* controller bit is set, the *Tx packet ready* flag is set automatically when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the *Tx packet ready* flag must always be set through the CPU.

When the *Tx packet ready* flag is set, either through the CPU or automatically, the FIFO *not empty* flag is also set. The *Tx packet ready* flag is then immediately cleared, and an interrupt is generated (if enabled). A second packet can now be loaded into the Tx FIFO and *Tx packet ready* set again (automatically if the packet is of maximum size). Both packets are then ready to be sent.

When the first packet has been successfully sent, the *Tx packet ready* flag is cleared and the appropriate Tx endpoint interrupt generated (if enabled) to signal that another packet can now be loaded into the Tx FIFO. The state of the FIFO *not empty* flag at this point indicates how many packets may be loaded. If it is set, there is another packet in the FIFO and only one more packet can be loaded. If it is clear then there are no packets in the FIFO and two more packets can be loaded.

High-bandwidth isochronous/interrupt endpoints

In high-speed mode, isochronous Tx endpoints can transmit up to three USB packets in any micro frame, with a payload of up to 1 024 bytes in each packet, corresponding to a data transfer rate of up to 3 072 bytes per micro frame. The OTG controller supports this by allowing the user to load data packets of up to 3 072 bytes ($3 * 1024$ bytes) into the associated FIFO in a single transaction. From the software and CPU point of view, the operation is then exactly as described for single- or double-packet buffering except that the *Tx packet ready* flag must always be set through the CPU as auto-set does not operate with high-bandwidth isochronous transfers.

Any data packet loaded into the FIFO that is larger than the maximum payload is automatically split into USB packets of the maximum payload, or smaller, for transmission over the USB. The OTG controller can be programmed with the number of USB packets transmitted per micro frame and the maximum payload in each packet. Together, these set the maximum size of packet that can be loaded into the FIFO.

At least one USB packet is always sent, the number of further USB packets sent in the same micro frame depending on the amount of data loaded into the FIFO. The *Tx packet ready* flag is cleared and an interrupt generated when all the packets have been sent. Each USB packet is sent in response to an IN token. If, at the end of a micro frame, the controller has not received enough IN tokens to send all the USB packets (for example, because one of the tokens received was corrupted), the remaining data is flushed from the FIFO. The *Tx packet ready* flag is then cleared and the incomplete Tx data flag set to indicate that not all of the data loaded into the FIFO was sent.

Optional special handling

The packets transferred in bulk operations are defined by the USB specification as either 8, 16, 32, 64 or 512 bytes in size (the 512 byte option only applies to high-speed transfers). For some system designs, it might be more convenient for the application software to write larger amounts of data to an endpoint in a single operation than is normally transferred in a single USB operation. A typical case is where the same endpoint is used for high-speed transfers of 512 bytes under some circumstances, but for full-speed transfer in all other cases. When operating at full speed, the maximum amount of data transferred in a single operation is 64 bytes. To cater for such circumstances, the OTG controller includes a configuration option that allows larger data packets to be written to bulk Tx endpoints, which are then split into packets of an appropriate (specified) size for transfer across the USB. (There is a similar option for reading from bulk Rx endpoints in larger volumes than individual USB packets.)

With this option, the Tx maximum packet size setting for the endpoint is increased to 16 bits, with 11-bit payload and 5-bit multiplier definitions. The application software then writes data packets of size multiplier*payload to the FIFO which the controller splits into individual packets of the stated payload for transmission over the USB.

From the application software point of view, the resulting operation is no different to the transmission of a single USB packet except in the size of the packet written. This facility is offered as a configuration option rather than as a standard feature because it increases the gate count.

- Note:*
- 1 *This feature is only for use with Bulk endpoints and, in accordance with the USB specification, the payload must be either 8, 16, 32, 64 or 512 bytes with the 512 byte option only applicable for high-speed transfers.*
 - 2 *The Tx maximum payload setting in the OTG controller must match the maximum packet size setting of the standard endpoint descriptor for the endpoint.*
 - 3 *The associated FIFO must be large enough to accommodate the data packet prior to being split.*

21.6.3 Peripheral mode OUT transactions

When the OTG controller operates in peripheral mode, data for OUT transactions is handled through the controller Rx FIFOs. The sizes of the Rx FIFOs for endpoints 1 to 15 are determined either by configuration constants in the core configuration file or, where dynamic FIFO sizing is selected, through the programmed controller settings. The maximum amount of data received by an Rx endpoint in any frame or micro frame (in high-speed mode) is individually programmable for that endpoint.

maximum payload = number of transactions/micro frame (where applicable)

Except when dynamic FIFO sizing is used, when the maximum packet size is set to less than, or equal to, half the FIFO size, double packet buffering is enabled for OUT transactions. When the maximum packet size is greater than half the FIFO size, single packet buffering is enabled. (Where dynamic FIFO sizing is selected, the use of single or double packet buffering is part of the specification for the endpoint FIFO.)

When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When single packet buffering is enabled, only one packet can be buffered even if the packet is less than half the FIFO size.

- Note:* *The maximum packet size must not exceed the FIFO size.*

Single packet buffering

If the size of the Rx endpoint FIFO is less than twice the maximum packet size for the endpoint (programmed in the OTG controller), only one data packet can be buffered in the FIFO and single packet buffering is enabled. When a packet is received and placed in the Rx FIFO, the *Rx packet ready* and the *FIFO full* flags are set and the appropriate Rx endpoint is generated (if enabled) to signal that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, the *Rx packet ready* flag must be cleared before further packets can be received. If the *auto clear* flag in the controller is set and a maximum-sized packet is unloaded from the FIFO, the *Rx packet ready* and *FIFO full* flags are also cleared. For packet sizes less than the maximum, *Rx packet ready* must be cleared through the CPU.

Double packet buffering

If the size of the Rx endpoint FIFO is at least twice the maximum packet size for the endpoint (as programmed in the OTG controller), two data packets can be buffered, and double packet buffering is enabled. When the first packet to be received is loaded into the Rx FIFO, the *Rx packet ready* flag is set and the appropriate Rx endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO.

Note: *The FIFO full flag is not set at this point. It is only set if a second packet is received and loaded into the Rx FIFO.*

After the first packet has been unloaded, the *Rx packet ready* flag must be cleared before further packets can be received. If the *auto clear* flag is set and a maximum-sized packet is unloaded from the FIFO, the *Rx packet ready* flag is cleared automatically. For packet sizes less than the maximum, the *Rx packet ready* flag must be cleared through the CPU. If the *FIFO full* flag is set when the *Rx packet ready* flag is cleared, the core first clears the *FIFO full* flag, then sets the *Rx packet ready* flag again to indicate that another packet is waiting in the FIFO to be unloaded.

High-bandwidth isochronous endpoints

In high-speed mode, isochronous Rx endpoints can receive up to three USB packets in any micro frame, with a payload of up to 1024 bytes in each packet, corresponding to a data transfer rate of up to 3072 bytes per micro frame.

Note: *High-bandwidth Interrupt transactions are not supported in peripheral mode. The core supports this automatically by combining all the USB packets received during a micro frame into a single packet of up to 3072 bytes (3*1024 bytes) within the Rx FIFO.*

From the programming point of view, the operation is exactly as described above for single- or double-packet buffering except that *Rx packet ready* must always be cleared by the CPU as auto clear does not operate with high-bandwidth isochronous transfers. The maximum quantity and payload of USB packets that can be received in any micro frame can be programmed in the OTG controller. The number of USB packets sent in any micro frame depends on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated number of packets has not been received by the end of a micro frame, the *incomplete Rx* flag is set to indicate that the data in the FIFO is incomplete. An interrupt is still generated to allow the data that has been received to be read from the FIFO.

Optional special handling

The packets transferred in bulk operations are defined by the USB specification as 8, 16, 32, 64 or 512 bytes, the 512-byte option only applying to high speed transfers. For some system designs, it may be more convenient for the application software to read more data from an endpoint in a single operation than can normally be transferred in a single USB operation. A typical case is where the same endpoint is used for high-speed transfers of 512 bytes in certain cases but for full-speed transfers under other circumstances.

When operating at full speed, the maximum amount of data transferred in a single operation is 64 bytes. To cater for such circumstances, the core includes a configuration option which, if selected, causes the core to amalgamate the packets received across the USB bus into larger data packets, prior to reading by the application software. (A similar option exists for writing to bulk Tx endpoints in larger volumes than individual USB packets).

With this option, the *Rx maximum packet size* setting for the endpoint is increased to 16 bits, with 11-bit payload and 5-bit multiplier definitions. The OTG controller amalgamates the appropriate number of USB packets received into a single data packet of size multiplier *payload within the FIFO, before asserting the *Rx packet ready* flag to signal to the application software the presence of a packet to be read from the FIFO. From the application software point-of-view, the resulting operation is no different to the receipt of a single USB packet, except for the size of the packet read. This facility is offered as a configuration option rather than as a standard feature because it increases the gate count.

- Note:
- 1 *This feature is only used with bulk endpoints and, in accordance with the USB specification. The payload must be either 8, 16, 32, 64 or 512 bytes, the 512-byte option only applying to high-speed transfers.*
 - 2 *The payload recorded Rx maximum packet size must also match the maximum packet size of the standard endpoint descriptor programmed for the endpoint. The associated FIFO must also be large enough to accommodate the amalgamated data packet.*
 - 3 *The Rx packet ready flag is only set when the specified number of packets have been received, or a short USB packet is received (if a packet of less than the specified payload for the endpoint). If a protocol is used whereby the endpoint receives bulk transfers that are a multiple of the recorded payload size with no terminating short packet, the OTG controller must not be programmed to expect more packets than there are in the transfer (otherwise the software will not be interrupted at the end of the transfer).*

21.6.4 Additional actions

The OTG controller core responds automatically to certain conditions on the USB bus or actions by the host. The details are given below.

Stall issued control transfer

The core automatically issues a STALL handshake to a control transfer under the following conditions:

- The host sends more data during an OUT data phase of a control transfer than was specified in the device request during the setup phase. This condition is detected by the controller when the host sends an OUT token (instead of an IN token) after the CPU has unloaded the last OUT packet and set DataEnd.
- The host requests more data during an IN data phase of a control transfer than was specified in the device request during the setup phase. This condition is detected by the core when the host sends an IN token (instead of an OUT token) after the CPU has cleared the Tx packet ready and set DataEnd in response to the ACK issued by the host to what should have been the last packet.
- The host sends more than the maximum packet data with an OUT data token.
- The host sends the wrong PID for the OUT status phase of a control transfer.
- The host sends more than a zero length data packet for the OUT status phase.

Zero-length OUT data packets in control transfers

A zero-length OUT data packet indicates the end of a control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred (that is, after the CPU has set DataEnd). If, however, the host sends a zero-length OUT data packet before the entire length of device request has been transferred, this signals the premature end of the transfer. In this case, the controller automatically flushes any IN tokens loaded by CPU, ready for the data phase from the FIFO and set SetupEnd.

Peripheral mode suspend

When no activity has occurred on the USB for 3 ms, the OTG controller enters suspend mode. If the suspend interrupt has been enabled, an interrupt is generated at this time. When in Suspend mode, the SUSPENDM output goes low (if enabled). When resume-signalling is detected, the OTG controller exits suspend mode. If the resume interrupt is enabled, an interrupt is generated. The CPU can also force the OTG controller to exit suspend mode by setting the *resume* flag. When this flag is set, the OTG controller exits suspend mode and drives resume signaling on the bus. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end resume signalling. No resume interrupt is generated when suspend mode is exited by the CPU.

Start-of-frame

When the OTG controller is operating in peripheral mode, it receives a start-of-frame packet from the host once every millisecond in full-speed mode, or every 125 µs when in high-speed mode. When the SOF packet is received, the 11-bit frame number contained in the packet is stored, and an output pulse, lasting one CLK bit-period, is generated on SOF_PULSE. An SOF interrupt is also generated (if enabled in the OTG controller).

Once the OTG controller has started to receive SOF packets, it expects one every millisecond (or every 125 µs). If no SOF packet is received after 1.00358 ms (or 125.125 µs), it is assumed that the packet has been lost and an SOF_PULSE (with an SOF

interrupt, if enabled) is generated, though the frame number is not updated. The OTG controller continues to generate an SOF_PULSE every millisecond (or 125 µs) and resynchronizes these pulses to the received SOF packets when they are successfully received again.

21.7 Operation in host mode

When the host-mode flag is set to 1, the OTG controller operates as a host either for point-to-point communications with another USB device or, when attached to a hub, for communication with a range of devices in a multi-point set-up. High-speed, full-speed and low-speed USB functions are supported, both for point-to-point communication and for operation through a hub. (Where necessary, the OTG controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub.) control, bulk, isochronous or interrupt transactions are supported.

This section describes the OTG controller actions concerning Tx endpoints, Rx endpoints, transaction scheduling, entry to and exit from suspend mode and reset, that apply when the OTG controller is used as a host.

Host mode is automatically selected where the OTG controller is connected to a hub. The conditions under which the OTG controller operates in host mode for point-to-point operations are explained in [Section 21.8.3: Host negotiation \(HNP\) on page 271](#).

21.7.1 Host mode device set-up

Before accessing a device as a host, whether for point-to-point or multi-point communication via a hub, the function address of the device being accessed must be recorded for each Rx or Tx endpoint.

Where a full- or low-speed device is connected to the OTG controller via a high-speed USB 2.0 hub, the Tx or Rx hub address and the hub port details must also be set, so that the OTG controller can support split transactions. In addition to the speed at which the device operates (high, full or low), the Tx or Rx type for each endpoint accessed by the device must be set in the OTG controller.

For multi-point communications, these settings record the current allocation of the OTG controller endpoints to the functions associated with the attached devices. To maximize the number of devices supported, this allocation can be changed dynamically by updating the address and speed information within the OTG controller. Any changes in the allocation of endpoints to device functions must be made following the completion of any on-going transactions on the endpoints affected.

21.7.2 Host mode IN transactions

When the OTG controller is operating as a host, IN transactions are handled in a similar manner to peripheral-mode OUT transactions, except that the transaction must first be initiated by setting the request packet flag in the OTG controller. This indicates to the transaction scheduler that a transaction is active on this endpoint. The transaction scheduler then sends an IN token to the target function. When the packet is received and placed in the Rx FIFO, the *Rx packet ready* flag is set and the appropriate Rx endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, *Rx packet ready* flag should be cleared. The *auto clear* flag can be used to cause the *Rx packet ready* flag to be cleared automatically when a maximum sized packet has been unloaded from the FIFO.

There is also an *auto request* flag which causes the *request packet* flag to be set automatically when the *Rx packet ready* flag is cleared. The *auto clear* and *auto request* flags can be used with an external DMA controller to perform complete bulk transfers without CPU intervention. If the target function responds to a bulk/interrupt IN token with a NAK, the OTG controller keeps retrying the transaction until any NAK limit that has been set has been reached. If the target function responds with a STALL, the OTG controller does not retry the transaction, but interrupts the CPU by setting the *Rx stall* flag. If the target function does not respond to the IN token within the required time (or there was a CRC or bit-stuff error in the packet), the OTG controller retries the transaction. If after three attempts the target function has still not responded, the OTG controller clears the *request packet* flag and interrupt the CPU and an *Error* flag is set.

Note: *In the case of high-bandwidth Interrupt transactions, the host attempts 2 or 3 transactions during a single micro frame and generates an interrupt when all packets have been received. If any of these transactions is not ACKed by the target, no further transactions are attempted during the same micro frame.*

21.7.3 Host mode OUT transactions

When the OTG controller is operating as a host, OUT transactions are handled in a similar manner to IN transactions when the OTG controller is operating as a peripheral. The *Tx packet ready* flag must be set as each packet is loaded into the Tx FIFO. The *auto set* flag can be used to set the *Tx packet ready* flag automatically when a maximum sized packet has been loaded into the FIFO. The *auto set* flag can be used with an external DMA controller to perform complete bulk transfers without CPU intervention. If the target function responds to the OUT token with a NAK, the OTG controller keeps retrying the transaction until any NAK limit that has been set has been reached. If the target function responds with a STALL, the OTG controller does not retry the transaction but interrupts the CPU by setting the *Rx stall* flag. If the target function does not respond to the OUT token within the required time (or there was a CRC or bit-stuff error in the packet), the OTG controller retries the transaction. If after three attempts the target function has still not responded, the OTG controller flushes the FIFO and interrupts the CPU and an error flag is set.

21.7.4 Host mode transaction scheduling

When operating as a host, the OTG controller maintains a frame/micro frame counter. If the target function is a full-speed or high-speed device, the OTG controller automatically sends an SOF/uSOF packet at the start of each frame/micro frame. If the target function is a low-speed device, a K state is transmitted as a keep-alive to stop the low-speed device entering Suspend mode.

After the SOF/uSOF packet has been transmitted, the OTG controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as an Rx endpoint for which the request packet flag is set or a Tx endpoint for which the *Tx packet ready* flag is set. An active isochronous or interrupt transaction is only started if it is on the first transaction scheduler cycle of a frame/ micro frame, and if the interval counter for that endpoint has counted down to zero.

This ensures that only one interrupt/isochronous transaction occurs per endpoint per **n** frames/micro frames (or up to three if high-bandwidth support is selected), **n** being the

interval set in the OTG controller for that endpoint. An active bulk transaction starts immediately, provided that there is sufficient time left in the frame/micro frame to complete the transaction before the next SOF/uSOF packet is due. If the transaction needs to be retried (for example because a NAK was received or the target function did not respond) the transaction is not retried until the transaction scheduler has checked all the other endpoints for active transactions. This ensures that an endpoint that sends a lot of NAKs does not block other transactions on the bus. The OTG controller also allows the user to specify a time limit during which NAKs may be received from a particular target before the endpoint is timed out.

Babble

The OTG controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay, and it does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, the OTG controller assumes that the function it is connected to has malfunctioned and suspends all transactions and generates a babble interrupt.

21.7.5 Host mode reset

If the power-reset flag is set while the OTG controller is in host mode, the OTG controller generates reset signalling on the bus. The CPU must keep this flag set for 20 ms to ensure correct resetting of the target device. After the CPU has cleared the bit, the OTG controller starts its frame counter and transaction scheduler.

21.7.6 Host mode suspend

If the *suspend mode* flag is set, the OTG controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated. To exit suspend mode, the CPU must set the *resume* flag and clear the power *suspend* flag. While the *resume* flag is high, the OTG controller generates resume signaling on the bus. After 20 ms, the CPU must clear the resume bit, at which point the frame counter and transaction scheduler start. While in suspend mode, the system clock is stopped to reduce power. The SUSPENDM output also goes low, if enabled. This may be used to power-down the USB drivers. However, if remote wake-up is to be supported, power to the PHY must be maintained so that the OTG controller detects resume signalling on the bus.

21.8 OTG session request (SRP)

To conserve power, the USB OTG supplement only requires VBus to be powered up when the bus is in use, and to be turned off otherwise. VBus is always supplied by the A device on the bus. the OTG controller determines whether it is the A device or the B device by sampling the IDDIG input from the UTMI+ PHY. This signal is pulled low when an A-type plug is sensed (signifying that the OTG controller is the A device) and taken high when a B-type plug is sensed (signifying that the OTG controller is the B device).

21.8.1 SRP starting a session

When the device containing the OTG controller requires a session-start, the CPU must set the controller's *Session* flag. The OTG controller then enables ID pin sensing. This results in the *IDDIG* input being taken low if an A-type connection is detected or high if a B-type connection is detected.

The *B-Device* flag is also set to indicate whether the OTG controller has adopted the role of the A device or the B device. If the OTG controller is the A device, it enters host mode (the A device is always the default host), turns on VBus and waits for VBus to go above the VBus valid threshold, as indicated by the *VBUSVALID* input going high. (This event also causes the two *Vbus* flags to be set high.) The OTG controller then waits for a peripheral to be connected.

When a peripheral is detected, a *Connect* interrupt is generated (if enabled) and either the *FS device* or *LS device* flag in the controller is set depending on whether a high-speed/full-speed peripheral or a low-speed peripheral is detected. The CPU must then reset this peripheral. If both *FS device* and *HS enable* flags are set, the OTG controller monitors *LINESTATE* during reset to see if a high-speed chirp is received from the peripheral. If a chirp is received, the OTG controller responds with high-speed chirps and enters high-speed mode. To end the session, the CPU must clear the *Session* flag.

If the OTG controller is the B device, it requests a session using the session-request protocol defined in the USB OTG supplement, that is, it first discharges VBus. When VBus has gone below the session-end threshold (as indicated by internal signalling and the controller *VBus[1:0]* flags going low) and the line state has been *SE0* for more than 2 ms, the OTG controller pulses the data line, then pulses VBus. At the end of the session, the *Session* flag is cleared by the OTG controller (it may also be cleared by the CPU if the application software performs a software disconnect). The OTG controller then causes the PHY to disconnect the pull-up resistor. This signals to the A device to end the session.

21.8.2 SRP detecting activity

When a non-controlling device on the USB-OTG set-up needs to start a session, it either raises VBus above the session-valid threshold if it is the A device (as indicated by internal signalling and the *VBus[1:0]* flags in the controller going to 10b) or, if it is the B device, it first pulses the data line, then pulses VBus.

Depending on which of these actions occurs, the OTG controller determines whether it is the A or the B device in the current set-up and acts accordingly as follows:

If VBus is raised above the session-valid threshold, then the OTG controller is the B device, and the OTG controller sets the *Session* flag. When reset signaling is detected on the bus, a *reset* interrupt is generated (if enabled) which the CPU must interpret as the start of a session.

The OTG controller is in peripheral mode at this point as the B device is the default peripheral. At the end of the session, the A device turns off the power to VBus. When VBus drops below the session-valid threshold (as indicated by internal signalling and the controller *VBus[1:0]* flags going to 01), the OTG controller detects this and clears the *Session* flag to indicate that the session has ended.

A *Disconnect* interrupt is also generated (if enabled). If data-line/VBus pulsing is detected, the OTG controller is the A device. It generates a *Session request* interrupt (if enabled) to indicate that the B device is requesting a session. The CPU must then start a session by setting the *Session* flag.

21.8.3 Host negotiation (HNP)

When the OTG controller is the A device, it automatically enters host mode when a session starts. When the OTG controller is the B device, it automatically enters peripheral mode when a session starts.

The CPU may request that the OTG controller becomes the host by setting the *host request* flag. This bit can be set either at the same time as a session start is requested by setting the *Session* flag, or at any time after a session has started. When the OTG controller next enters suspend mode (no activity on the bus for 3 ms). Assuming the *host request* flag remains set, it enters host mode and begins host negotiation (as specified in the USB OTG supplement) by switching TERMSEL, causing the PHY to disconnect the pull-up resistor on the D- line.

This causes the A device to switch to peripheral mode and connect its own pull-up resistor. When the OTG controller detects this, it generates a *Connect* interrupt if this is enabled. It also sets the *reset* (power) flag to begin resetting the A device.

Note: *The OTG controller begins this reset sequence automatically to ensure that reset is started as required within 1 ms of the A device connecting its pull-up resistor.*

The CPU waits at least 20 ms, then clears the reset bit and enumerates the A device. When the OTG controller-based B device has finished using the bus, the CPU puts it into suspend mode by setting the *Suspend mode* flag. The A device detects this and either terminates the session or reverts to host mode. If the A device is core-based, it generates a *Disconnect* interrupt (if this is enabled).

21.8.4 DMA channels

The OTG controller supports DMA access to the FIFOs for Tx endpoints [1:15] and Rx endpoints [1:15] either by the built-in DMA controller (if implemented) or by an external DMA controller. There is a separate DMA request line for each Tx endpoint and each Rx endpoint.

If a total of N Tx endpoints and M Rx endpoints are defined (in addition to endpoint 0):

- DMA_REQ[0: N-1] are associated with Tx endpoints [1:N]
- DMA_REQ[N: N+M-1] are associated with Rx endpoints [1:M]

The DMA request lines are individually enabled through the Rx or Tx DMA request enable flags. These may be used in two possible modes:

- DMA request mode 0,
- DMA request mode 1.

The operating mode is selected through the Rx or Tx *DMA request mode* flag. For Rx endpoints operating in request mode 0, the DMA request line goes high when a data packet is available in the endpoint FIFO and goes low one CLK period after the last read request is issued. Alternately, the request line goes low when the CPU clears the Rx packet ready flag.

The DMA request modes are summarized below.

DMA request mode 0

- The appropriate endpoint interrupt is generated to prompt the loading of all packets and following the receipt of all packets.
- This mode is equally useful for bulk, interrupt or isochronous transfers.
- If the endpoint is configured for isochronous transfers, DMA request mode 0 should always be selected where DMA is used.

DMA request mode 1

This mode is most useful where large blocks of data are transferred to a bulk endpoint. The USB protocol requires such packets to be split into a series of packets of the maximum packet size for the endpoint (512 bytes for high speed, 64 bytes for full speed).

It can be used, with a suitably programmed DMA controller, to avoid the overhead of having to interrupt the processor after each individual packet: instead the processor is only interrupted after the transfer has completed. In some cases, the block of data transferred will comprise a pre-defined number of these packets, that the controlling software counts through the transfer process. In other cases, the last packet in the series may be less than the maximum packet size and the receiver may use this short packet to signal the end of the transfer. (If the total size of the transfer is an exact multiple of the maximum packet size, the transmitting software should send a null packet for the receiver to detect.). DMA transfers may be 8-bit, 16-bit, or 32-bit as required. However, all the transfers associated with one packet (with the exception of the last) must be of the same width so that the data is consistently byte-, word- or double-word-aligned. The last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

- The DMA request line only goes high when the packet received is of the maximum packet size (as set in the controller), otherwise the DMA request line stays low.
- For Tx endpoints operating in either request mode, the DMA request line goes high when the endpoint FIFO is able to accept a data packet. It goes low when Tx Max P bytes have been loaded into the FIFO. Alternatively, the request line goes low if the CPU sets the *Tx packet ready* flag. The mode selected also affects the generation of endpoint interrupts (if enabled).
- The endpoint interrupt is suppressed except following the receipt of a short packet (that is, one of less than *Rx maximum packet size* bytes).

Note:

When programming the DMA for HS USB OTG, the burst size of the DMA has to be equal to the maximum packet size of the USB endpoint and the transfer size for the DMA transfer (even for each linked list item) has to be programmed in multiples of the burst size.

21.8.5 Dynamic FIFO sizing

The OTG controller has an overall FIFO size of 12 Kbytes, areas of which may then be allocated to the different endpoints when the OTG controller is initialized.

Note:

It is strongly recommended that this feature be used only where the OTG controller is used in a device that requires different FIFO sizes in different contexts. If the FIFO sizes do not need to change, it is preferable to set these sizes through the standard configuration options.

The allocation of FIFO space to the different endpoints requires the following items to be specified for each Tx and Rx endpoint:

- the start address of the FIFO within the RAM block,
- the maximum size of packet to be supported,
- whether double-buffering is required.

Note:

The last two items define the amount of space that needs to be allocated to the FIFO

21.8.6 USB-OTG signals

Table 100. USB-OTG high speed and ULPI external PHY interface signals

Signal	Type	Description	Comment
ULPISTP	O	ULPI stop signal	
ULPIDIR	I	ULPI direction signal	
ULPINXT	I	ULPI next signal	
ULPICLK	I	ULPI clock signal	
ULPID0	IO	ULPI data bit 0	Used in both single data rate (SDR) and double data rate (DDR) modes.
ULPID1	IO	ULPI data bit 1	
ULPID2	IO	ULPI data bit 2	
ULPID3	IO	ULPI data bit 3	
ULPID4	IO	ULPI data bit 4	Used in single data rate (SDR) mode only.
ULPID5	IO	ULPI data bit 5	
ULPID6	IO	ULPI data bit 6	
ULPID7	IO	ULPI data bit 7	

Table 101. USB-OTG full speed external PHY interface signals

Signal	Type	Description
USBOEn	O	USB output enable (active low).
USBINTn	I	USB external transceiver interrupt input.
USBVP	IO	USB single-ended input/output from/to D+ receiver.
USBRCV	I	USB single-ended data from differential data receiver.
USBVM	IO	USB single-ended input/output from/to D- receiver.
USBSDL	IO	I2C clock line.
USBSDA	O	I2C data line.

21.8.7 GPIO mapping

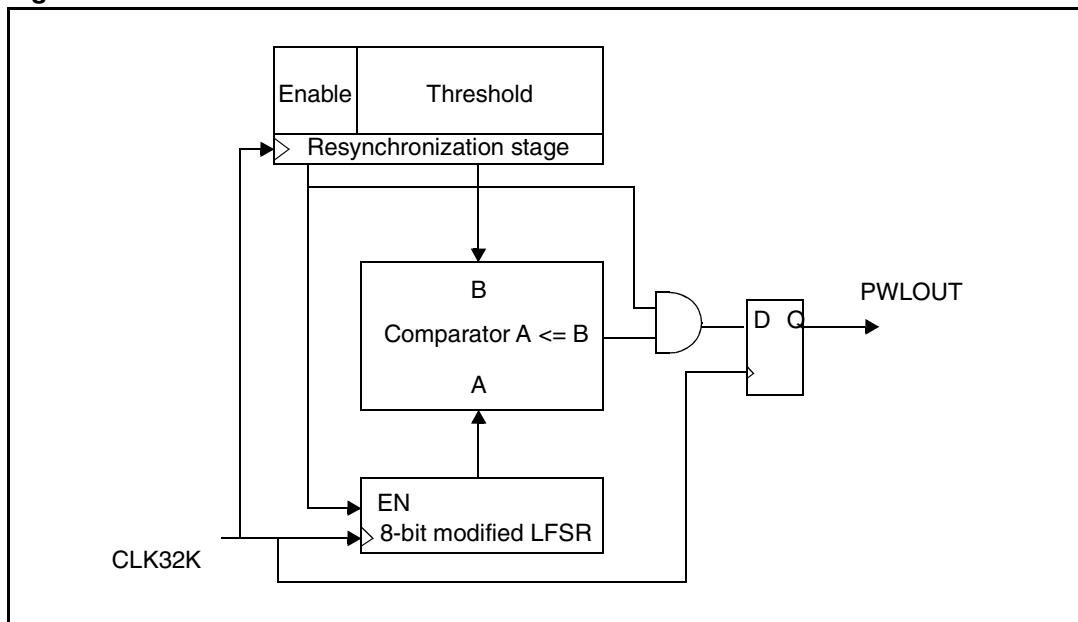
The transceiver signals are mapped onto GPIO alternate pin functions. Alternate function A should be activated for high-speed connection, and alternate function B serves the full-speed. The application software must configure all the GPIO pins to the required alternate function. If full-speed interface is used GPIO[73:78] are free to be used for other purposes. It is not necessary to switch interface when switching to full-(low-) speed mode from the high-speed one. For details of the GPIO mapping of the STn8815A12 device, refer to [Section 3.3: GPIO alternate functions on page 69](#).

22 Pulse width light modulator (PWL)

The pulse width light modulator (PWL) can be used to control the backlighting of an I/O pad or an LCD panel.

The PWL produces a series of pulses that can be fed to the backlighting. The programmed threshold value determines the perceived lighting level. The PWL module features an 8-bit random sequence data generator and a programmable threshold comparator, and uses the 32.768 kHz clock input, as shown in [Figure 63](#).

Figure 63. PWL architecture



The pseudo-random 8-bit data generator is built using a linear feedback shift register (LFSR) modified to generate all 2^8 values. It generates a white normal-law random value between 0 and 255. The original LFSR polynomial generator is:

$$P(x) = x^8 + x^4 + x^3 + x^2 + 1$$

The modified LFSR random value is presented to an 8-bit (unsigned) comparator. Here it is compared with the programmed threshold value. The comparator delivers:

- 0 if the random value is greater than the programmed threshold
- 1 if the random value is less than or equal to the programmed threshold

Assuming the random sequence is normal, this generates a sequence of 0s and 1s with a mean value proportional to the programmed threshold.

The PWL produces only one output signal, PWLOUT, which can go directly to an I/O pad or LCD display (even when the platform is in sleep mode).

Table 102. PWL signal

Signal	I/O	Description
PWLOUT	O	Pulse-width light modulator output.

23 Synchronous serial port (SSP)

The STn8815A12 platform has one synchronous serial port (SSP). This section describes the functionality and signals of the SSP. The SSP can act as a master or slave interface, allowing synchronous serial communication with slave or master peripherals.

23.1 Features

- Motorola SPI-compatible interface
- Texas Instruments synchronous serial interface
- National Semiconductor MicroWire interface
- Unidirectional interface
- Parallel-to-serial conversion of data written to an internal, 32-bit wide, 32-location deep, transmit FIFO
- Serial-to-parallel conversion of received data, which is buffered in a 32-bit wide, 32-location deep, receive FIFO
- Programmable data frame size from 4 to 32 bits
- Programmable clock bit-rate and prescaler
- Programmable clock phase and polarity in SPI mode
- Support for direct memory access (DMA) transfers

23.2 Overview

In both master and slave configurations, the SSP receives and transmits serial data, performing all the necessary conversions.

- Parallel-to-serial conversion of data written by the CPU for transmission on the SSPTXD pin. The data is written to an internal, 32-bit wide, 32-location deep, transmit FIFO.
- Serial-to-parallel conversion of received data from a peripheral device on the SSPRXD pin. The data is buffered in a 32-bit wide, 32-location deep, receive FIFO.

The transmit and receive paths are buffered with internal FIFO memories, allowing up to thirty two 32-bit values to be stored independently in both transmit and receive modes. The FIFOs can be burst-loaded, emptied by the system or serviced by a DMA transfer, with from one to eight words per transfer. Each 32-bit word from the system fills one entry in the FIFO.

The SSP includes a programmable bit-rate clock divider and prescaler to generate the serial output clock SSPCLK from the on-chip clock SSPCLKI (48 MHz for this device).

A single combined interrupt is delivered, which is derived from several internal, maskable events. In addition to this interrupt request line, a set of DMA signals are provided for interfacing with the DMA controller.

23.3 Functional description

23.3.1 Supported frame formats

Three frame formats are supported:

- Motorola's SPI
- Texas Instruments' synchronous serial protocol
- National Semiconductor's MicroWire

For the SPI and MicroWire formats, the SSPFRM pin functions as a chip-select to enable the external device (target of the transfer), and is held active-low during the data transfer.

For the Texas Instruments format, SSPFRM is pulsed high for one (serial) data period at the start of each frame.

The function and use of the serial clock SSPCLK is different for each format.

- For the MicroWire format, both data sources switch (change to the next bit) outgoing data on the falling edge of SSPCLK and sample incoming data on the rising edge.
- For the Texas Instruments format, data sources switch outgoing data on the rising edge of SSPCLK and sample incoming data on the falling edge.
- For the SPI format, the user has the choice of which edges of SSPCLK to use for switching outgoing data and for sampling incoming data. In addition, the user can alter the phase of SSPCLK, shifting it one half-period earlier or later.

The Texas Instruments format and Motorola's SPI are full-duplex protocols.

MicroWire uses a half-duplex master-slave messaging protocol. At the start of a frame, a single-byte control message is transmitted from the controller to the peripheral; no data is sent by the peripheral while the controller is transmitting this message. The peripheral interprets the message and responds with the requested data, one clock after the last bit of the requesting message. The returned data (part of the same frame) can be from 4 to 32 bits in length. The total frame length is 13 to 25 bits.

Note that the serial clock (SSPCLK), if driven by the SSP, only toggles while an active data transfer is underway. At other times, it is held in the inactive state, as defined by the specified protocol under which it operates.

Motorola's SPI frame format

The SPI frame format allows the SSP to comply with Motorola's SPI protocol.

A system conforming to this protocol has a master-slave configuration. The SPI protocol is a 4-wire interface composed of serial data in (master in, slave out or MISO), serial data out (master out, slave in or MOSI), a shift clock (SCK) and an active-low slave enable signal (SSn).

Communication between the master and the slave is determined by the presence or absence of the master clock. Data transfer is initiated by the detection of the master clock and is terminated by the absence of the master clock. The slave has to be enabled during this period of transfer.

When the SSP is the master, the slave enable is derived from the master transmit frame sync pulse, MTFS. In SPI modes, MTCK also drives its own internal receive clock MRCK via an internal multiplexer.

Figure 64 and *Figure 65* show the SSP in SPI master and slave mode, respectively.

Figure 64. SSP as an SPI master

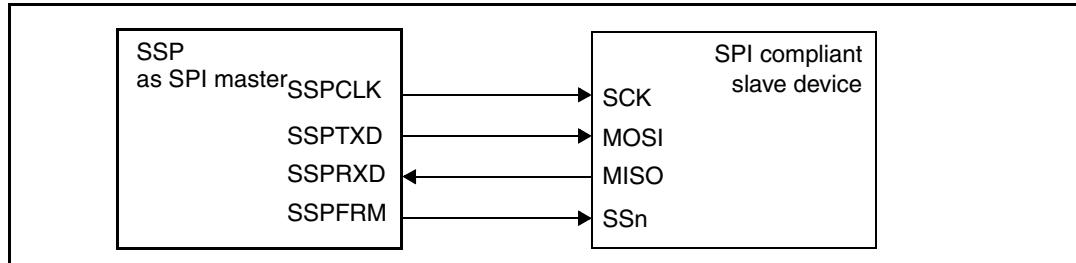
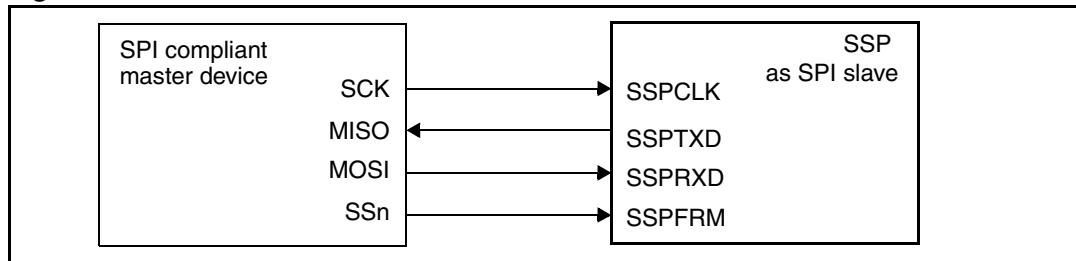


Figure 65. SSP as an SPI slave



There are four possible sub-modes, depending on the edges selected for switching driven data and for sampling received data, and the selection of the phase (see *Section 23.3.2 on page 287*). These sub-modes are numbered 0 to 3, and are described in *Table 103*, and illustrated in the figures (*Figure 66* to *Figure 73*).

Table 103. Sub-modes of Motorola's SPI frame format

Sub-mode			
0	1	2	3
During idle periods: SSPCLK is forced low when the SSP is configured as master, otherwise SSPCLK is an input (HiZ). SSPFRM is forced high. SSPTXD is in HiZ.		During idle periods: SSPCLK is forced high when the SSP is configured as master, otherwise SSPCLK is an input (HiZ). SSPFRM is forced high. SSPTXD is in HiZ.	
When the SSP is enabled and there is valid data within the transmit FIFO, SSPFRM goes low and stays low for the remainder of the frame. This causes slave data to be enabled on the SSPRXD input line, and the SSPTXD line to be driven.			

Table 103. Sub-modes of Motorola's SPI frame format (continued)

Sub-mode			
0	1	2	3
<p>One half SSPCLK period later, valid master data is transferred to the SSPTXD pin.</p> <p>Now that both the master and slave data have been set, the SSPCLK master clock pin goes high after one further half SSPCLK period and continues toggling for the remaining data bits.</p>	<p>One half SSPCLK period later, valid master data is transferred to the SSPTXD pin.</p> <p>At the same time, the SSPCLK master clock pin goes high and continues toggling for the remaining data bits.</p>	<p>One half SSPCLK period later, valid master data is transferred to the SSPTXD pin.</p> <p>Now that both the master and slave data have been set, the SSPCLK master clock pin goes low after one further half SSPCLK period and continues toggling for the remaining data bits.</p>	<p>One half SSPCLK period later, valid master data is transferred to the SSPTXD pin.</p> <p>At the same time, the SSPCLK master clock pin goes low and continues toggling for the remaining data bits.</p>
Data is captured on the rising edges and propagated on the falling edges of the SSPCLK signal. From 4 to 32 bits may be transferred per frame.	Data is captured on the falling edges and propagated on the rising edges of the SSPCLK signal. From 4 to 32 bits may be transferred per frame.	Data is captured on the rising edges and propagated on the falling edges of the SSPCLK signal. From 4 to 32 bits may be transferred per frame.	
For a single word transmission, after all bits of the data word have been transferred, SSPFRM is returned to its idle high state one SSPCLK period after the last bit has been captured.			
For continuous back-to-back transmissions, SSPFRM must be pulsed high between each data word transfer. Therefore, the master device must raise SSPFRM between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, SSPFRM is returned to its idle state one SSPCLK period after the last bit has been captured.	For continuous back-to-back transmissions, SSPFRM is held low between successive data word transfers, and termination is the same as for the single-word transfer.	For continuous back-to-back transmissions, SSPFRM must be pulsed high between each data word transfer. Therefore, the master device must raise SSPFRM between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, SSPFRM is returned to its idle state one SSPCLK period after the last bit has been captured.	For continuous back-to-back transmissions, SSPFRM is held low between successive data word transfers, and termination is the same as for the single-word transfer.
For illustrations of the above, see: Figure 66 (single transfer), Figure 67 (continuous transfer).	For illustrations of the above, see: Figure 68 (single transfer), Figure 69 (continuous transfer).	For illustrations of the above, see: Figure 70 (single transfer), Figure 71 (continuous transfer).	For illustrations of the above, see: Figure 72 (single transfer), Figure 73 (continuous transfer).

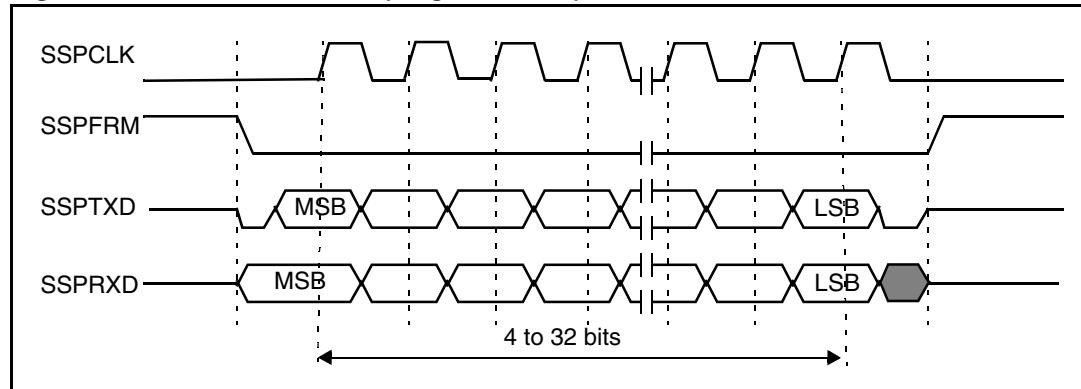
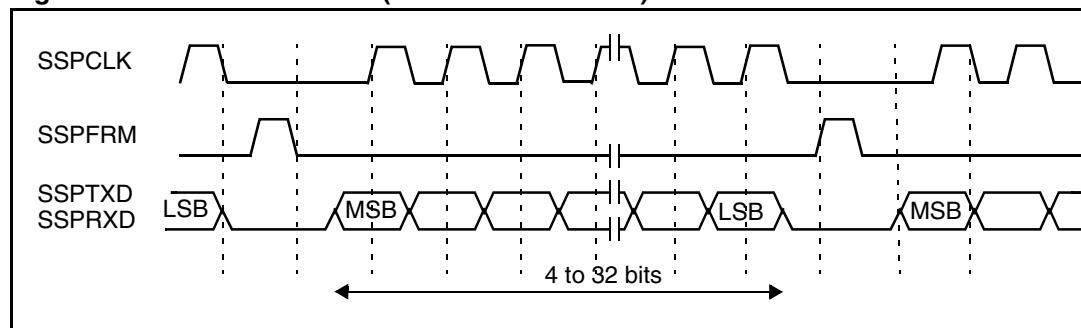
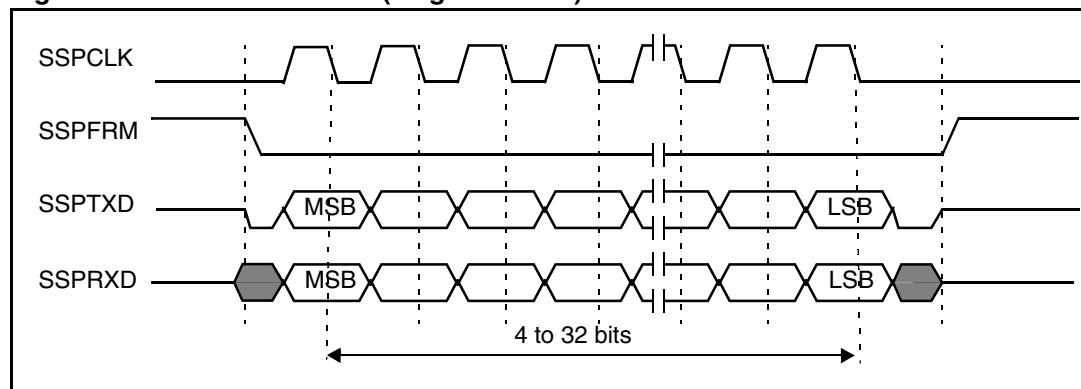
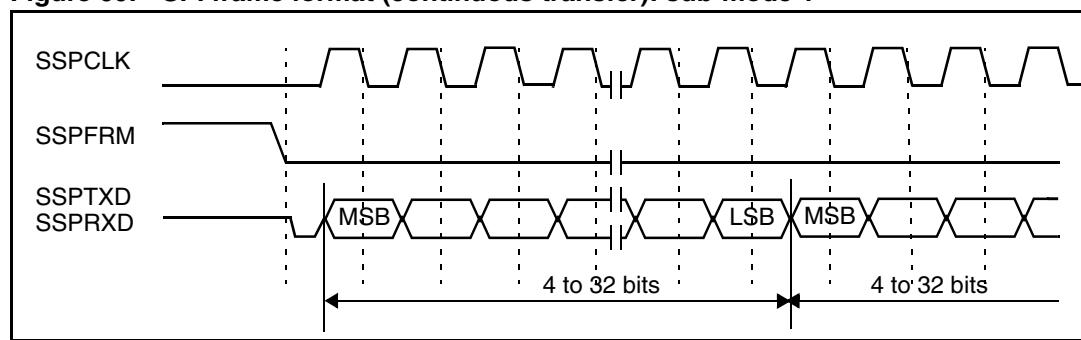
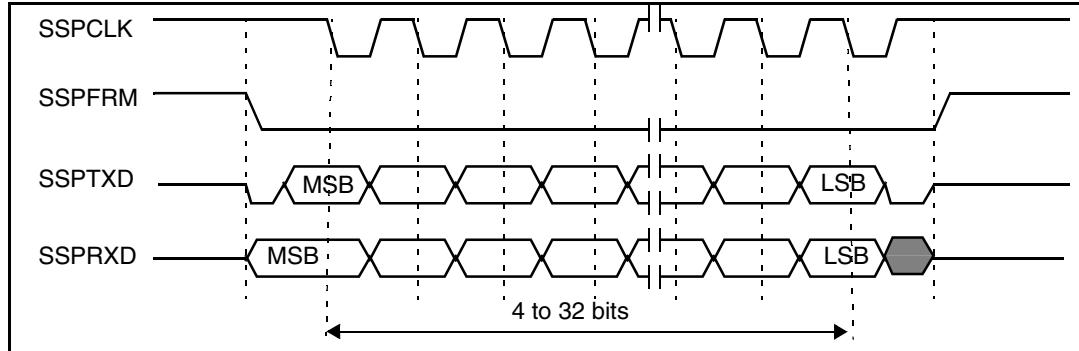
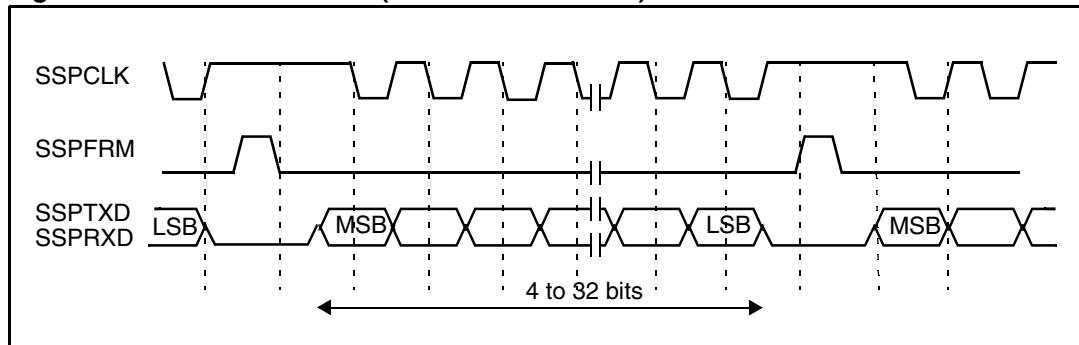
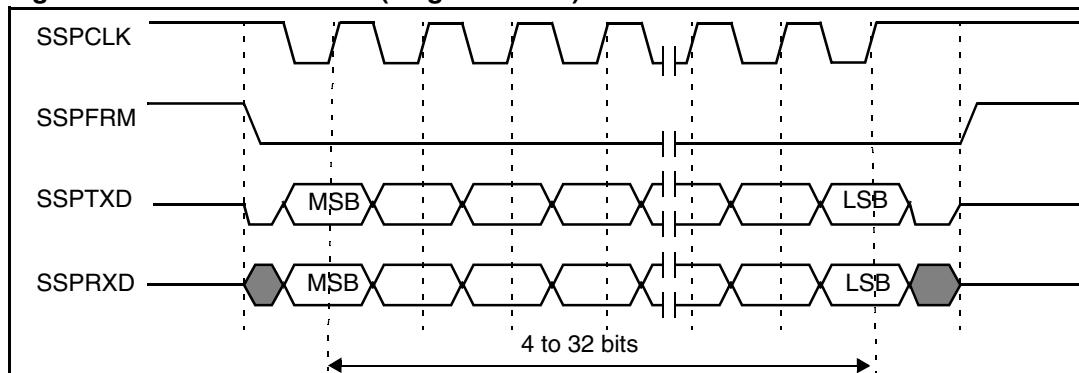
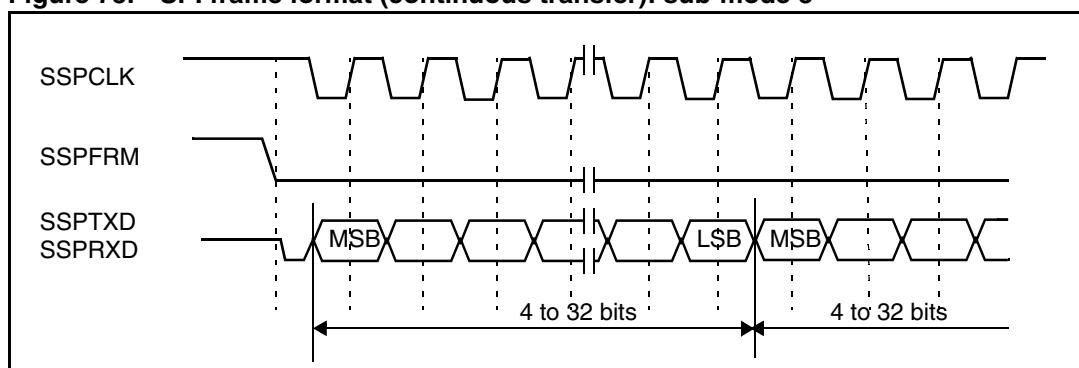
Figure 66. SPI frame format (single transfer): sub-mode 0**Figure 67.** SPI frame format (continuous transfer): sub-mode 0**Figure 68.** SPI frame format (single transfer): sub-mode 1**Figure 69.** SPI frame format (continuous transfer): sub-mode 1

Figure 70. SPI frame format (single transfer): sub-mode 2**Figure 71.** SPI frame format (continuous transfer): sub-mode 2**Figure 72.** SPI frame format (single transfer): sub-mode 3**Figure 73.** SPI frame format (continuous transfer): sub-mode 3

Texas Instruments frame format

Master and slave modes of operation for the Texas Instruments (TI) frame format are described in [Table 104](#).

Table 104. Texas Instruments' frame format, master and slave modes

Master mode	Slave mode
Whenever the SSP is idle, SSPCLK and SSPFRM are forced low, and the transmit data line SSPTXD is tri-stated.	Whenever the SSP is idle, SSPCLK and SSPFRM are inputs, and the transmit data line SSPTXD is tri-stated.
When the outgoing data is ready to be transmitted (that is, the transmit FIFO contains at least one valid data item), SSPFRM is pulsed high for one SSPCLK clock period. On the following clock, the data to be transmitted is driven on SSPTXD, one bit at a time, with the most significant bit first. Similarly, the peripheral drives data on the SSPRXD pin.	Once SSPFRM is pulsed high by the external master, on the following clock the data to be transmitted is driven on SSPTXD, one bit at a time, with the most significant bit first. Similarly, the external master drives data on the SSPRXD pin.
Word length may be from 4 to 32 bits. All transitions take place on the rising edge of SSPCLK and data sampling is done on the falling edge. At the end of the transfer, SSPTXD is tri-stated.	
<i>Figure 74</i> shows the Texas Instruments synchronous serial frame format for a single transmitted/received frame.	
<i>Figure 75</i> shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted/received.	

Figure 74. TI synchronous serial frame format (single transfer)

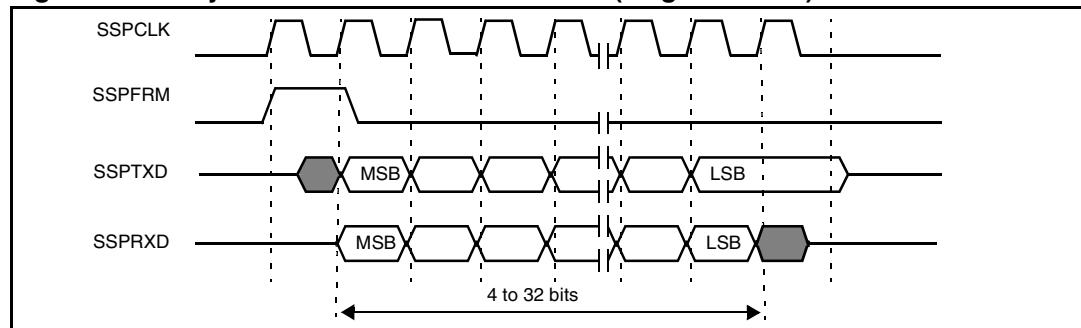
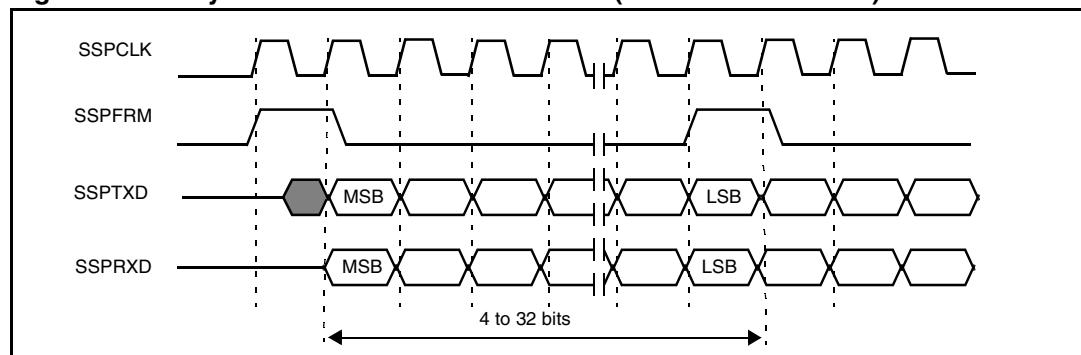


Figure 75. TI synchronous serial frame format (continuous transfer)



National Semiconductor MicroWire frame format

The MicroWire format is very similar to the SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with a 4- to 32-bit control word that is transmitted from the master SSP to the off-chip slave device. During this transmission, no incoming data is received by the master SSP. After the message has been sent, the off-chip slave decodes it and, immediately or after a delay (of one serial clock after the last bit of the control message has been sent), responds with the required data. The returned data is 4 to 32 bits in length, making the total frame length anywhere from 4 to 64 bits, with one or no dead bits.

Figure 76 and *Figure 77* show the SSP as a MicroWire master in full-duplex and half-duplex modes, respectively.

Figure 76. SSP as a MicroWire master (full-duplex)

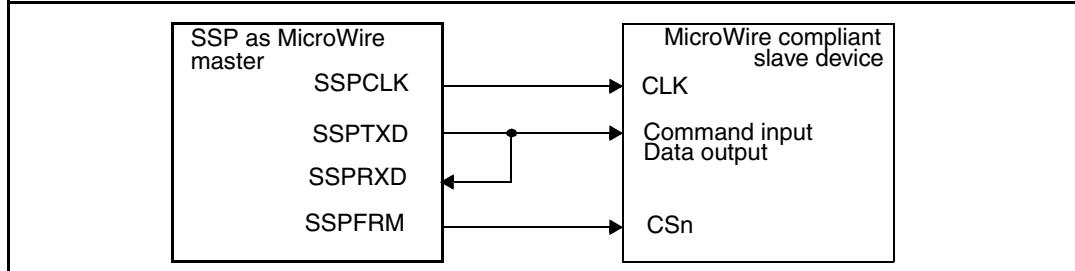


Figure 77. SSP as a MicroWire master (half-duplex)

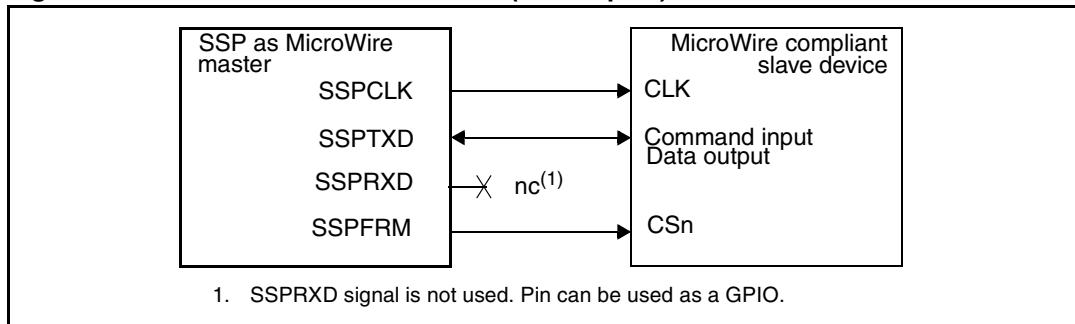


Figure 78 and *Figure 79* show the SSP as a MicroWire slave in full-duplex and half-duplex modes, respectively.

Figure 78. SSP as a MicroWire slave (full-duplex)

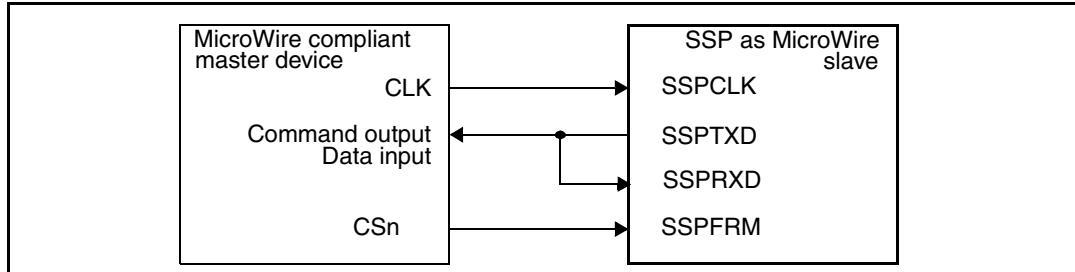
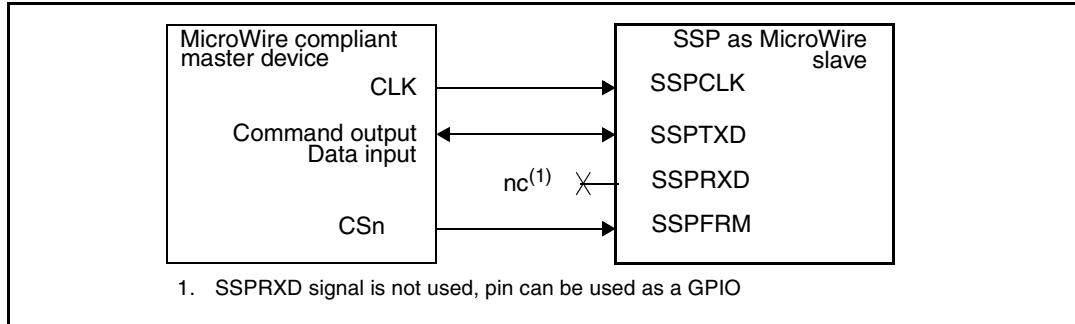
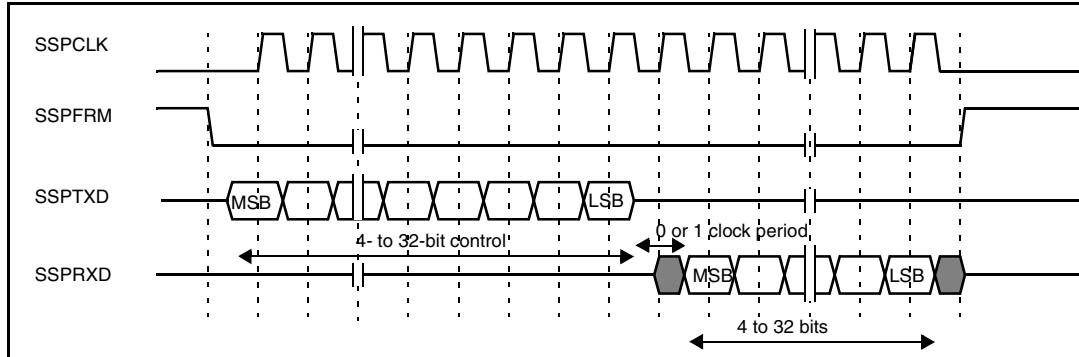
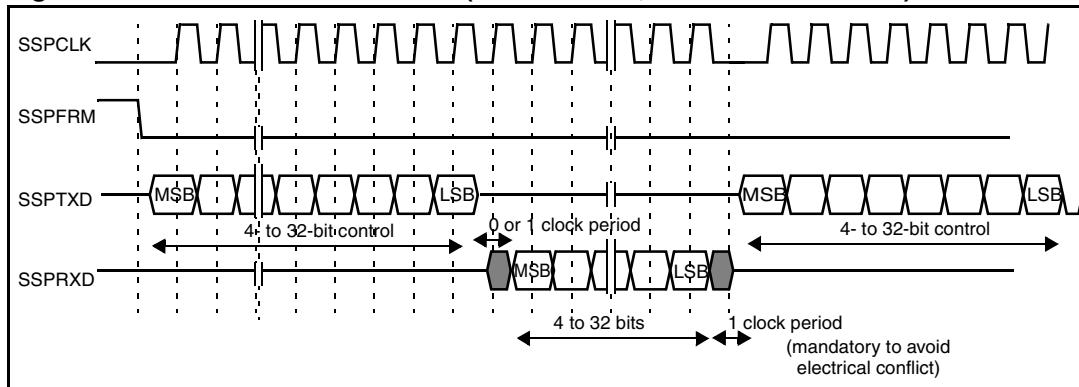


Figure 79. SSP as a MicroWire slave (half-duplex)

Master and slave modes of operation for the MicroWire frame format are described in [Table 105](#).

Table 105. MicroWire frame format, master and slave modes

Master mode	Slave mode
Whenever the SSP is idle, SSPCLK is forced low, SSPFRM is forced high and the transmit data line SSPTXD is tri-stated.	Whenever the SSP is idle, SSPCLK and SSPFRM are inputs, and the transmit data line SSPTXD is tri-stated.
When the outgoing data is ready to be transmitted, SSPFRM goes low for the duration of the frame. Then, one clock cycle later, the 4- to 32-bit command word to be transmitted is driven on SSPTXD, one bit at a time with the most significant bit first. Data transitions are on the falling edge of SSPCLK.	Once SSPFRM is asserted low by the external master, on the following clock, the 4- to 32-bit command is received on the SSPRXD pin or on the SSPTXD pin, one bit at a time with the most significant bit first. Sampling is done on the rising edge of SSPCLK.
After the last command bit has been sent, a zero or single wait-state is inserted, then the peripheral returns the serial data requested, MSB first, on the SSPRXD pin or SSPTXD pin. Sampling is done on the rising edge of SSPCLK. The last falling edge of SSPCLK coincides with the end of the last data bit (LSB) on the SSPRXD pin or SSPTXD pin, and it remains low after that (if it is the only word or last word of the transfer). SSPFRM is de-asserted high half a clock period later.	After the last command bit has been sent, a zero or single wait-state is inserted, then the SSP begins the transmission of the data on the SSPTXD pin. Word length may be from 4 to 32 bits. All transitions take place on the falling edge of SSPCLK. SSPTXD is tri-stated either on the falling edge of SSPCLK after the LSB has been transmitted, or when SSPFRM goes high (inactive).
For illustrations of the above for master mode, see: Figure 80 (single transfer), Figure 81 (continuous transfer).	

Figure 80. MicroWire frame format (master mode, single transfer)**Figure 81.** MicroWire frame format (master mode, continuous transfer)**Note:**

The off-chip slave device can tri-state the SSPRXD line either on the falling edge of SSPCLK, after the LSB has been latched by the receive shifter, or when the SSPFRM pin goes high.

Timing constraints (for example, output access time; that is, the time for the slave device to start driving the MSB on the SSPRXD line after the falling edge of the clock) can be adjusted, using bit rate generation, by defining the SSPCLK frequency.

Unidirectional frame format

This unidirectional format is very similar to the MicroWire format described previously, except that transmission is limited to the control word transmitted from the master SSP to the off-chip slave device, and is not followed by reception. Each serial transmission consists of just a 4- to 32-bit control word. During this transmission, no incoming data is received by the master SSP.

[Figure 82](#) shows the SSP as a unidirectional master (in half-duplex mode).

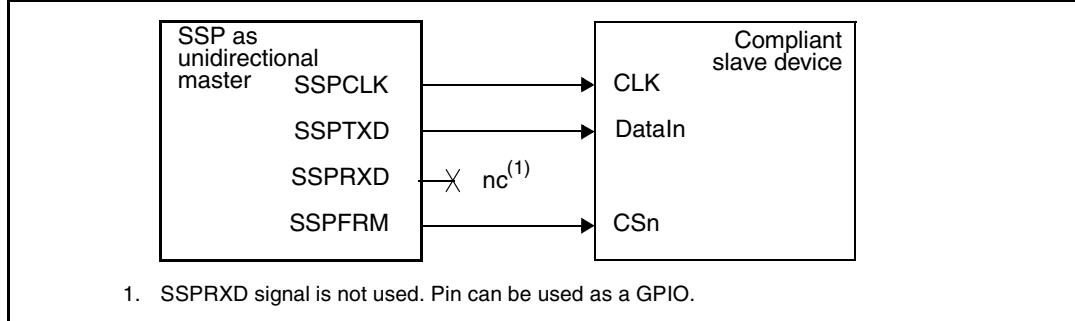
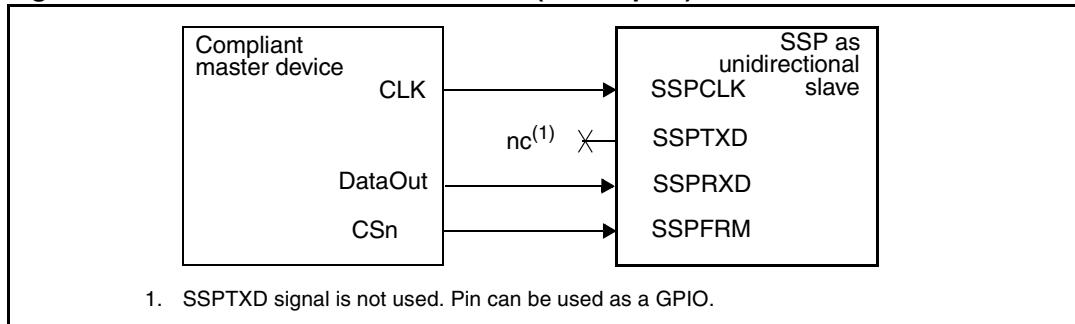
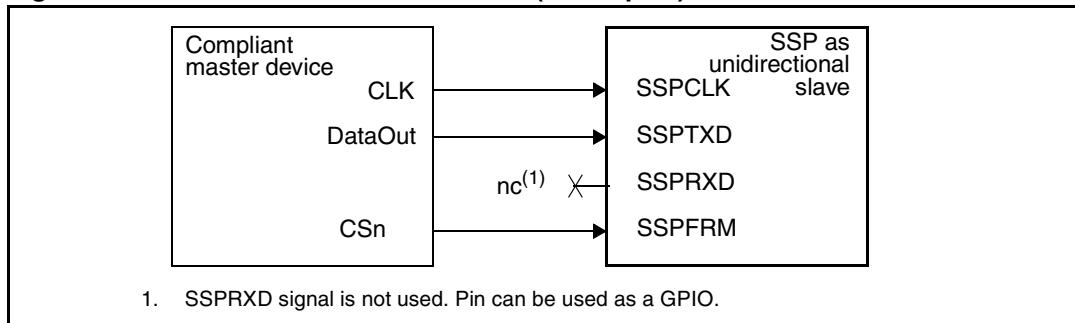
Figure 82. SSP as a unidirectional master (half-duplex)

Figure 83 and *Figure 84* show the SSP as a unidirectional slave, in full-duplex and half-duplex modes respectively.

Figure 83. SSP as a unidirectional slave (half-duplex)**Figure 84. SSP as a unidirectional slave (full-duplex)**

Master and slave modes of operation for the unidirectional frame format are described in [Table 106](#)

Table 106. Unidirectional frame format, master and slave modes

Master mode	Slave mode
Whenever the SSP is idle, SSPCLK is forced low, SSPFRM is forced high and the transmit data line SSPTXD is tri-stated. SSPRXD is not used in this mode.	SSPCLK and SSPFRM are inputs, and the transmit data line SSPTXD is not used in this mode (tri-stated).
When the outgoing data is ready to be transmitted, SSPFRM goes low for the duration of the frame. Then, one clock cycle later, the 4-to 32-bit command word to be transmitted is driven on SSPTXD, one bit at a time with the most significant bit first. Data transitions are on the falling edge of SSPCLK. SSPTXD is tri-stated either on the falling edge of SSPCLK after the LSB has been transmitted, or when SSPFRM goes high (inactive).	Once SSPFRM has been asserted low by the external master, on the following clock, the 4-to 32-bit command is received on SSPRXD or on SSPTXD, one bit at a time with the most significant bit first. Sampling is done on the rising edge of SSPCLK. After the last command bit has been sent, a zero or single wait-state is inserted, then the SSP begins the transmission of the data. Word length may be from 4 to 32 bits. All transitions take place on the falling edge of SSPCLK. SSPTXD is tri-stated either on the falling edge of SSPCLK after the LSB has been transmitted, or when SSPFRM goes high (inactive).
When the transmit FIFO is empty, SSPFRM is de-asserted high half a clock period after the last command bit is sent.	
For illustrations of the above for master mode, see: Figure 85 (single transfer), Figure 86 (continuous transfer).	

Figure 85. Unidirectional frame format (master mode, single transfer)

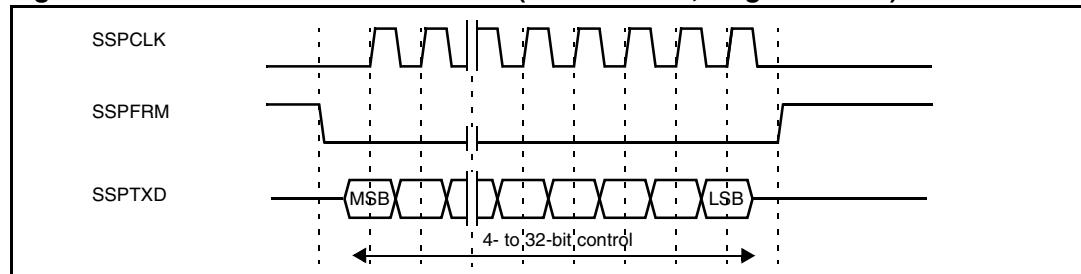
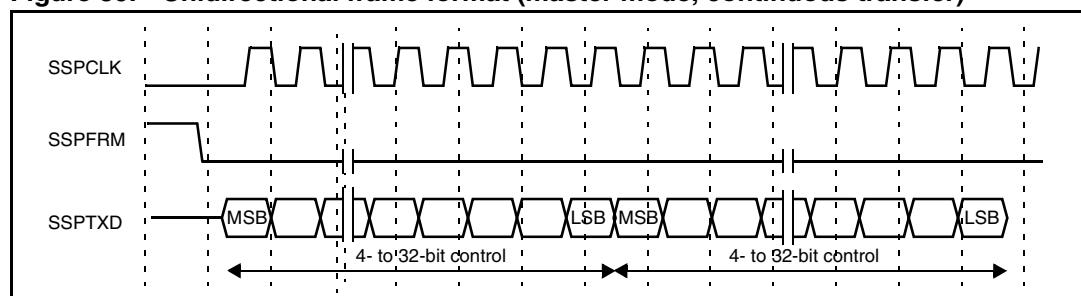


Figure 86. Unidirectional frame format (master mode, continuous transfer)



23.3.2 SPI clock configuration

This section is applicable to the Motorola SPI frame format only.

For SPI, the SSP permits the configuration of the SSPCLK clock that is used to capture data, by selection of the clock phase and polarity:

SPI clock phase

The SSP permits the configuration of the clock edge that captures data and that allows it to change state. This mostly affects the first bit transmitted, by either allowing or not allowing a clock transition before the first data capture edge.

SPI clock polarity

The SSP permits the configuration of the clock polarity; that is, the steady state (low or high) of the SSPCLK signal.

Together, the above configurations allow the clock to be stopped between transfers, using one of four possible timing variations listed in [Table 107](#).

Table 107. Possible timing variations of SSP clock for SPI

SSPCLK stopped level	First rising SSPCLK rising edge occurrence	SSP transmit-data start	SSP receive-data start
low	In the middle of the first data bit (with a delay).	Half a cycle ahead of the rising edge of SSPCLK.	On the rising edge of SSPCLK.
high	In the middle of the first data bit (with a delay).	Half a cycle ahead of the falling edge of SSPCLK.	On the falling edge of SSPCLK.
low	At the start of the first data bit (no delay).	On the rising edge of SSPCLK.	On the falling edge of SSPCLK.
high	At the start of the first data bit (no delay).	On the falling edge of SSPCLK.	On the rising edge of SSPCLK.

23.3.3 SSP bit rate generation

The SSP bit clock, SSPCLK, is derived as described below for master and slave modes.

Master mode

In master mode, SSPCLK is derived by dividing down the clock SSPICLK, the 48 MHz clock from PLL2. This is done in two stages.

- The clock is first divided by an even prescale value, CPSDVSR, which can take any even value from 2 to 254.
- The clock is further divided by an integer value from 1 to 256; this is SCR + 1, where SCR can take any integer value from 0 to 255.

The frequency of the delivered bit clock is therefore given by the following formula:

$$f_{SSPCLK} = f_{SSPICLK} / [CPSDVR \times (SCR + 1)]$$

In master mode, the maximum possible value of the SSPCLK frequency is then one half of the SSPICLK frequency; 24 MHz for this device.

Slave mode

In slave mode, SSPCLK is taken directly from the external master via the SSPCLK pin. SSPCLK can have a frequency of up to 4 MHz.

Note: The SSPCLK duty cycle is always 50%.

23.3.4 SSP data endianness

All transfers can be sent and received with the most significant bit (MSB) first or least significant bit (LSB) first. This is individually configurable for transmit and receive. In both cases, the default after reset is MSB first.

23.4 DMA interface

The SSP provides an interface to connect to the DMA controller. The DMA interface includes the following signals.

23.4.1 Transmit DMA signals

SSPTXDMASREQ: Single 32-bit data DMA transfer request, asserted by the SSP. This signal is asserted when there is at least one empty location in the transmit FIFO.

SSPTXDMABREQ: Burst DMA transfer request, asserted by the SSP. This signal is asserted when the transmit FIFO contains less 32-bit data than the programmed watermark level.

SSPTXDMACLR: DMA request clear, asserted by the DMA controller to clear the transmit request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

23.4.2 Receive DMA signals

SSPRXDMASREQ: Single 32-bit data DMA transfer request, asserted by the SSP. This signal is asserted when the receive FIFO contains at least one valid data item.

SSPRXDMABREQ: Burst DMA transfer request, asserted by the SSP. This signal is asserted when the receive FIFO contains more 32-bit data than the programmed watermark level.

SSPRXDMACLR: DMA request clear, asserted by the DMA controller to clear the receive DMA request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

Burst transfer and single transfer request signals

The burst transfer and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there is more data than the watermark level in the receive FIFO, the burst transfer request and the single transfer request are asserted. When the amount of data left in the receive FIFO is less than the watermark level, the single request only is asserted.

This is useful for situations where the number of words left to be received in the stream is less than a burst. For example: say that nineteen 32-bit words have to be received, and the

watermark level is programmed to be four. The DMA controller then transfers four bursts of four words, and three single transfers to complete the stream.

Note: For the remaining three words, the SSP cannot assert the burst request.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is de-asserted, a request signal can become active again, depending on the conditions described above. All request signals are de-asserted if the SSP is disabled or DMA transfers are disabled.

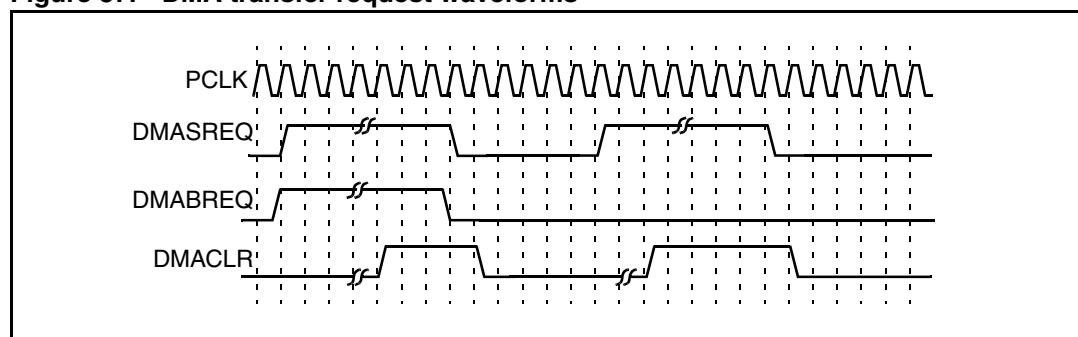
Data transfers can be made by either single or burst transfers, depending on the programmed watermark level and the amount of data in the FIFO. [Table 108](#) shows the trigger points for DMABREQ depending on the watermark level, for both the transmit and receive FIFOs.

Table 108. SSP DMA trigger points for the transmit and receive FIFOs

Watermark level	Burst length	
	Transmit (number of empty locations)	Receive (number of filled locations)
1/64	1	1
1/16	4	4
1/8	8	8
1/4	16	16

[Figure 87](#) shows the timing diagram for both a single transfer request and a burst transfer request with the appropriate DMA clear signal. The signals are all synchronous to PCLK. For clarity, it is assumed that there is no synchronization of the request signals in the DMA controller.

Figure 87. DMA transfer request waveforms



23.5 SSP signals

23.5.1 Interrupt signals

There are four individual maskable interrupts generated by the SSP:

- Receive interrupt
- Transmit interrupt
- Timeout interrupt
- Receive overrun interrupt

All four sources are combined into a single interrupt signal, SSPINTR, which is the only interrupt signal from the SSP that drives the vectored interrupt controller (VIC). This combined SSPINTR interrupt, which is an OR function of the individual masked sources, is asserted if any of the individual interrupts described below are enabled and asserted.

Receive interrupt (SSPRXINTR)

The receive interrupt is asserted when the number of data items in the receive FIFO reaches the programmed trigger watermark level. The receive interrupt is cleared by reading data from the receive FIFO until the amount of data is less than the programmed watermark level.

Transmit interrupt (SSPTXINTR)

The transmit interrupt is asserted when the number of data items in the transmit FIFO becomes less than the programmed watermark level. It is cleared by performing writes to the transmit FIFO until the number of data items rises above the programmed watermark level.

Transmit data can be written to the FIFO before or after interrupts are enabled.

- Data can be written to the transmit FIFO prior to enabling the SSP and the interrupts.
- The SSP and the interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

Timeout interrupt (SSPTINTR)

The receive timeout interrupt is asserted when the receive FIFO is not empty and no further data is received over a 32-bit period. This mechanism ensures that the user is aware that data is still present in the receive FIFO and requires servicing. The receive timeout interrupt is cleared in any of the following cases.

- The FIFO becomes empty through reading all the data.
- New data is received on the SSPRXD pin.
- The interrupt is cleared in software.

Receive overrun interrupt (SSPRORINTR)

The receive overrun interrupt is asserted when the receive FIFO is already full and an additional data frame is received, causing an overrun of the FIFO. Data is over-written in the receive shift register, but not in the FIFO.

23.5.2 Interface signals

The SSP receives serial data on its SSPrxD input pin and transmits serial data on its SSPTxD output pin. Transmission and reception can operate simultaneously and independently for full-duplex operation.

Serial transfer and receive operations are synchronized to an internally generated clock (master mode) or incoming external clock signal (slave mode) SSPCLK.

Each individual word or group of words transmitted or received are signaled by a frame synchronization signal. The SSP can generate (master mode) or receive (slave mode) its transmit frame sync signal on SSPFRM pin.

The SSP interface signals are summarized in [Table 109](#).

Table 109. SSP signals

Signal	I/O	Description
SSPCLK	I/O	SSP serial bit clock. This is delivered by the SSP when configured in master mode, and is an input when the SSP is in slave mode.
SSPFRM	I/O	SSP frame synchronization. This is an output driven by the SSP when configured in master mode and is an input when the SSP is in slave mode.
SSPrxD	I	SSP received serial data. On reset, this pin is in HiZ and GPIO mode.
SSPTxD	O	SSP transmitted serial data. On reset, this pin is in HiZ and GPIO mode.

24 Vectored interrupt controller (VIC)

The vectored interrupt controller (VIC) allows the operating system interrupt handler to quickly dispatch interrupt service routines in response to peripheral interrupts. This provides a software interface to the interrupt system and improves response times to interrupt requests.

24.1 Features

- 64 interrupt lines
- Software interrupt generation
- Normal and fast interrupts
- 16 vectored interrupts
- Privileged mode

24.2 Functional description

VIC signals are not directly accessible from the STn8815A12 pins. However, interrupts can be raised from any of the general purpose input/output (GPIO) pins.

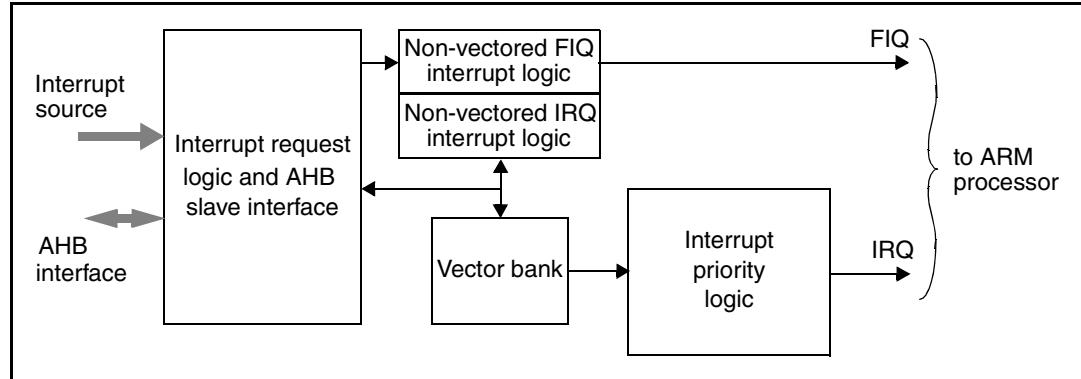
Two levels of interrupt are available:

- fast interrupt request (FIQ) for fast, low latency, interrupt handling
- interrupt request (IRQ) for more general interrupts

There are 64 software-controlled interrupt lines and 16 vectored interrupts that only generate an IRQ interrupt. The vectored IRQ interrupt provides an address for an interrupt service routine (ISR).

Interrupt priority is regulated by hardware. The FIQ interrupt has the highest priority, followed by interrupt vector 0 to interrupt vector 15. Non-vectored IRQ interrupts have the lowest priority.

Figure 88. VIC functional overview



24.2.1 Fast interrupt request (FIQ)

A fast interrupt request (FIQ) is handled in a similar way to standard IRQs, except that the interrupt handler does not check for new interrupt requests while the FIQ is being serviced. FIQs are available only to programs. Typically, a system uses only one FIQ source at a time. This has two benefits.

- The ISR can be executed directly without determining the source of the interrupt.
- Interrupt latency is reduced.

FIQ interrupts have the highest priority, and are not nested. For best results, start the FIQ handler at the FIQ vector address, 0x1C (no jump needed).

24.2.2 Interrupt request (IRQ)

With standard IRQs, the current ISR updates the interrupt priority hardware, which masks the current and any lower priority interrupt requests. When the ISR flags that the current interrupt is serviced, lower priority interrupts go active.

To reduce interrupt latency, the IRQ interrupts in the processor can be re-enabled after the ISR is entered. When that happens, the current ISR is interrupted, and any higher priority ISRs are executed. If a higher priority interrupt goes active, the current ISR is interrupted, and the higher priority ISR is executed.

IRQs can be nested. Nesting increases the interrupt latency.

25 Multichannel serial ports (MSPx)

The STn8815A12 includes four multichannel serial ports: MSP0, MSP1, MSP2 and MSP3. These are synchronous receive and transmit serial interfaces.

- MSP0 provides a full set of interface signals (see [Section Table 116.: MSP signals on page 321](#)).
- For MSP1, MSP2 and MSP3 the following restrictions apply:
 - MSPrCK or MSPrFS signals are not provided,
 - MSPSCK2 and MSPCK3 are grounded, and so cannot clock the baud-rate generator.

25.1 Features

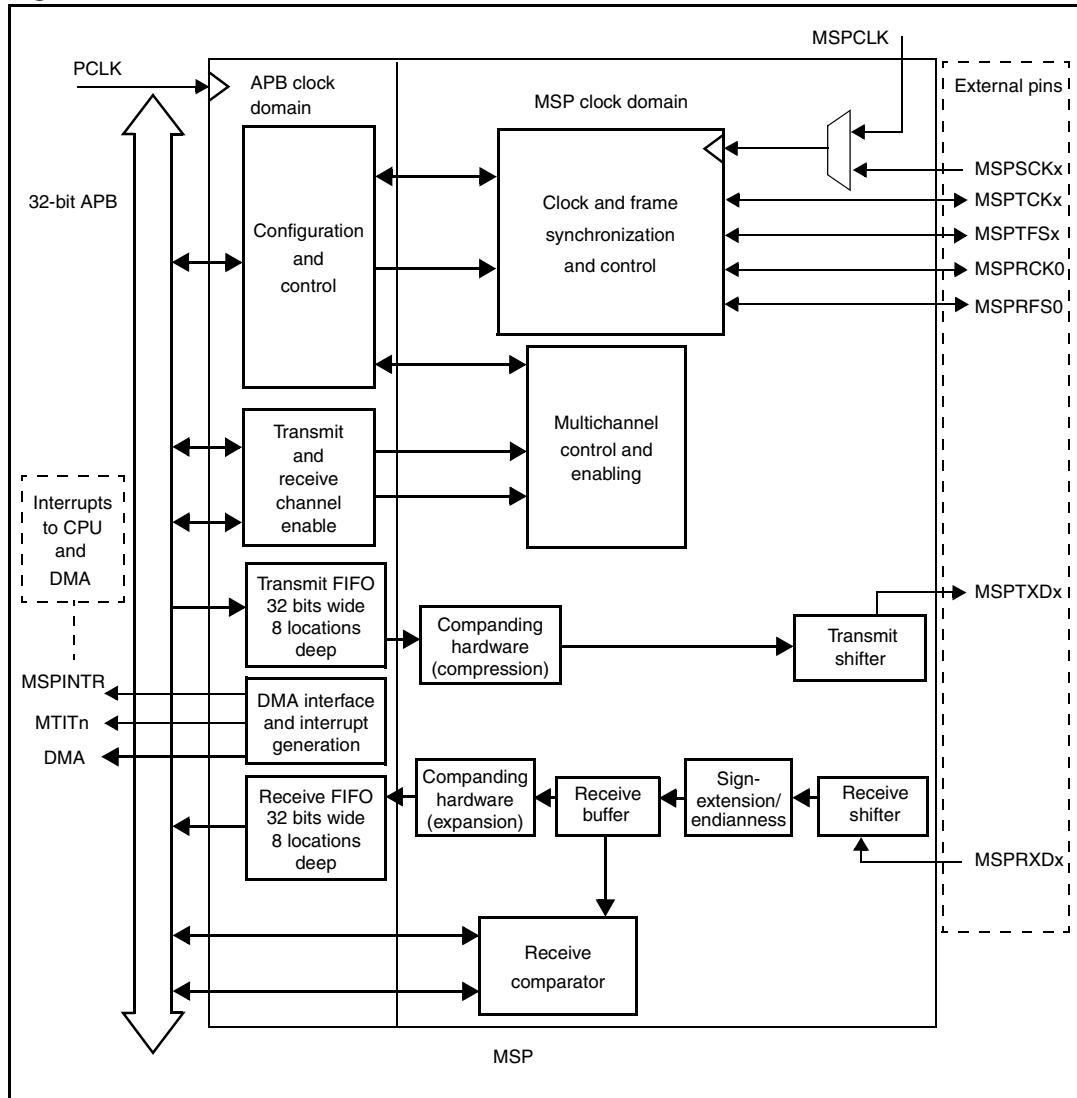
The multichannel serial port (MSP) is a synchronous receive and transmit serial interface. The MSP provides:

- Full-duplex communication
- Multichannel transmit and receive of up to 128 channels
- Element (data) sizes of 8, 10, 12, 14, 16, 20, 24, and 32 bits, LSB or MSB first
- μ -Law and A-Law companding
- Independent framing and clocking for receive and transmit
- External shift clock or an internal, programmable frequency shift clock for data transfer
- Direct interface to:
 - industry-standard codecs and serially connected A/D and D/A devices
 - IIS compliant devices
 - SPI compliant devices
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation
- Acceptance of received data through 32-bit internal comparison with masked bits
- Separate transmit and receive first-in, first-out memory buffers (FIFOs), 32 bits wide, 8 locations deep
- Support for direct memory access (DMA)

25.2 Overview

Figure 89 illustrates the architecture of MSP0.

Figure 89. MSP0 architecture



25.3 Functional description

This section describes:

- the MSP configuration,
- transmitting data,
- receiving date,
- transmitter/receiver resets,
- the sample rate generator.

MSP configuration

- The transmit and receive clocks define the boundaries between bits for transmission and reception: see [Section 25.4](#).
- The frame synchronization signals, MSPTFSx and MSPRFSx, mark the beginning of an element transfer: see [Section 25.5.2](#).

The following parameters can be configured independently for transmission and reception:

- number of elements per frame, for each phase independently: see [Section 25.5.1](#).
- number of bits per element, for each phase independently: see [Section 25.5.1](#).
- polarities of MSPTCKx, MSPRCKx, MSPTFSx, and MSPRFSx: see [Section 25.5.4](#).
- number of phases per frame (single-phase or dual-phase): see [Section 25.5.5](#).
- whether subsequent frame synchronization restarts the serial data stream or is ignored. see [Section 25.5.8](#).
- data delay from frame synchronization to first data bit: see [Section 25.5.6](#).
- MSB or LSB first, and sign extension or zero filling for received data: see [Section 25.5.12](#).

25.3.1 Transmitting data

Data to be transmitted is buffered in a 32-bit wide, 8-location deep FIFO stack. This transmit FIFO:

- can be disabled so that it acts like a one-word buffer,
- is not cleared when the transmitter is disabled.

Transmit data can be written to the FIFO before enabling the transmitter. Alternatively, the transmitter and interrupts can be enabled, so that data can be written to the transmit FIFO by an interrupt service routine.

Data is transmitted on the MSPTXD_x pin. Between the transmit FIFO and the MSPTXD_x pin, there is a transmit shift buffer (shifter) which stores the data element that is about to be transmitted.

When the transmitter is enabled, if there is no data in the shifter, then the value in the FIFO is copied into the shifter. Otherwise, the value in the FIFO is copied to the shifter only when the last bit of data has been moved to the MSPTXD_x pin.

A typical transmit operation proceeds as follows.

1. The transmit frame is synchronized.
2. Data is copied to the transmit FIFO, and then to the transmit shifter.
3. Data is copied from the shifter to the MSPTXD_x pin, after the programmed data delay.
4. More data is copied from the FIFO to the shifter.

25.3.2 Receiving data

Received data is buffered in a 32-bit wide, 8-location deep FIFO stack. This receive FIFO:

- can be disabled so that it acts like a one-word buffer,
- is not cleared when the receiver is disabled.

Received data can be read from the receive FIFO after disabling the receiver.

Received data arrives on the MSPRXD_x pin. Between the MSPRXD_x pin and the receive FIFO, there is a receive shift buffer (shifter). Each data bit arriving on MSPRXD_x is immediately moved into the receive shifter.

Once a full data element is received in the shifter, the element is copied to an intermediate receive buffer. The content of the receive buffer is then copied to the FIFO.

Data can be transferred from the receive shifter to the receive buffer only if a previous receive buffer to receive FIFO copy has completed.

A typical receive operation proceeds as follows.

1. The receive frame is synchronized.
2. Data is copied from the MSPRXD_x pin to the shifter, after the programmed data delay.
3. Data is copied from the shifter to the intermediate buffer, and then to the FIFO.
4. More data is copied from the MSPRXD_x pin to the shifter.

25.3.3 Transmitter/receiver resets

An MSP can be reset in the following ways:

- Hardware reset. When the MSP is reset:
 - All internal state machines are reset to their initial state.
 - All internal counters, FIFOs, flags and interrupts are cleared.
 - The output-only pin, MSPTXD_x, is put in the high-impedance state. All input-only pins and 3-state pins should be in a known state.
 - The sample rate generator is reset.
- When the device is pulled out of reset, the MSP remains in the reset condition.
- Software reset. The transmitter and receiver can be reset independently in software.
When this happens:
 - All activity on the relevant transmitter or receiver stops.
 - All input-only pins, and all other pins that are configured as inputs, are in a known state.
 - If either MSPRFS_x or MSPTFS_x is an output, it is driven to its inactive state.
 - If MSPRCK_x or MSPTCK_x is an output, it can be driven by the sample rate generator.
- Note that the MSPTXD_x pin is put in the high-impedance state when the transmitter is disabled.
- Sample rate generator software reset. The sample rate generator can be reset by disabling it. The sample rate generator clock (GCLK) and the frame synchronization signal (GFS) are then driven inactive (low).
When the sample rate generator is re-enabled, GFS is driven active (high) after eight GCLK cycles.

The tables below give the states of the MSP transmit and receive pins when the MSP is under reset, or when the MSP transmitter or receiver is disabled.

Table 110. MSP transmitter pin states after hardware reset or when disabled

MSP pins	Hardware reset	Transmitter disabled
MSPTXD _x	High impedance	High impedance.
MSPTCK _x	Input	Known state if input; MSPTCK _x if output.
MSPTFS _x	Input	Known state if input; transmit polarity value (inactive state) if output.
MSPSCK _x	Input	Input.

Table 111. MSP receiver pin states after hardware reset or when disabled

MSP pins	Hardware reset	Receiver disabled, MSP pins not in GPIO mode
MSPRXD _x	Input	Input.
MSPRCK _x	Input	Known state if input; MSPRCK _x if output.
MSPRFS _x	Input	Known state if input; receive polarity value (inactive state) if output.
MSPSCK _x	Input	Input.

25.3.4 Sample rate generator

The sample rate generator can produce clocking (GCLK) and framing (GFS) signals independently for transmit and receive.

GCLK can be generated using:

- the CPU clock,
- the external clock source on the MSPSCK_x pin.

When the external clock is selected:

- The selected input clock can be divided by a value between 1 and 256 to produce GCLK. The programmable divider is clocked by the rising edge of the selected clock source.
- The signal can be inverted.
- GCLK can be one of:
 - free-running,
 - synchronized with the receive frame synchronization pin (MSPRFS_x).

Synchronizing GCLK with the MSPRFS_x signal ensures that the MSP and the device with which the MSP is communicating are dividing down the MSPSCK_x clock with the same phase relationship; an inactive-to-active transition on MSPRFS_x triggers a re-synchronization of GCLK and the generation of GFS.

GCLK always begins in the high state after synchronization. In addition, MSPRFS_x is always detected on the same edge of MSPSCK_x that generates GCLK, regardless of the length of the MSPRFS_x signal. Although an external MSPRFS_x is provided, GFS can still be configured to drive internal receive frame synchronization.

Figure 90 and *Figure 91* show this operation for:

- the two polarities of MSPRFSx,
- divide factors of 1 and 2,
- synchronization of GCLK by the MSPRFSx signal.

In synchronization of GCLK by the MSPRFSx signal, the transmitter can operate synchronously with the receiver if the following two conditions are met.

- MSPTFSx is driven by GFS. The receive input MSPRFSx can be used if it can be sampled by the falling edge of GCLK. In that case, the MSPRFSx pin and MSPTFSx pin must be connected together externally.
- MSPTCKx and MSPRCKx are driven by GCLK. The MSPTCKx and MSPRCKx pins are then output pins.

Figure 90. GCLK synchronization and GFS generation (divide factor = 1)

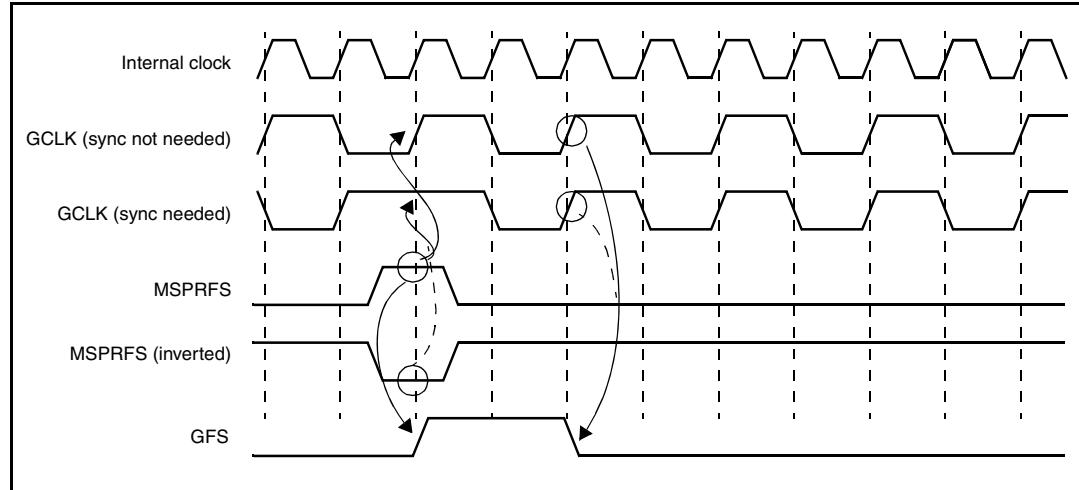
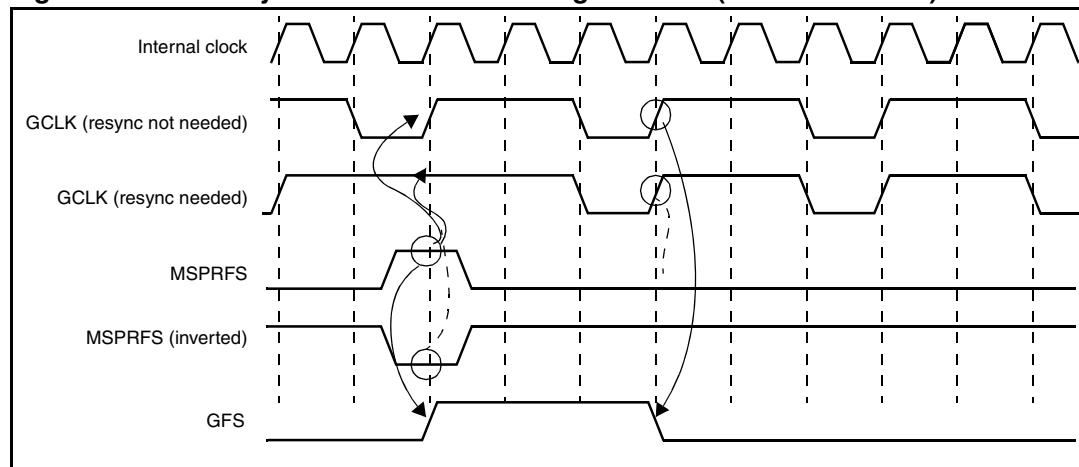


Figure 91. GCLK synchronization and GFS generation (divide factor = 2)



25.4 Clock operation

The transmit and receive clocks can be programmed as inputs or outputs. Their polarities can also be configured; that is, whether the internal clock signals are inverted before being sent to the external input or output pins.

When the same clock is used for the receiver and transmitter, the clock transmit and receive polarities must have the same settings. The receiver uses the edge opposite to that used by the transmitter to ensure valid setup and hold times of the data around this edge. [Figure 92](#) and [Figure 93](#) show clocking for transmit and receive data.

Figure 92. Transmit data clocking

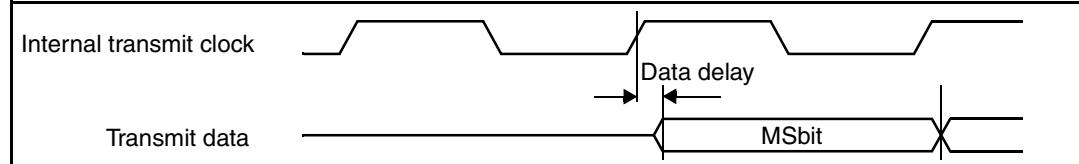
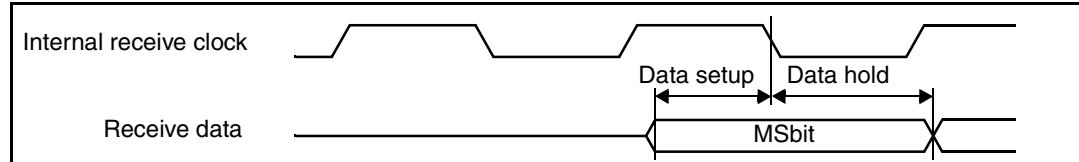


Figure 93. Receive data clocking



25.4.1 Transmit clock selection

The transmit clock can be configured in either of the following ways:

- As input

The MSPTCK_x pin is an input driven by the external clock, and the signal can be inverted before driving the internal clock.
- As output

the sample rate generator clock drives the internal clock, and the signal can be inverted before driving the MSPTCK_x pin.

25.4.2 Receive clock selection

The receive clock can be configured in any of the following ways:

- As input
 - The MSPRCK_x pin is an input driven by the external clock, and the signal can be inverted before driving the internal clock.
 - The internal transmit clock drives the internal receive clock. In this case, although the MSPRCK_x pin is an input, it is not used.
- As output
 - The sample rate generator clock drives the internal receive clock, and the signal can be inverted before driving the MSPRCK_x pin.
 - The internal transmit clock drives the internal receive clock, and the signal can be inverted before driving the MSPRCK_x pin.

25.5 Frame operation

25.5.1 Frame and element length

- Frame length: The frame length is the number of elements (or timeslots or channels) transferred per frame synchronization signal.
Each phase of a frame can have up to 128 elements, so a single-phase frame can have up to 128 elements, and a dual-phase frame can have up to 256 elements.
- Element length: The element length can be 8, 10, 12, 14, 16, 20, 24 or 32 bits.

25.5.2 Frame synchronization signals

The source of the frame synchronization signal for transmit (MSPTFSx) and receive (MSPRFSx) can be selected in software.

For transmit (MFPTFSx) the choices are:

- external signal on the MSPTFSx pin. MSPTFSx is an input signal, and is detected on the falling edge of the internal transmit clock.
- sample rate generator synchronization signal (GFS). MSPTFSx is an output signal, and its level is changed on the rising edge of the internal transmit clock.
- transmit shifter. MSPTFSx is an output signal, and is activated when data is moved to the transmit shifter.

For receive (MFPRFSx) the choices are:

- directly from an external signal on the MSPRFSx pin. MSPRFSx is an input signal, and is detected on the falling edge of the internal receive clock.
- indirectly from an external signal on the MSPRFSx pin. MSPRFSx is an input signal. The external signal is used to synchronize the sample rate generator clock (GCLK). GCLK generates the signal GFS. The GFS signal drives the internal receive frame synchronization signal (RFS).
- sample rate generator synchronization signal (GFS). MSPRFSx is an output signal, and its level is changed on the rising edge of the internal transmit clock.
- internal transmit frame synchronization signal (TFS)

When the RFS is driven by the TFS, there are three options for the MSPRFSx pin:

- high impedance.
- input, used to synchronize GCLK and generate GFS.
- output for the RFS.

25.5.3 Frame synchronization signal generation

The transmit and receive frame synchronization signals, MSPTFSx and MSPRFSx, can be configured independently for all data delay values. The following options can be set:

- period between synchronization signals, and the active width.
- synchronization on data copy to the transmit shifter. Maximum packet frequency requires a data delay of 0.

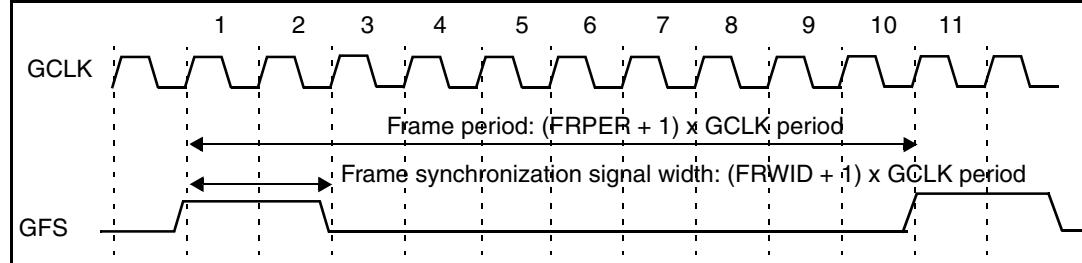
When the sample rate generator is disabled, its frame synchronization signal (GFS) is in an inactive (low) state.

The frame period ranges from 1 to 8 192 data bits.

The frame synchronization signal width ranges from 1 to 64 data bit clocks.

[Figure 94](#) shows a frame of 10 GCLK periods, with a frame synchronization signal width of 2 GCLK periods.

Figure 94. Frame synchronization signal generation: period and width



25.5.4 Frame synchronization signal polarity

The internal frame synchronization signals are active high. The polarities between the internal and external signals can be inverted in software, whether the signals are input or output.

25.5.5 Frame phase (single or dual)

The datastream following frame synchronization is configurable as follows.

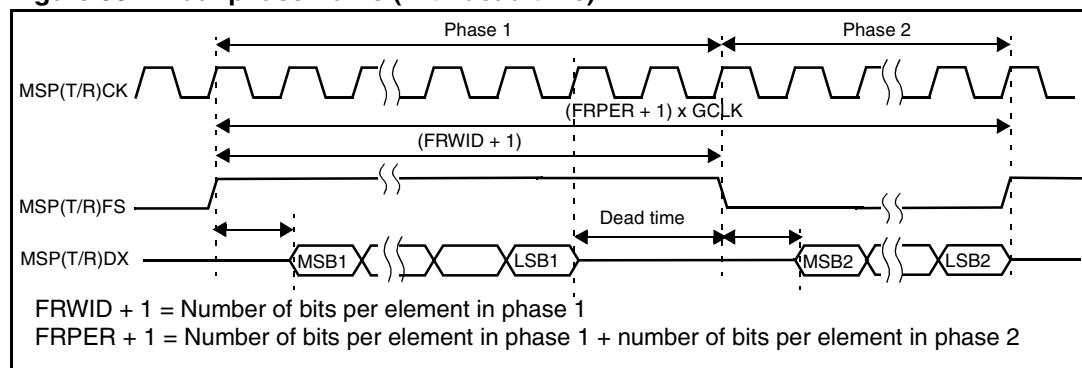
- It can have one or two phases, set independently for transmit and receive.
- The number of elements in each phase can be different.
- The number of bits per element for each phase can be different.

For dual-phase frames, the point at which the second phase starts can be configured.

- Phase 2 starts as soon as phase 1 has finished.
- Phase 1 starts as soon as the frame synchronization signal (MSPTFSx or MSPRFSx) becomes active. Phase 2 starts when the frame synchronization signal reaches the edge opposite to the one that started phase 1, as shown in [Figure 95](#).

When MSPTFSx or MSPRFSx is an output pin, the duration of phase 1 and the total frame period for the two phases are configurable. In [Figure 95](#), they are referred to as FRWID+1 and FRPER+1. If MSPTFSx or MSPRFSx is an input pin, the MSP detects the signal transition after phase 1, and starts transmission/reception of phase 2. Dead time between phase 1 and phase 2 can exist, as shown in [Figure 95](#).

Figure 95. Dual-phase frame (with dead time)



25.5.6 Data delay

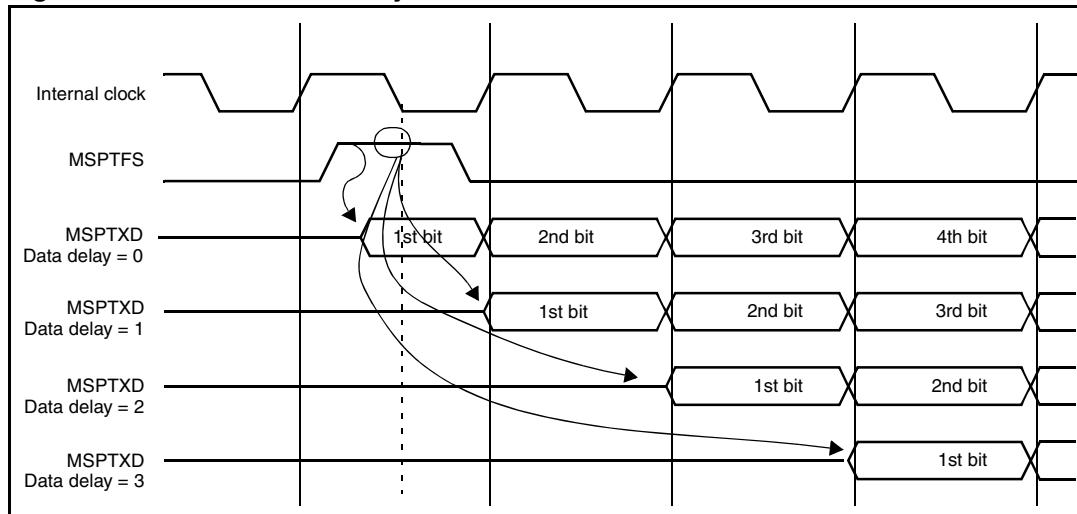
The start of a frame is defined by the first clock cycle in which the relevant (transmit or receive) frame synchronization signal is active.

A delay can be introduced between the start of the frame and the start of data transmission/reception. This delay is called the data delay, and is configurable from zero to three clock cycles.

Transmit data delay

Figure 96 shows the effect of the data delay on transmission.

Figure 96. Transmit data delay



When the transmit frame synchronization signal (MSPTFS_x) is an input, the external frame synchronization signal is sampled, and data can be transmitted on the next cycle. However, in the case of a zero-bit data delay, the data must be ready for transmission on the same clock cycle. Therefore, the first data bit is assumed to be in the transmit shifter, and when MSPTFS_x goes active, it immediately starts driving the first bit from the shifter onto the MSPTXD_x pin.

Transmit data extra delay

A further delay can be introduced for the MSPTXD_x pin turn-on time. This feature is useful for MSP multichannel operations, such as in a time-division multiplexed (TDM) system where several MSPs are used to transmit data on the same TDM line. By increasing the delay, bus contention can be avoided.

When the extra delay is enabled, the MSPTXD_x pin is maintained at high impedance for one extra PCLK clock cycle before the first data bit is delivered on the TDM line. Only the first bit of the data (sub-)frame is delayed.

The extra delay is illustrated in *Figure 97*.

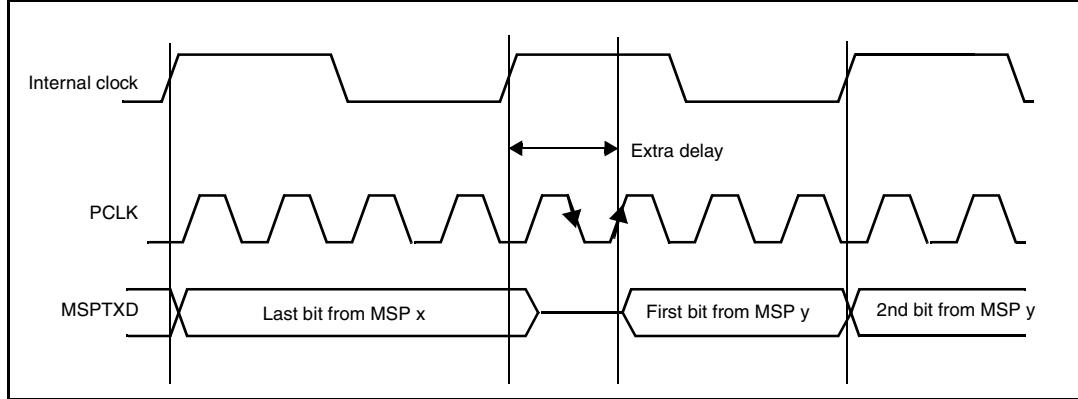
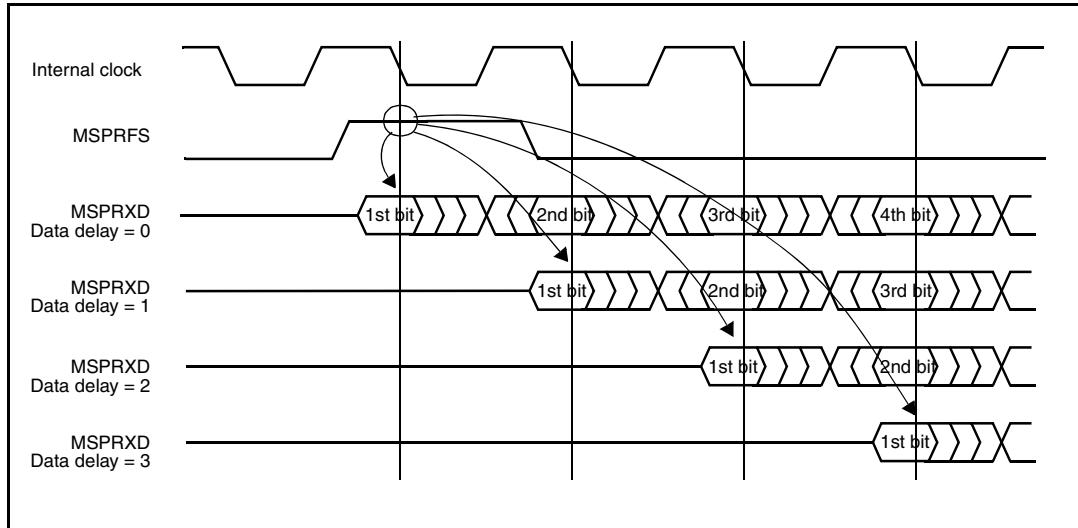
Figure 97. Transmit extra data delay**Receive data delay**

Figure 98 shows the effect of the data delay on reception.

Figure 98. Receive data delay

When the receive frame synchronization signal (MSPRFS_x) is an input, the external receive frame synchronization signal is sampled, and data can be received on the same or a subsequent cycle.

25.5.7 Maximum frame frequency

The frame frequency is determined by the following formula:

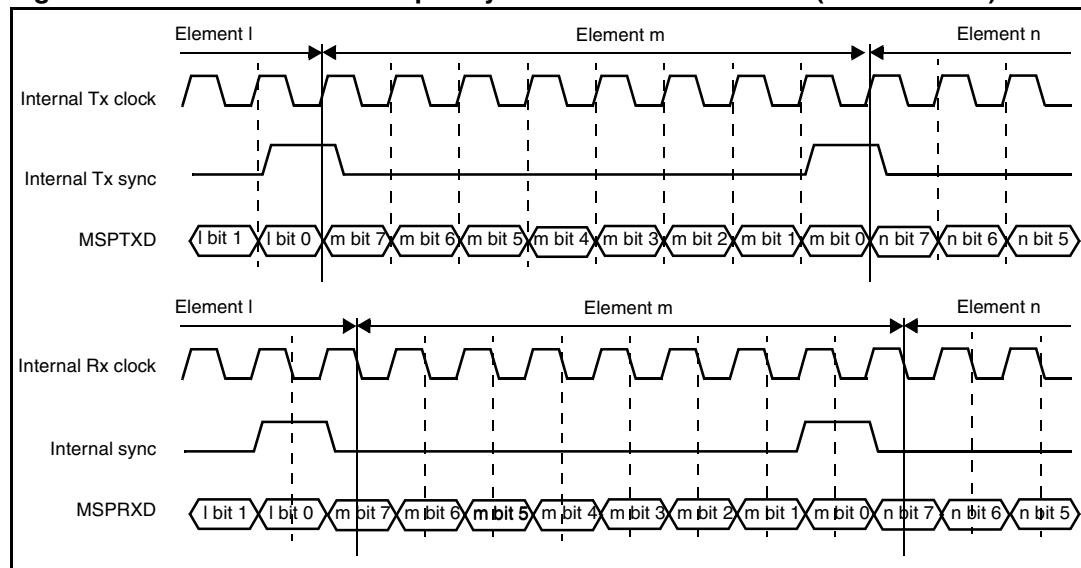
$$\text{Frame frequency} = (\text{Bit clock frequency}) / (\text{Number of bit clocks between frame synchronization signals})$$

The frame frequency can be increased by decreasing the time between frame synchronization signals in bit clocks. The minimum time between frame synchronization signals corresponds to the number of bits transferred per frame. This time also defines the maximum frame frequency, which is calculated using the following formula:

$$\text{Maximum frame frequency} = (\text{Bit clock frequency}) / (\text{Number of bits per frame})$$

The data bits in consecutive frames are transmitted continuously with no inactivity between bits. If there is a 1-bit data delay, as shown, the frame synchronization signal overlaps the last bit transmitted in the previous frame. See [Figure 99](#).

Figure 99. Maximum frame frequency for transmit and receive (8-bit element)



25.5.8 Unexpected frame synchronization ignore

The MSPs can be configured to ignore frame synchronization signals that occur one or more bit clocks earlier than the programmed data delay.

- During transmission. There are two options for dealing with these unexpected signals.
 - Abort the transmission, generate the transmit frame synchronization error interrupt and re-transmit the current element.
 - Ignore the signals.
- During reception. There are two options for dealing with these unexpected signals.
 - Abort the data transfer, generate the receive frame synchronization error interrupt and begin receiving a new element.
 - Ignore the signals.

25.5.9 Data packing

There are two methods for packing data:

- By manipulating frame length and element length.

This is applicable when each frame contains more than one element. For example, if there are four 8-bit elements to be transferred in a single-phase frame, it requires four reads and four writes for each frame (one read and one write for each byte transfer). But if the frame and element length are set to view the stream as one 32-bit element, it requires only one read and one write for each frame.

- By ignoring frame synchronization signals.

This is applicable when each frame contains fewer than the maximum number of elements. For example, if each frame has only a single 8-bit element, data can be packed by ignoring three frame syncs out of four: then the four 8-bit elements can be packed into one 32-bit data element on receipt. Thus, only one read transfer and one write transfer is needed every 32 bits.

25.5.10 Data swapping

Bytes within a word can be swapped in various ways between the Tx FIFO and the Tx shifter for the transmit path, and between the receive buffer and Rx FIFO for the receive path.

25.5.11 Loopback mode

Loopback mode allows the testing of serial port code with a single MSP device. In loopback mode:

- the transmit frame synchronization signal is used as the receive frame sync signal.
- the pins MSPRXD_x, MSPRFS_x and MSPRCK_x are internally connected through multiplexers to pins MSPTXD_x, MSPTFS_x and MSPTCK_x, as shown in [Figure 100 on page 308](#).

25.5.12 Data companding

Companding is the compression and expansion of data. The MSP allows companding in either μ -law or A-law format. The specifications for μ -law and A-law are part of the CCITT G.711 recommendation.

- μ -law allows a dynamic range of 14 bits.
- A-law allows a dynamic range of 13 bits.

Any value outside these ranges is set to the highest or lowest permissible value.

Note: Due to the data justification format selected for the companding, the data must be at least 16 bits wide.

Companding assumes an 8-bit element length. If a different element length is specified, the difference is ignored: companding continues as if the element length is eight bits.

Transmit data is compressed between the transmit FIFO and shifter. Receive data is expanded between receive buffer and FIFO. See [Figure 100 on page 308](#).

The MSPs can be configured to determine whether receive data is to be sign-extended, zero-filled or expanded using μ -law or A-law. [Table 112](#) summarizes the possible expansion/zero-fill/sign-extension combinations.

Table 112. Sign extension/comanding

Expansion	Sign extension
None	Zero-fill most significant bits (MSBs)
None	Sign-extend MSBs
μ -law	Sign-extend MSBs
A-law	Sign-extend MSBs

25.5.13 Using companding hardware as a co-processor

If an MSP is not used for transmitting and/or receiving data, the companding hardware can be used as a co-processor to:

- convert linear data to the appropriate μ -law or A-law format,
- convert μ -law or A-law data to linear format,
- analyze the quantization effects on data when compressed and then re-expanded using the same companding law.

There are two ways to compand data, direct and loopback. The direct method is faster, but no interrupts are generated.

Direct mode

In direct mode:

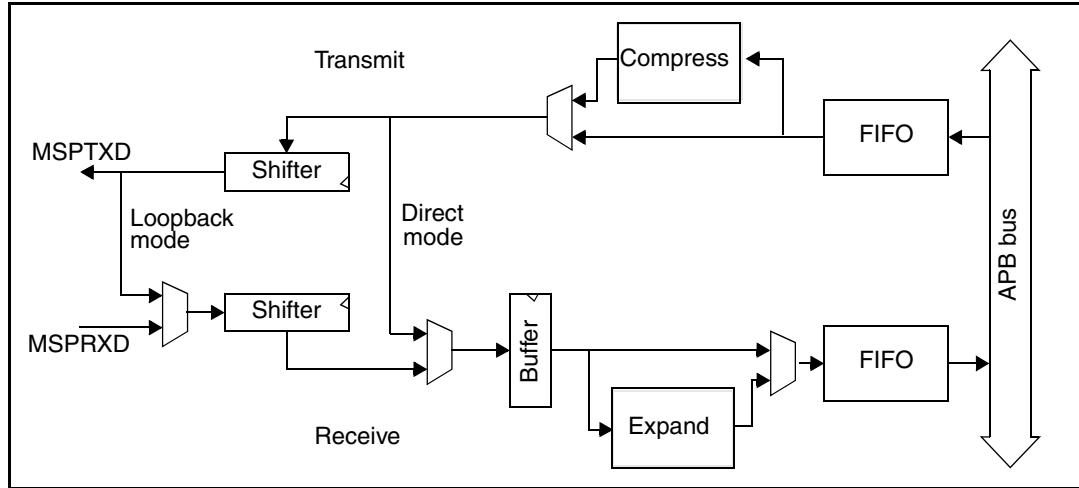
- both the transmitter and receiver are disabled.
- the transmit and receive FIFOs are internally connected through the companding logic.
- the MSP flags are set and cleared according to the states of the two FIFOs, allowing software synchronization by polling.
- the transmit and receive FIFO service interrupts are not generated.

Loopback mode

In loopback mode:

- both the transmitter and receiver are enabled.
- the transmit and receive FIFO service interrupts can be generated.

Figure 100 illustrates the two methods.

Figure 100. Companding hardware as a co-processor

25.5.14 Data endianness

All transfers can be sent or received with the MSB first or LSB first.

The default data endianness after reset is MSB first. This endianness selection is independent of the data type (sign-extension/companding) and element length.

25.5.15 Multichannel operations

When using an MSP with a single-phase frame, multiple channels can be independently selected for transmission and reception. Each frame represents a time-division multiplexed datastream, where an element within the frame corresponds to a channel. Up to 128 channels can be enabled at any one time.

If a transmit element is not enabled, the MSPTXD_x pin is kept in the high-impedance state.

Receive data can be compared bit-to-bit with a stored value, and the result (either match or non-match) used to accept or reject the received data. A receive interrupt can also be generated.

Note: If companding is enabled, the comparison is performed on the received data before expansion.

25.5.16 SPI clock modes for SPI protocols

The SPI clock modes allow the MSPs to comply with Motorola's SPI protocol, which has a master-slave configuration. The 4-wire interface handles:

- data in: master in, slave out (MISO),
- data out: master out, slave in (MOSI),
- shift clock (SCK),
- active-low slave enable signal (SS_n).

Data transfer starts when the master clock is detected, and stops when the master clock cannot be detected. The slave has to be enabled during this period of transfer. When the MSP is the master, the slave enable signal is derived from the master transmit frame synchronization signal (MSPTFS_x). In SPI modes, the MSP transmit clock (MSPTCK_x) drives the receive clock (MSPRCK_x).

For the MSP to operate in SPI modes, it must be configured in single-frame mode with one element per frame.

Figure 101 and *Figure 102* show the MSP in SPI master and slave modes.

Figure 101. MSP as SPI master

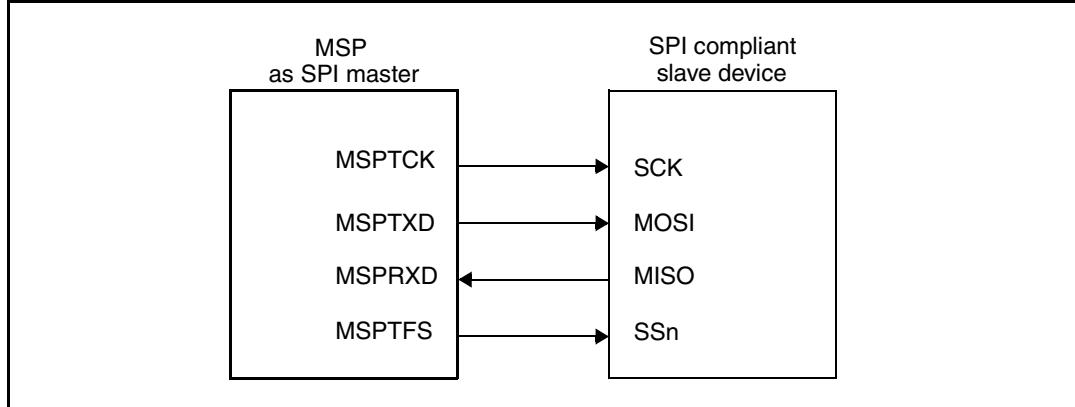
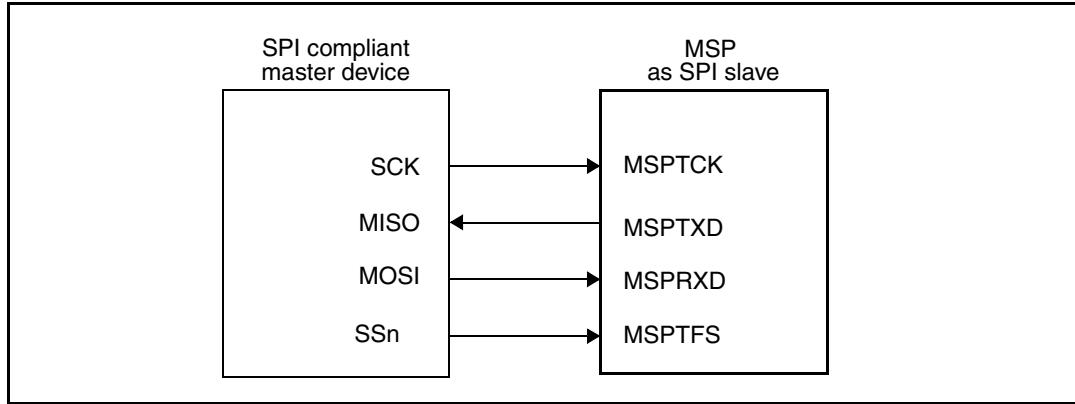


Figure 102. MSP as SPI slave



SPI clock modes

The MSPs support two SPI transfer formats, with delay and without delay. The MSPs also allow:

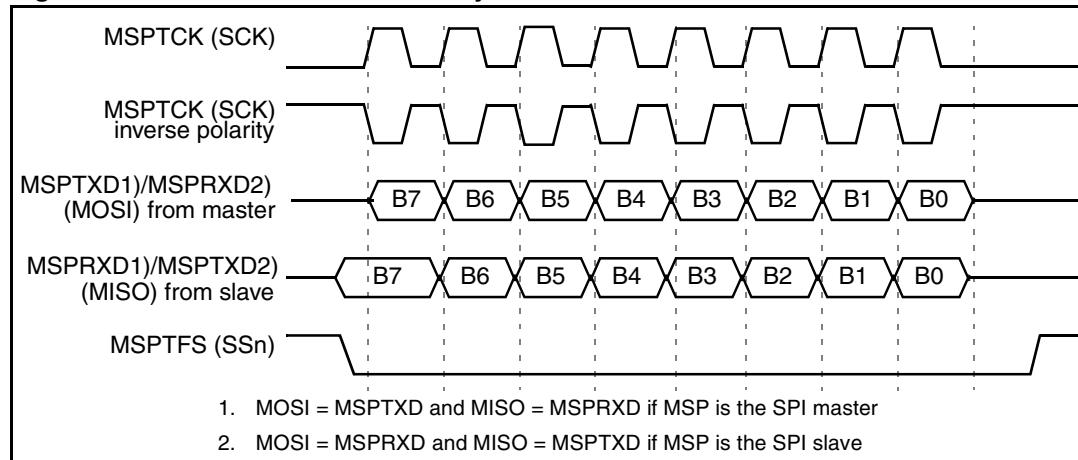
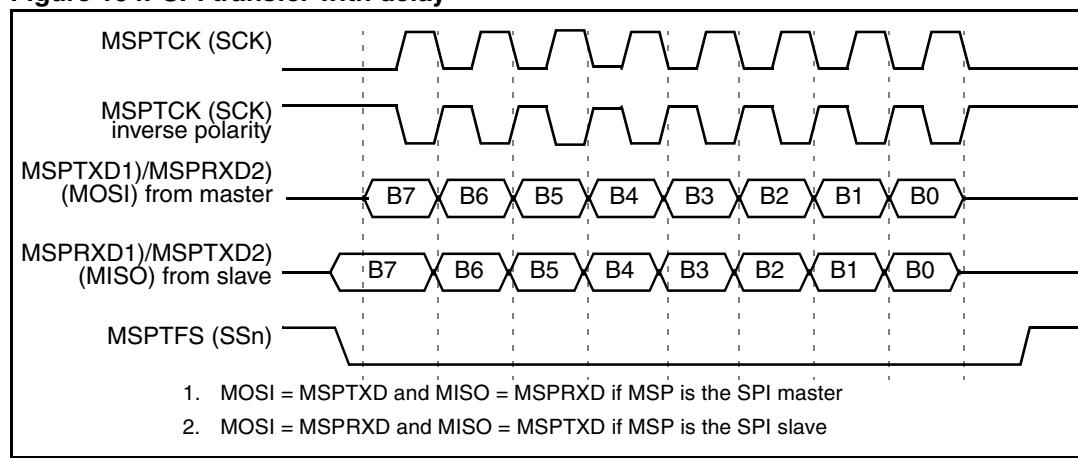
- serial clocks to be stopped between transfers using four possible timing variations,
- the edge to be selected on which data is transmitted (driven) and received (sampled).

The four clock modes are described in *Table 113*.

Table 113. MSP clock modes for SPI protocol

Clock mode			
0	1	2	3
MSPTCKx is held at the low inactive level between transfers.	MSPTCKx is held at the high inactive level between transfers.	MSPTCKx is held at the low inactive level between transfers.	MSPTCKx is held at the high inactive level between transfers.
MSPTCKx's first rising edge occurs at the start of the first data bit (no delay).		MSPTCKx's first rising edge occurs in the middle of the first data bit (with delay).	
The MSP transmits data on the rising edge of MSPTCKx and receives data on the falling edge of MSPRCKx.	The MSP transmits data on the falling edge of MSPTCKx and receives data on the rising edge of MSPRCKx.	The MSP transmits data one-half cycle ahead of the rising edge of MSPTCKx and receives data on the rising edge of MSPRCKx.	The MSP transmits data one-half cycle ahead of the falling edge of MSPTCKx and receives data on the falling edge of MSPRCKx.

Figure 103 and *Figure 104* show the timing diagrams of the two SPI transfer formats and the four timing variations.

Figure 103. SPI transfer without delay**Figure 104. SPI transfer with delay**

The above diagrams correspond to the following cases.

- *Figure 103* is the timing diagram in the case of a clock without a delay. The transition of the first clock edge marks the beginning of data transfer, provided that the slave enable signal (SSn) is already asserted. The data transfer is synchronized on the first clock edge.
- *Figure 104* is the timing diagram in the case of a clock with a delay. Data transfer begins before the transition of the serial clock. The transition of the slave enable signal (SSn) from high to low, marks the beginning of the transfer.

Both the transmitter and the receiver operate together as a master or a slave. The MSP is a master when it generates clocks.

The slave enable signal, MSPTFSx (SSn), enables the serial data input and output driver on the slave device.

MSP as SPI master

As an SPI master, the MSP must be configured as follows:

- Single-frame mode.
- One element per frame.
- MSPTCKx as an output. The sample rate generator clock (GCLK) drives the master clock MSPTCKx. The clock divide ratio must be programmed to generate MSPTCKx at the required SPI data rate. GCLK also drives MSPRCKx and SCK.
- MSPTFSx as an output. The sample rate generator synchronization signal (GFS) drives the slave enable signal MSPTFSx, which should be:
 - connected to the slave enable input (SSn),
 - generated when data is copied to the transmit shifter. The data delay must be set to 1 clock cycle.
- ‘MSPRFSx to receive data’ delay must be set to one MSPRCKx clock cycle.

See [Section 25.5.6: Data delay on page 303](#).

MSP as SPI slave

As an SPI slave, the MSP must be configured as follows:

- single-frame mode.
- one element per frame.
- MSPTCKx as an input. An external SPI master drives the master clock MSPTCKx, as shown in [Figure 102 on page 309](#). MSPTCKx drives MSPRCKx and SCK.
- MSPTFSx as an input, connected to the slave enable input (SSn). MSPTFSx drives MSPRFSx. The external SPI master must assert MSPTFSx (low) before the transfer of data begins.
- ‘MSPTFSx to transmit data’ and ‘MSPRFSx to receive data’ delay must be zero. This ensures that the first data to be transmitted is available on the MSPTXDx pin, and that the MSP is ready to receive data from the SPI master as soon as it detects MSPTCKx. Depending on the clock stop mode used, data is received at various clock edges according to [Table 113 on page 310](#).
- internal clock drives the sample rate generator clock (GCLK). Both must have the same frequency. Although MSPTCKx is generated externally, the sample rate generator must be enabled to provide GCLK, which is used to synchronize MSPTCKx with MSPTFSx. GCLK must be at least eight times the SPI data rate. This

rate is achieved by programming the sample rate generator to its maximum speed for all SPI transfer rates.

25.5.17 MSP error detection

The MSPs can signal four kinds of error:

- transmit underrun,
- transmit frame synchronization error,
- receive overrun,
- receive frame synchronization error.

These errors can trigger an interrupt.

Transmit underrun

This error occurs in either of the following situations:

- During transmission:
 - When the transmit FIFO has not been loaded since the last data was moved to the shifter.
 - When all bits in the shifter have been moved onto the MSPTXD_x pin.
- When the transmitter has been started, and an external transmit frame synchronization signal occurs before the transmit FIFO is loaded.

During the underrun condition, the transmitter continues to transmit the old data at every new frame synchronization signal.

Transmit frame synchronization error

This error occurs when a synchronization signal is detected *DD* bit clocks earlier than the last transmitted bit of the previous frame, where *DD* is the data delay.

These errors can be handled in one of the following ways:

- The error is ignored and transmission continues.
- The current element is retransmitted.

Receive overrun

This error occurs in any of the following situations:

- the receive FIFO is full and has not been read since the last data was received.
- the receive buffer is full and no data has been transferred to the receive FIFO.
- the receive shifter is full and no data has been transferred to the receive buffer.

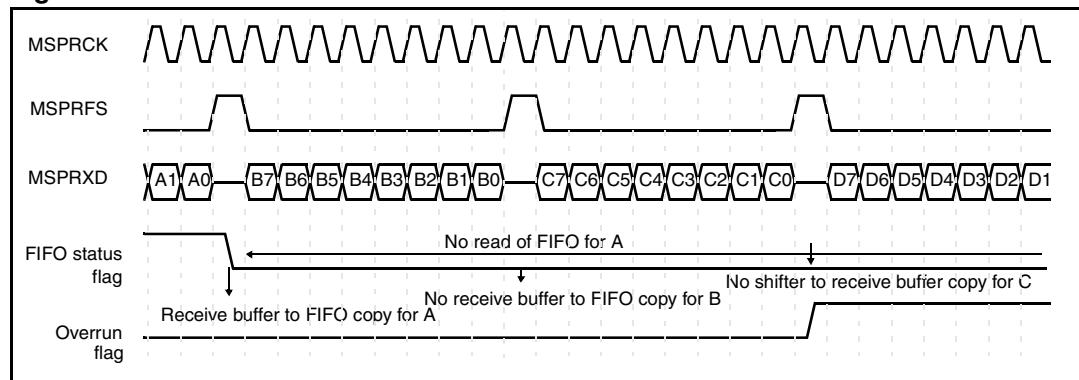
During the overrun condition, new data arriving on the MSPRXD_x pin is moved into the receive shifter and the previous contents of the receive shifter are lost.

Once the receiver starts running from reset:

- If the receive FIFO is disabled, a minimum of three elements must be received before the receive overrun flag can be set.
- If the receive FIFO is enabled, a minimum of ten elements must be received before the receive overrun flag can be set.

Figure 105 shows an example of a receive overrun condition when the receive FIFO is disabled. Note the following sequence of events.

1. Since element A is not read before the reception of element B is complete, B is not yet transferred to the FIFO.
2. Another element, C, arrives and fills the receive shifter.
3. The FIFO is finally read, but not earlier than 2.5 cycles before the end of element C.
4. New data, D, overwrites the previous element C in the receive shifter.
5. If the receive overrun flag remains set after the FIFO has been read, the next element can overwrite D if the FIFO is not read in time.

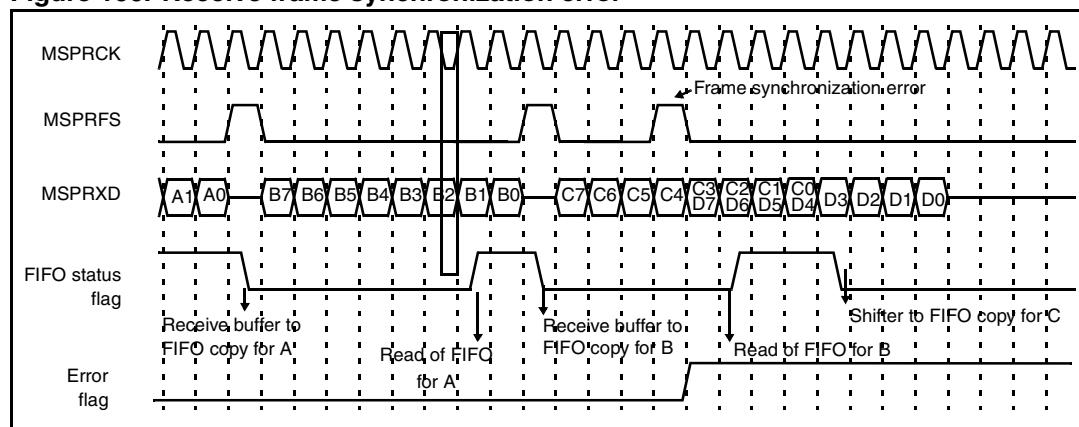
Figure 105. Receive overrun

Receive frame synchronization error

This error occurs when a synchronization signal is detected DD bit clocks earlier than the last received bit of the previous frame, where DD is the data delay. See [Figure 106](#).

These errors can be handled in one of the following ways:

- The error is ignored and reception continues.
- The receive frame synchronization error flag is set.

Figure 106. Receive frame synchronization error

Note:

The received data does not take into account the unexpected frame synchronization signal. Therefore, in the example shown above, the value written to the receive FIFO is composed of the eight bits received following the valid frame synchronization signal [C7:C0].

25.6 DMA interface

Each MSP provides an interface to the DMA controller. The DMA interface includes the signals shown in [Table 114](#).

Table 114. MSP DMA signals

Signal	I/O	Description
MSPTDMASREQ	O	Transmit DMA single word request (active high).
MSPTDMABREQ	O	Transmit DMA burst request (active high).
MSPTDMACLR	I	Transmit DMA clear, asserted by the DMA controller to clear the transmit request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.
MSPRDMASREQ	O	Receive DMA single word request (active high).
MSPRDMABREQ	O	Receive DMA burst request (active high).
MSPRDMACLR	I	Receive DMA clear, asserted by the DMA controller to clear the receive request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

Transmit DMA signals

- MSPTDMASREQ: single-word DMA transfer request, asserted by the MSP. This signal is asserted when there is at least one empty location in the transmit FIFO (whether enabled or disabled).
- MSPTDMABREQ: burst DMA transfer request, asserted by the MSP. This signal is asserted when the transmit FIFO is enabled and contains four words or less.
- MSPTDMACLR: DMA request clear, asserted by the DMA controller to clear the transmit request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

Receive DMA signals

- MSPRDMASREQ: single-word DMA transfer request, asserted by the MSP. This signal is asserted when the receive FIFO contains at least one word (whether enabled or disabled).
- MSPRDMABREQ: burst DMA transfer request, asserted by the MSP. This signal is asserted when the receive FIFO is enabled and contains four words or more.
- MSPRDMACLR: DMA request clear, asserted by the DMA controller to clear the receive DMA request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data item in the burst.

Note: The MSPTDMASREQ and MSPRDMASREQ signals are each forced low when the associated part of the MSP is disabled.

The burst transfer and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there is more data than the watermark level in the receive FIFO, the burst transfer request and the single transfer request are asserted. When the amount of data left in the receive FIFO is less than the watermark level, the single request only is asserted.

This is useful for situations where the number of words left to be received in the stream is less than a burst. Say nineteen words have to be received and the watermark level is

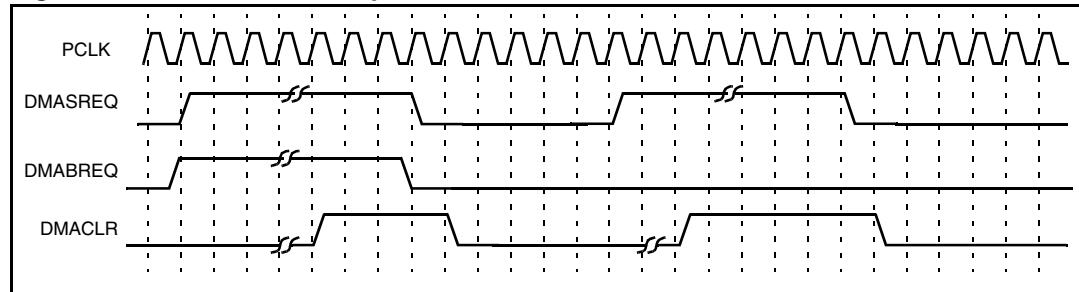
programmed to be four. The DMA controller then transfers four bursts of four words and three single transfers to complete the stream.

Note: For the remaining three words, the MSP cannot assert the burst request.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is de-asserted, a request signal can become active again, depending on the conditions described above. All request transmitter/receiver signals are de-asserted if the MSP transmitter/receiver is disabled or the transmitter/receiver DMA transfers are disabled.

Figure 107 shows the timing diagram for both a burst transfer request and a single transfer request with the appropriate DMA clear signal. The signals are all synchronous to PCLK.

Figure 107. DMA transfer request waveforms



25.7 MSP configuration examples

Multi-phase frame example

Figure 108 shows an example of the Audio Codec standard, which uses the dual-phase frame feature. The first phase consists of a single 16-bit element. The second phase consists of twelve 20-bit elements. The phases are configured as follows:

- dual-phase frame,
- one element per frame in phase 1,
- 16 bits per element in phase 1,
- 12 elements per frame in phase 2,
- 20 bits per element in phase 2,
- receive data is sampled on the falling edge of MSPRCKx,
- transmit data is clocked on the rising edge of MSPTCKx,
- active-high frame synchronization signals are used,
- data delay of one bit clock is used.

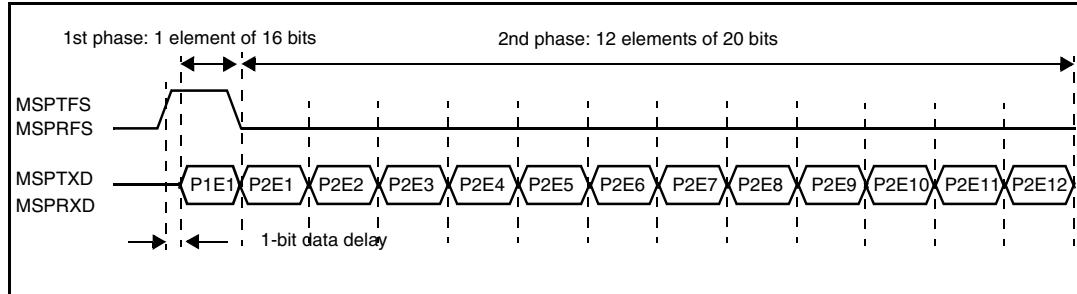
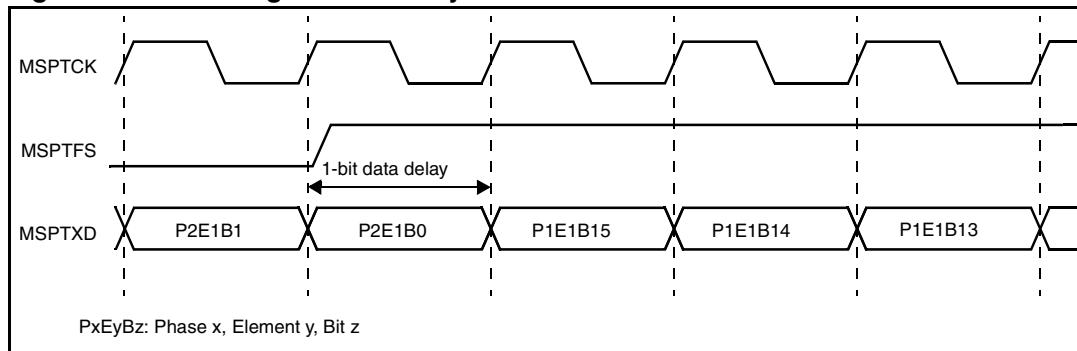
Figure 108. Dual-phase frame format

Figure 109 shows the timing near frame synchronization. Note that the frame synchronization signal overlaps the first element. The inactive-to-active transition of the frame synchronization signal indicates frame synchronization; the signal can then remain high for an arbitrary number of bit clocks.

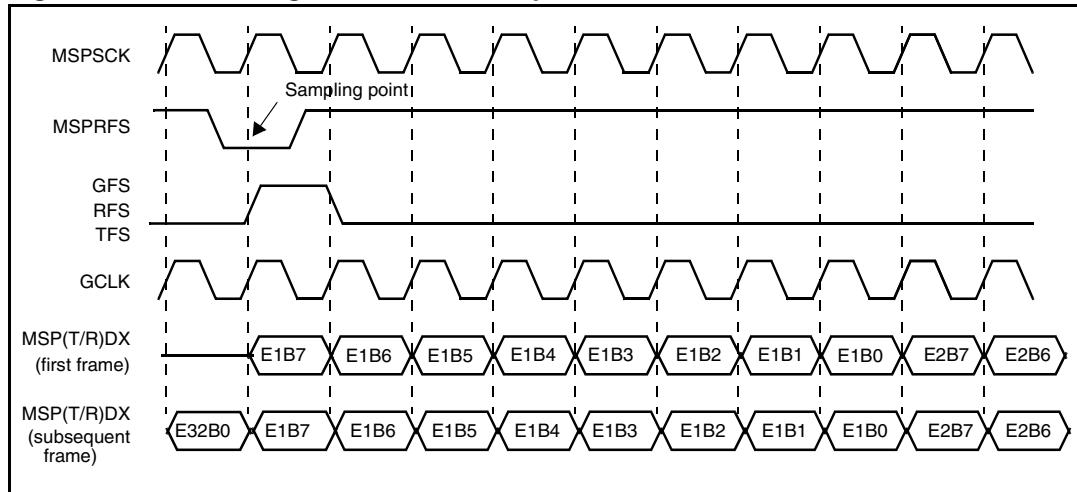
In *Figure 109*, there is a 1-bit data delay. Regardless of the data delay, transmission can occur without gaps. The last data bit of the previous (last) element in phase 2 is immediately followed by the first data bit of the first element in phase 1 of the next data frame.

Figure 109. Bit timing near frame synchronization

STBus single-rate clock example

Figure 110 shows the MSP timing required for a Mitel STBus with a single-rate clocking scheme. The MSP is configured as follows:

- single-phase frame,
- 32 elements per frame,
- 8 bits per element,
- no data delay,
- active-low frame synchronization signal,
- an external clock (MSPSCK) drives the sample rate generator clock (GCLK), which is synchronized with the external frame synchronization signal input on MSPRFSx,
- sample rate generator drives the internal transmit and receive clocks,
- Internal synchronization signals GFS, TFS and RFS are generated on the rising edge of MSPSCK.

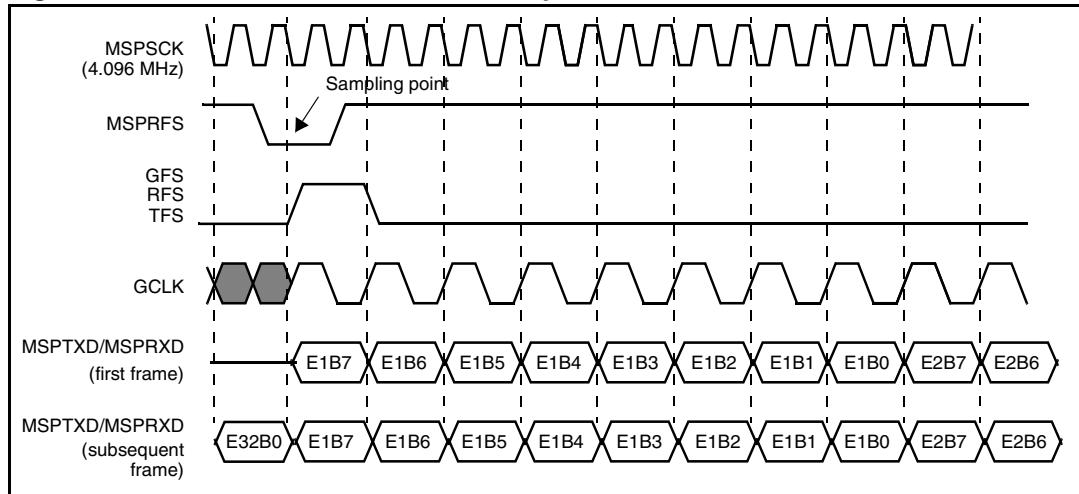
Figure 110. STbus single-rate clock example

The rising edge of MSPSCKx detects MSPRFS. This external frame synchronization signal re-synchronizes the internal clocks and generates the frame synchronization signal for internal use. The internal frame synchronization signal is generated such that it is wide enough to be detected on the falling edges of the internal clocks.

STBus double-rate clock example

Figure 111 shows the MSP timing required for a Mitel STBus with a double-rate clocking scheme. The MSP runs at the maximum frame frequency and is configured as follows:

- single-phase frame,
- 32 elements per frame,
- 8 bits per element,
- no data delay,
- active-low frame synchronization signal,
- external clock MSPSCK drives the sample rate generator clock (GCLK), which is synchronized with the external frame synchronization signal input on MSPRFSx,
- sample rate generator drives the internal transmit and receive clocks,
- internal synchronization signals GFS, TFS and RFS are generated on the falling edge of MSPSCK,
- frequency of receive clock is half of MSPSCK frequency.

Figure 111. STbus double-clock rate example

25.8 MSP signals

25.8.1 Interrupt signals

Each MSP has eight maskable interrupts, and a combined interrupt.

Table 115. MSP interrupts

Signal	I/O	Description
MSPTXINTR	O	Transmit FIFO service request (active high).
MSPRXINTR	O	Receive FIFO service request (active high).
MSPTEINTR	O	Transmit error (active high).
MSPREINTR	O	Receive error (active high).
MSPTFSINTR	O	Transmit frame synchronization (active high).
MSPRFSINTR	O	Receive frame synchronization (active high).
MSPINTR	O	Combined interrupt generated as an OR function of the eight maskable interrupts (active-high). This interrupt is output to the vectored interrupt controller (VIC).
MSPTFOINTR	O	Transmit FIFO not full (active high).
MSPRFOINTR	O	Receive FIFO not empty (active high).

Provision of the individual outputs as well as a combined interrupt output allows use of either a global interrupt service routine or modular device drivers to handle interrupts.

The status of the individual interrupt sources can be determined in software.

Transmit FIFO service request (MSPTXINTR)

The transmit interrupt is asserted high when one of the following conditions occurs.

- The transmit FIFO is enabled, and contains four words or less.
- The interrupt is cleared by writing to the FIFO until it holds at least four words, or by clearing the interrupt in software.
- The transmit FIFO is disabled and contains no data.
- The interrupt is cleared by a single write to the FIFO.

This interrupt is based on a level comparison, rather than on a level transition.

Receive FIFO service request (MSPRXINTR)

The receive interrupt is asserted high when one of the following conditions occurs.

- The receive FIFO is enabled and contains four or more words.
- The interrupt is cleared by reading data from the FIFO until it holds less than four words.
- The receive FIFO is disabled and contains data.
- The interrupt is cleared by a single read of the FIFO.

This interrupt is based on a level comparison, rather than on a level transition.

Transmit error (MSPTEINTR)

The transmit error interrupt is asserted high when one of the following errors occurs in the transmitter:

- transmit underrun error; see [page 312](#).
- transmit frame synchronization error; see [page 312](#),

The cause of the interrupt can be determined in software. The interrupt can be cleared in software.

Receive error (MSPREINTR)

The receive error interrupt is asserted when one of the following errors occurs in the receiver:

- receive overrun error; see [page 312](#).
- receive frame synchronization error; see [page 313](#),

The cause of the interrupt can be determined in software. The interrupt can be cleared in software.

Transmit frame synchronization (MSPTFSINTR)

The transmit frame synchronization interrupt is asserted high when a transmit frame synchronization signal is detected.

This interrupt can operate when the transmitter is disabled: the appropriate source and polarity of frame synchronization can still be selected. So, even when the transmitter is disabled, the frame synchronization signal MSPTFS_x is synchronized to the CPU clock and sent to the CPU in the form of the MSPTFSINTR signal. This means that the CPU can wait for the detection of a new frame synchronization signal, after which it can safely enable the transmitter. The interrupt can be cleared in software.

Receive frame synchronization (MSPRFSINTR)

The receive frame synchronization interrupt is asserted high when a receive frame synchronization signal is detected.

This interrupt can operate when the receiver is disabled: the appropriate source and polarity of frame synchronization can still be selected. So, even when the receiver is disabled, the frame synchronization signal MSPRFS_x is synchronized to the CPU clock and sent to the CPU in the form of the MSPRFSINTR signal. This means that the CPU can wait for the detection of a new frame synchronization signal, after which it can safely enable the receiver.

This interrupt can be cleared in software.

Transmit FIFO not full (MSPTFOINTR)

The MSPTFOINTR interrupt is asserted high while the transmit FIFO is not full.

Receive FIFO not empty (MSPRFOINTR)

The MSPRFOINTR interrupt is asserted high while the receive FIFO is not empty.

25.8.2 Interface signals

The MSP signals are summarized in [Table 116](#).

Table 116. MSP signals

Signal	I/O	Description
MSPTXDO	O	MSP0 transmitted serial data.
MSPTFS0	I/O	MSP0 transmit frame sync. This is an output or an input, depending on the MSP0 transmitter configuration.
MSPTCK0	I/O	MSP0 transmit serial bit clock. This is an output or an input, depending on the MSP0 transmitter configuration.
MSPRXD0	I	MSP0 received serial data.
MSPRFS0	I/O	MSP0 receive frame sync. This is an output or an input, depending on the MSP0 receiver configuration.
MSPRCK0	I/O	MSP0 receive serial bit clock. This is an output or an input, depending on the MSP0 receiver configuration.
MSPSCK0	I	MSP0 sample rate external clock.
MSPTXD1	O	MSP1 transmitted serial data.
MSPTFS1	I/O	MSP1 transmit frame sync. This is an output or an input, depending on the MSP1 transmitter configuration.
MSPTCK1	I/O	MSP1 transmit serial bit clock. This is an output or an input, depending on the MSP1 transmitter configuration.
MSPRXD1	I	MSP1 received serial data.
MSPSCK1	I	MSP1 sample rate external clock.
MSPTXD2	I/O	MSP2 transmitted serial data.
MSPTFS2	I/O	MSP2 transmit frame sync. This is an output or an input, depending on the MSP2 transmitter configuration.
MSPTCK2	I/O	MSP2 transmit serial bit clock. This is an output or an input, depending on the MSP2 transmitter configuration.
MSPRXD2	I	MSP2 received serial data.
MSPTXD3	I/O	MSP3 transmitted serial data.
MSPTFS3	I/O	MSP3 transmit frame sync. This is an output or an input, depending on the MSP3 transmitter configuration.
MSPTCK3	I/O	MSP3 transmit serial bit clock. This is an output or an input, depending on the MSP3 transmitter configuration.
MSPRXD3	I	MSP3 received serial data.

26 Fast IrDA controller (FIrDA)

The fast IrDA controller interfaces the platform to an off-chip infrared transceiver. It performs the modulation and demodulation of infrared signals, and the wrapping of the IrLAP frames.

26.1 Features

Supported standards

- IrDA serial infrared physical layer specification (IrPHY) version 1.3 (see note below)
- IrDA link access protocol (IrLAP) version 1.1 (see note below)

Supported infrared modes and baud rates

- serial infrared (SIR): 9.6 Kbit/s, 19.2 Kbit/s, 38.4 Kbit/s, 57.6 Kbit/s, and 115.2 Kbit/s
- medium infrared (MIR): 576 Kbit/s, and 1.152 Mbit/s
- fast infrared (FIR): 4 Mbit/s

Integrated functionality

- half-duplex infrared frame transmission and reception
- 16-bit CRC algorithm for SIR and MIR
- 32-bit CRC algorithm for FIR
- bit and character stuffing
- preamble, start, and stop flag generation
- RZI and 4PPM modulation and demodulation
- synchronization by means of a DPLL in FIR mode
- bus interface:
 - 32-bit register AMBA APB interface,
 - input or output FIFO (8 x 32-bit word depth; that is, 32 characters deep),
 - payload data transfer controllable by CPU or DMA controller.

Transceiver interface

- compliant with all IrDA transceivers.
- configurable polarity of transmit and receive signals.

Note:

The full titles of the supported IrDA standards specifications are:

Infrared Data Association (IrDA), IrDA Serial Infrared Physical Layer Specification (IrPHY) Version 1.3, October 15, 1998.

Infrared Data Association (IrDA), IrDA Link Access Protocol (IrLAP) Version 1.1, June 16, 1996.

26.2 Hardware overview

The fast IrDA controller consists of six main units, plus a bus interface. The units are:

- wrapper unit,
- modulation unit,
- synchronization unit,
- demodulation unit,
- baud generation unit,
- FIFO unit.

The functions of the different units are described below.

26.2.1 Wrapper unit

The wrapper unit marks the beginning and end of an IrLAP frame, and checks the reliability of the data. The wrapping schemes are specified in the IrLAP specification V1.1.

- If the controller is transmitting data, the wrapper unit appends a beginning flag, an end flag, and a cyclic redundancy check (CRC) flag to the IrLAP frame. This results in the transmit (Tx) frame which drives the modulation unit.
- If the controller is receiving data, the wrapper unit retrieves the IrLAP frame from the receive (Rx) frame and examines the CRC.

The wrapper unit is described in more detail in [Section 26.5 on page 327](#).

26.2.2 Modulation unit

The modulation unit takes the Tx frame from the wrapper unit and modulates the bits to create the Tx signal that drives the infrared transceiver. The modulation schemes used are:

- four pulse position modulation (4PPM) for FIR,
- return-zero-inverted (RZI) for SIR and MIR.

The modulation schemes are specified in the IrPHY specification V1.3. The modulation unit is described in more detail in [Section 26.6 on page 332](#).

26.2.3 Synchronization unit

The synchronization unit detects and synchronizes the Rx signal from the infrared transceiver. It is described in more detail in [Section 26.7 on page 335](#).

26.2.4 Demodulation unit

The demodulation unit demodulates the synchronized Rx signal in order to retrieve the Rx frame. It is described in more detail in [Section 26.8 on page 335](#).

26.2.5 Baud rate generation unit

The baud rate generation unit creates the enable signals EN_SYMB and EN_PULSE.

- EN_SYMB determines the symbol rate during transmission. It is also used to sample the Rx signal during reception in SIR and MIR modes.
- EN_PULSE determines the pulse width during transmission.

The baud rate generation unit is described in more detail in [Section 26.9 on page 336](#).

26.2.6 FIFO unit

Since IrDA supports only half-duplex communication, a single FIFO is used for transmission and reception of data. The FIFO unit is described in more detail in [Section 26.10 on page 338](#).

26.3 Data flow overview

Data transfer and control comprises:

- data transfer and communication,
- IrDA protocol stack layers,
- IrLAP frames.

26.3.1 Data transfer and communication

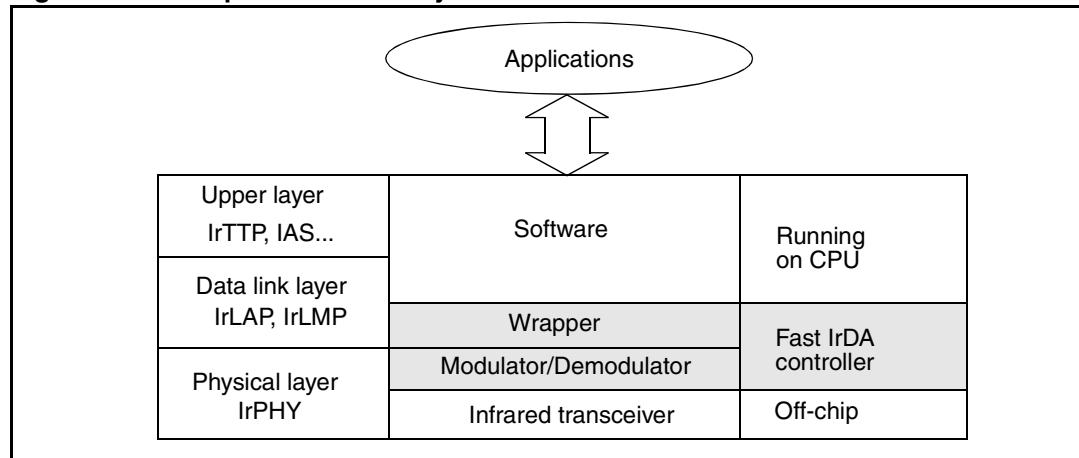
Communication between the fast IrDA controller and the internal bus is performed by means of 32-bit registers. The bus interface can easily be adapted to different bus systems.

The transfer of payload data between the fast IrDA controller and memory can either be controlled by the CPU or by the DMA controller. The interaction of the fast IrDA controller and the DMA controller is described in [Section 26.11 on page 340](#).

26.3.2 IrDA protocol stack layers

The core tasks that are implemented in the fast IrDA controller are illustrated in [Figure 112](#), along with the IrDA protocol stack layers.

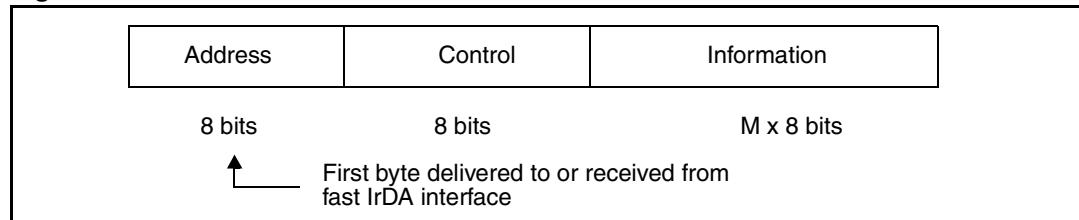
Figure 112. IrDA protocol stack layers



26.3.3 IrLAP frames

The input/output from/to the software are IrLAP frames, as shown in [Figure 113](#) (and specified in the IrLAP specification V1.1).

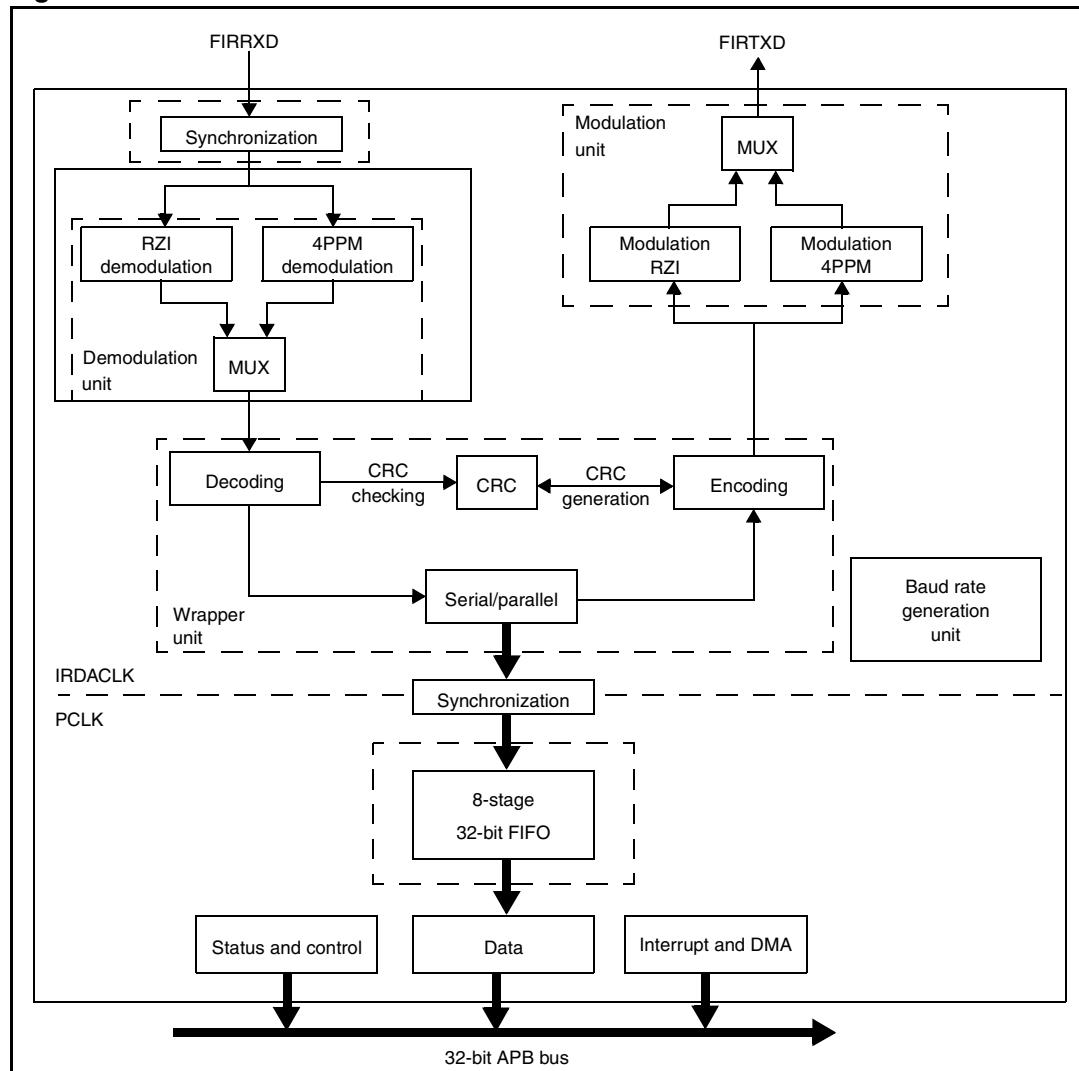
Figure 113. IrLAP frame



The input/output from/to the infrared transceiver are electrical signals as specified in the IrPHY specification V1.3.

The overall structure and data flow of the fast IrDA controller are shown in [Figure 114](#).

Figure 114. Fast IrDA architecture



26.4 Functional description

This section describes:

- the IrDA operating states,
- the IrDA clock domains.

26.4.1 Operating states

The fast IrDA controller has four different operational states:

- inactive,
- listening,
- receive,
- transmit.

Inactive state

In the inactive state, the controller cannot transmit or receive. From this state, it can be switched to the listening state.

Listening state

In the listening state, the controller waits for a signal to be received from the IrDA transceiver. The synchronization unit constantly scans for a rising edge of the active-high FIRRxD signal. If the signal is detected, the controller switches to the receive state and indicates this by generating a signal detected interrupt, SDINTR.

In the listening state, if the software indicates that it needs to transmit (and therefore copy data to the FIFO), the controller automatically switches to the transmit state. In the transmit state, the controller can transmit a frame. It switches back to the listening state when the frame has been completely transmitted. Two other events can cause the controller to come out of the transmit state, as follows.

- If the controller detects a transmission error, the transmission is aborted and the controller switches back to the listening state. It indicates this to the software by generating an FIINTR interrupt.
- The software can switch the controller directly to the inactive state. The controller then generates an FIINTR interrupt.

Receive state

In the receive state, the controller can receive a frame. Once the received frame has been completely fetched from the FIFO, the controller switches back to the listening state. Two other events can cause the controller to come out of the receive state, as follows.

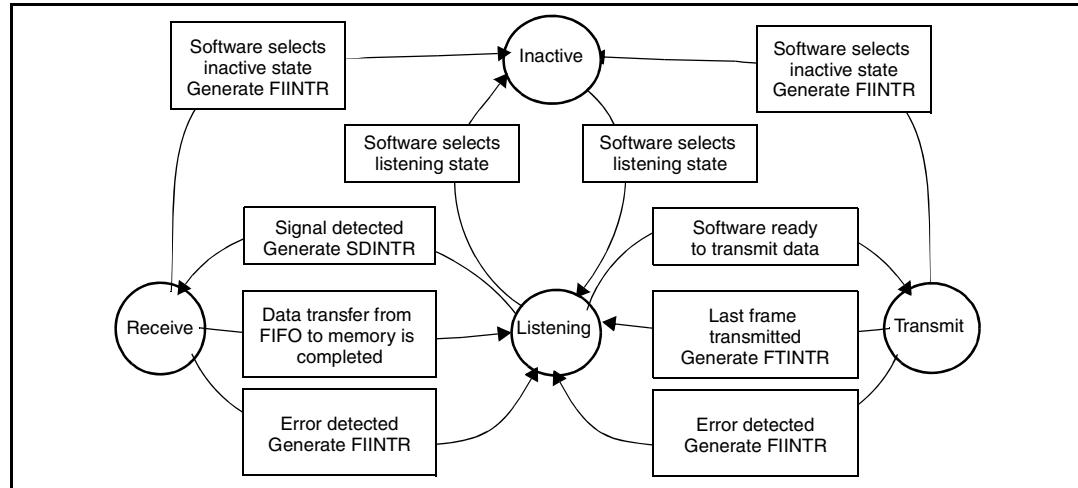
- If the controller detects an invalid frame, the reception is aborted and the controller switches back to the listening state. It indicates this to the software by generating a frame invalid interrupt, FIINTR.
- The software can switch the controller directly to the inactive state. The controller then generates an FIINTR interrupt.

Transmit state

While in the transmit state, it is not possible to receive data from the IrDA transceiver. Equally, while in the receive state, it is not possible to copy data into the FIFO (and transmit).

The above states are summarized in [Figure 115](#).

Figure 115. Top level state machine



[Table 117](#) shows which units are used in each operational state.

Table 117. Use of the fast IrDA controller units in different states

State	Sync unit	Modulation unit	Demod unit	Wrapper unit	FIFO unit	Baud rate generator unit
Inactive	Not used	Not used	Not used	Not used	Not used	Not used
Listening	Active	Not used	Not used	Not used	Not used	Not used
Reception	Active	Not used	Active	Active	Active	Active
Transmission	Active	Active	Not used	Active	Active	Active

26.4.2 Clock domains

While the quality requirements of the infrared signal in terms of frequency deviation and jitter put restrictions on the kernel clock that can be used, the fast IrDA controller provides a flexible bus interface by means of two independent clock domains.

- The controller is clocked by IRDACLK (IRDACLK frequency = 48 MHz).
- The FIFO is clocked by PCLK, which should have a frequency of at least 13 MHz.

26.5 Wrapper unit

The wrapper unit is introduced in [Section 26.2 on page 323](#).

It is active in the transmit and receive states. In the listening state, it is not used.

26.5.1 Transmission state

In the transmit state, the wrapper unit builds the Tx frame out of the IrLAP frame. The data is sent least significant bit (LSB) first. The encoding method depends on the infrared mode used.

Serial infrared (SIR)

- CRC generation

The raw data to be transmitted is scanned from the least significant to the most significant bit of each byte written into the parallel/serial converter. A 16-bit CRC-CCITT is calculated for the IrLAP frame and is appended to the end of the data.

The CRC is initialized to all ones. This allows detection of any missed or inserted zero bits at the beginning of a block (missed or inserted ones are still detected).

The one's complement of the CRC is transmitted rather than the CRC itself. This allows detection of slippage-type errors. The CRC is sent most significant bit (MSB) first.

The CRC-CCITT polynomial is defined as follows:

$$\text{CRC}(x) = x^{16} + x^{12} + x^5 + 1$$

The CRC is sent MSB first, to be consistent with other manufacturers.

For example, if the four bytes 0xF5, 0xF1, 0xA7 and 0xC1 are built into the IrLAP frame to be sent out then the CRC bytes are 0x38 and 0x06, which are added to the IrLAP frame.

IrLAP frame

0xC1 11000001 MSB	0xA7 10100111	0xF1 11110001	0xF5 11110101 LSB
-------------------------	------------------	------------------	-------------------------

IrLAP frame/CRC

0x06 00000110 MSB	0x38 00111000	0xC1 11000001	0xA7 10100111	0xF1 11110001	0xF5 11110101 LSB
-------------------------	------------------	------------------	------------------	------------------	-------------------------

- Character stuffing

Data bytes that would otherwise be interpreted as flags or other control characters are transformed into non-flag/control characters prior to transmission. A control escape (CE) byte is defined as 0x7D. For each byte encountered with the same value as a flag or a CE byte (0xC0, 0xC1 or 0x7D), the encoder performs the following tasks:

- inserts a control escape (CE) byte before the byte,
- complements bit 5 of the byte.

For example:

IrLAP frame/CRC

0x06 00000110 MSB	0x38 00111000	0xC1 11000001	0xA7 10100111	0xF1 11110001	0xF5 11110101 LSB
-------------------------	------------------	------------------	------------------	------------------	-------------------------

IrLAP frame/CRC/CE

0x06 00000110 MSB	0x38 00111000	0xE1 11100001	0x7D 01111101	0xA7 10100111	0xF1 11110001	0xF5 11110101 LSB
-------------------------	------------------	------------------	------------------	------------------	------------------	-------------------------

- Beginning-of-frame (BOF) and end-of-frame (EOF)

The BOF and EOF flags enclose the frame. The BOF flag serves as a reference for the position of the payload data. The EOF flag delimits the end of the CRC and marks the end of frame.

The first BOF flag before the data is obligatory and is defined as 0xC0. Additional consecutive BOF flags, referred to as XBOF flags, can be optionally added before the

BOF flag. An additional XBOF flag is defined as 0xFF. The number of additional XBOF flags sent with each frame can be programmed. The maximum number of additional flags is 48.

The EOF flag is defined as 0xC1.

For example:

IrLAP frame/CRC/CE

0x06	0x38	0xE1	0x7D	0xA7	0xF1	0xF5
00000110	00111000	11000001	01111101	10100111	11110001	11110101
MSB						LSB

Tx bytes (with one additional XBOF flag)

0xC1	0x06	0x38	0xE1	0x7D	0xA7	0xF1	0xF5	0xC0	0xFF
11000001	00000110	00111000	11000001	01111101	10100111	11110001	11110101	11000000	11111111
MSB						LSB			

Note that the receiver can also receive several consecutive BOF flags (0xC0) after optional XBOF (0xFF) flags. The extra BOF flags are skipped.

- Start and stop bits

Each Tx byte is enclosed by a start bit and a stop bit. This is necessary to be compatible with IrDA controllers that use a UART. The start bit is defined as 0 and the stop bit is defined as 1.

For example:

Tx bytes

0xC1	0x06	0x38	0xE1	...	0xF1	0xF5	0xC0	0xFF
11000001	00000110	00111000	11100001	...	11110001	11110101	11000000	11111111
MSB						LSB		

Tx frame

11100000010	1000001100	1001110000	1111000010	...	1111100010	1111101010	1110000000	1111111110
Last bits to be transmitted						First bits to be transmitted		

Medium infrared (MIR)

- CRC generation: The CRC generation is the same as for the SIR, described above.
- Bit stuffing: A 0 is inserted after five consecutive 1s are transmitted in order to distinguish the flag from the data. The 0 insertion is done for every field except the flags.

Using the same example as for SIR:

IrLAP frame/CRC

0x06	0x38	0xC1	0xA7	0xF1	0xF5
00000110	00111000	11000001	10100111	11110001	11110101
MSB					LSB

IrLAP frame/CRC/zero insertion

0000 0110	00111000	11000001	10100110	11111000	01111101	01
Last bits to be transmitted					First bits to be transmitted	

- Beginning (STA) and end (STO) flags: Finally, two start (STA) flags and one stop (STO) flag are appended to the Tx data. The STA and STO both take the value 0x7E.

Additional consecutive STA flags are optional. The number of additional STA flags sent with each packet can be programmed. The maximum number of additional flags is 48.

Note that there can only be 0 or 2 additional STA flags in MIR mode.

Using the same example as for SIR:

IrLAP frame/CRC/zero insertion

0000011000111000110000011010011011110000111110101						
Last bits to be transmitted					First bits to be transmitted	

Tx Frame (with no additional beginning flags)

0x7E	0000011000111000110000011010011011110000111110101	0x7E	0x7E
01111110	0000011000111000110000011010011011110000111110101	01111110	01111110
Last bits to be transmitted			First bits to be transmitted

Fast infrared (FIR)

In fast infrared mode, the wrapper unit generates a 32-bit CRC. The preamble, start and stop flags in this mode are described in the modulation unit section (see [Section 26.6 on page 332](#)).

- CRC generation: For a FIR transmission, a 32-bit CRC code is applied using the CRC32 polynomial:

$$\text{CRC}(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The CRC32 result for each packet is treated as four data bytes, where each byte is encoded in the same way as payload data. Payload data bytes are input to this calculation in LSB first format. The 32-bit CRC register is preset to all 1s prior to calculation of the CRC on the transmit data stream.

For example, if the two data bytes 0x1B and 0xA4 were to be sent out in sequence then the four bytes of the CRC would be 0x94, 0xBE, 0x54 and 0x39.

IrLAP frame					
Tx frame					
0xA4			0x1B		
10100100			00011011		
MSB			LSB		
0x39	0x54	0xBE	0x94	0xA4	0x1B
00111001	01010100	10111110	10010100	10100100	00011011
Last bits to be transmitted					First bits to be transmitted

26.5.2 Reception state

In the receive state, the wrapper unit retrieves the IrLAP frame and the CRC bytes out of the Rx frame. If an error is detected in the demodulation unit (for example, the receipt of an invalid symbol) or in the FIFO unit (for example, a FIFO overflow), this decoding process is aborted. After decoding, the IrLAP and CRC bytes are shifted to the FIFO unit.

Serial infrared (SIR)

In SIR mode, the start and stop bits are removed from the Rx frame in order to retrieve the Rx bytes. If the stop bit is not 1 then an frame invalid interrupt (FIINTR) is generated and the wrapper unit switches to the listening state.

Note: *There can be transmission pauses between stop and start bits.*

The wrapper unit recognizes the beginning and the end of an Rx frame by means of the beginning-of-frame (BOF) flag (0xC0) and end-of-frame (EOF) flag (0xC1). The occurrence of the first BOF flag is indicated to the software by a frame detected interrupt (FDINTR). Multiple flags are treated as a single flag.

Note: *In the receive state, an 0xFF is treated as any byte. After the first BOF flag is received, any extra BOF flags are skipped.*

As soon as the Rx bytes are detected, the wrapper unit starts decoding in order to retrieve the bytes of the IrLAP frame. Incoming CE flags are discarded, and the IrLAP and CRC bytes are moved to the FIFO via the serial/parallel converter.

The IrLAP frame and the received CRC field are also shifted into the CRC unit, which calculates a new CRC on all the protected bits and the received CRC field. After receiving an EOF flag, it compares the new CRC to the known constant 0xF0B8. In the case of a mismatch, the received IrLAP frame is no longer valid and an FIINTR interrupt is generated. After a successful CRC, the frame complete signal of the FIFO is set.

Medium infrared (MIR)

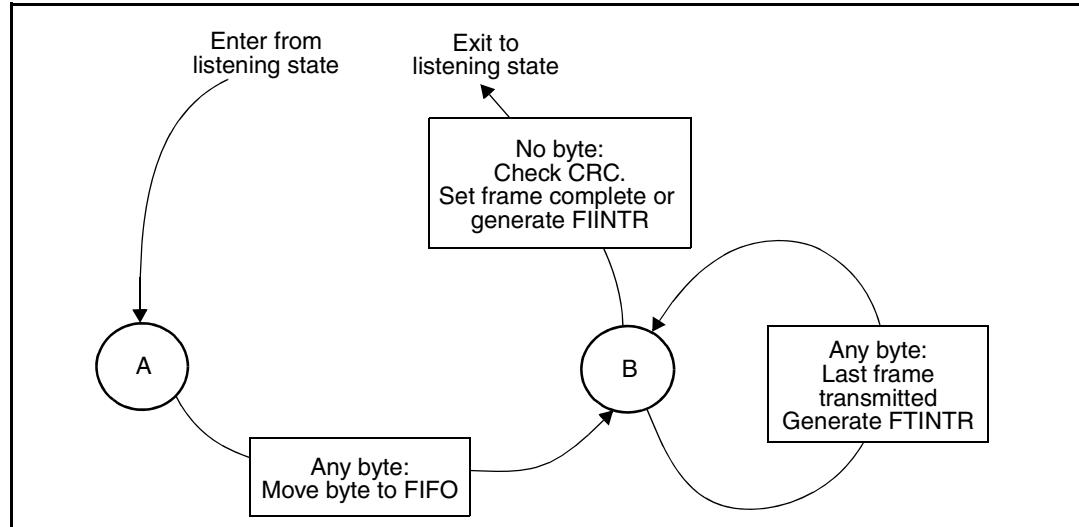
For MIR mode, the decoding process is very similar to that of SIR. The CRC is also calculated in the same way. The occurrence of the first STA flag is indicated to the software by a frame detected interrupt (FDINTR).

A frame can be aborted for a number of reasons; due to blocking of the IR transmission path in the middle of the frame, a random introduction of infrared noise or intentional termination by the transmitter. Regardless of what caused the aborted frame, the wrapper unit treats a frame as an aborted frame when seven or more consecutive 1s (no optical signal) are received. The abort terminates the frame immediately without processing the CRC or end flag. In addition, a frame invalid interrupt (FIINTR) is generated.

Fast infrared (FIR)

Figure 116 shows the state machine of the wrapper unit in the case of receiving an FIR frame.

Figure 116. State machine for receiving FIR frames



The state machine in FIR mode is quite simple due to the lack of beginning and end flags. The calculated 32-bit CRC field is compared to the constant 0xDEBB20E3.

26.6 Modulation unit

The modulation unit is introduced in [Section 26.2 on page 323](#).

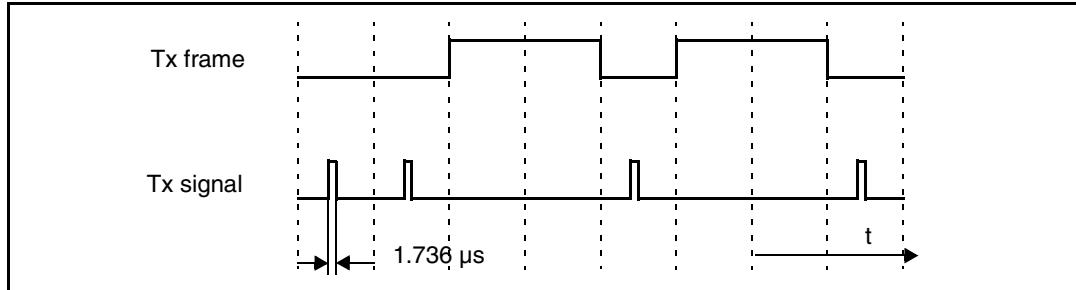
The modulation unit is active in the transmit state only. It is responsible for the modulation of the Tx frames from the wrapper unit in order to generate the Tx signal. The polarity of the Tx signal can be configured; that is, whether a low or high state of the Tx signal corresponds to the LED 'on' state.

The Tx signal is generated by means of the enable signals EN_SYMB and EN_PULSE. These signals are generated by the baud rate generation unit, as described in [Section 26.9 on page 336](#), and they determine the baud rate of the Tx signal.

Once a frame has been completely transmitted, a frame transmitted interrupt (FTINTR) is generated and the controller switches back to the listening state.

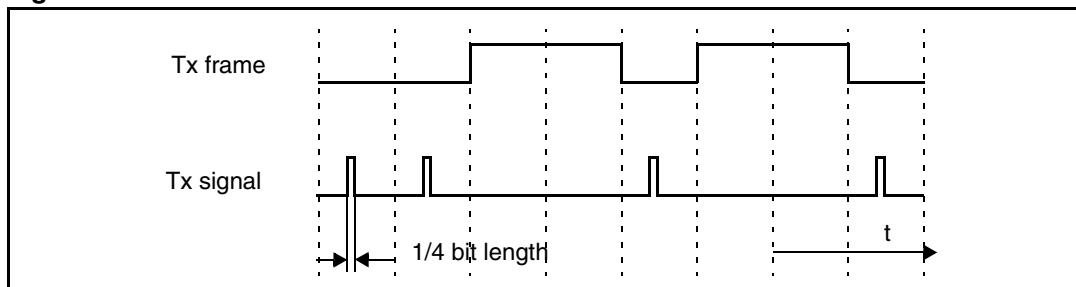
26.6.1 Serial infrared (SIR)

In SIR mode, a return-to-zero-inverted (RZI) modulation is employed. This means that for a 0 bit in the Tx frame, a high pulse is generated and for a 1 bit in the Tx frame, no pulse is generated. For bit rates up to 115.2 Kbit/s, a pulse duration of 1.736 µs is used. For example, if the byte to be transmitted is 0x6C then 01101100 is shifted into the modulation unit with LSB first. *Figure 117* illustrates the signal actually transmitted for data rates of up to 115.2 Kbit/s.

Figure 117. RZI modulation for SIR

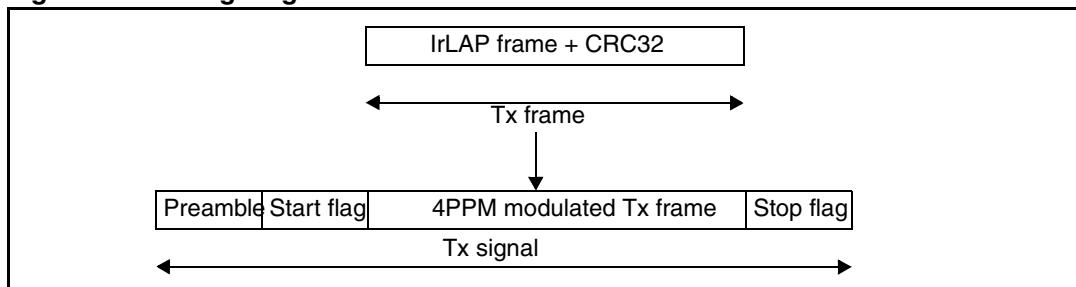
26.6.2 Medium infrared (MIR)

MIR mode also uses an RZI modulation, but the pulse length is 1/4 of a bit. For example, if the byte to be transmitted is 0x6C then 01101100 is shifted into the modulation unit with LSB first. [Figure 118](#) illustrates the signal actually transmitted for 0.576 and 1.152 Mbit/s rates.

Figure 118. RZI modulation for MIR

26.6.3 Fast infrared (FIR)

In FIR mode, a four pulse position modulation (4PPM) is used. Additionally, a preamble (PA), a start flag (STA) and a stop flag (STO) are added, as shown in [Figure 119](#). These flags are detailed later in this subsection.

Figure 119. Tx signal generation at 4 Mbit/s

4PPM encoding is achieved by defining a data symbol duration (D_t) and subsequently dividing D_t into four equal time slices, called chips. A chip marks the time-space when an infrared pulse can be sent. Within a data symbol only one pulse is allowed, thus four different data symbols exist, representing the four possible combinations of two bits. The data symbols used are listed in [Table 118](#) along with the corresponding data bit pairs.

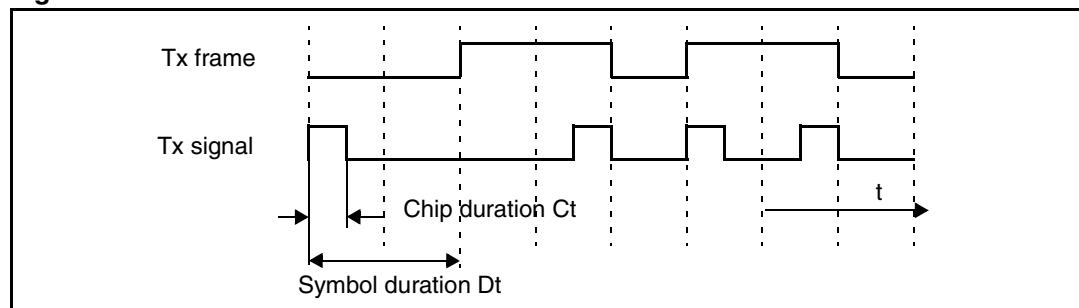
Table 118. 4PPM encoding scheme

Data bit pair MSB LSB	4PPM data symbol sent first to last
00	1000
01	0100
10	0010
11	0001

The 4PPM data encoding scheme defines only the legal encoded payload data symbols. All other chip combinations are, by definition, illegal symbols for encoded payload data. Some of these illegal symbols are used in the definition of the preamble, start flag and stop flag fields because they are unambiguously not data.

With a bit rate of 4 Mbit/s, the resulting data symbol duration D_t is 500 ns and the resulting chip duration C_t is 125 ns.

For example, if the byte to be transmitted is 0x6C then 01101100 is shifted into the modulation unit with LSB first. [Figure 120](#) illustrates the signal actually transmitted at a rate of 4 Mbits.

Figure 120. 4PPM for FIR

- Preamble field definition

The preamble field consists of sixteen repeated transmissions of the following stream of symbols.

1000	0000	1010	1000
------	------	------	------

First chips to be transmitted Last chips to be transmitted

- Start flag definition

The start flag consists of a single transmission of the following stream of symbols.

0000	1100	0000	1100	0110	0000	0110	0000
------	------	------	------	------	------	------	------

First chips to be transmitted Last chips to be transmitted

Note that there is only one start flag in FIR mode. Thus, if additional start flags have been configured, they are ignored by the controller.

- Stop flag definition

The stop flag consists of a single transmission of the following stream of symbols.

0000	1100	0000	1100	0000	0110	0000	0110
------	------	------	------	------	------	------	------

First chips to be transmitted Last chips to be transmitted

26.7 Synchronization unit

The synchronization unit is introduced in [Section 26.2 on page 323](#).

This unit synchronizes the Rx signal from the off-chip IrDA infrared transceiver. To synchronize the signal, the Rx signal is sampled at the rising edge of the clock IRDACLK.

If the synchronization unit detects activity of the Rx signal while in the listening state, the fast IrDA controller switches to the receive state. In addition, a signal detected interrupt (SDINTR) is generated.

If the synchronization unit detects no activity of the Rx signal for more than 10 ms in the receive state, the fast IrDA controller switches to the listening state. In addition, a frame invalid interrupt (FIINTR) is generated. This behavior corresponds to the handling of a frame abort; the reception abort timer is programmable.

26.8 Demodulation unit

The demodulation unit is introduced in [Section 26.2 on page 323](#).

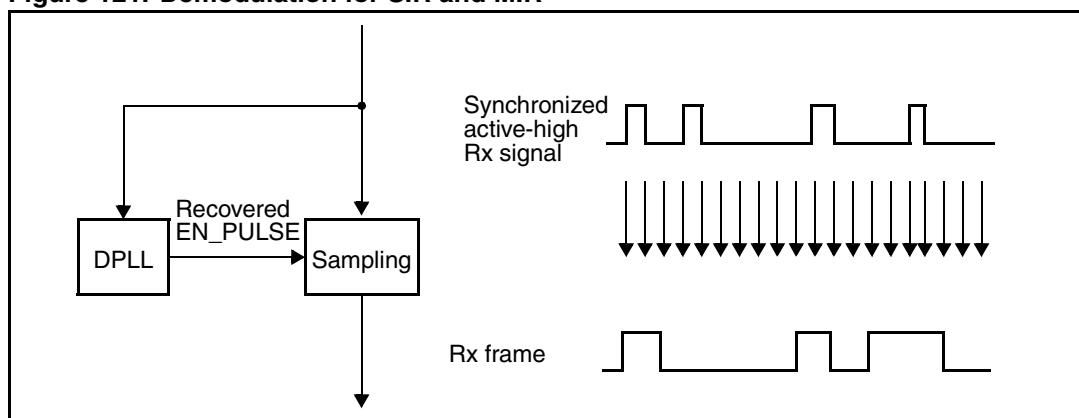
This unit is active in the receive state only. The demodulation unit is responsible for the demodulation of the synchronized, active-high Rx signal from the synchronization unit, in order to obtain the Rx frame.

The unit must be configured according to the polarity of the Rx signal; that is, whether a low or high state of the Tx signal indicates the LED ‘on’ state.

26.8.1 Serial and medium infrared (SIR and MIR)

The RZI demodulation for SIR and MIR is shown in [Figure 121](#).

Figure 121. Demodulation for SIR and MIR

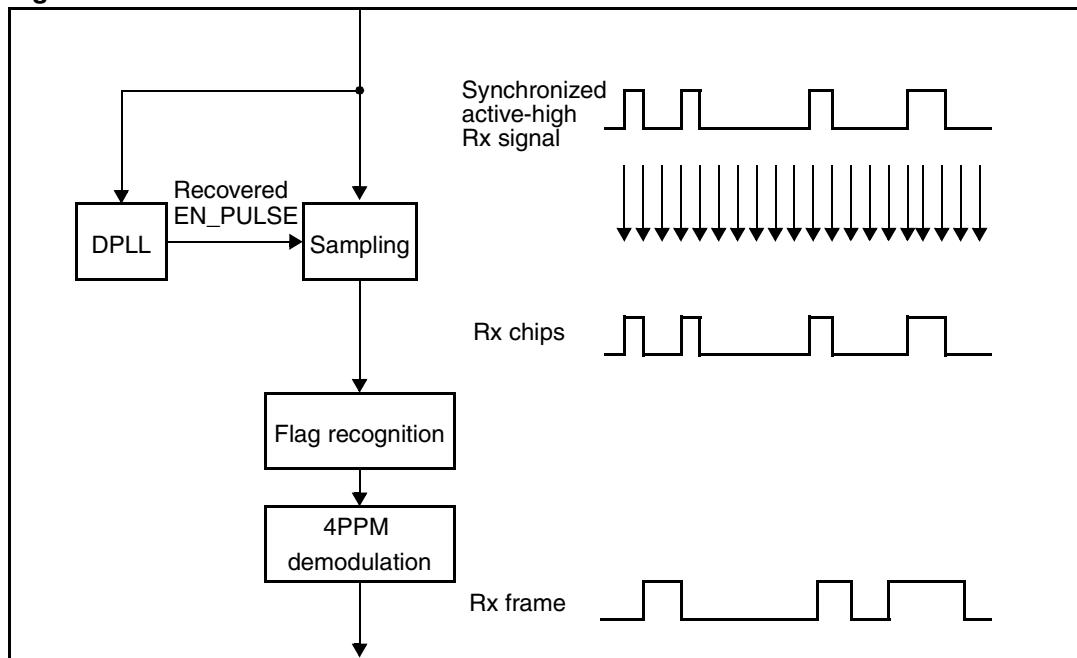


The pulses of the synchronized active-high signal are extended in order to eliminate the influence of jitter. The signal is then sampled with the signal EN_SYMB. The phase of EN_SYMB is determined by the first rising edge of the Rx signal and is readjusted with every rising edge.

26.8.2 Fast infrared (FIR)

The demodulation for FIR is shown in [Figure 122](#).

Figure 122. Demodulation for FIR



The preamble field (PA) of the synchronized Rx signal is used by the receiver to establish a phase lock. To achieve this, the 8 MHz signal EN_PULSE is recovered from the Rx signal by means of a DPLL (digital phase locked loop). The incoming Rx signal is then sampled with the recovered enable signal EN_PULSE.

During the PA, the receiver begins to search for the start flag, STA, in order to establish symbol synchronization. If an STA flag is received correctly, the receiver begins to demodulate the 4PPM modulated Rx frame and generates a frame detected interrupt (FDINTR). The receiver continues to receive and demodulate until the stop flag, STO, is recognized. The STO flag indicates the end of the frame.

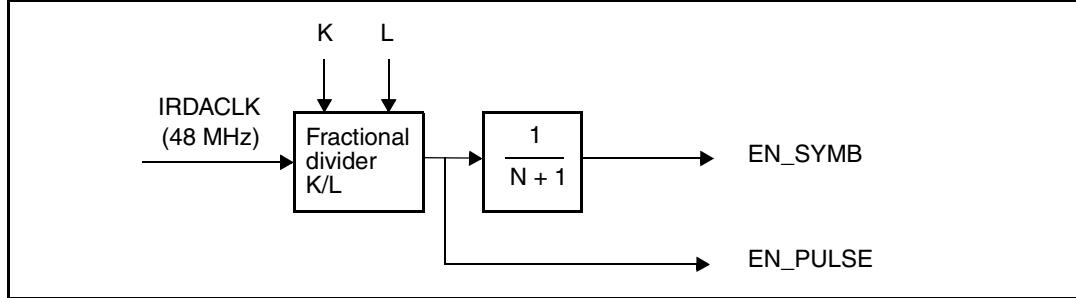
26.9 Baud rate generation unit

The baud rate generation unit is introduced in [Section 26.2 on page 323](#).

This unit creates the enable signals EN_SYMB and EN_PULSE.

- The EN_SYMB signal is used as follows.
 - During transmission, the signal EN_SYMB determines the symbol rate, which is defined as the number of transmitted symbols per second.
 - In the receive state for SIR and MIR, the EN_SYMB signal is used to sample the synchronized and inverted Rx signal.
- The EN_PULSE signal is used to create the pulses of the Tx signal during transmission.

The signals EN_SYMB and EN_PULSE are derived from the clock IRDACLK using cascaded clock dividers, as shown in [Figure 123](#).

Figure 123. Baud rate generation

Thus, the resulting frequencies of EN_PULSE and EN_SYMB are:

$$f_{EN_PULSE} = f_{IRDACLK} * K/L$$

$$f_{EN_SYMB} = f_{EN_PULSE} / (N+1)$$

The values of K, L and N can be configured in software.

The fractional divider causes jitter with a maximum of $1/(2*f_{IRDACLK})$, which is equal to 10.417 ns for SIR and MIR. This value is within the tolerances of the IrPHY specification.

26.9.1 Serial infrared (SIR)

Since for each SIR symbol one bit is transmitted, the bit rate and the symbol rate are equal for SIR. Thus, the baud rate generation unit must produce the following symbol rates, f_{EN_SYMB} :

9.6 kHz, 19.2 kHz, 38.4 kHz, 57.6 kHz and 115.2 kHz.

In the case of a SIR transmission, a pulse duration of 1.736 μ s is used. Thus, the baud rate generation unit must create a pulse rate f_{EN_PULSE} of 576 kHz.

Table 119 provides a list of settings of K, L and N+1 for $f_{IRDACLK}$ equal to 48 MHz.

Table 119. Settings of K, L and N+1 for SIR ($f_{IRDACLK} = 48$ MHz)

Bit rate (Kbit/s)	f_{EN_PULSE} (kHz)	f_{EN_SYMB} (kHz)	K	L	N+1
9.6	576	9.6	3	250	60
19.2	576	19.2	3	250	30
38.4	576	38.4	3	250	15
57.6	576	57.6	3	250	10
115.2	576	115.2	3	250	5

26.9.2 Medium infrared (MIR)

Since for each MIR symbol one bit is transmitted, the bit rate and the symbol rate are equal for MIR. Thus, the baud rate generation unit must produce the following symbol rates, f_{EN_SYMB} : 576 kHz and 1.152 MHz.

In the case of a MIR transmission, the pulse duration is a quarter of the symbol duration. Thus, the baud rate generation unit must create a pulse rate f_{EN_PULSE} equal to $4*f_{EN_SYMB}$.

Table 120 provides a list of settings of K, L and N + 1 for $f_{IRDACLK}$ equal to 48 MHz.

Table 120. Settings of K, L and N +1 for MIR ($f_{IRDACLK} = 48 \text{ MHz}$)

Bit rate (Kbit/s)	f_{EN_PULSE} (kHz)	f_{EN_SYMB} (kHz)	K	L	N +1
576	2 304	576	6	125	4
1152	4 608	1 152	12	125	4

26.9.3 Fast infrared (FIR)

Since for each FIR symbol two bits are transmitted, the symbol rate is one half of the bit rate for FIR. Thus, the baud rate generation unit must produce a symbol rate f_{EN_SYMB} of 2 MHz.

In the case of a FIR transmission, the pulse duration is a quarter of the symbol duration. Thus, the baud rate generation unit must create a pulse rate f_{EN_PULSE} equal to $4*f_{EN_SYMB}$.

Table 121 provides the settings of K, L and N + 1 for $f_{IRDACLK}$ equal to 48 MHz.

Table 121. Settings of K, L and N +1 for FIR ($f_{IRDACLK} = 48 \text{ MHz}$)

Bit rate (Kbit/s)	f_{EN_PULSE} (kHz)	f_{EN_SYMB} (kHz)	K	L	N +1
4000	8000	2000	1	6	4

26.10 FIFO unit

The FIFO unit is introduced in [Section 26.2 on page 323](#). This unit controls data transfers between a peripheral and memory, and buffers data that is being received and transmitted.

The fast IrDA controller generates the following request signals to control data transfers to and from memory:

- Burst request signal (BREQINTR): Requests a burst transfer of a programmed number of words.
- Last burst request signal (LBREQINTR): Requests a final burst transfer.
- Single request signal (SREQINTR): Requests a transfer of a single word.
- Last single request signal (LSREQINTR): Requests a final single-word transfer.

These signals can either be used as interrupt requests or as DMA requests. They are reset on the occurrence of the request clear signal (REQCLR), which is either configured in software or generated by the DMA controller.

The burst size is programmable. It is typically set to 4 words for a FIFO size of 8 words, and this is the default value.

For both transmission and reception, the FIFO unit packs bytes into 32-bit words, using the little-endian convention.

Note: *The same request signals are used for transmission and reception. The software is responsible for providing a proper interrupt service or for correctly programming the DMA channel.*

26.10.1 Transmit state

The software indicates its need to transmit data by specifying the frame size to the fast IrDA controller. The controller then switches to the transmit state and the FIFO unit asserts burst requests (using BREQINTR) until the amount of data still to be transferred is less than or equal to the burst size. At this point:

- If the amount of remaining data is equal to the burst size, a last burst request is issued, (using LBREQINTR).
- If the amount of remaining data is less than the burst size, single requests are issued (using SREQINTR) until the last data item is ready, and then a last single request is issued (using LSREQINTR).

Note: Since the size of the frame to be transmitted is not necessarily a multiple of 4, the last word must be filled up with dummy (upper) bytes. The controller transmits only the valid bytes of the final word.

The data is buffered in an 8-stage, 32-bit shift register before it is processed by the fast IrDA controller. The data valid signal indicates to the wrapper unit if there are still bytes of the frame currently in the buffer that are ready to be moved to the wrapper unit. If a FIFO underflow occurs before all bytes of a frame have been shifted into the FIFO then a frame invalid interrupt (FIINTR) is generated, the transmission is aborted and all pending bytes in the peripheral are discarded.

The next frame to be transmitted can be copied into the buffer when all bytes of the current frame have been completely transferred to the wrapper unit.

26.10.2 Receive state

The bytes received in a single frame are packed into a 32-bit word in the FIFO where the data is buffered. The received bytes are counted by a 12-bit counter. The counter value can be read by software. If the number of received bytes is greater than the programmed maximum for the number of bytes that can be received, the current frame becomes invalid. This is indicated by a frame invalid interrupt (FIINTR). This interrupt is also generated if a buffer overflow occurs.

When all the data of the current received frame has been moved into the buffer, the frame complete signal is asserted. The bytes of this frame must be moved out of the buffer by the software before the next received frame can be shifted into the buffer. If this is not done, the next frame received is completely discarded.

Note: Between any two frames there are start and stop flags. Due to these flags, there is always enough time for the software to flush the FIFO, even when the frames are sent back-to-back.

The buffered data can be moved out of the buffer at the full bus speed using DMA burst transfers. The FIFO unit asserts burst requests (using BREQINTR) until the amount of data still to be transferred is less than or equal to the burst size. At this point:

- if the amount of remaining data is equal to the burst size, a burst request is issued (using LBREQINTR),
- if the amount of remaining data is less than the burst size, single requests are issued (using SREQINTR) until the last word is ready, and then a last single request is issued (using LSREQINTR).

The occurrence of a frame invalid interrupt (FIINTR) during reception indicates that the received data has become invalid. The buffer content is then cleared without sending further

requests. Once the received frame has been completely read out of the buffer, the fast IrDA controller switches back to the listening state.

Note: *Since the size of the received frame is not necessarily a multiple of 4, the upper bytes of the last word can be invalid. The software must check for invalid bytes in the final word.*

Along with the IrLAP bytes, the CRC bytes are also transferred to memory. The CRC bytes can be double-checked by the software for the purpose of testing.

26.11 Interaction with the DMA controller

The platform allows DMA transfers between an IrDA peripheral and on-chip memory, in either direction. A single DMA channel is available.

The recommended configuration settings for a DMA transfer are given in [Table 122](#).

Table 122. DMA parameter recommendations

Parameter	Recommendations
Transfer size	Since the DMA controller is not the flow controller, the transfer size value is not used.
IrDA burst size	The IrDA burst size should be set to 4 words. Note that an IrDA burst size of 1 word is also supported.
Memory burst size	The memory burst size should be set to the IrDA burst size.
IrDA transfer width	The IrDA transfer width should be set to 32 bits.
Memory transfer width	The memory transfer width should be set to 32 bits.
IrDA address increment	The IrDA address should not be incremented after a transfer.
Memory address increment	The memory address should be incremented after each transfer.

26.12 IrDA signals

The IrDA signals consist of:

- interface signals,
- interrupts.

26.12.1 Interface signals

The signals of the fast IrDA controller are summarized in [Table 123](#).

Table 123. IrDA signals

Signal	I/O	Description
FIRRXD	I	IrDA SIR/MIR/FIR received serial data.
FIRTXD	O	IrDA SIR/MIR/FIR transmitted serial data.

26.12.2 Interrupts

There are eight individual maskable interrupt sources. All eight sources are combined into a single interrupt signal, IRDAINTR, which is the only interrupt signal from the controller that is routed to the platform's vectored interrupt controller (VIC). This combined IRDAINTR interrupt is an OR function of the individual masked sources (and is therefore asserted if any of the individual interrupts are enabled and asserted).

Table 124 provides a summary of the interrupts.

Table 124. Interrupt summary

Interrupt	Description
FDINTR	Frame detected interrupt: A frame has been detected during the receive state. FDINTR must have a lower priority than FIINTR.
FIINTR	<p>Frame invalid interrupt: In the receive state, the frame currently being received is invalid. This can be due to:</p> <ul style="list-style-type: none"> – a CRC error, – the receipt of an invalid flag, – a frame abort, – the receipt of an invalid symbol, – the receipt of a frame that is too long, – a FIFO overflow, – or an abort by software. <p>In the interrupt service routine, the software should discard the bytes of the current frame which have already been transferred to memory.</p> <p>In the transmit state, the current frame has not been sent completely. This can be due to a FIFO underflow or an abort by software.</p> <p>FIINTR must have a higher priority than FDINTR.</p>
SDINTR	Signal detected interrupt: A signal has been detected during the listening state.
FTINTR	Frame transmitted interrupt: A frame has been completely transmitted.
BREQINTR	Burst request interrupt: A burst transfer of a programmed number of words from/to memory is requested. This request can be used either as an interrupt request or as a DMA requests.
LBREQINTR	Last burst request interrupt: A final burst transfer from/to memory is requested. This request can be used either as an interrupt request or as a DMA requests.
SREQINTR	Single request interrupt: The transfer of a word from/to memory is requested. This request can be used either as an interrupt request or as a DMA requests.
LSREQINTR	Last single request interrupt: A final single transfer from/to memory is requested. This request can be used either as an interrupt request or as a DMA requests.
IRDAINTR	Combined interrupt: OR of the eight previous individual interrupt lines.

The burst transfer and single transfer request signals can both be asserted at the same time. For example, during reception, when there is more data in the FIFO than the programmed burst size, the burst transfer request and the single transfer request are both asserted. When the amount of data left in the FIFO is less than the programmed burst size, only the single transfer request is asserted. This is useful for situations in which the number of words left to be received in the stream is less than a burst.

27 I2C high speed controller

27.1 Introduction

The I2C controller interface supports the physical and data-link layer according to the I2C standard revision 2.1 (January 2000). The I2C bus is a 2-wire serial bus that provides a low cost interconnection between ICs. The media access control (MAC) services include:

- Data encapsulation/decapsulation (transmit and receive):
 - framing,
 - addressing,
 - error detection.
- Media access management:
 - medium allocation,
 - contention resolution.
- I2C controller operation mode:
 - slave mode: the device cannot initiate transactions on the bus, but it responds to a master device when addressed.

The application interface for frame transmission/reception and configuration is implemented as for the APB bus. The DMA interface can be used to upload and download Tx/Rx data as an alternative to the CPU.

Table 125. Terms and definitions

Term	Definition
Transmitter	The device which sends data to the bus
Receiver	The device which receives data from the bus
Master	The device which initiates a transfer, generates the clock and terminates the transfer
Slave	The device addressed by a master
Synchronization	Procedure to synchronize the clock signal between two or more devices

27.2 Features

- Two-wire I2C serial interface,
- Baud rates:
 - standard mode (100 Kbit/s)
 - fast mode (400 Kbit/s)
 - high-speed mode (3.4 Mbit/s)
- Operating mode supported:
 - I2C slave-only mode
- I2C transactions supported:
 - 7 or 10-bit addressing
 - 7 or 10-bit addressing with combined formats
 - general call command
 - start byte procedure
 - spike digital filtering on the SDA and SCL lines
 - main bus interfaces
 - APB interface for configuration and data transaction transfer,
- DMA interface for data transaction transfer.
 - DMA capability (1 channel) to manage the Tx FIFO
 - DMA capability (1 channel) to manage the Rx FIFO.
- IP I2C kernel IP clocked at 48 MHz

27.3 I2C protocol overview

Two signals, serial data (SDA) and serial clock (SCL) carry information between the devices connected to the bus. Each device has an unique address and operates as either a transmitter or receiver, depending on the function of the device. A master is the device which initiates a data transfer on the bus and generates the clock signal. Any addressed device is then considered to be a slave.

The arbitration procedure uses a wired-AND connection of all I2C interfaces to the I2C bus. If two or more masters try to put information onto the bus, the first to produce a logic 1 when the other produces a logic 0 loses the arbitration. The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired- AND connection to the SCL line.

Generation of clock signals on the I2C bus is always the responsibility of master devices. Each master generates its own clocks when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding-down the clock line, or by another master when arbitration occurs.

27.3.1 General

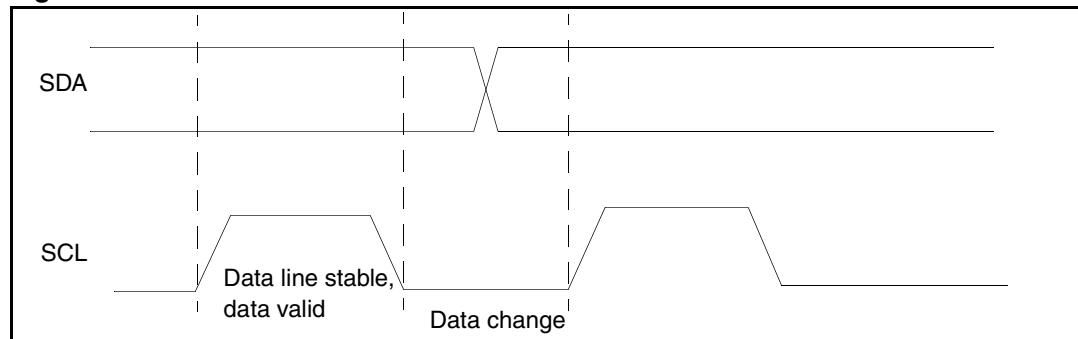
SDA and SCL are bi-directional lines, connected to a positive supply voltage via a pull up resistor. Both lines are high in idle state (bus free). The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired- AND function. Data on the bus can be transferred at rates of up to 100 Kbit/s in the standard mode, up to 400 Kbit/s in the fast mode, or up to 3.4_Mbit/s in high-speed mode.

The number of devices connected to the bus is limited by the maximum bus capacitance of 400 pF.

27.3.2 Bit transfer

Two logic levels 0 (low) and 1 (high) are defined on the I2C bus. One clock pulse is generated for each data bit transferred. The data must be stable during the high period of the clock. The high or low state of the data line can only change when the clock signal on the SCL line is low as described in [Figure 124](#).

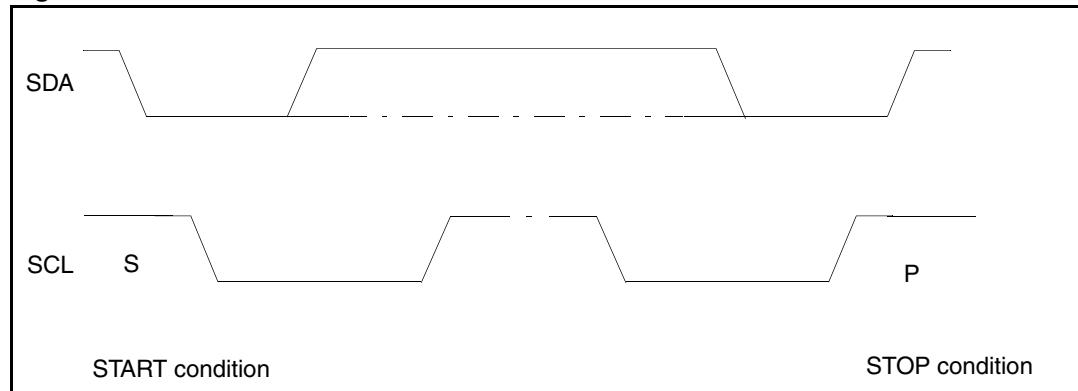
Figure 124. Bit transfer on the I2C bus



Two unique situations, known as START (S) and STOP (P) conditions, can violate the rule above defined as shown in [Figure 125](#). A high to low transition on the SDA line while SCL is high, indicates a START condition. A low to high transition on the SDA line while SCL is high, indicates a STOP condition. START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition, the bus is considered to be free again a certain time after the STOP condition.

The bus is frozen in busy state if a repeated START (Sr) is generated instead of a STOP condition. The START (S) and repeated START (Sr) conditions are functionally identical.

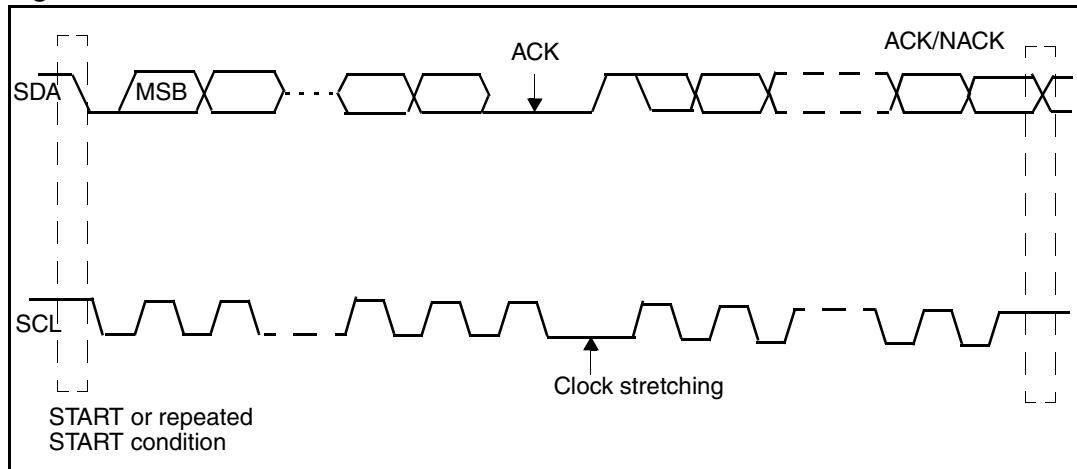
Figure 125. START and STOP conditions



27.3.3 Data transfer

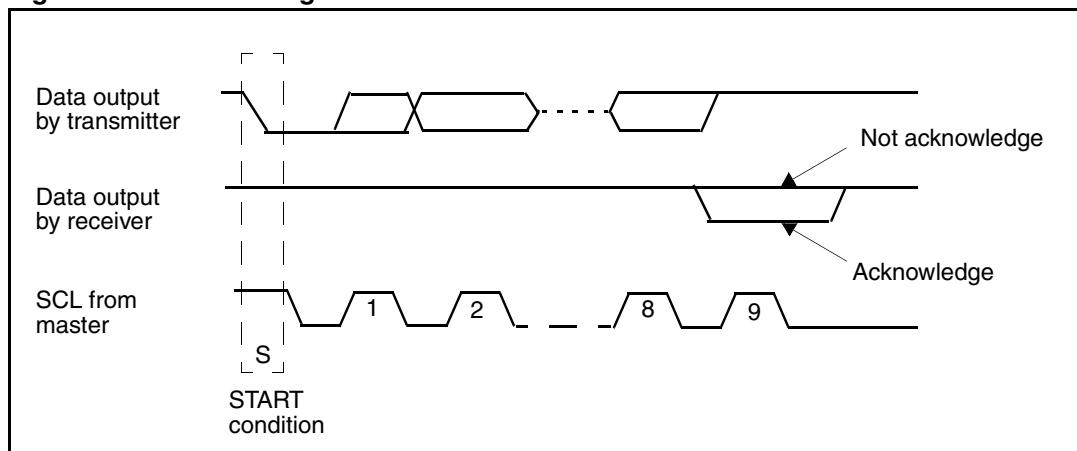
Data are transferred on the bus at byte level (the MSB is sent first). The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an acknowledge bit (see [Figure 126](#)). The slave can hold the clock line (SCL low) when it cannot transmit or receive data forcing the master into a wait state. Data transfer with acknowledge is obligatory.

Figure 126. Data transfer on the I2C bus



The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (high) during the acknowledge clock pulse. The receiver must pull down the SDA line during the acknowledge clock pulse ([Figure 127](#)). If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition.

Figure 127. Acknowledge on the I2C bus

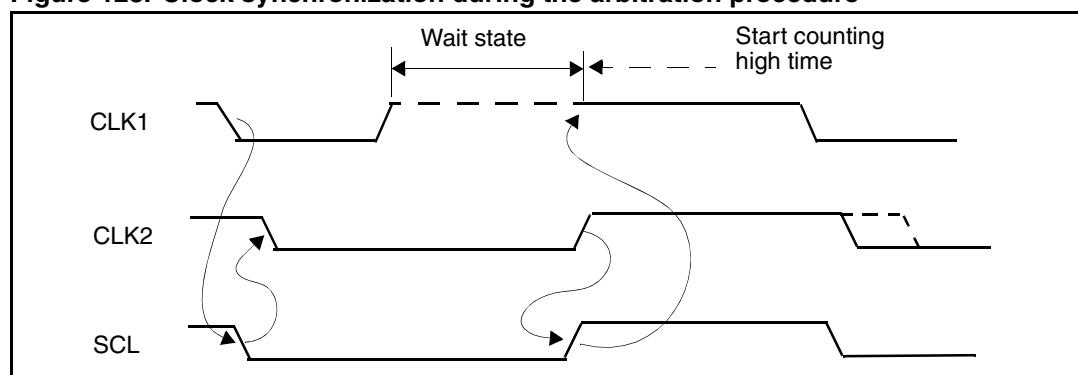


27.3.4 Arbitration and clock generation

All masters generate their own clock on the SCL line to transfer messages on the I2C bus. Data is only valid during the high period of the clock. Clock synchronization is performed using a wired-AND connection of I2C interfaces to the SCL line. A high-to-low transition on the SCL line causes the devices concerned to start counting down their low period. Once a device clock has gone low, it holds the SCL line in that state until the clock high state is reached (*Figure 128*).

However, the low-to-high transition of this clock does not change the state of the SCL line if another clock is still within its low period. The SCL line is therefore held low by the device with the longest low period. Devices with shorter low-periods enter a high wait state during this time. When all the devices concerned have counted down their low period, the clock line is released and goes high. Therefore all the device clocks are aligned to the state of the SCL line (high), and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again. In this way, a synchronized SCL clock is generated with its low period determined by the device with the longest clock-low period, and its high period determined by the one with the shortest clock-high period.

Figure 128. Clock synchronization during the arbitration procedure



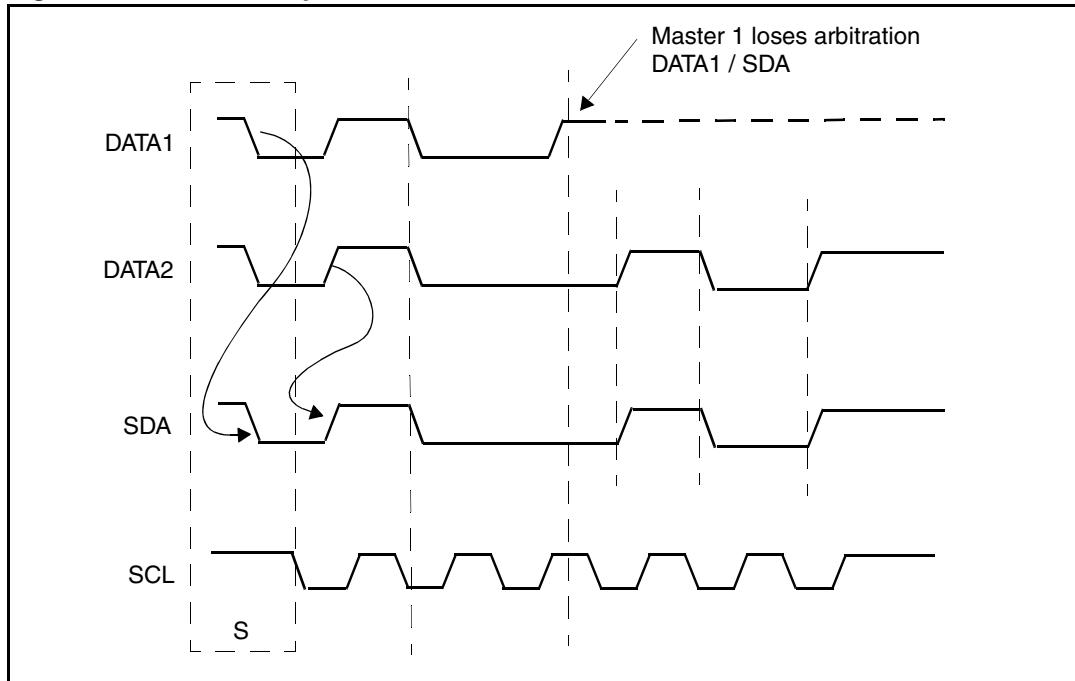
A master can only start a transfer if the bus is free. Two or more masters can generate a START within the minimum hold time of the START condition, which results in a defined START condition on the bus. Arbitration takes place on the SDA line while the SCL line is high. This is done such that the master which transmits a high level while another master is transmitting a low level, switches off its DATA output stage because the level on the bus does not correspond to its own level. Arbitration can continue for many bits, from the addressing phase up to the data phase.

Because address and data information on the I2C bus is determined by the winning master, no information is lost during the arbitration process. A master that loses the arbitration can generate clock pulses until the end of the byte in which the arbitration is lost. Since the Hs-mode master has a unique 8-bit master code, it always finishes the arbitration during the first byte. If a master also includes the slave function and it loses arbitration during the addressing stage, it is possible that the winning master is trying to address it. The losing master must therefore switch immediately into slave mode.

Arbitration is not allowed in the following cases:

- a repeated START condition and a data bit,
- a STOP condition and a data bit (TBC),
- a repeated START condition and a STOP condition.

Note: Slaves are not involved in the arbitration procedure.

Figure 129. Arbitration procedure of two masters

As well as being used during the arbitration procedure, the clock synchronization mechanism enables receivers to cope with fast bit or byte transfers.

- Byte level: if a device receives data at a fast rate, but needs more time to store a received byte (or prepare another byte to be transmitted), slaves can hold the SCL line low after reception and acknowledgment of a byte. This forces the master into a wait state until the slave is ready for the next byte transfer in a form of handshake procedure.
- Bit level: a device can slow down the bus clock by extending each clock low period. The speed of any master is thereby adapted to the internal operating rate of this device.

In Hs mode, this handshake feature can only be used on byte level.

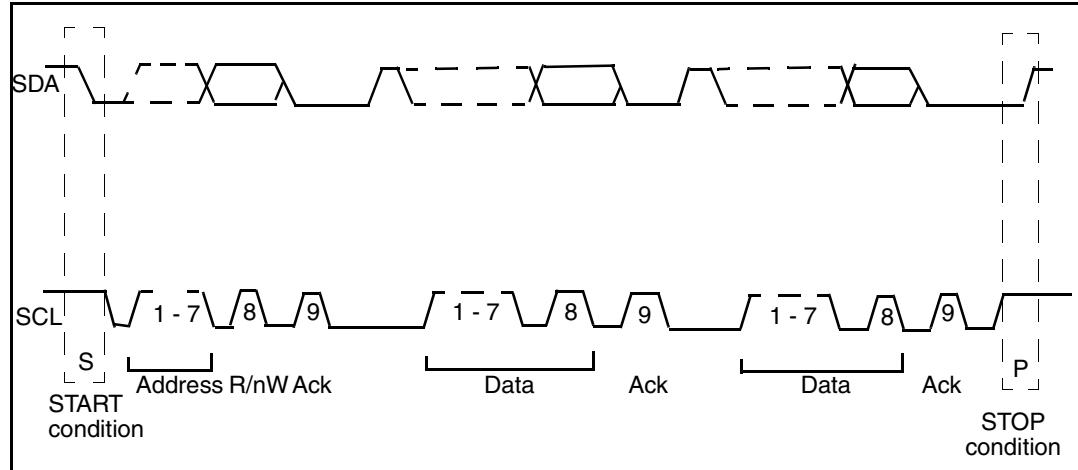
27.3.5 Formats with 7-bit addressing mode

Referring to [Figure 130](#) the sequence of the basic data transfer is as follows.

- A START condition to initiate the transfer.
- A 7-bit address followed by an eight data direction bit (R/nW). A 'zero' indicates a transmission (write), a 'one' indicates a request for data (read).
- Data transmission phase including n bytes.
- STOP condition to terminate the transfer.

If a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition

Figure 130. A complete data transfer



When an address is sent, each device in the system compares the first seven bits after the START condition with its own address. If they match, the device considers itself addressed by the master as a slave-receiver or slave-transmitter, depending on the state of the R/nW bit. Several data transfer are supported in 7-bits addressing mode. These are described below.

- Master transmit to slave-receiver: the transfer direction is not changed (see [Figure 131](#))
 - Master-receive from slave-transmitter: the transfer direction is changed after the first acknowledge.
 - Combined format: several transactions of the previous type (master-transmitter and/or master-receiver) are interleaved by repeated START conditions. If a master-receiver sends a repeated START condition, it has previously sent a not-acknowledged bit.

Figure 131. Master-transmitter (7-bit address)

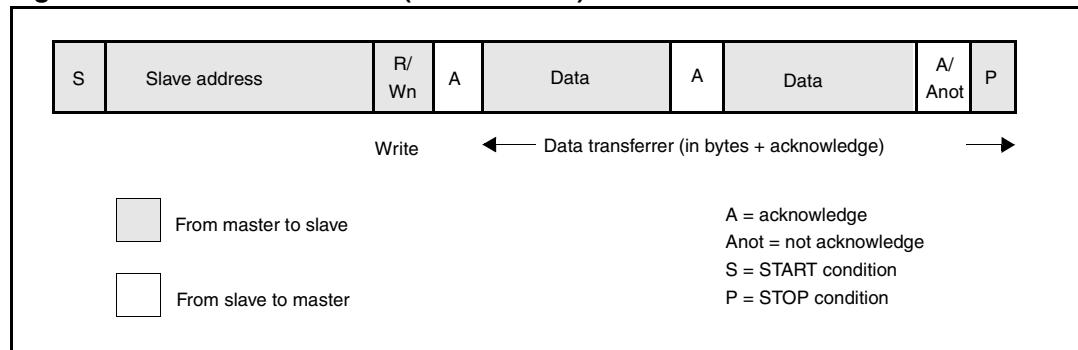


Figure 132. Master-receiver (7-bit address)

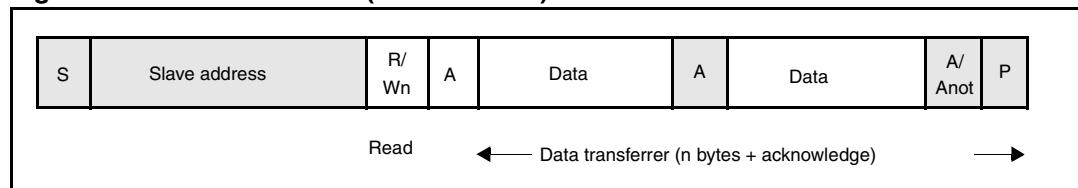
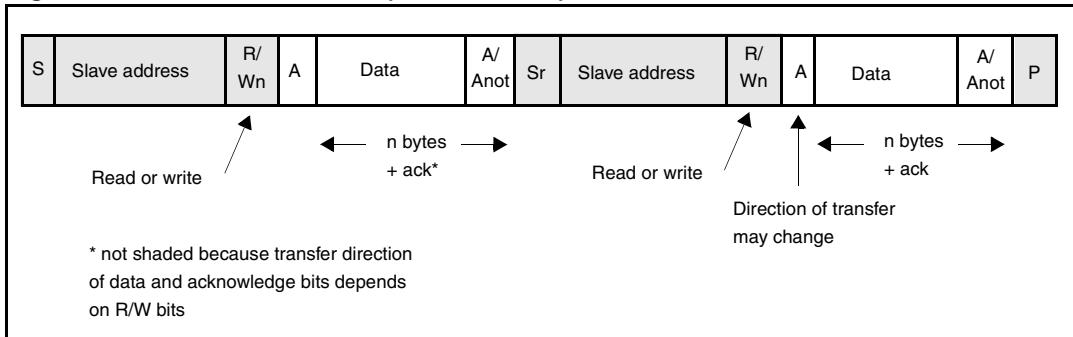


Figure 133. Combined format (7-bit address)

I2C bus devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address, even if these START conditions are not positioned according to the proper format. A START condition immediately followed by a STOP condition (void message) is an illegal format.

27.3.6 Address byte definition

The first byte of the data transfer (after the START condition) defines the target address. Two groups of eight addresses (0000xxx and 1111xxx) are reserved for the purposes described in [Table 126](#).

Table 126. Definition of the first byte (7-bit address)

Slave address	R/nW bit	Description
000 0000	0	General call address
000 0000	1	START byte ⁽¹⁾
000 0001	X	CBUS address ⁽²⁾
000 0010	X	Reserved for different ⁽³⁾ bus format
000 0011	X	Reserved
000 01XX	X	Hs-master code
111 11XX	X	Reserved
111 10XX	X	10-bit slave addressing format

1. No devices must not acknowledge at the reception of the START byte.
2. I2C bus devices must not respond on reception of this address.
3. The address reserved for a different bus format is included so that I2C and other protocols can be mixed.

27.3.7 General call address

The general-call address (see [Figure 134](#)) addresses every device connected to the I2C bus. If a device does not need any of the data supplied within the general-call structure, it ignores this address (that is, it does not issue an acknowledge). If a device requires data from a general-call address, it acknowledges the address and behaves as a slave receiver. The second and successive bytes are acknowledged by every slave-receiver capable of handling the data. A slave which cannot process one of these bytes must ignore it (not acknowledge). The meaning of the general-call address is specified by the second byte. There are two cases:

- When the least-significant bit is zero:
 - 0x06: reset and write the programmable part of the slave address by hardware. On receiving this 2-byte sequence, all devices able to respond to the general call address reset and modify the programmable part of their address (hw_addr[9:0]).
 - 0x04: write programmable part of slave address by hardware. On receiving this 2-byte sequence, all devices in which the programmable part of the address is defined by hardware (and which respond to the general-call address) latch the programmable part. The device does not reset.
- Remaining codes have not been fixed and devices must ignore them.
- When the least significant bit is a ‘one’, the 2-byte sequence is a *hardware general call*. This means that the sequence is transmitted by a hardware master device, which cannot be programmed to transmit a desired slave address. The hardware master identifies itself to the system, inserting its address (7-bits) in the address byte see [Figure 135](#)

Figure 134. General call address format

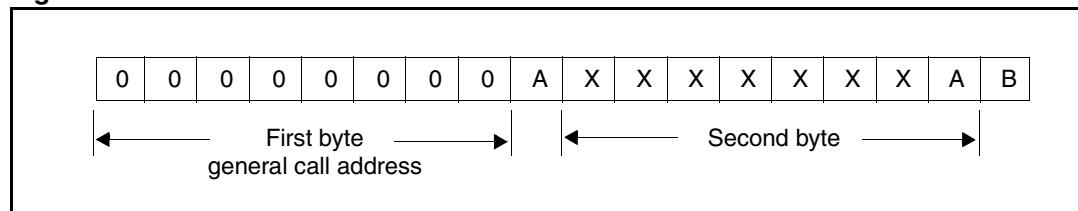
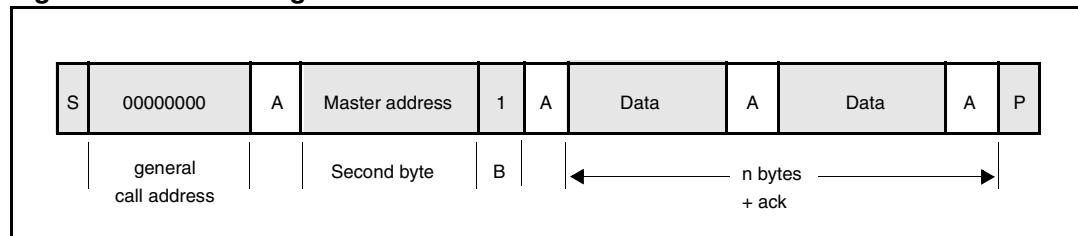


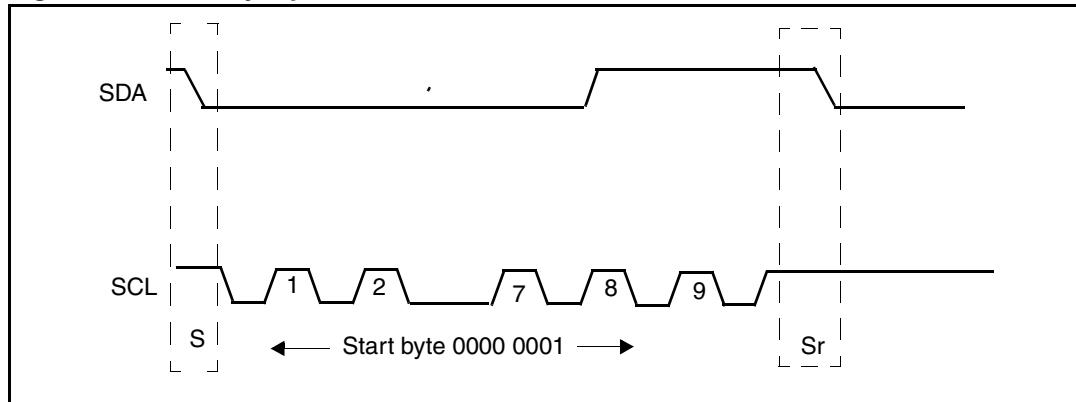
Figure 135. Hardware general call



27.3.8 Start byte

The start byte procedure is defined for frame synchronization to slow devices (micro-controllers) which use software polling to monitor the bus. In this case, a data transfer can be preceded by a start procedure which is much longer than normal (see [Figure 136](#))

Figure 136. Start byte procedure



The start procedure consists of:

- a START condition,
- a START byte (0x01),
- a not-acknowledge bit (NACK),
- a repeated START condition (Sr).

After the START condition has been transmitted by a master which requires bus access, the START byte (0x01) is transmitted. Another micro-controller can then sample the SDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this low level on the SDA line, the micro-controller may switch to a higher sampling rate to find the repeated START condition, Sr, which is then used for synchronization.

An acknowledge-related clock pulse is generated after the START byte. Devices cannot acknowledge the START byte.

27.3.9 CBUS compatibility

CBUS receivers can be connected to the standard-mode I2C bus. In a mixed bus structure, I2C bus devices must not respond to the CBUS message. For this reason, a special CBUS address to which no I2C-bus compatible device responds has been reserved. After the STOP condition, all devices are again ready to accept data.

27.3.10 Fast mode

Fast-mode devices can receive and transmit at up to 400 Kbit/s. The fast-mode devices are downward-compatible and can communicate with standard mode devices in a 0 to 100 Kbit/s I2C bus system. Standard mode devices are not upward compatible, and should not be included in a fast mode I2C bus system. The fast mode I2C bus specification includes the following additional information:

- Maximum bit rate of 400 Kbit/s.
- Timing of the serial data (SDA) and serial clock (SCL) signals has been adapted.
- Fast-mode devices incorporate spike suppression and a Schmidt trigger at the SDA and SCL inputs.
- The output buffers of fast-mode devices include slope control of the falling edges of the SDA and SCL signals.
- If the power supply of a fast-mode device is switched-off, the SDA and SCL IO pins must float so that they do not drive the bus lines.
- External pull-up devices connected to the bus lines must be adapted to accommodate the maximum permissible rise time for the fast-mode I2C bus.

27.4 High-speed controller

High-speed mode (Hs-mode) devices transfer information at up to 3.4 Mbit/s. They remain fully downward compatible with fast or standard-mode devices for bi-directional communication in a mixed-speed bus system. The protocol and data format is maintained as with the F/S-mode system, with the exception that arbitration and clock synchronization is not performed during the Hs-mode transfer.

27.4.1 High speed transfer

To achieve a bit transfer of up to 3.4 Mbit/s the following improvements have been made to the regular I2C bus specification:

- Hs-mode master devices have an open-drain output buffer for the SDAH signal and a combination of an open-drain pull-down and current-source pull-up circuit on the SCLH output. This current-source circuit shortens the rise time of the SCLH signal. Only the current-source of one master is enabled at any one time, and only in Hs-mode.
- Hs-mode master devices generate a serial clock signal with a high-to-low ratio of 1 to 2. This relieves the timing requirements for setup and hold times.
- As an option, Hs-mode master devices can have a built-in bridge which separates the high-speed serial data and clock lines (SDAH, SCLH) from the SDA and SCL lines of F/S-mode devices.
- The only difference between Hs-mode slave and F/S-mode devices is the operating speed. Hs-mode slave devices have open-drain output buffers on the SCLH and SDAH outputs. Optional pull-down transistors on SCLH can be used to stretch the low level of the SCLH signal. This is only allowed after the acknowledge bit in Hs-mode transfers.
- The inputs of Hs-mode devices include spike suppression and a Schmidt trigger at the SDAH and SCLH inputs.
- The output buffers of Hs-mode devices include slope control of the falling edges of the SDAH and SCLH signals.

27.4.2 Serial data transfer format in Hs-mode

Serial data transfer format in Hs-mode commences with the following sequence:

- START condition (S),
- 8-bit master code (0000 1XXX),
- not-acknowledge bit (no device is allowed to acknowledge the master code).

The master code has two main functions:

- Arbitration and clock synchronization between competing masters at F/S-mode speeds, resulting in one winner master
- Delineation of a Hs-mode transfer

Hs-master codes are reserved 8-bit codes, and are not used for slave addressing or other purposes. Arbitration and clock synchronization only take place during the transmission of the master code and not-acknowledge bit. After this phase a winning master remains active. The master code indicates that an Hs-mode transfer is starting and the connected devices must meet the Hs-mode specification.

After the not-acknowledged bit, and the SCLH line have been pulled to a high level, the active master switches to Hs mode and enables the current-source pull-up circuit for the SCLH signal (at time tH in [Figure 138](#)). As other devices can delay the serial transfer before tH by stretching the low period of the SCLH signal, the active master enables its current-source pull-up circuit when all devices have released the SCLH line and the SCLH signal has reached a high level, thus speeding up the last part of the rise time of the SCLH signal.

The active master sends a repeated START condition (Sr) followed by a 7-bit (or 10-bit) slave address with a R/nW bit, and receives an acknowledge bit from the selected slave.

After a repeated START condition and after each acknowledge or not-acknowledge bit, the active master disables its current-source pull-up circuit. This enables other devices to delay the serial transfer by stretching the low period of the SCLH signal. The active master re-enables its current-source pull-up circuit when all devices have released and the SCLH signal reaches a high level.

Data transfer continues in Hs mode after the next repeated START (Sr), and only switches back to F/S mode after a STOP condition (P). To reduce the overhead of the master code, a master may link a number of Hs-mode transfers, separated by repeated START conditions (Sr).

Figure 137. Data transfer format in Hs mode

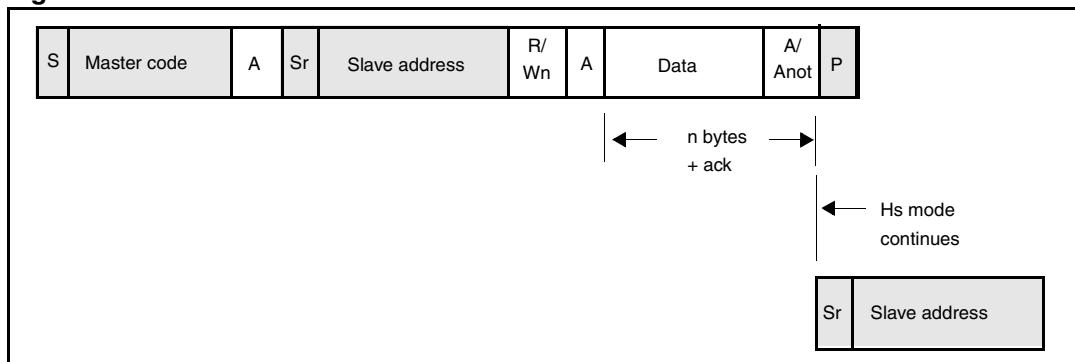
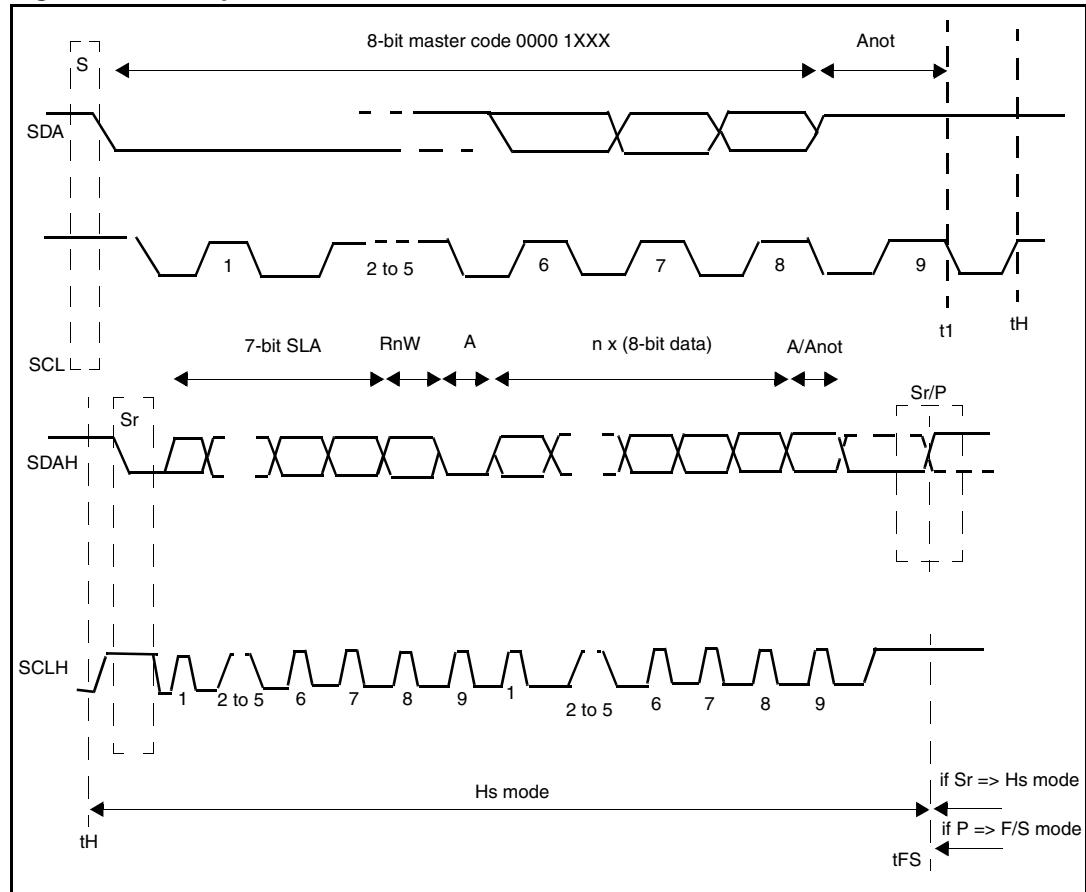


Figure 138. Complete Hs-mode transfer



27.4.3 Switching from F/S to Hs mode and back

After reset and initialization, Hs-mode devices must be in F/S mode. Each Hs-mode device can switch from F/S to Hs mode and back and is controlled by the serial transfer on the I2C bus.

Before time t_1 , each connected device operates in F/S mode. Between times t_1 and t_{H} (this interval can be stretched by any device) each connected device must recognize the sequence S 0000 1XXX nA and has to switch from the F/S-mode setting to the Hs-mode setting. The winning master performs the following actions:

- Adapts its SDAH and SCLH input filters according to the spike suppression requirement in Hs mode.
- Adapts the set-up and hold times according to the Hs-mode requirements.
- Adapts the slope control of its SDAH and SCLH output stages according to Hs-mode requirements.
- Switches to the Hs-mode bit rate, which is required after time t_{H} .
- Enables the current source pull-up circuit of its SCLH output stage at time t_{H} .

The non-active, or losing masters must:

- Adapt their SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
- Wait for a STOP condition to detect when the bus is free again.

All slaves must:

- Adapt their SDAH and SCLH input filters according to the spike suppression requirement in Hs mode.
- Adapt the setup and hold times according to the Hs mode requirements.
- Adapt the slope control of their SDAH output stages as necessary.

At time tFS in [Figure 138](#), each connected device must recognize the STOP condition (P) and switch its internal circuit from the Hs-mode setting back to the F/S mode setting, according to the state before time t1. It must be completed within the minimum bus-free time tBUF as specified in F/S mode.

27.4.4 Hs-mode devices at lower speed modes

High-speed mode devices are fully downward compatible, and can be connected to an F/S mode I2C-bus system. In such configurations no master codes are transmitted, therefore all Hs-mode master devices stay in F/S mode and communicate at F/S-mode speeds with their current source disabled.

27.4.5 Mixed speed modes on one serial bus system

If a system has a combination of Hs and F/S-mode devices, an interconnection bridge can be used to have different bit rates between devices. One bridge is required to connect and disconnect an Hs-mode section to or from a F/S-mode section at the appropriate time.

27.4.6 Formats with 10-bit addressing mode

10-bit addressing mode is compatible, and can be combined with, 7-bit addressing mode. Devices with 7-bit and 10-bit addresses may be connected to the same I2C bus. Both 7-bit and 10-bit addressing can be used in F/S-mode and Hs-mode systems.

27.4.7 Address format

The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr). The first byte includes the sequence 111 10XX of which the last two bits (XX) are the most-significant bits of the 10-bit address, the eighth bit being the RnW bit. If the RnW bit is ‘zero’, the second byte contains the remaining 8 bits of the 10-bit address. If the RnW is ‘one’, the next byte contains data transmitted from a slave to a master.

27.4.8 Data transfer format

Several combinations of read/write formats are possible within a transfer that includes 10-bit addressing. Some example data transfers are listed below:

- Master-transmitter transmits to slave-receiver with a 10-bit slave address ([Figure 139](#)).

The transfer direction does not change. After the START condition, each slave compares the first seven bits of the first byte of the slave address (111 10XX) with its own address and tests if the eighth bit (RnW) is 0. It is possible that more than one device finds a match and generates an acknowledge (A1). All slaves that find a match compare the eight bits of the second byte of the slave address to their own addresses, but only one slave will find a match and generates an acknowledge (A2). The

matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

- Master-receiver reads slave-transmitter with a 10-bit slave address ([Figure 140](#)). The transfer direction changes after the second RnW bit. Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks if the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and tests if the eight bit (RnW) is 1. If a match occurs, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or until it receives another repeated START condition (Sr) followed by a different slave address. After a repeated START condition (Sr), all the other slave devices compare the first seven bits of the first byte of the slave address (111 10XX) to their own addresses, and test the eight (RnW) bits. However, none of them are addressed because RnW is '1' (for 10-bit devices), or the 111 10XX slave address (for 7-bit devices) does not match.
- Combined format, master transmits data to slave then reads data from the same slave ([Figure 141](#)).

The same master occupies the bus all the time. The transfer changes after the second RnW bit.

- Combined format ([Figure 142](#)).
- A master transmits data to one slave and then transmits to another slave. The same master occupies the bus all the time.
- Combined format ([Figure 143](#)).
- A master transmits data to a 7-bit address slave then transmits data to another 10-bit address slave. The same master occupies the bus all the time.
- I2C-bus devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address.

Figure 139. slave-receiver (10-bit address)

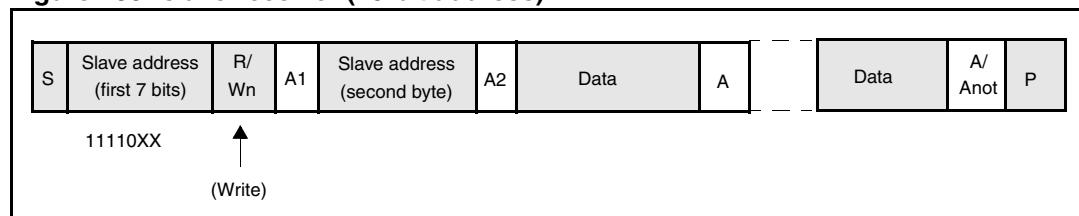


Figure 140. Master-receiver (10-bit address)

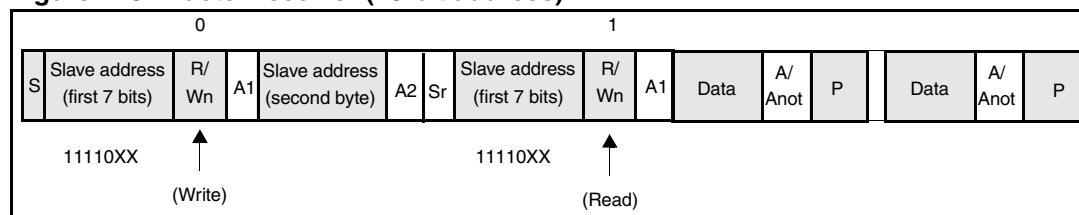
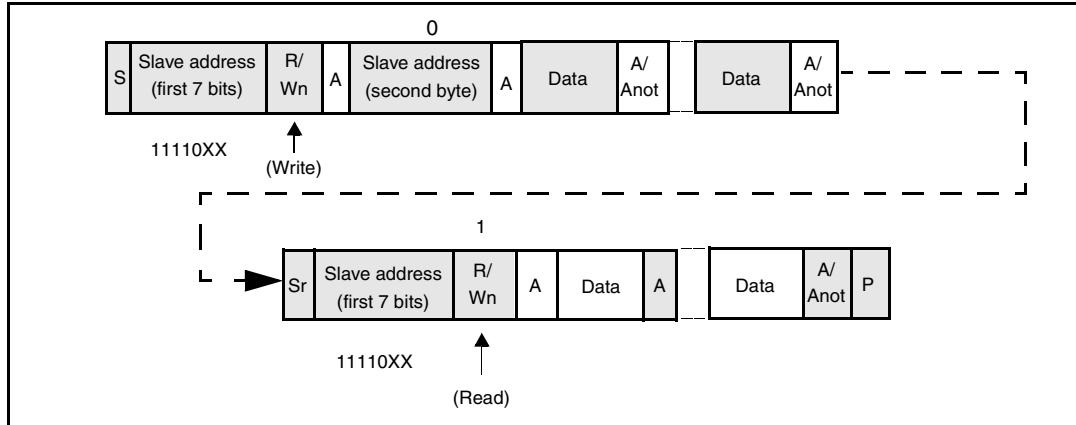
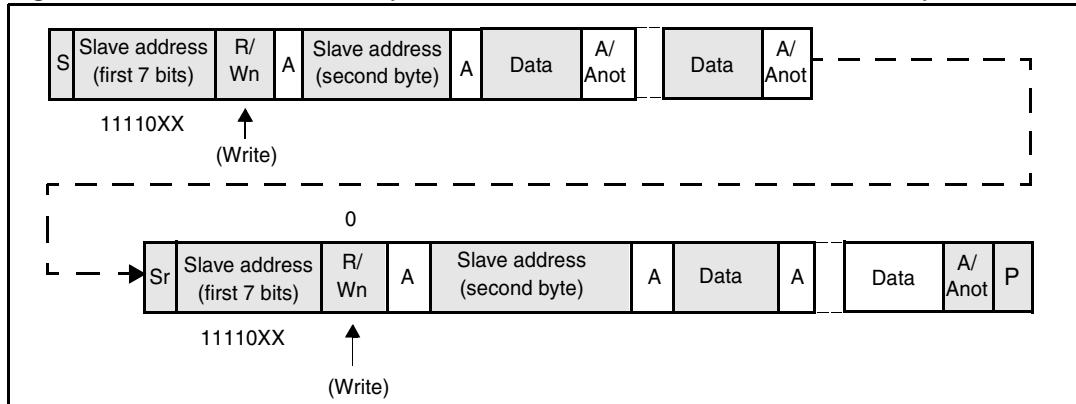
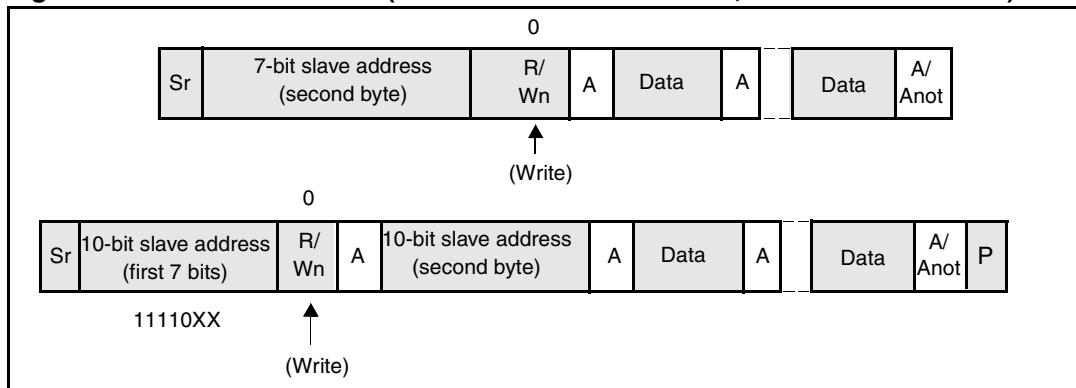


Figure 141. Combined format (communication to and from a 10-bit address slave)**Figure 142. Combined format (transmission to two 10-bit address slaves)****Figure 143. Combined format (transmission to two slaves, 10 and 7-bit address)**

27.4.9 General call address and start-byte with 10 bit-addressing

In the 10-bit addressing procedure for the I2C bus, the first two bytes after the START condition (Sr) usually determine which slave is to be selected by the master. The exception is the 'general call' address (0x00). Slave devices with 10-bit addressing react to a 'general call' in the same way as slave devices with 7-bit addressing. Hardware masters may transmit their 10-bit address after a general call (two successive bytes after the general call address).

Table 127. SDA and SCL bus-line characteristics for F/S -mode I2C devices

Parameter	Symbol	Standard mode		Fast mode		Unit
		Min.	Max.	Min.	Max.	
SCL clock frequency.	fSCL	0	100	0	400	kHz
Hold time (repeated) START condition. After this period, the first clock pulse is generated.	tHD;STA	4.0	-	0.6	-	μs
Low period of the SCL clock.	tLOW	4.7	-	1.3	-	μs
High period of the SCL clock.	tHIGH	4.0	-	0.6	-	μs
Set-up time for a repeated START condition.	tSU;ST	4.7	-	0.6	-	μs
Data hold time for I2C-bus devices.	tHD;DAT	0 ⁽¹⁾	3.45 ⁽¹⁾	0 ⁽¹⁾	0.9 ⁽¹⁾	μs
Data set-up time	tSU;DAT	250	-	100 ⁽²⁾	-	ns
SDA and SCL rise time	tR	-	1000	20 + 0.1 Cb ⁽³⁾	300	ns
SDA and SCL fall time	tF	-	300	20 + 0.1 Cb ⁽⁴⁾	300	ns
Set-up time for STOP condition	tSU;STO	4.0	-	0.6	-	μs
Bus free time between STOP and START condition	tBUF	4.7	-	1.3	-	μs
Capacitive load for each bus line	Cb	-		1.3	-	μs
Low level input voltage						
Fixed input levels	Vil	-0.5	1.5	n/a	n/a	V
Vdd-related input levels		-0.5	0.3 Vdd	-0.5	0.3 Vdd	
High level input voltage						
Fixed input levels	Vih	3.0	(5)	n/a	n/a	V
Vdd-related input levels		0.7 Vdd		0.7 Vdd		

1. Refer to (b)
2. A fast-mode I2C bus device can be used in a standard-mode I2C-bus system, but the requirement tSU;DAT >= 250 ns must then be met
3. Cb = total capacitance of one bus line in pF
4. Refer to (f)
5. Maximum Vih = Vddmax + 0.5 V

Table 128. SDAH and SCLH bus-line characteristics (Hs mode)

Parameter	Symbol	Standard mode		Fast mode		Unit
		Min	Max	Min	Max	
SCLH clock frequency	fSCL	0	3.4	0	1.7	kHz
Hold time (repeated) START condition	tSU;STA	160	-	160	-	ns
Setup time (repeated) START condition	tHD;STA	160	-	160	-	ns
SCLH low time	tLOW	160	-	320	-	μs
SCLH high time	tHIGH	60	-	120	-	μs
Data setup time	tSU;DAT	10	-	10	-	μs
Data hold time	tHD;DAT	0	70	0	150	ns
Setup time for STOP condition	tSU;STO	160	-	160	-	ns

27.5 I2C signals

The I2C interface signals are described in [Table 129](#).

Table 129. I2C signals

Signal	I/O	Description
I2CSCL0	IO	I2C bus clock signal
I2CSDA0	IO	I2C0 bus data signal
I2CSCL1	IO	I2C1 bus clock signal
I2CSDA1	IO	I2C0 bus data signal

28 1-wire master (OWM)

The STn8815A12 has a 1-Wire™ master (OWM) that implements the hardware layer of the 1-Wire or HDQ™ protocols⁽¹⁾. The OWM is a master or slave interface that enables synchronous serial communication with slave or master peripherals.

28.1 Features

- generates interrupts to provide more efficient programming
- supports standard and overdrive 1-Wire communication speeds
- supports HDQ standard
- supports single-bit transmissions
- provides added support for long line conditions

This implementation eliminates CPU **bit-banging** by internally generating all 1-Wire timing and control signals.

28.2 Overview

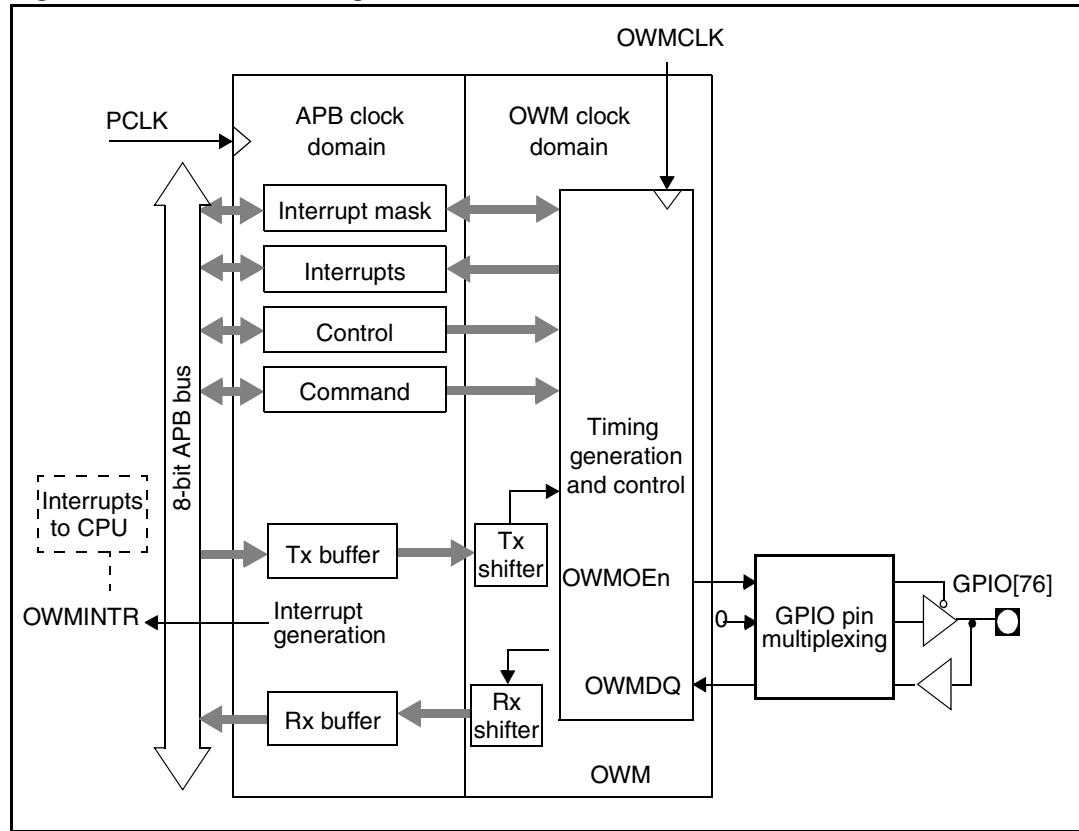
The OWM handles communication to 1-Wire devices without CPU loading. The 1-Wire protocol uses a single wire to communicate between the master and slave. It employs a return-to-1 mechanism, where the line is pulled up by an external pull-up resistor after a command or data is sent or received. The OWM implements only the hardware layer of the 1-Wire protocol.

The host CPU loads commands, reads and writes data, and sets interrupt control. All timing and control of the 1-Wire bus are generated by the OWM.

When bus activity has generated a response that the CPU needs to receive, the 1-Wire master sets a status bit and, if enabled, generates an interrupt to the CPU.

1. 1-Wire protocol is a Dallas Semiconductor trademark. HDQ protocol is a Benchmark/TI trademark.

Figure 144. OWM block diagram



28.3 Functional description

Operation of the 1-Wire bus is described in detail in the book of iButton standards (refer to *Dallas Semiconductor*). Each slave device, in general, has its own set of commands that are described in detail in the device's datasheet. The user is referred to these documents for details on specific slave implementations.

28.3.1 I/O signaling

The 1-Wire bus requires strict signaling protocols to ensure data integrity.

The four protocols used by the 1-Wire master are:

- the initialization sequence (reset pulse followed by presence pulse),
- write 0,
- write 1,
- read data.

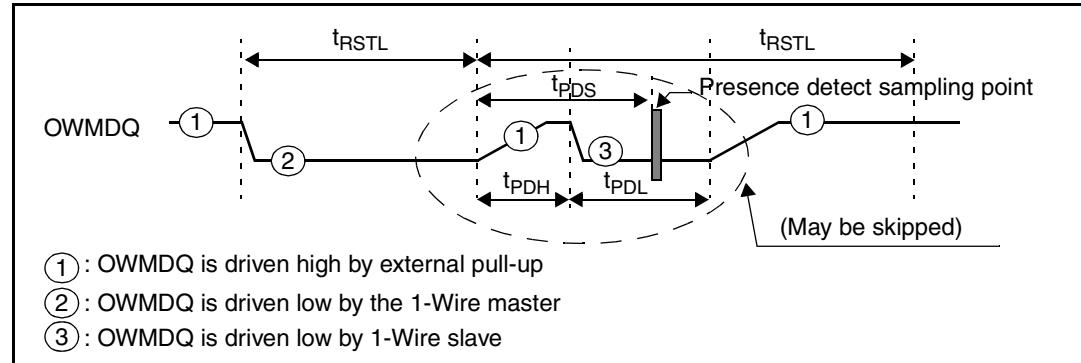
The master initiates all of these types of signaling except the presence pulse.

1-wire initialization sequence

The initialization sequence required to begin any communication with the bus slave is shown in [Figure 145](#). A presence pulse following a reset pulse indicates that the slave is ready to accept a command. The 1-Wire master transmits a reset pulse for t_{RSTL} . The 1-Wire bus line

is then pulled high by the pull-up resistor. After detecting the rising edge on the OWDQ pin, the slave waits for t_{PDH} and then transmits the presence pulse for t_{PDL} . The master samples the bus at t_{PDS} after the slave responds to test for a valid presence pulse or after waiting for t_{PDHCNT} following the start of t_{RSTH} . The result of this sample is stored. The reset time slot ends t_{RSTL} after the master releases the bus. Refer to [Section 28.3.2](#) for timing specification of t_{RSTL} , t_{RSTH} , and t_{PDS} .

Figure 145. 1-wire initialization sequence: reset pulse and presence pulse

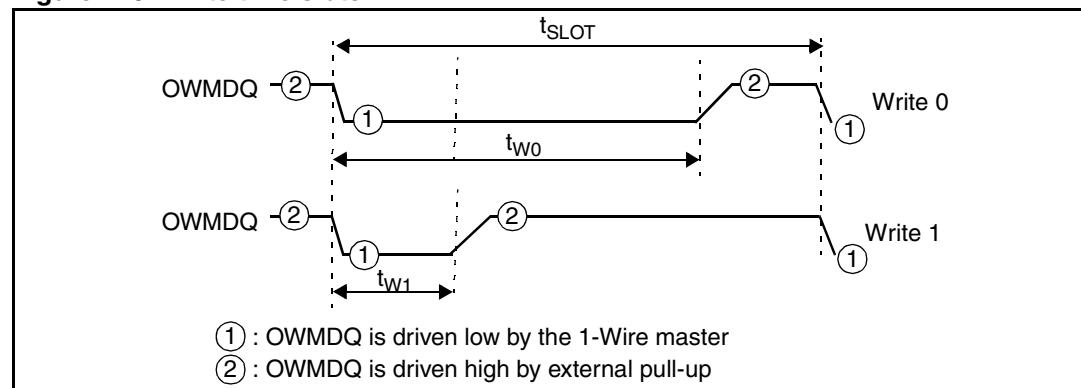


The presence detection phase may be programmed to be skipped.

1-wire write time slots

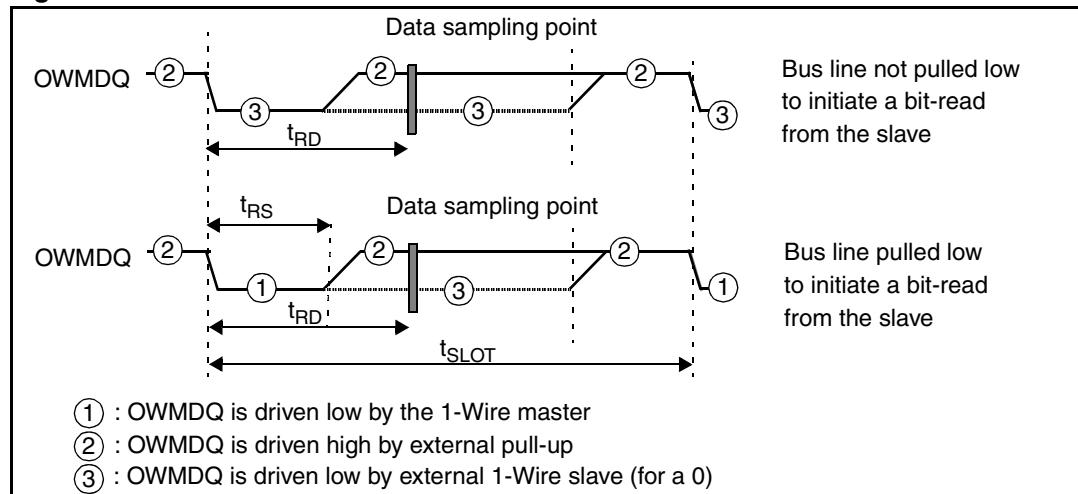
A write time slot is initiated when the 1-Wire master pulls the 1-Wire bus line from a logic high (inactive) level to a logic low level. The master generates a Write 1 time slot by releasing the line at t_{W1} and allowing the line to pull up to a logic high level. The line is held low for t_{W0} to generate a Write 0 time slot. A slave device samples the 1-Wire bus line between 15 and 60 μ s after the line falls. If the line is high when sampled, a Write 1 occurs. If the line is low when sampled, a Write 0 occurs. See [Figure 146](#) for more information.

Figure 146. Write time slots



1-wire read time slots

A read time slot is initiated when the 1-Wire master pulls the bus low for at least 1 μ s and then releases it. If the slave device responds with a 0, it continues to hold the line low for up to t_{RD} ; otherwise it releases it immediately. The master samples the data after t_{RD} from the start of the read time slot. The master ends the read slot after a time of t_{SLOT} . See [Figure 147](#) for more information.

Figure 147. Read time slots

28.3.2 OWM bit time slots: TSLOT in OWM_CR

The OWM serial bit rate is derived by dividing down the peripheral clock OWMCLK (48 MHz in this device) by 48, to get the internal timebase period $\tau = 1 \mu\text{s}$. Then, according to programming of the TSLOT value, the following timings are produced.

Table 130. Time slots ($\tau = 1 \mu\text{s}$)

TSLOT	Bit time slot (t_{SLOT})	Write 0 time (t_{W0})	Write 1 time (t_{W1})	Read start (t_{R0}) ⁽¹⁾	Read strobe (t_{RD})	Reset time high (t_{RSTH})	Reset time low (t_{RSTL})	Presence detect strobe (t_{PDS}) ⁽²⁾	Time-out ⁽³⁾
00	11τ	8τ	1τ	1τ ⁽¹⁾	2τ	50τ	61τ	3τ ⁽³⁾	6τ ⁽³⁾
01	73τ	63τ	6τ	1τ ⁽¹⁾	15τ	500τ	488τ	30τ ⁽³⁾	60τ ⁽³⁾
10	209τ	127τ	6τ	1τ ⁽¹⁾	79τ	500τ	488τ	30τ ⁽³⁾	60τ ⁽³⁾
11	209τ	127τ	6τ	1τ ⁽¹⁾	79τ	50τ	223τ	30τ ⁽³⁾	360τ ⁽³⁾

1. Read start phase occurs if the bus line is pulled low to initiate a bit-read from the slave.
2. Presence detect phase may be skipped.
3. The 1-Wire master may be programmed to wait for a falling edge on the OWDQ line to be detected after a reset for a time up to this time-out value, or to wait for a falling edge.

28.3.3 Data endianness

All bytes are sent and received with the LSB first.

In one-bit mode, only bit 0 is sent and received. The other bits are ignored for transmit, and are undefined when reading the receive buffer.

28.4 OWM signals

28.4.1 Interface signals

The OWM transmits and receives serial data on its OWMDQ bidirectional pin.

Table 131. OWM interface signal

Signal	I/O	Description
OWMDQ	I/OD	1-Wire™ interface signal. On reset, this pin is pulled up and is in GPIO mode.

28.4.2 Interrupts

There are seven individual maskable interrupt sources generated by the OWM. All these seven sources are combined into a single interrupt signal, OWMINTR.

OWMINTR interrupt

The OWMINTR is the only interrupt signal from the OWM that drives the vectored interrupt controller (VIC). This combined OWMINTR interrupt, which is an OR function of the individual masked sources, is asserted if any of the individual interrupts listed below are asserted and enabled.

- One wire low interrupt: OWMLINTR
If not masked, this interrupt is set high when the 1-Wire line is low while the master is in idle, and indicates that a slave device has issued a presence pulse on the 1-Wire (DQ) line.
- One wire short interrupt: OWMSINTR
If not masked, this interrupt is set high when the 1-Wire line was low before the master was able to send out the beginning of a reset or a time slot.
- Receive shifter full interrupt: OWMRSRFINTR
If not masked, this interrupt is set high when there is a byte of data waiting in the receive shifter. This signal is cleared by hardware when data in the receive shifter is transferred to the receive buffer.
- Receive buffer full interrupt: OWMRBFINTR
If not masked, this interrupt is set high when there is a byte of data waiting to be read in the receive buffer. This signal is cleared when the byte is read from the receive buffer, except if another byte is waiting in the receive shifter.
- Transmit shifter empty interrupt: OWMTSREINTR
If not masked, this interrupt is set high when there is nothing in the transmit shifter and it is ready to receive the next byte of data to be transmitted from the transmit buffer. This signal is cleared when data is transferred from the transmit buffer to the transmit shifter.
- Transmit buffer empty interrupt: OWMTBEIFINTR
If not masked, this interrupt is set high when there is nothing in the transmit buffer and it is ready to receive the next byte of data. This signal is cleared when data is written to the transmit buffer.
- Presence detect interrupt: OWMPDINTR
After a 1-Wire reset has been issued, if not masked, this interrupt is set high after the appropriate amount of time for a presence detect pulse to have occurred.

29 SD-Card host interface (SDI)

The SD/MMC card host interface (SDI) provides an interface between the APB peripheral bus and MultiMediaCards (MMCs), SD-Memory Cards, and CE-ATA devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association web site at www.mmca.org, published by the MMCA technical committee.

SD memory card system specifications are available through the SD Card Association web site at www.sdcards.org.

CE-ATA system specifications are available through the CE-ATA Workgroup web site at www.ce-ata.org.

29.1 Features

The SDI features are:

- Full compliance with MultiMediaCard System Specification Version 4.2; card support for 3 different data bus modes:
 - 1 bit (default)
 - 4 bits
 - 8 bits
- Full compatibility with previous versions of MultiMediaCards (forward compatibility)
- Full compliance with *SD Memory Card Specifications - Part 1 - Physical Layer Specification Version 2.0*
- Full support of the CE-ATA features (full compliance with CE-ATA Digital protocol Rev1.0 2-March-2005)
- Data transfer up to 24 MHz in 8-bit mode and up to 48 MHz in 4-bit mode
- Data and command output enable signals to control external bi-directional drivers

Note: SPI compatible communication controller is not implemented in this controller.

Note: SD Memory Card protocol is a super-set of the MultiMediaCard protocol as defined in MultiMediaCard system specification v2.11. Several commands required for SD Memory devices are not supported by the I/O portion of Combo cards. The interface electrical and signaling definition is as defined in the MMC reference.

29.2 Overview

The MultiMediaCard/SD bus connects the cards to the controller.

The current version of the SDI supports a single SD/MMC4.2 card or a stack of MMCv3.31 (or previous) cards. The SDI can interface with several SD/MMC4.2 cards with support of GPIOs for card selection and external buffers.

The communication over the bus is based on command and data transfers.

- **Command:** a command is a token which starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast commands are available for MMC v3.31 or previous). A command is transferred serially on the CMD line. All commands have a fixed length of 48 bits. The

general format for a command token for MultiMediaCard, SD-Memory card is shown in [Table 132](#). CE-ATA commands are an extension of the MMC commands v4.2, and therefore have the same format.

Table 132. Command format

Bit Position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7
0	1	1	End bit

- **Response:** a response is a token which is sent from an addressed card (or synchronously from all connected cards for MMCv3.31 or previous), to the host as an answer to a previously received command. A response is transferred serially on the CMD line. The SDI supports two response types. Both use CRC error checking:
 - 48-bit short response ([Table 133](#))
 - 136-bit long response ([Table 134](#))

Table 133. Short response format

Bit Position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7(or 1111111)
0	1	1	End bit

Table 134. Long response format

Bit Position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127	-	CID or CSD (including internal CRC7)
0	1	1	End bit

- **Data:** data can be transferred from the card to the host or vice versa. Data is transferred via the data lines. They are stored in a FIFO of 32 words, each word is 32 bits wide.

Table 135. Data token format

Description	Start bit	Data	CRC16	End bit
Block data	0	-	yes	1
Stream data	0	-	no	1

The basic transaction on the MultiMediaCard/SD bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers to/from the SD Memory Cards are done in block mode. Data transfers to/from MMC are done in block or in stream mode. Data transfers to/from the CE-ATA Devices are done in block mode.

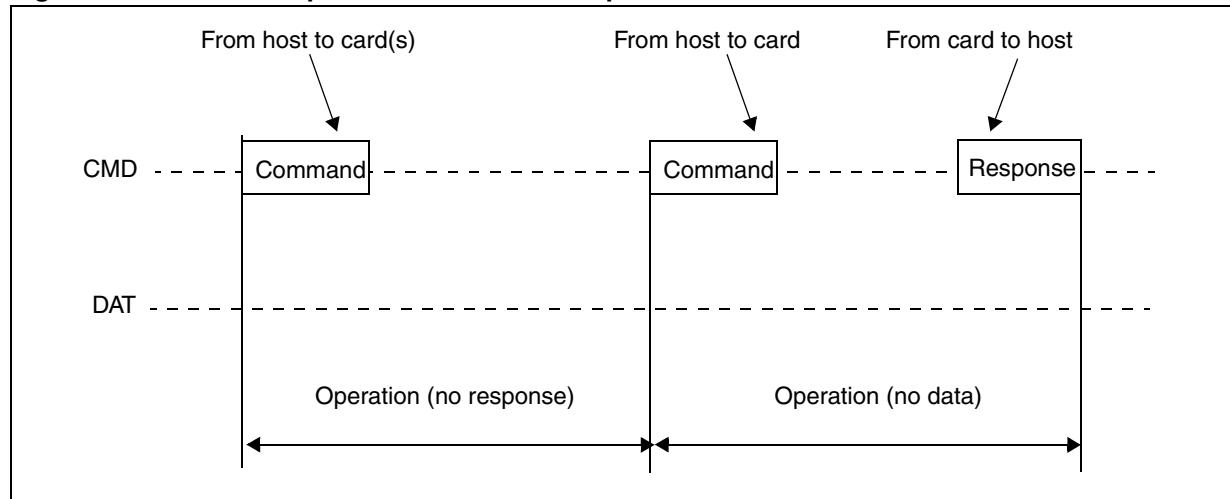
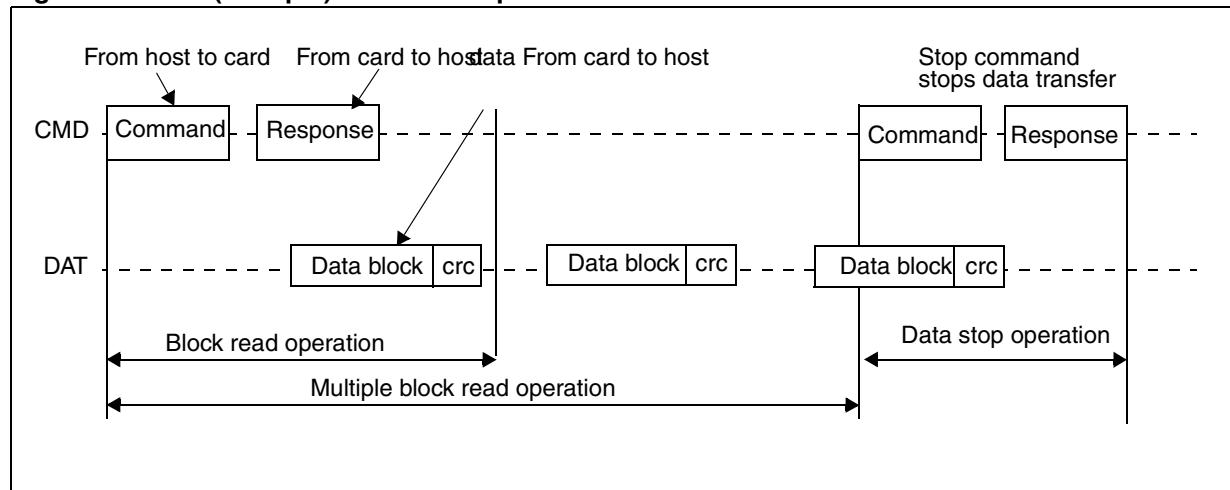
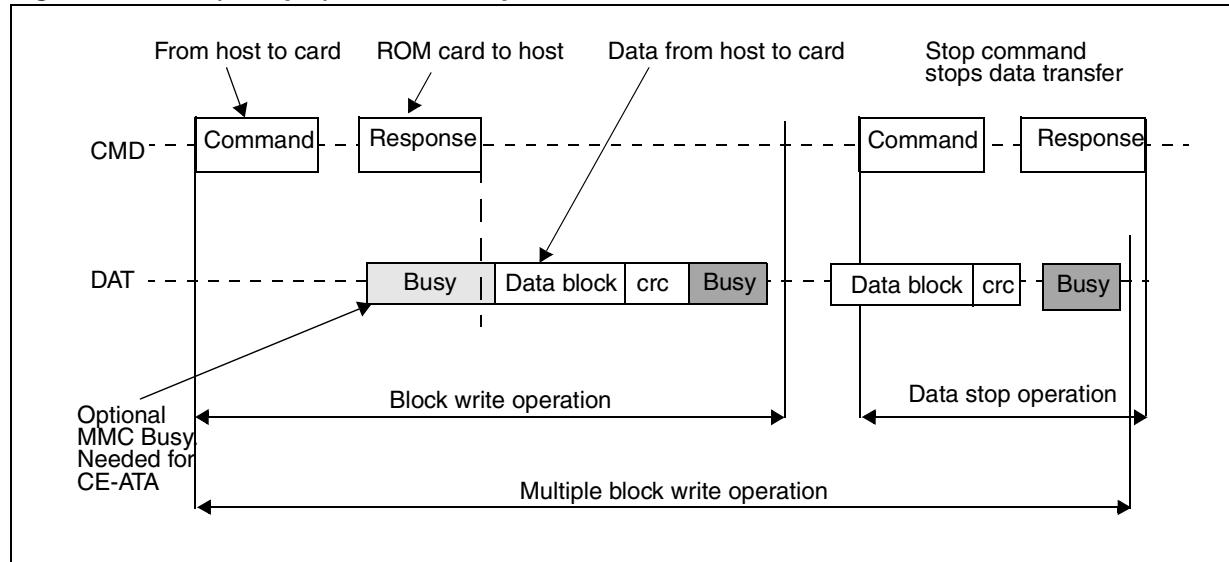
Figure 148. SDI “no response” and “no data” operations**Figure 149. SDI (multiple) block read operation**

Figure 150. SDI (Multiple) block write operation

SDI will not send any data as long as the MMCBusy is asserted.

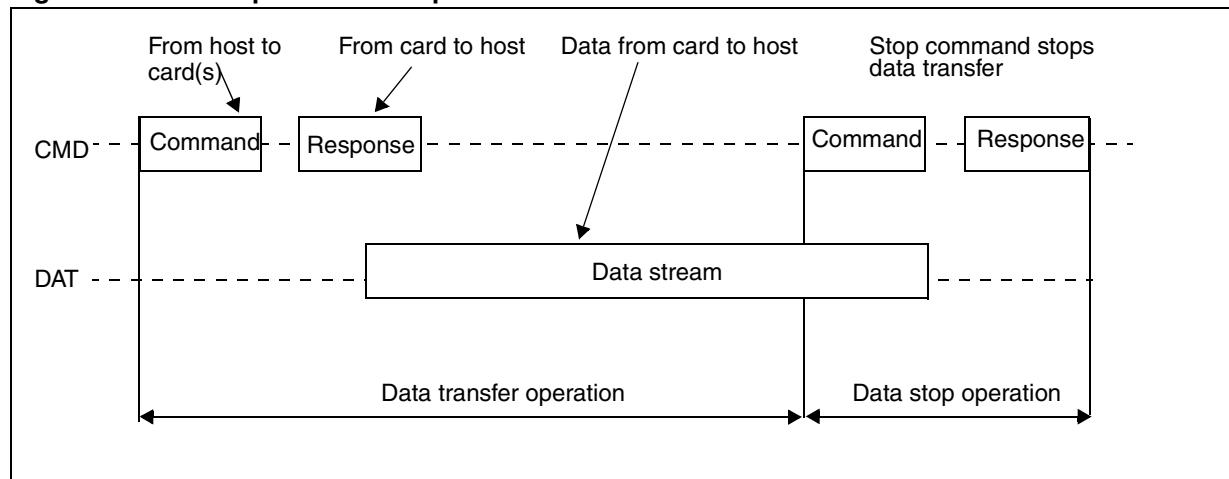
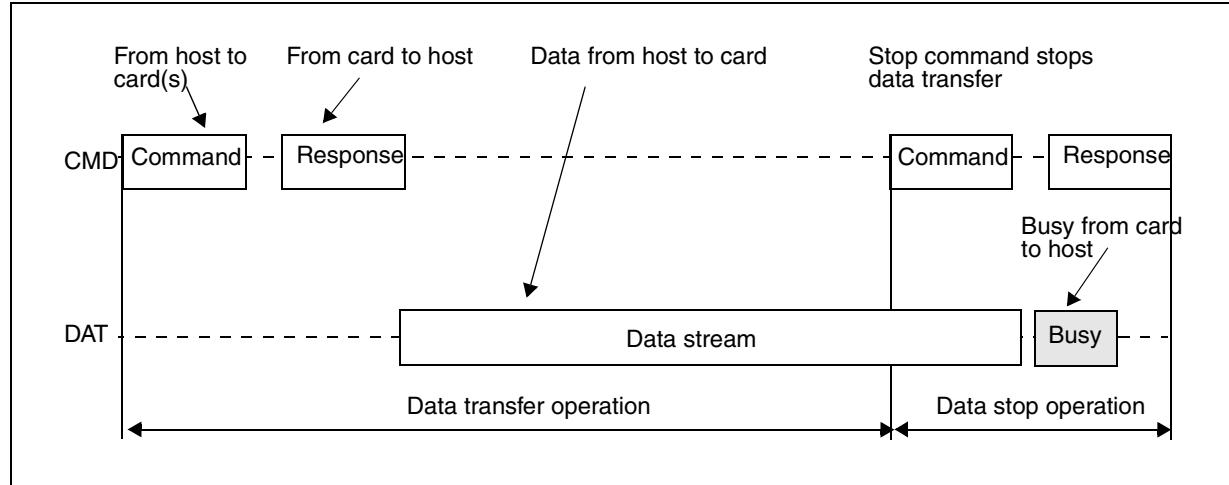
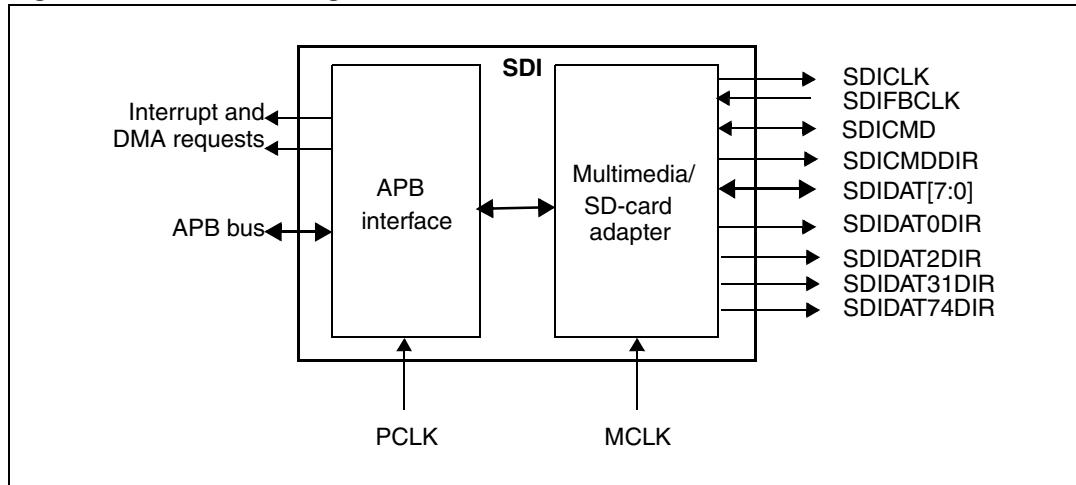
Figure 151. SDI sequential read operation

Figure 152. SDI sequential write operation

29.3 Functionality

The SDI consists of two parts:

- The SDI adapter block provides all functions specific to the MMC/SD card such as the clock generation unit, command and data transfer.
- The APB interface accesses the SDI adapter registers, and generates interrupt and DMA request signals.

Figure 153. SDI block diagram

By default **SDIDAT[0]** is used for data transfer. After initialization, the host can change the data bus width.

If a MultiMediaCard is connected to the bus, **SDIDAT[0]** or **SDIDAT[3:0]** or **SDIDAT[7:0]** can be used for data transfer. MMC v3.31 or previous supports only 1 bit of data, so only **SDIDAT[0]** can be used.

If an SD card is connected to the bus, data transfer can be configured by the host to use SDIDAT[0] or SDIDAT[3:0]. All data lines operate in push-pull mode.

SDICMD has two operational modes:

- Open-drain for initialization (only for MMCv3.31 or previous)
- Push-pull for command transfer (SD-Card MMC4.0 use push-pull drivers also for initialization)

SDICLK is the clock to the card: one bit is transferred on both command and data lines with each clock cycle. The clock frequency can vary between 0 and 20 MHz (for a MultiMedia card v3.31) and between 0 and 48 MHz for a MultiMediaCard v4.0, or 0 and 25 MHz (for an SD card)

The SDI uses two clock signals.

- SDI adapter clock (MCLK)
- APB bus clock (PCLK)

The relationship between MCLK and PCLK is defined below:

$$f_{PCLK} \geq 3/8 f_{MCLK}$$

29.4 SDI interface signals

The following signals are used on the MultiMediaCard/SD-Card bus:

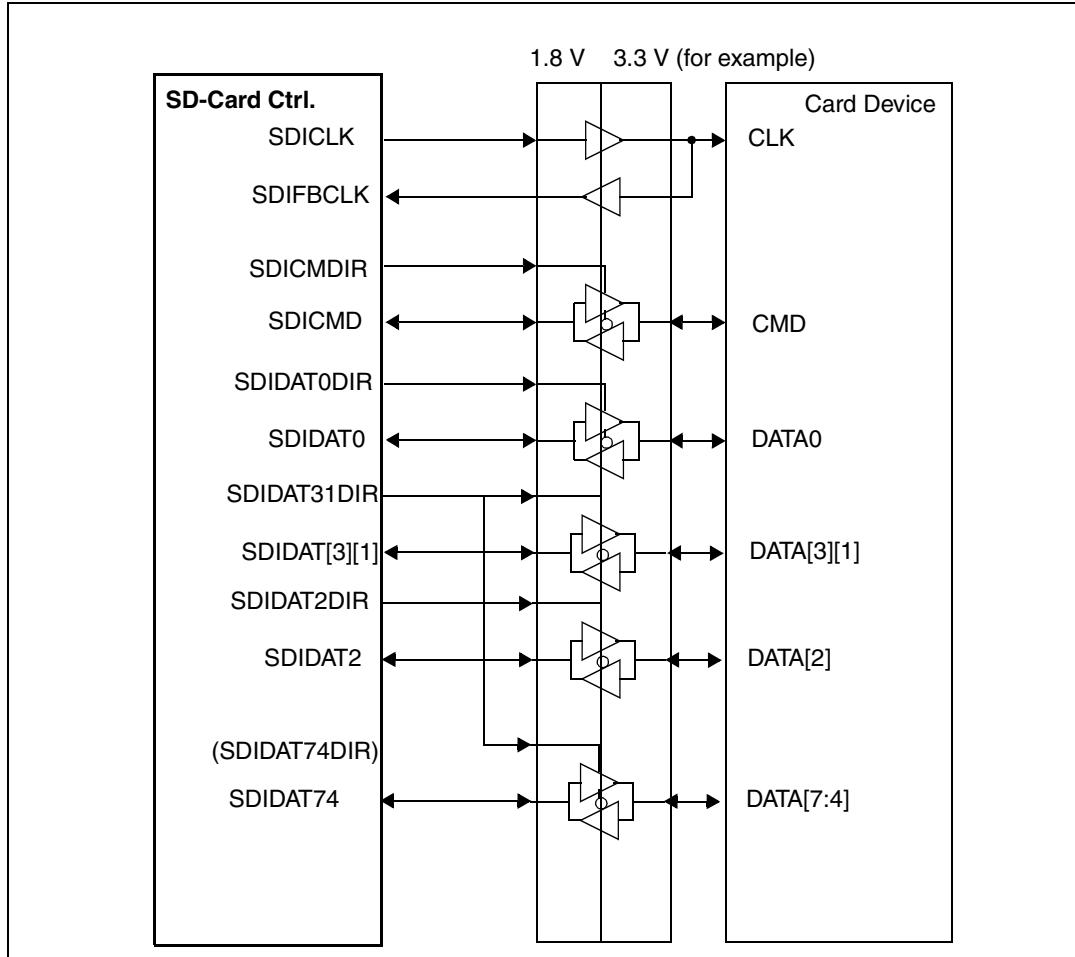
Table 136. SDI signal definition (GPIO alternate function enabled)

Pin	Direction	Description
SDICLK	Output	MultiMediaCard/SD card clock This signal is the clock from host to card.
SDIFBCLK	Input	MultiMediaCard/SD card feedback clock This signal is the feedback clock from card to host.
SDICMD	Bidirectional	MultiMediaCard/SD card command This signal is the bidirectional command/response signal.
SDIDAT[7:0]	Bidirectional	MultiMediaCard/SD card Data This signal is the bidirectional data bus.
SDICMDDIR	Output	SDICMD direction signal This signal is set HIGH when the SDICMD signal is driven by the SD-Card host interface.
SDIDAT0DIR	Output	SDIDAT0 direction signal This signal is set HIGH when the SDIDAT[0]signal is driven by the SD-Card host interface.
SDIDAT2DIR	Output	SDIDAT2direction signal This signal is set HIGH when the SDIDAT[2] signal is driven by the SDI
SDIDAT31DIR	Output	SDIDAT31 direction signal This signal is set HIGH when the SDIDAT[3]and SDIDAT[1] signals are driven by the SDI
SDIDAT74DIR	Output	SDIDAT74 direction signal This signal is set HIGH when the SDIDAT[7:4] signals are driven by the SDI ⁽¹⁾

1. This signal is not currently delivered. SDIDAT31DIR must be used instead.

Figure 154 shows the typical connection of the SDI to external buffers (voltage level shifter), the showed card device has all data lines.

Figure 154. SDI typical connection to external bi-directional drivers



Note:

SDIFBCLK is optional: an internal multiplexer allows selection between the internal SDICLK clock signal or an external SDIFBCLK signal to latch the incoming signals on MMC/SD-Card data bus. This multiplexer is controlled by bit 7 (FBCLKEN) of the SDI_PWR register.

The SDIFBCLK signal should be used to latch SD-Card input data bus when the propagation delays introduced by the external drivers are not compatible with AC timings of the SDI interface.

30 AC/DC characteristics

These characteristics are only guaranteed under the following operating conditions:

- Case temperature range (T_C): -25°C to 85°C
- Supply voltage range (V_{DD}): 1.14V to 1.47V

30.1 Absolute maximum ratings

Table 137. Absolute maximum ratings

Parameter	Symbol	Limits		Unit
		Minimum	Maximum	
I/O supply voltage for VDDIOC	V_{DDIOC}	$V_{SS} - 0.3$	$V_{SS} + 2.3$	V
I/O supply voltage for VDDIOD	V_{DDIOD}	$V_{SS} - 0.3$	$V_{SS} + 2.7$	V
Other I/O supply voltages	V_{DDIO}	$V_{SS} - 0.3$	$V_{SS} + 3.2$	V
Logic supply voltage	V_{DD_CORE}	$V_{SS} - 0.3$	$V_{SS} + 1.6$	V
PLL supply voltage	V_{DDA}	$V_{SS} - 0.3$	$V_{SS} + 3.2$	V
OTP supply voltage	V_{OTP}	$V_{SS} - 0.3$	$V_{SS} + 3.2$	V
Voltage applied to any pin	V_{IP}	$V_{SS} - 0.3$	$V_{SS} + 3.2$	V
Storage temperature	T_{AST}	-40	125	°C
Electrostatic discharge, human body model (condition: leakage < 1µA)	V_{ESD}	-2000	+2000	V
Electrostatic discharge, robotic charge device model (condition: leakage < 1µA)		- 500	+ 500	

Note:

Stresses above those listed in Table 137 may cause permanent damage to the device. These are stress ratings only and operation of the device in these or more extreme conditions is not advised. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

30.2 Electrostatic discharge (ESD) standards

The STn8815 complies with the following EMC standards:

- ANSI ESD STM5.3.1(1999): Charged device model (CDM)
- EIA/JESD22-A114-B (June 2000): Human body model (HBM)

30.3 Recommended DC operating conditions

Table 138. Recommended DC operating conditions

Parameter	Symbol	Limits			Unit
		Min.	Typical	Max.	
I/O supply voltage for VDDIO[C/F] ⁽¹⁾	V _{DDIO}	1.70	1.80	1.90	V
I/O supply voltage for VDDIO[A/B/D/E]	V _{DDIO}	1.70	-	2.70	V
Logic supply voltage ⁽²⁾	V _{DD_CORE}	1.14	1.20	1.36	V
Logic supply voltage (overdrive)	V _{DD_CORE}	1.26	1.30	1.47	V
PLL supply voltage	V _{DDA}	1.70	1.80	1.90	V
OTP supply voltage	V _{OTP}	2.25	2.50	2.75	V
Supply ground	V _{SS}	0	0	0	V
Operating case temperature	T _c ⁽³⁾	-25		85	°C

1. If the SubLVDS I/O buffers are not used, the VDDIOF power pins can use the full range (1.70V to 2.70V).

2. V_{DD_CORE} only supplies the multimedia processor die. V_{DD_CORE} = VDD12

3. T_c = operating case temperature

Warning: All AC/DC characteristics provided in the following subsections are quoted for use within the limited I/O supply voltage range of 1.70V to 1.90V for the VDDIOA, VDDIOB, VDDIOC, VDDIOD, VDDIOE and VDDIOF power domains.

30.4 DC characteristics

Table 139. DC operating characteristics (functional pins)

Parameter	Symbol	Limits			Unit	Test condition
		Minimum	Typical	Maximum		
Low level input voltage ⁽¹⁾	V _{IL}	-0.3		0.35 V _{DDIO}	V	—
High level input voltage	V _{IH}	0.65 V _{DDIO}		V _{DDIO} + 0.3	V	—
Schmitt-trigger hysteresis ⁽²⁾	V _{HYST}	0.40		0.60	V	—
Low level output voltage ⁽³⁾	V _{OL}			0.45	V	I _{OL} = x mA ⁽⁴⁾
High level output voltage ⁽³⁾	V _{OH}	V _{DDIO} - 0.45			V	I _{OH} = x mA ⁽⁴⁾
Low level output voltage (JDEC wide range)	V _{OL}			0.2	V	I _{OL} = 100 µA
High level output voltage (JDEC wide range)	V _{OH}	V _{DDIO} - 0.2			V	I _{OH} = 100 µA
Pull-down current	I _{PD}	22	50	94	µA	V _{IN} = V _{DDIO}
Pull-up current	I _{PU}	40	52	86	µA	V _{IN} = 0 V
Input leakage current	I _{OZ}	-1		1	µA	0 < V _{IN} < V _{DDIO}
Input pin capacitance ⁽⁵⁾	C _I			3	pF	f = 1 MHz T _c ⁽⁶⁾ = 25 °C
Output pin capacitance ⁽⁵⁾	C _O			3	pF	f = 1 MHz T _c ⁽⁶⁾ = 25 °C
Operating CPU frequency	F _{CLK}			264	MHz	@V _{DD_CORE} = 1.14V T _c ⁽⁶⁾ = 85 °C
Operating CPU frequency	F _{CLK}			332	MHz	@V _{DD_CORE} = 1.26V T _c ⁽⁶⁾ = 85 °C

1. Except oscillator inputs SXTALI and MXTALI. Refer to the oscillator electrical specifications.

2. Applies to all I/Os except SubLVDS (CSI signals), SXTAL and MTAL I/Os

3. JDEC normal range.

4. See [Table 5: Pin characteristics on page 59](#)

5. Parameter guaranteed by design.

6. T_c = operating case temperature

Table 140. DC operating characteristics (power pins)

Parameter	Symbol	Limits			Unit	Test condition
		Minimum	Typical	Maximum		
Analog power (total V_{DDA})	P_{ANA}			5 ⁽¹⁾	mA	$T_C^{(2)} = 25^\circ\text{C}$ $V_{DDA} = 1.8\text{ V}$
Normal mode supply current (total $V_{DD_CORE} + V_{DDIO}$)	P_{NORMAL}		250 ⁽³⁾	500 ⁽⁴⁾	mA	@ 264 MHz, $T_C^{(2)} = 25^\circ\text{C}$ $V_{DD_CORE} = 1.2\text{ V}$ $V_{DDIO} = 1.8\text{ V}$
Slow mode supply current (total $V_{DD_CORE} + V_{DDIO}$)	P_{SLOW}		12	25	mA	@ 19.2 MHz, $T_C^{(2)} = 25^\circ\text{C}$ $V_{DD_CORE} = 1.2\text{ V}$ $V_{DDIO} = 1.8\text{ V}$
Input bias DC power consumption, operating mode	$PLVDSEON$	0.5	0.7	1.0	mW	CSI enabled by SAA
Input bias DC power consumption, power down mode	$PLVDSEOFF$		50		mW	
Doze mode supply current (V_{DD_CORE})	P_{DOZE}		5	15	mA	@ 32.768 kHz, $T_C^{(2)} = 25^\circ\text{C}$ $V_{DD_CORE} = 1.2\text{ V}$ $V_{DDIO} = 1.8\text{ V}$
Sleep mode supply current (V_{DD_CORE})	I_{SLEEP}		5	15	mA	
Deep-sleep mode supply current (V_{DD_CORE})	I_{DSLEEP}			170	μA	

1. With PLL1 and PLL2 in lock condition, 19.2 MHz input clock,
 $f_{PLL1} = 264\text{ MHz}$ ($f_{VCO1} = 1056\text{ MHz}$), $f_{PLL2} = 864\text{ MHz}$ ($f_{VCO2} = 864\text{ MHz}$)

2. T_C = Operating case temperature

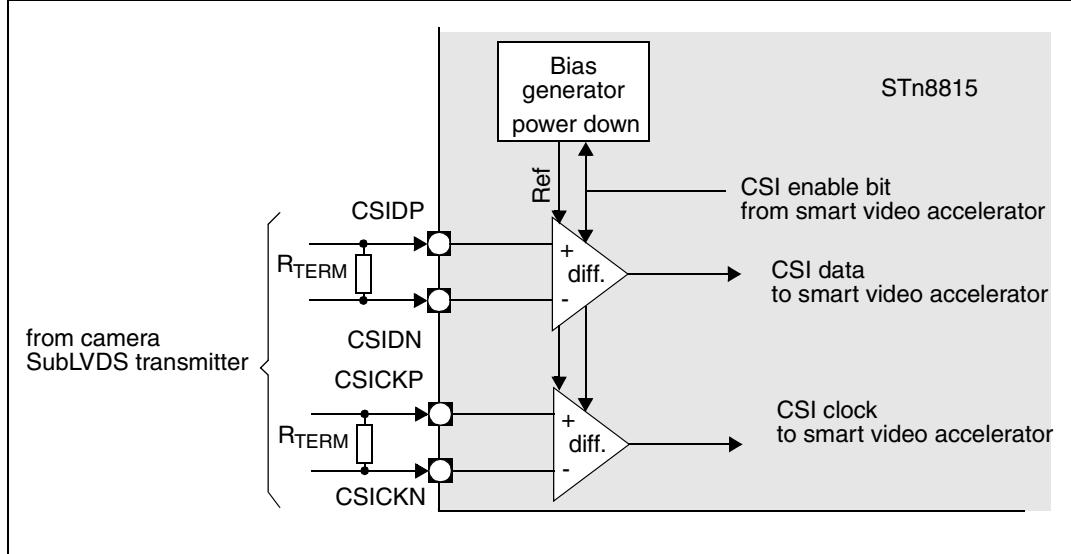
3. Stopped peripherals, CPU caches enabled with a miss rate of approximately 10%, smart audio accelerator and smart video accelerator stopped, program and data in external DDR-SDRAM (at 133 MHz).

4. All peripherals with maximum activity (worst case estimated from design).

SubLVDS receiver characteristics

Table 141 lists the functional operating characteristics for the SubLVDS receiver inputs of the CSI (serial camera interface). SubLVDS provides 1.8 V operated differential signaling for short distances, allowing data rates up to 208 MHz with low EMI.

The signal on CSI differential inputs is understood as logic 1 when the current flows from the positive (non inverting, CSIDP and CSICKP) terminal to the negative (inverting, CSIDN and CSICKN) terminal, and as logic 0 when the current flows in the opposite direction.

Figure 155. CSI SubLVDS input connections**Table 141. SubLVDS DC characteristics**

Parameter	Symbol	Limit values			Unit	Test condition
		Min	Typ	Max		
Input voltage range	V _{LVDSIN}	V _{DDIO} /2 - 0.4	V _{DDIO} /2	V _{DDIO} /2 + 0.4	V	
Input differential threshold ($V_{CSIxP} - V_{CSIxN}$)	V _{LVDSTHDIF}	50		200	mV	
Termination resistor value	R _{TERM}	80	100	150	Ω	
Operating frequency	F _{LVDS}			325	MHz	
Power up time	T _{PWRUP}			5	μs	When SVA enables the CSI interface
Input leakage current	I _{OZ}			0.1	μA	0 < V _{IN} < V _{DDIO} , T _C ⁽¹⁾ = 25 °C, for one differential input pair
Input pin capacitance (guaranteed by design)	C _I			3	pF	f = 1 MHz, T _C ⁽¹⁾ = 25 °C

1. T_C = Operating case temperature

30.5 AC characteristics

The AC test conditions are shown in [Table 142](#).

Table 142. AC test conditions

Parameter	Value
Input pulse levels	0 to +1.7 V
Input rise and fall times (10% to 90%)	2.0 ns
Input timing reference levels	+0.85 V
Output timing reference levels	+0.85 V
Output load	See Figure 157 .

Figure 156. AC test conditions: input waveforms

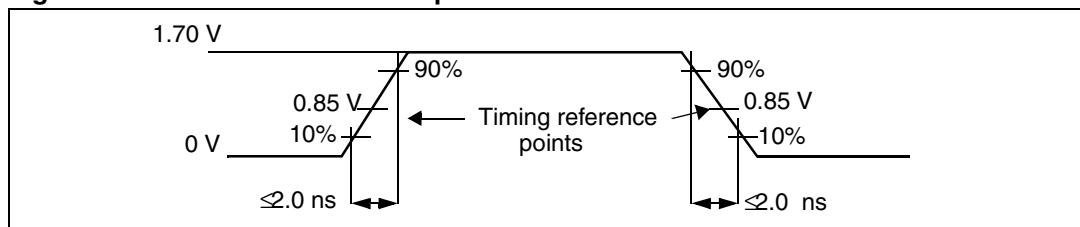
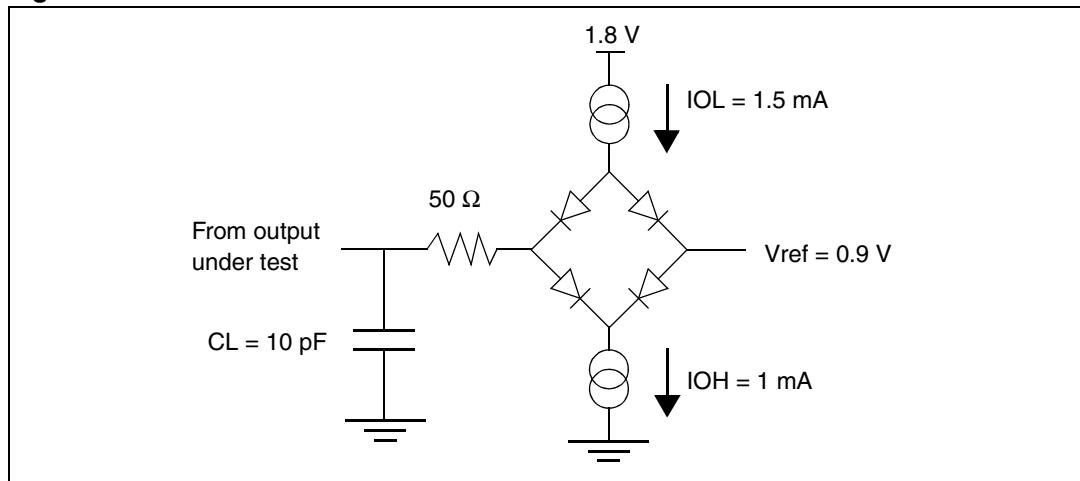


Figure 157. AC test conditions: load circuit



30.6 Internal clock cycle definition

The internal clock is referred to hereafter as the CPU clock. Its frequency is derived from the input clock (on pad CLKIN) according the configuration of the PLL.

Its period is denoted t_{CPU} and can vary from 3.79 ns (264 MHz) to infinity (0 Hz).

The AHB clock is derived from the CPU clock by a programmable divider (divide by 1, 2, 3 or 4). Its period, t_{AHB} , can vary from 7.57 ns (132 MHz) to infinity (0 Hz).

30.7 Oscillator electrical specifications

This device contains two oscillators:

- a 32.768 kHz oscillator,
- a 13 to 26 MHz oscillator.

Each requires a specific crystal, with parameters that must be as close as possible to the recommended values given below.

30.7.1 32.768 kHz oscillator specifications

The 32.768 kHz oscillator is connected between SXTALI (oscillator amplifier input) and SXTALO (oscillator amplifier output). It also requires two external capacitors of 27 pF, as shown in *Figure 158*. The recommended oscillator specifications are shown in *Table 143*.

Table 143. Recommended oscillator specifications (32.768 kHz oscillator)

Parameter	Symbol	Limits			Unit
		Min.	Typical	Max.	
Crystal frequency	F_{SXTAL}	-	32.768	-	kHz
Motion inductance	L_{mSXTAL}	-	11.8	-	kHz
Motional capacitance	C_{mSXTAL}	-	2.0	20.0	fF
Motional resistance	R_{mSXTAL}	30	60	80	kΩ
Shunt capacitance	C_{oSXTAL}	-	1.5	-	pF
Load capacitance	C_{lSXTAL}	-	27	-	pF

The oscillator amplifier specifications are shown in *Table 144*.

Table 144. Oscillator amplifier specifications (32.768 kHz oscillator)

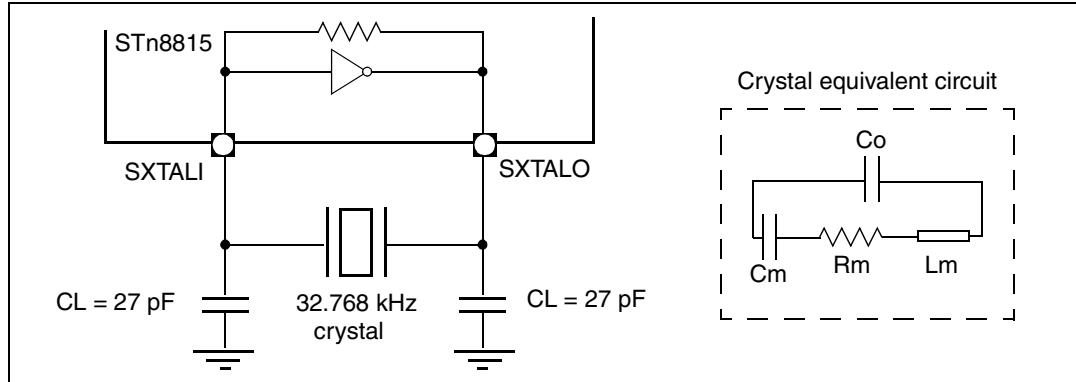
Parameter	Symbol	Limits			Unit
		Min.	Typical	Max.	
Low level input voltage	V_{ILSXTAL}	0.0	0.38	-	V
High level input voltage	V_{IHSXTAL}	-	-	1.95	V
Input hysteresis	V_{HYSTXTAL}	-	0.3	-	V
Startup time	T_{SXTAL}	0.5	-	5	s
Transconductance ⁽¹⁾	G_{mSXTAL}	21.3	-	-	μA/V

1. The transconductance of the crystal oscillator should be 3 times the transconductance calculated with the following formula (in the worst case):

$$G_{\text{mSXTAL}} = R_{\text{mSXTAL}} \cdot \omega^2 \cdot (C_{\text{lSXTAL}} + 2 C_{\text{oSXTAL}})^2$$

To drive the 32.768 kHz crystal pins from an external clock source:

- Disable the oscillator. This disables the internal inverter, thus reducing the power consumption to a minimum. This also allows the SXTALI input to be driven, even when a crystal is connected between the SXTALI and SXTALO pins.
- Drive the SXTALI pin with a square signal that has a low level V_{ILXTAL} and a high level V_{IHXTAL} , or a sine wave (maximum amplitude: $V_{\text{IHXTAL}}/2$; offset: $V_{\text{IHXTAL}}/2$).

Figure 158. 32.768 kHz crystal connection

30.7.2 13 to 26 MHz oscillator specifications

The 13 to 26 MHz oscillator is connected between MXTALI (oscillator amplifier input) and MXTALO (oscillator amplifier output). It also requires two external capacitors of 16 pF, as shown in [Figure 159](#). The recommended oscillator specifications are shown in [Table 145](#).

Table 145. Recommended oscillator specifications (13 to 26 MHz oscillator)

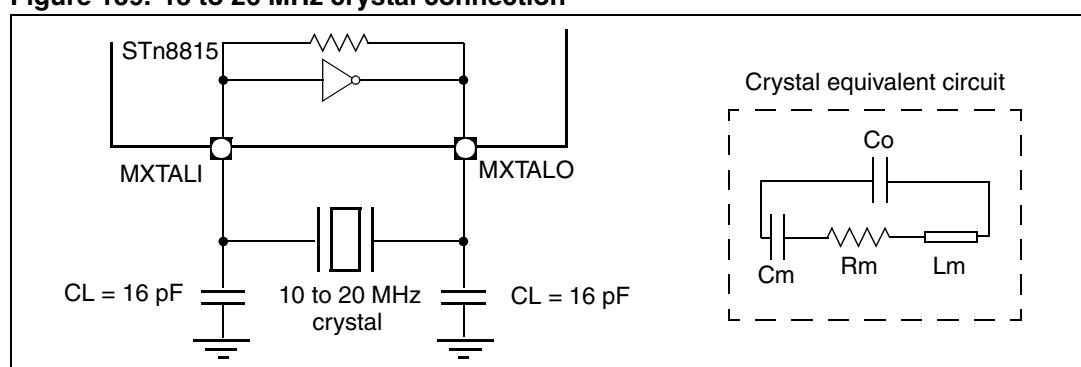
Parameter	Symbol	Limits			Unit
		Min.	Typical	Max.	
Crystal frequency	F_{MXTAL}	13	19.2	26	MHz
Motion inductance	L_m_{MXTAL}	-	5.6	-	MHz
Motional capacitance	C_m_{MXTAL}	-	7.9	20.0	fF
Motional resistance	R_m_{MXTAL}	-	8	50	Ω
Shunt capacitance	C_o_{MXTAL}	-	4.4	-	pF
Load capacitance	$C_{\text{L}}_{\text{MXTAL}}$	-	16	-	pF

The oscillator amplifier specifications are shown in [Table 146](#).

Table 146. Oscillator amplifier specifications (13 to 26 MHz oscillator)

Parameter	Symbol	Limits			Unit
		Min.	Typical	Max.	
Low level input voltage	$V_{ILMXTAL}$	0.0	0.38	-	V
High level input voltage	$V_{IHMXTAL}$	-	-	1.95	V
Input hysteresis	$V_{HYSTXTAL}$	-	0.3	-	V
Start-up time	T_{MXTAL}	0.5	-	50	ms
Transconductance ⁽¹⁾	Gm_{MXTAL}	1060	-	-	$\mu A/V$

1. The transconductance of the crystal oscillator should be 3 times the transconductance calculated with the following formula, in the worst case:
 $Gm_{MXTAL} = Rm_{MXTAL} \cdot \omega^2 \cdot (CL_{MXTAL} + 2 \cdot C_{oMXTAL})^2$

Figure 159. 13 to 26 MHz crystal connection

To drive the 13 to 26 MHz crystal pins from an external clock source:

- Disable the oscillator. This disables the internal inverter, thus reducing the power consumption to a minimum. This also allows the MXTALI input to be driven, even when a crystal is connected between the MXTALI and MXTALO pins.
- Drive the MXTALI pin with a square signal that has a low level $V_{ILMXTAL}$ and a high level V_{IHXTAL} , or a sine wave (maximum amplitude $V_{IHXTAL} / 2$, offset $V_{IHXTAL} / 2$).

30.7.3 Reset timings

This section provides the timing requirement for the assertion of PORn pin in the following situations:

- Power-on device reset (that is, starting from non-stabilized supplies),
- Hardware device reset (that is, with already stabilized supplies).

30.7.4 Timing requirements for device power-on reset

During power-on reset, there are no strict power supply sequencing constraints, but to avoid the PLLs to try locking on non-yet existing input clock signal, it is recommended to supply VDD12 at same time or before VDDA. This limits the power-consumption of the PLL to the minimum. If the same voltage regulator is supplying the I/Os and the PLL analog part, then the recommended sequencing is VDD12 applied at same time or before VDD18/VDDA.

The timing requirements in this section assume stable power supplies at the assertion of PORn and TRSTn signals.

Table 147. Timing requirements for device power-on reset

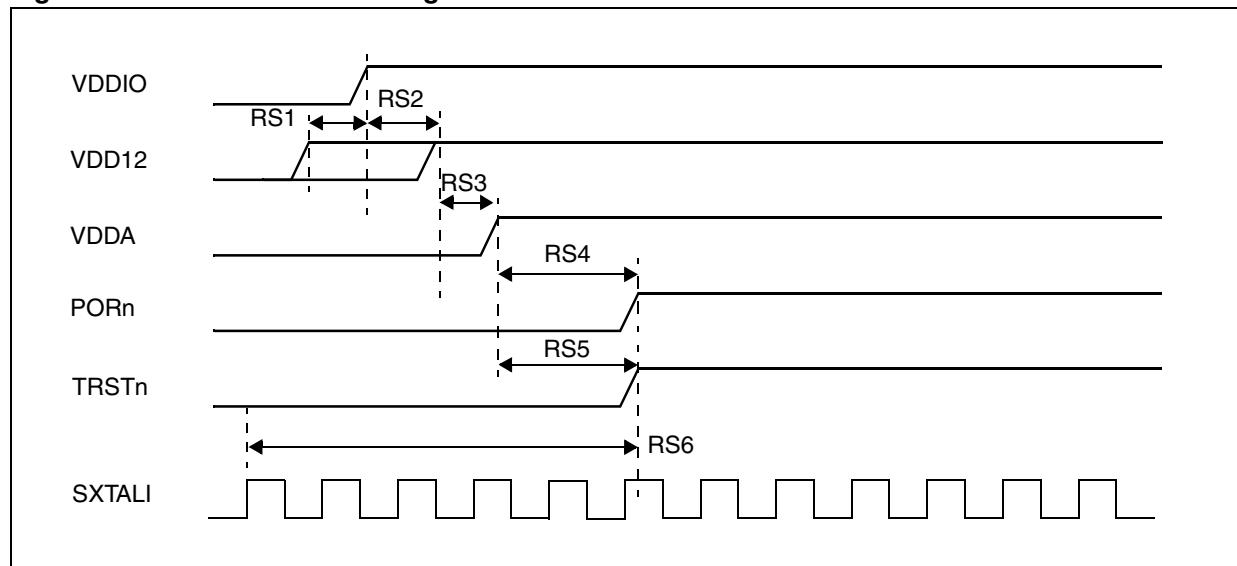
No.	Parameter	Symbol	Timing		Unit
			min.	max.	
RS1	VDD12 stable to VDDIO stable ⁽¹⁾	td(VDD12V - VDDIOV)	0	-	ns
RS2	VDDIO stable to VDD12 stable ⁽¹⁾	td(VDDIOV - VDD12V)	0	-	ns
RS3	VDD12 stable to VDDA stable ⁽²⁾	td(VDD12V - VDDAV)	0	-	ns
RS4	VDDIO, VDDA, VDD12 stable to PORn high	td(VxV - PORnH)	1000	-	ns
RS5	VDDIO, VDDA, VDD12 stable to TRSTn high	td(VxV - TRSTnH)	1000	-	ns
RS6	SXTALI cycles before PORn rising edge ⁽³⁾		5	-	cycle

1. RS1 + RS2 denotes that there are no constraints on power-on (and power-off) sequencing between VDDIO (including VDDIOext) and VDD12 voltages.

2. VDDA voltage should be applied at the same time or after the VDD12 voltage, and be turned off at the same time or before VDD12 to avoid unpredictable power consumption of the PLLs.

3. SXTALI must be stable and have at least 5 cycles before PORn release.

Figure 160. Power-on reset timings



30.8 Flexible static memory controller (FSMC) timings

All timing characteristics are relative to the SMCKO signal for muxed NOR-Flash memory accesses, and from the internal AHB clock for CompactFlash/CF+/NAND-Flash memory accesses.

30.8.1 Switching characteristics for muxed memory read and write cycles

Table 148. Switching characteristics for muxed memory read and write cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
FS0	SMCKO period	tw(CKO)	15 (sync)	-	ns
			7.5 (async)	-	
	SMCKO period (overdrive mode)	tw(CKO)	12 (sync)	-	ns
			6 (async)	-	
FS1	SMCKO high to SMCSxn low (x = 0...2)	td(CKOH-CSnL)	-	4	ns
FS2	SMCKO high to SMCSxn high (x = 0...2)	td(CKOH-CSnH)	0	-	ns
FS3	SMCKO high to SMADVn low	td(CKOH-ADVnL)	-	4	ns
FS4	SMCKO high to SMADVn high	td(CKOH-ADVnH)	0	-	ns
FS5	SMCKO high to SMADx valid (x = 16...25)	td(CKOH-ADV)	-	4	ns
FS6	SMCKO high to SMADx invalid (x = 16...25)	td(CKOH-ADIV)	0	-	ns
FS7	SMCKO high to SMWEn low	td(CKOH-WEnL)	-	4	ns
FS8	SMCKO high to SMWEn high	td(CKOH-WEnH)	0	-	ns
FS9	SMCKO high to SMOEn low	td(CKOH-OEnL)	-	4	ns
FS10	SMCKO high to SMOEn high	td(CKOH-OEnH)	0	-	ns
FS11	SMCKO high to SMADQ[15:0] valid	td(CKOH-ADQV)	-	4	ns
FS12	SMCKO high to SMADQ[15:0] invalid	td(CKOH-ADQIV)	0	-	ns

30.8.2 Timing requirements for muxed memory read cycles

Table 149. Timing requirements for muxed memory read cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
FS13	SMADQ[15:0] valid data before SMCKO high – in sync. burst mode – in async. mode	tsu(ADQV - CKOH)	2 8	- -	ns ns
FS14	SMADQ[15:0] valid data after SMCKO high – in sync. burst mode – in async. mode	th(CKOH - ADQV)	2 0	- -	ns ns
FS15	SMWAITn valid before SMCKO high – in sync. burst mode – in async. mode	tsu(WAITnV - CKOH)	2 8	- -	ns ns
FS16	SMWAITn valid after SMCKO high – in sync. burst mode – in async. mode	th(CKOH - WAITnV)	2 0	- -	ns ns

Figure 161. Muxed NOR Flash memory asynchronous read

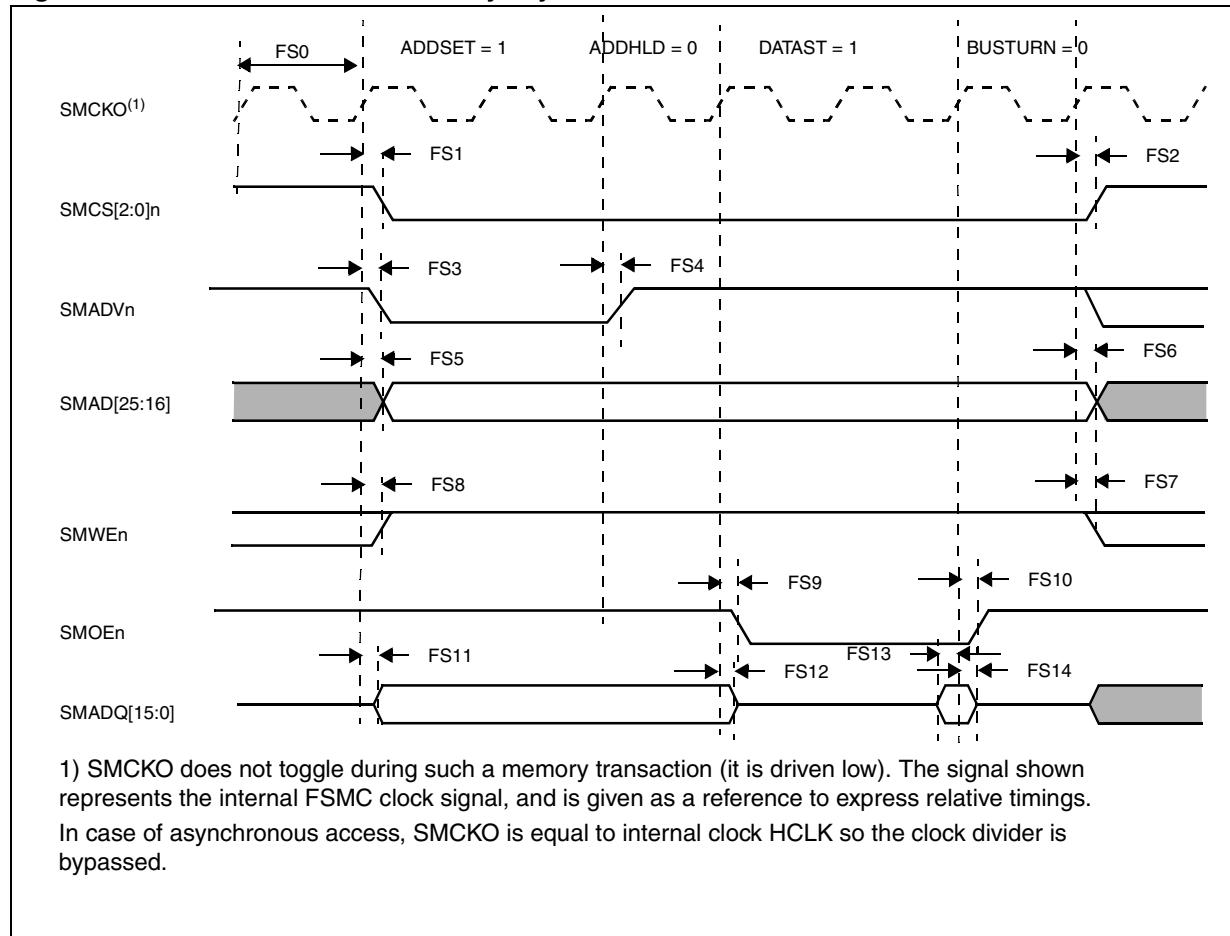


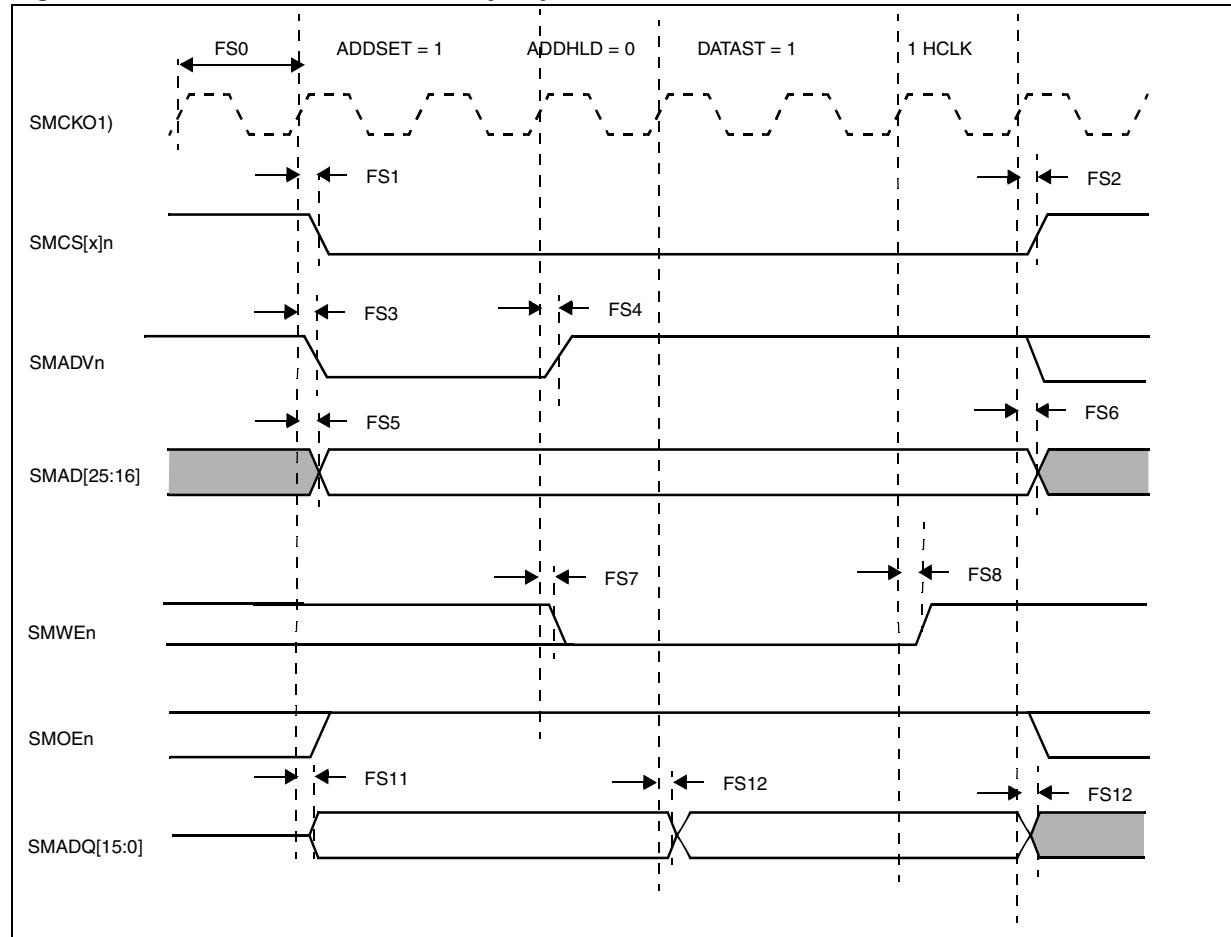
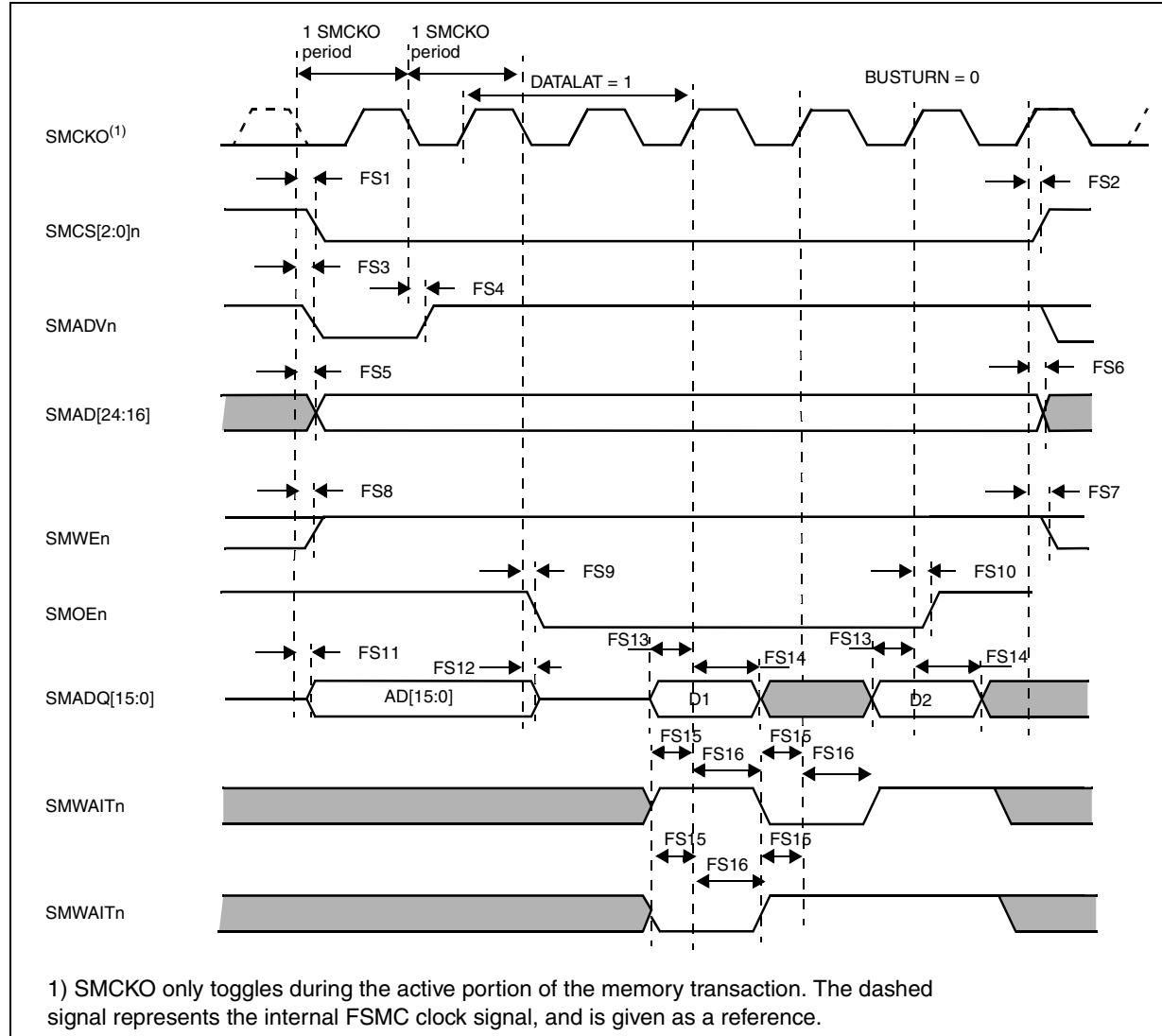
Figure 162. Muxed NOR Flash memory asynchronous write

Figure 163. Muxed NOR Flash memory synchronous read (CLKDIV > 0)

Note: All the timing characteristics are relative to the SMCKO signal.

30.8.3 Switching characteristics for CF/NAND Flash read and write cycles

Table 150. Switching characteristics for CF/NAND-Flash read and write cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
FS20	HCLK high to SMPSxn low (x = 0...1)	td(HCLKH-CSnL)	-	4	ns
FS21	HCLK high to SMPSxn high (x = 0...1)	td(HCLKH-CSnH)	0	-	ns
FS22	HCLK high to SMADx valid (x = 0...10)	td(HCLKH-ADV)	-	4	ns
FS23	HCLK high to SMADx invalid (x = 0...10)	td(HCLKH-ADIV)	-0.2	-	ns
FS24	HCLK high to SMPCExn valid (x = 1...2)	td(HCLKH-CEnV)	-	4	ns
FS25	HCLK high to SMPCExn invalid (x = 1...2)	td(HCLKH-CEnIV)	0	-	ns
FS26	HCLK high to SMPREGn low	td(HCLKH-IOREGnL)	-	4	ns
FS27	HCLK high to SMPREGn high	td(HCLKH-IOREGnH)	0	-	ns
FS28	HCLK high to SMIORn low	td(HCLKH-IORnL)	-	4	ns
FS29	HCLK high to SMIORn high	td(HCLKH-IORnH)	0	-	ns
FS30	HCLK high to SMIOWn low	td(HCLKH-IOWnL)	-	4	ns
FS31	HCLK high to SMIOWn high	td(HCLKH-IOWnH)	0	-	ns
FS32	SMOEn high to low jitter	td(OEnH-OEnL)	0	0.1	ns
FS33	SMOEn low to high jitter	td(OEnL-OEnH)	0	0.1	ns
FS34	HCLK high to SMWEn low	td(HCLKH-WEnL)	-	4	ns
FS35	HCLK high to SMWEn high	td(HCLKH-WEnH)	0	-	ns
FS36	HCLK highto SMADQ[15:0] valid	td(HCLKH-ADQV)	-	4.5	ns
FS37	HCLK high to SMADQ[15:0] invalid	td(HCLKH-ADQIV)	0	-	ns

30.8.4 Timing requirements for CF/NAND Flash memory read cycles

Table 151. Timing requirements for CF/NAND-Flash memory read cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
FS38	SMADQ[15:0] valid data before HCLK high	tsu(ADQV-HCLKH)	7.8		ns
FS39	SMADQ[15:0] valid data after HCLK high	th(HCLKH-ADQV)	0	-	ns
FS40	SMWAITn valid before HCLK high	tsu(WAITnV-HCLKH)	2	-	ns
FS41	SMWAITn valid after HCLK high	th(HCLKH-WAITnV)	2	-	ns
FS42	SMPIOIS16n valid before HCLK high	tsu(IOIS16nV - HCLKH)	6	-	ns

Figure 164. PC-Card controller timing for common memory read access

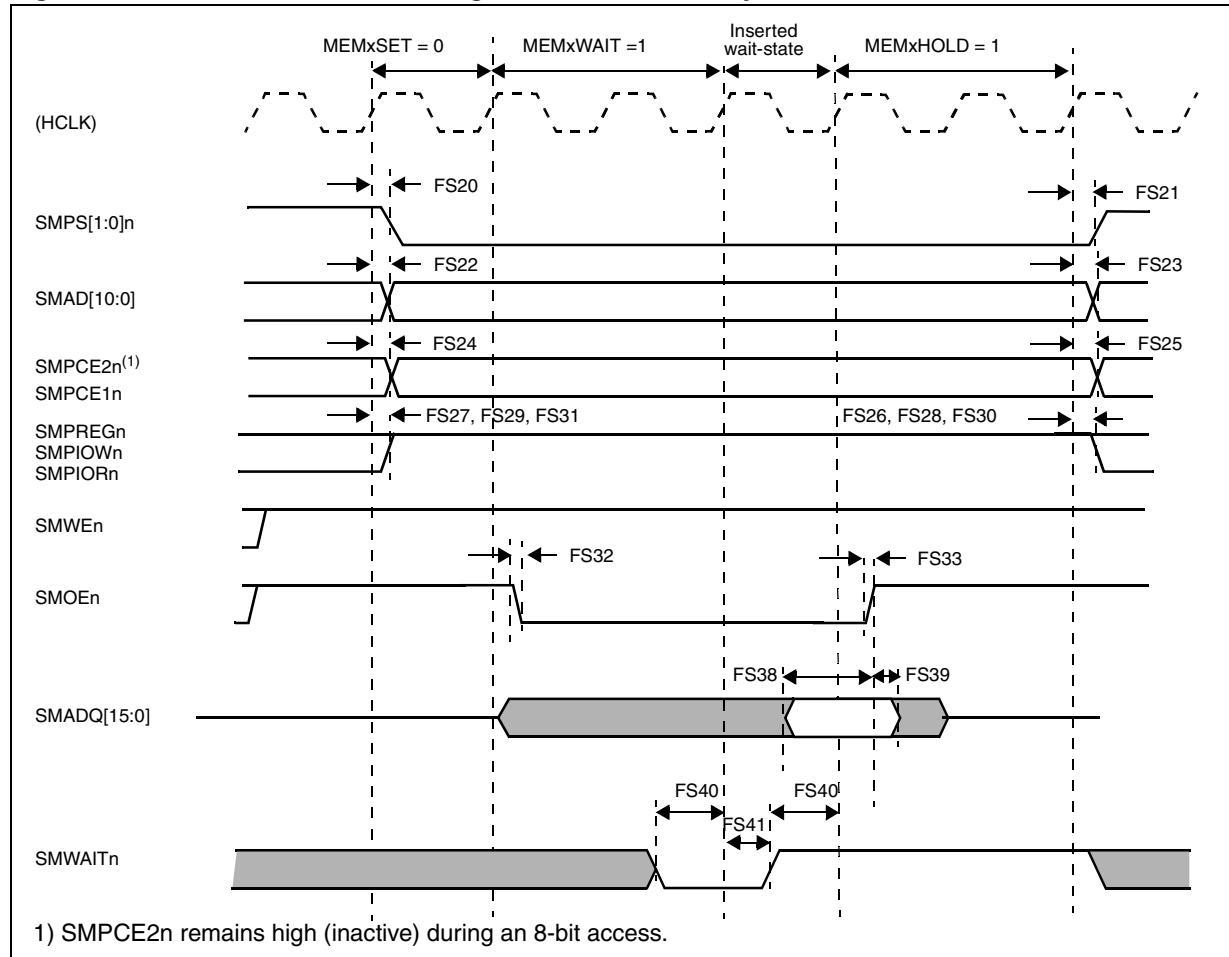


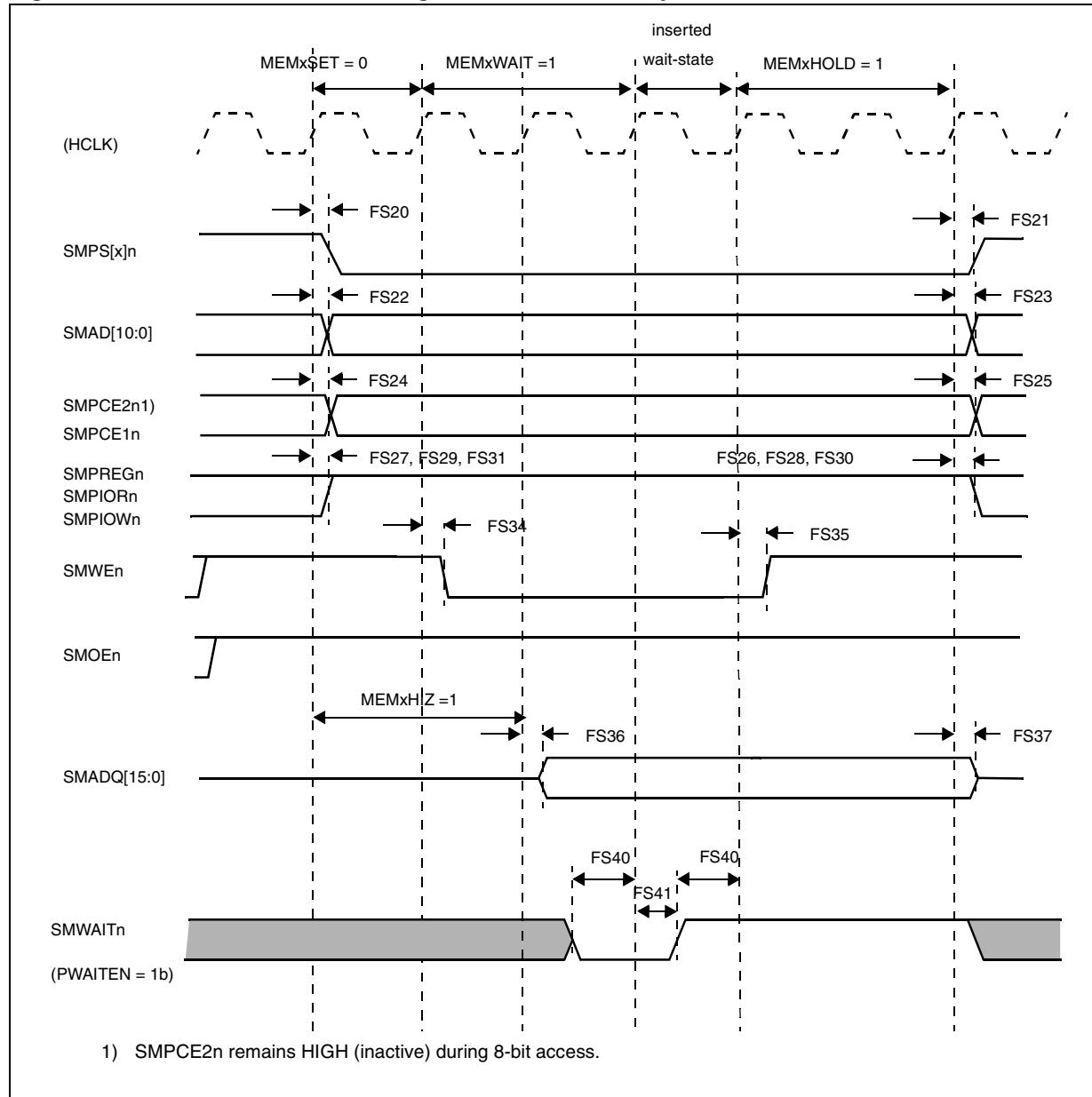
Figure 165. PC-Card controller timing for common memory write access

Figure 166. PC-Card controller timing for attribute memory read access

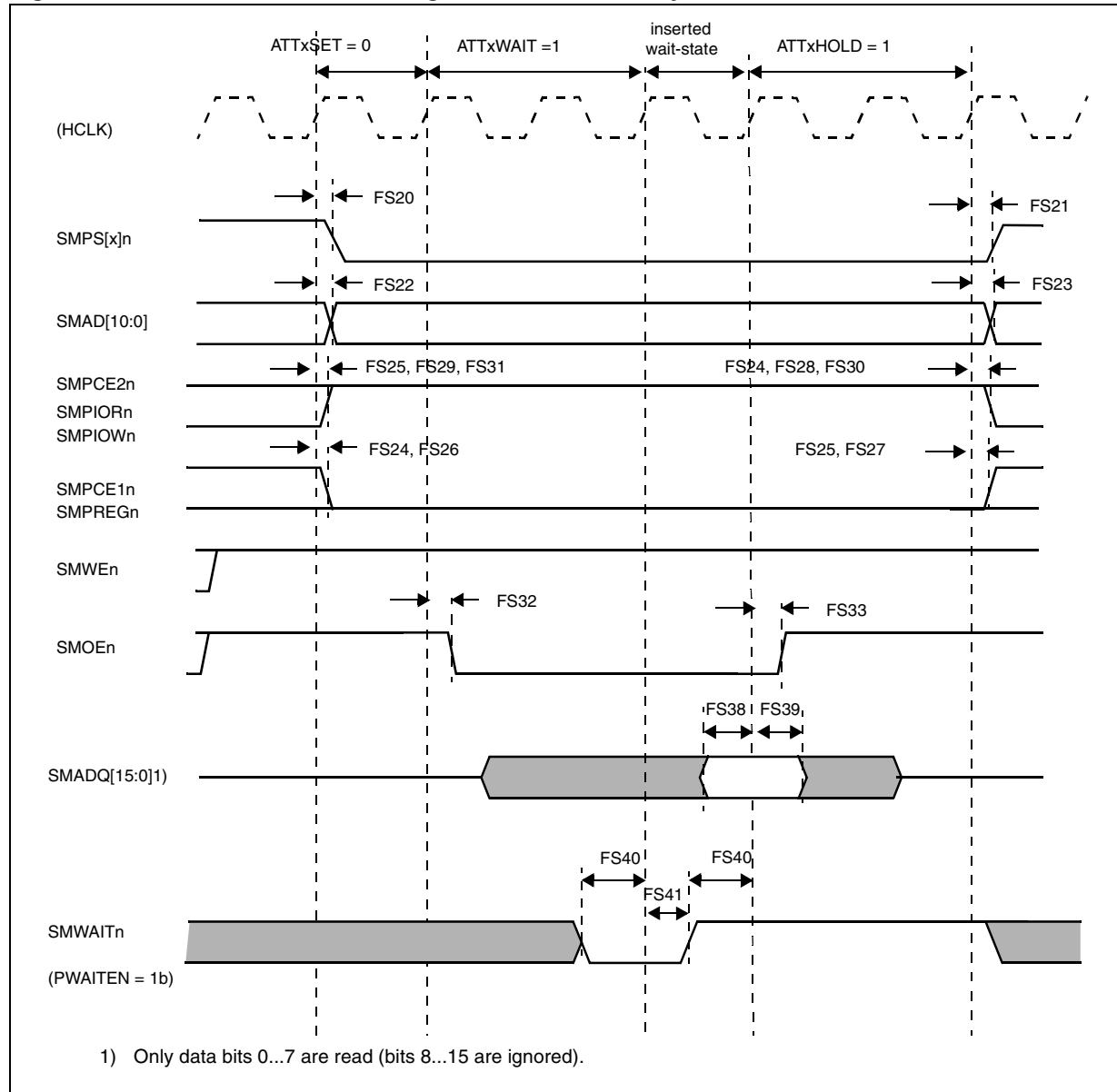


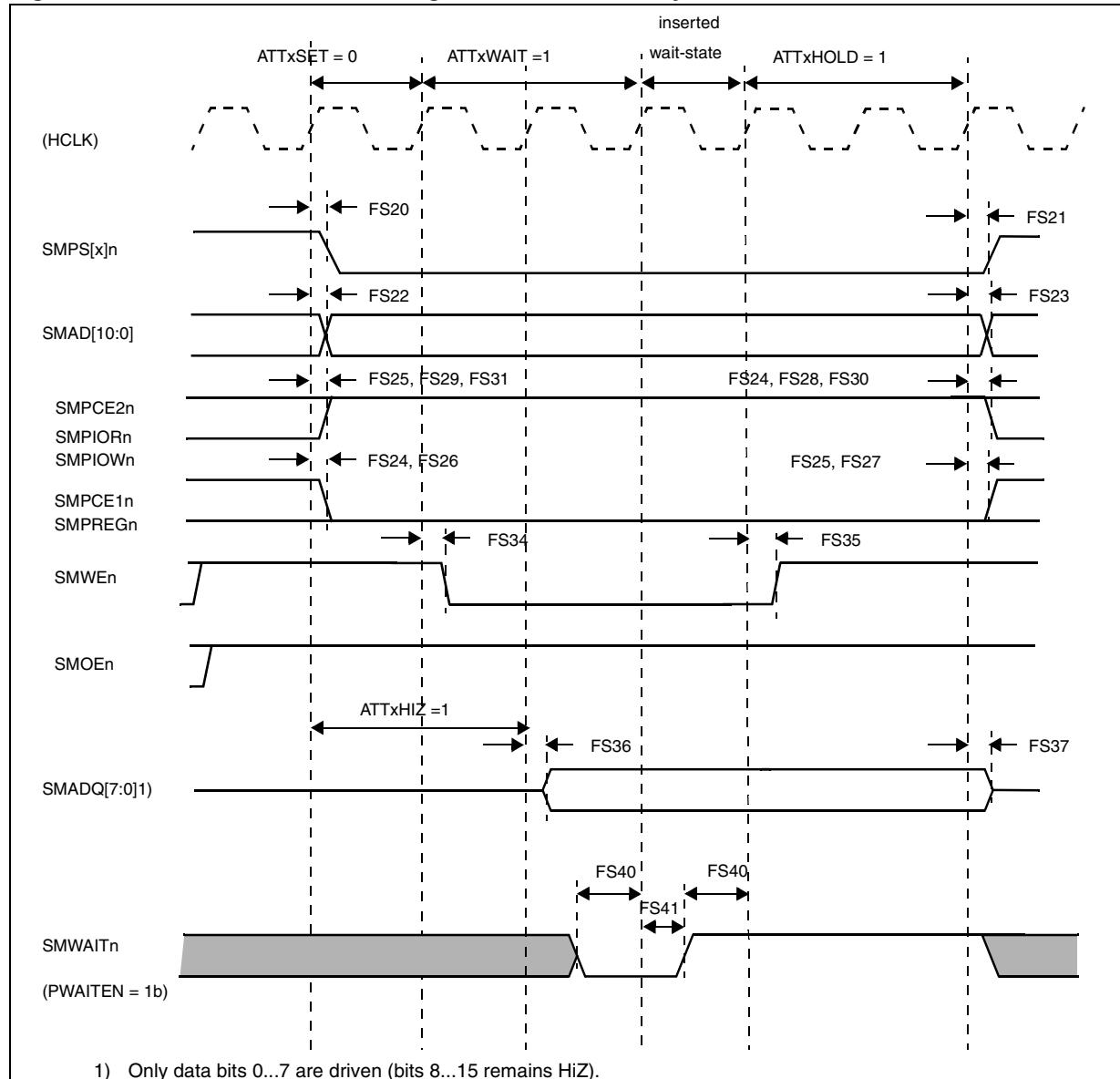
Figure 167. PC-Card controller timing for attribute memory write access

Figure 168. PC-Card controller timing for I/O space read access

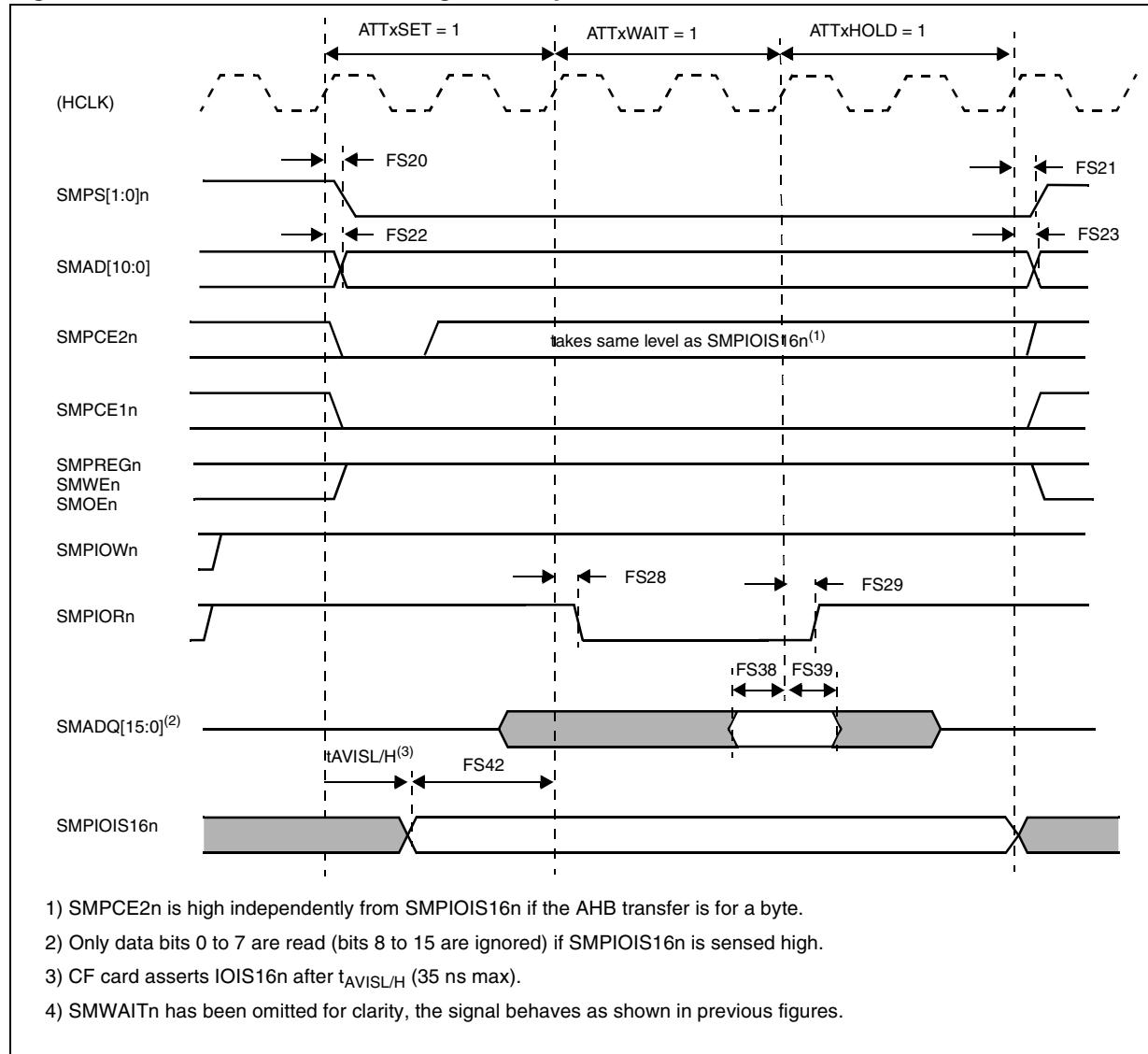
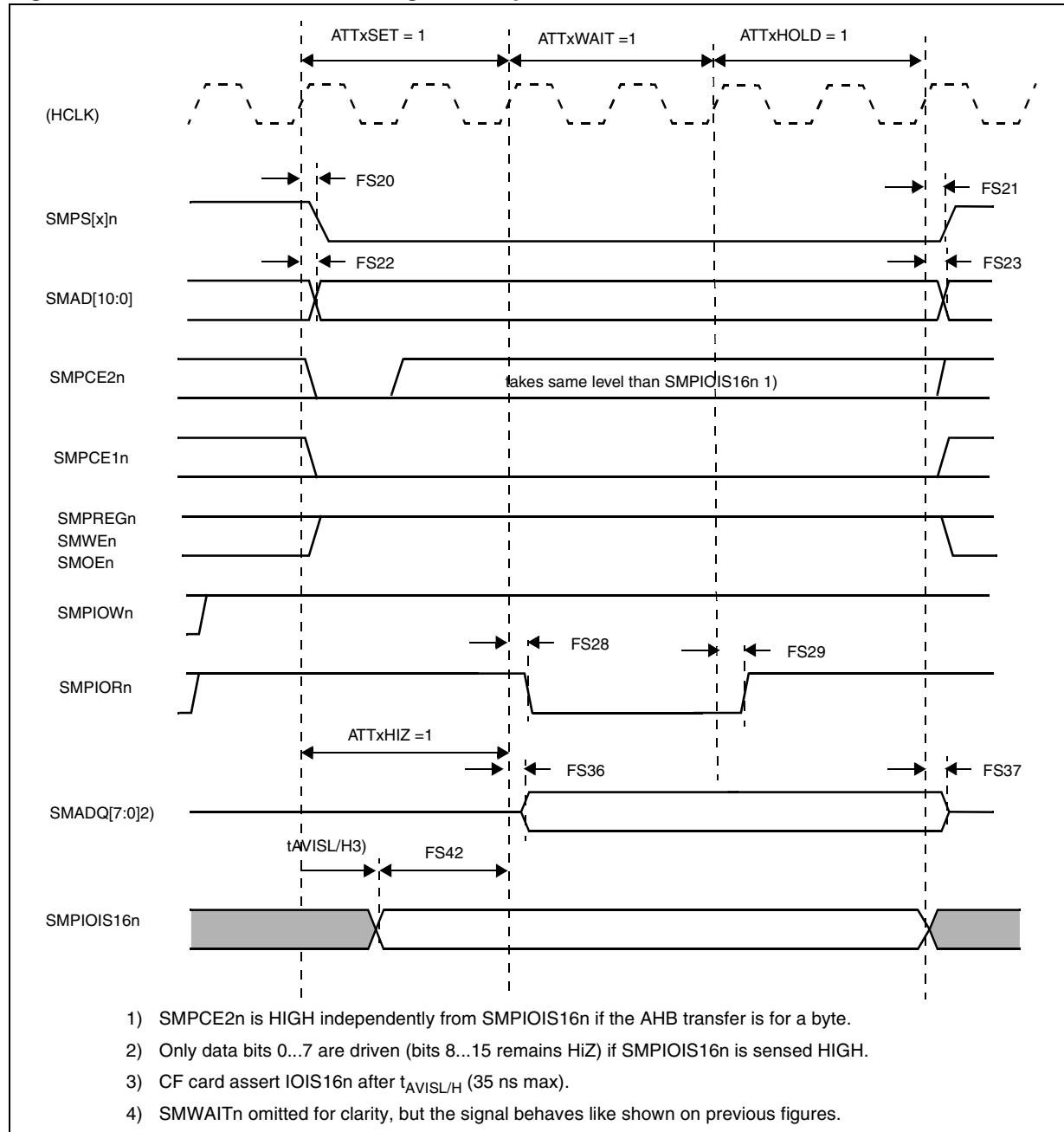


Figure 169. PC-Card controller timing for I/O space write access

30.9 SDRAM memory controller (SDMC) timings

30.9.1 Switching characteristics for SDR-SDRAM access

All the switching timing characteristics are relative to SDRCKP signal, and are provided for the Command Delay clock strategy.

Table 152. Switching characteristics for SDR-SDRAM access

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
SD1	SDRCKP cycle time (normal mode) SDRCKP cycle time (overdrive mode)	tc(CKP) = t _{CK}	7.5 6	-	ns
SD2	SDRCKP high pulse width	tw(CKPH)	0.46	0.54	t _{CK}
SD3	SDRCKP low pulse width	tw(CKPL)	0.46	0.54	t _{CK}
SD4	SDRCKP low to SDRCSn[1:0] low	td(CKPL-CSnL)	-	1.00	ns
SD5	SDRCKP low to SDRCSn[1:0] high	td(CKPL-CSnH)	-1.00	-	ns
SD6	SDRCKP low to SDRRASn low	td(CKP-RASnL)	-	1.00	ns
SD7	SDRCKP low to SDRRASn high	td(CKP-CASnH)	-1.00	-	ns
SD8	SDRCKP low to SDRCKE[1:0] high	td(CKPL-CKEH)	-	1.00	ns
SD9	SDRCKP low to SDRCKE[1:0] low	td(CKPL-CKEL)	-1.00	-	ns
SD10	SDRCKP low to SDRCASn low	td(CKPL-CASnL)	-	1.00	ns
SD11	SDRCKP low to SDRCASn high	td(CKPL-RASnL)	-1.00	-	ns
SD12	SDRCKP low to SDRAD[15:0] valid	td(CKPL-ADV)	-	1.00	ns
SD13	SDRCKP low to SDRAD[15:0] invalid	td(CKPL-ADIV)	-1.00	-	ns
SD14	SDRCKP low to SDRWRn low	td(CKPL-WRnL)	-	1.00	ns
SD15	SDRCKP low to SDRWRn high	td(CKPL-WRnH)	-1.00	-	ns
SD16	SDRCKP low to SDRDQMU/L valid	td(CKPL-DQMv)	-	1.00	ns
SD17	SDRCKP low to SDRDQMU/L invalid	td(CKPL-DQMIV)	-1.00	-	ns
SD18	SDRCKP low to SDRDQ[15:0] valid	td(CKPL-DQV)	-	1.00	ns
SD19	SDRCKP low to SDRDQ[15:0] invalid	td(CKPL-DQIV)	-1.00	-	ns

30.9.2 Timing requirements for low-power/mobile SDR-SDRAM read access

For low-power or mobile SDR-SDRAM memory read access, data returned by the memory is latched with the clock presented on SDFBCK pin. Thus, timing requirements for SDR-SDRAM read accesses are relative to SDFBCK signal.

Table 153. Timing requirements for low power/mobile SDR-SDRAM read access

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
SD30	SDRFBCK high pulse width	tw(FBCK)	2.5	-	ns
SD31	SDRFBCK low pulse width	tl(FBCK)	2.5	-	ns
SD32	SDRCKP high to SDRFBCK high	td(CKPH-FBCKH)	0	3	ns
SD33	SDRDQ[15:0] valid data before SDRFBCK high	tsu(DQV-FBCK)	0.5	-	ns
SD34	SDRDQ[15:0] valid data after SDRFBCK high	th(FBCK-DQV)	2	-	ns

30.9.3 Specific AC specification for mobile DDR-SDRAM access

When Mobile DDR-SDRAM memory is connected to the SDMC, the SDRCKN clock must be enabled. SDRCKP and SDRCKN are driven the differential clock inputs of the Mobile DDR-SDRAM memory. The crossing point of SDRCKP and SDRCKN has the following parameter:

Table 154. Specific AC specification for mobile DDR-SDRAM access

Parameter	Symbol	Limit values		Unit
		min.	max.	
Crossing point voltage, SDRCKP and SDRCKN outputs	VIX	0.80	1.00	V

30.9.4 Switching characteristics for DDR-SDRAM access

For write access to Mobile DDR-SDRAM, SDRDQ[15:0] and SDRDQM(L/U) signals are delivered on both rising and falling edge of an internal clock signal equivalent to SDRCKP signal shifted by 90 degrees ($t_{CK} / 4$). In addition, the SDRDQS(L/U) strobe signals are delivered to the Mobile DDR-SDRAM memory. The memory is latching incoming SDRDQ[15:0]/SDRDQM(L/U) signals on both rising and falling edge of SDRDQS(L/U).

Table 155. Switching characteristics for DDR-SDRAM access

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
SD1	SDRCKP cycle time (normal mode) SDRCKP cycle time (overdrive mode)	tc(CKP) = t_{CK}	7.5 6	-	ns
SD2	SDRCKP high pulse width	tw(CKPH)	0.46	0.54	t_{CK}
SD3	SDRCKP low pulse width	tw(CKPL)	0.46	0.54	t_{CK}
SD4	SDRCKP low to SDRCSn[1:0] low	td(CKPL-CSnL)	-	1.00	ns

Table 155. Switching characteristics for DDR-SDRAM access

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
SD5	SDRCKP low to SDRCSn[1:0] high	td(CKPL-CSnH)	-1.00	-	ns
SD6	SDRCKP low to SDRRASn low	td(CKP-RASnL)	-	1.00	ns
SD7	SDRCKP low to SDRRASn high	td(CKP-CASnH)	-1.00	-	ns
SD8	SDRCKP low to SDRCKE[1:0] high	td(CKPL-CKEH)	-	1.00	ns
SD9	SDRCKP low to SDRCKE[1:0] low	td(CKPL-CKEL)	-1.00	-	ns
SD10	SDRCKP low to SDRCASn low	td(CKPL-CASnL)	-	1.00	ns
SD11	SDRCKP low to SDRCASn high	td(CKPL-RASnL)	-1.00	-	ns
SD12	SDRCKP low to SDRAD[15:0] valid	td(CKPL-ADV)	-	1.00	ns
SD13	SDRCKP low to SDRAD[15:0] invalid	td(CKPL-ADIV)	-1.00	-	ns
SD14	SDRCKP low to SDRWRn low	td(CKPL-WRnL)	-	1.00	ns
SD15	SDRCKP low to SDRWRn high	td(CKPL-WRnH)	-1.00	-	ns
SD16	SDRCKP low to SDRDQMU/L valid	td(CKPL-DQMv)	-	1.00	ns
SD17	SDRCKP low to SDRDQMU/L invalid	td(CKPL-DQMIV)	-1.00	-	ns
SD20	SDRCKP transition to SDRDQS(L/U) transition	td(CKP-DQS)	-0.25	0.25	ns
SD21	SDRCKP high to SDRDQS(L/U) low	td(CKPH-DQSL)	-	3.00	ns
SD22	SDRCKP high to SDRDQS(L/U) HiZ	td(CKPL-DQSZ)	0.00	-	ns
SD23	SDRDQS(L/U) cycle time	tc(DQS)	0.91	1.09	ns
SD24	SDRDQS high pulse width	tw(DQSH)	0.42	0.58	t _{CK}
SD25	SDRDQS low pulse width	tw(DQSL)	0.42	0.58	t _{CK}
SD26	SDRCKP transition to SDRDQM(L/U) valid	td(CKP-DQMv)	-	t _{CK} / 4 + 0.25	ns
SD27	SDRCKP transition to SDRDQM(L/U) invalid	td(CKP-DQMIV)	t _{CK} / 4 - 0.25	-	ns
SD28	SDRCKP transition to SDRDQ[15:0] valid	td(CKP-DQV)	-	t _{CK} / 4 + 0.43	ns
SD29	SDRCKP transition to SDRDQ[15:0] invalid	td(CKP-DQIV)	t _{CK} / 4 - 0.25	-	ns

30.9.5 Timing requirements for mobile DDR-SDRAM read access

For mobile DDR-SDRAM memory read access, the data returned by the memory on SDRDQ[7:0] signals is latched with the SDRDQSL signal shifted by 90 degrees (by the embedded DLL), and data on SDRDQ[15:8] pins is latched with the SDRDQSU signal shifted by 90 degrees. Thus, timing requirements for DDR-SDRAM read accesses are relative to SDRDQS(L/U) signals.

Table 156. Timing requirements for mobile DDR-SDRAM read access

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
SD35	SDRDQS high data after SDRCKP high (normal mode)	td(CKPH-DQSH)	1.5	6.0	ns
	SDRDQS high data after SDRCKP high (overdrive mode)	td(CKPH-DQSH)	1.5	5.0	ns
SD36	SDRDQS preamble low pulse width	tw(DQSPREL)	5.0	-	ns
SD37	SDRDQS high pulse width	tw(DQSH)	$t_{CK} / 2 - 0.3$	$t_{CK} / 2 + 0.3$	ns
SD38	SDRDQS postamble low pulse width	tw(DQSPSTL)	2.5	-	ns
SD39	SDRDQS edge to SDRDQ[15:0] valid	td(DQS-DQV)	0	0.70	ns

Figure 170. SDR-SDRAM write (command delay strategy, burst length = 2)

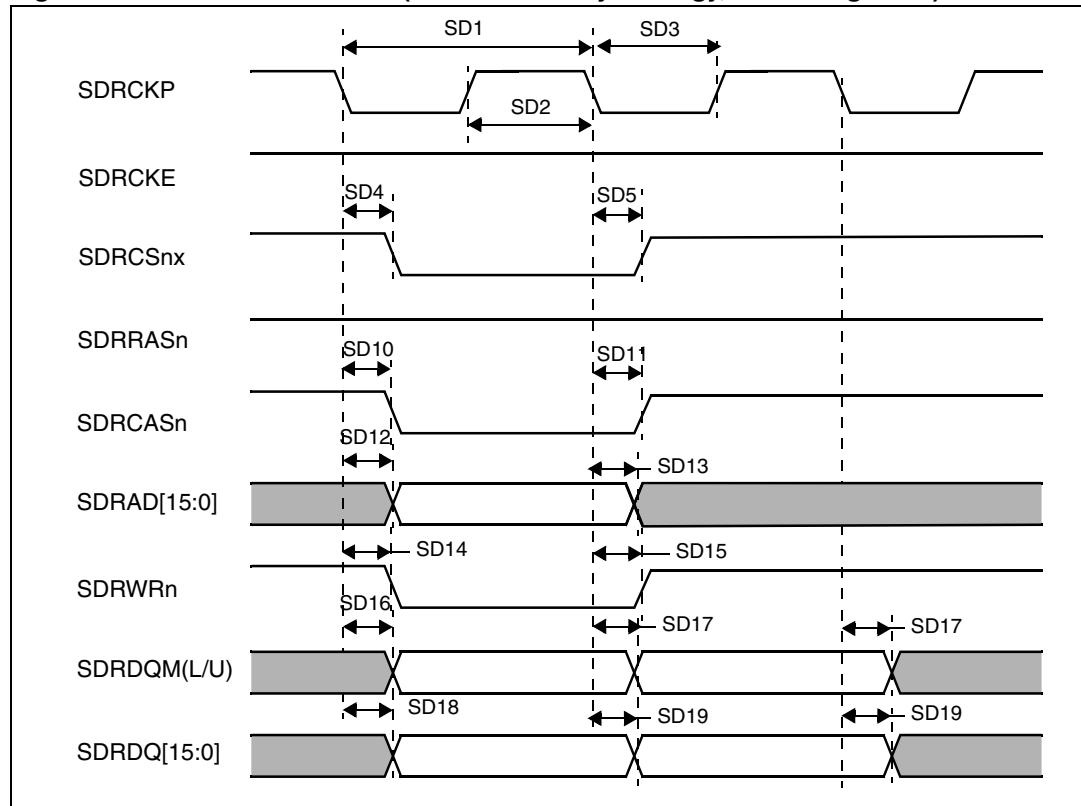


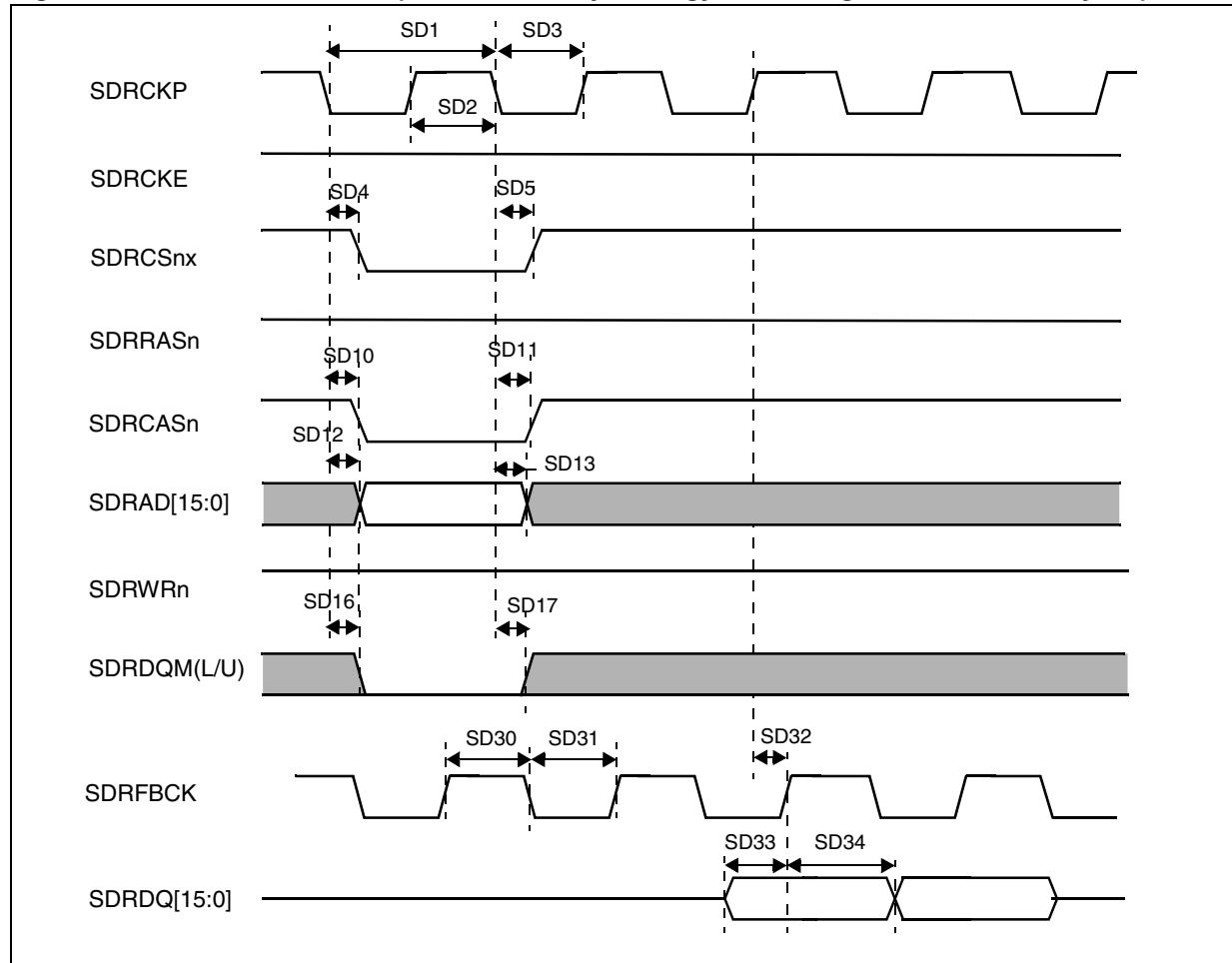
Figure 171. SDR-SDRAM read (command delay strategy, burst length = 2, CAS latency = 2)

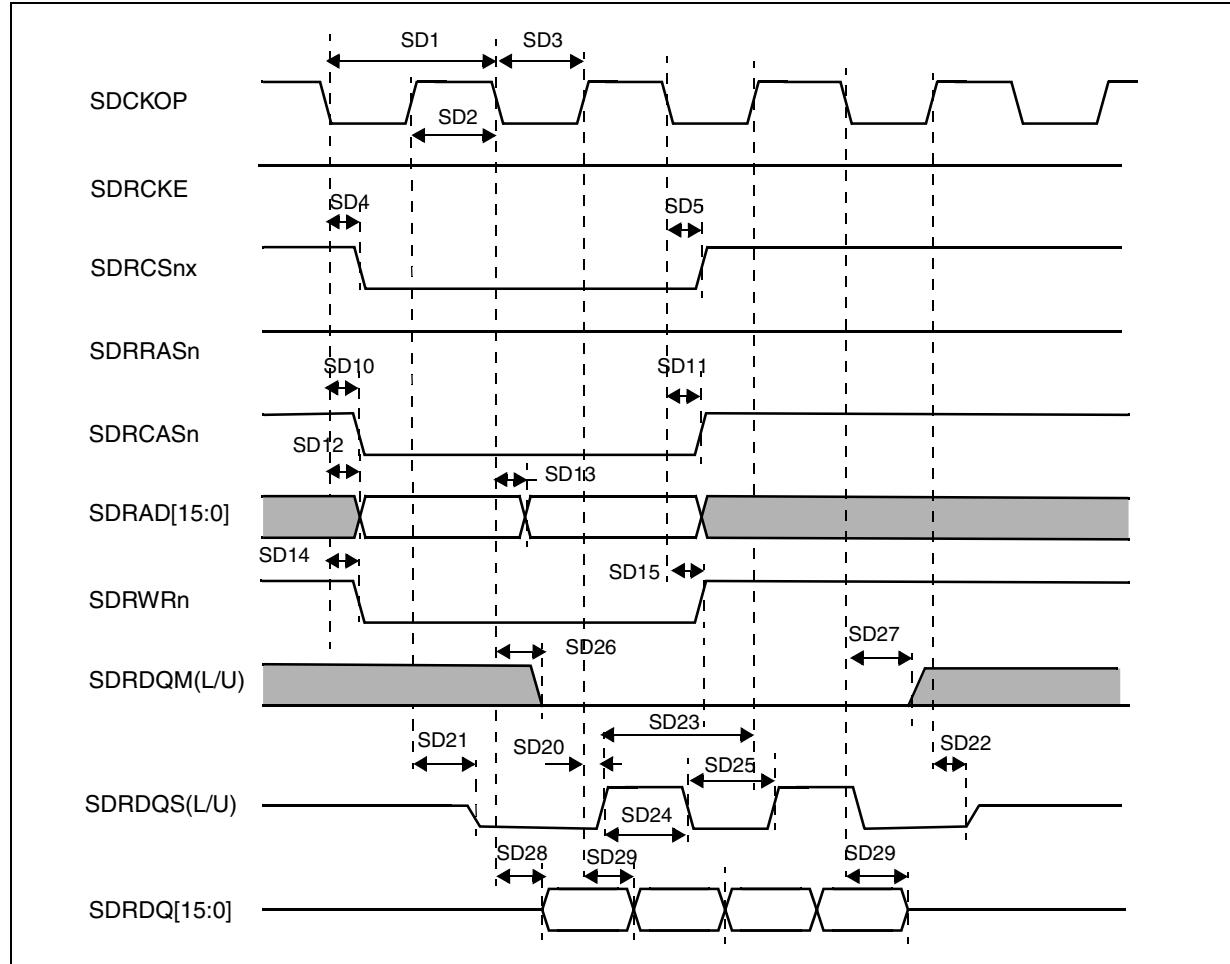
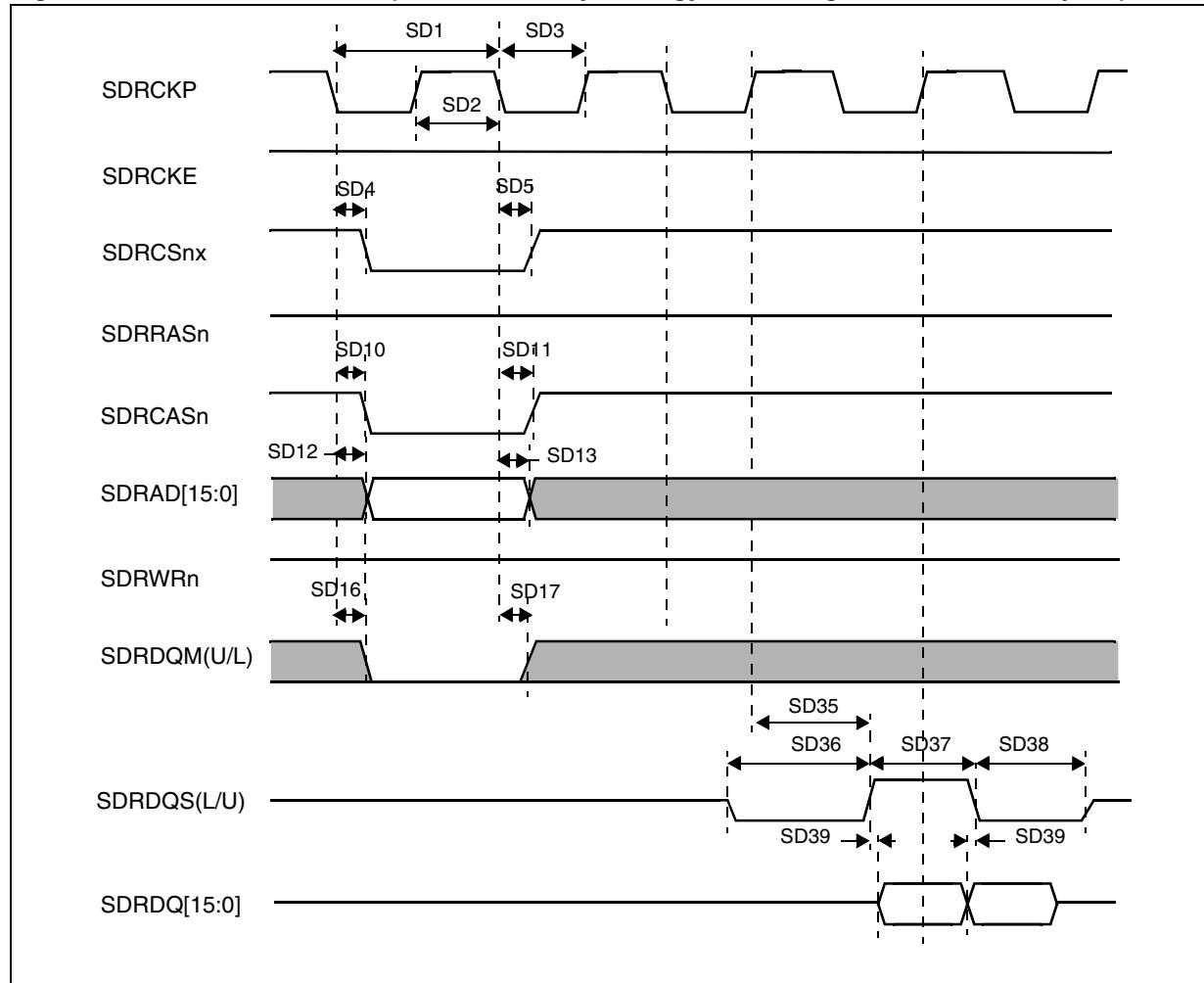
Figure 172. DDR-SDRAM write (command delay strategy, 2 consecutive bursts of length = 2)

Figure 173. DDR-SDRAM read (command delay strategy, burst length = 2, CAS latency = 3)



30.10 Color LCD controller (CLCD) timings

The switching characteristics for CLCD controller outputs are shown in [Table 157](#).

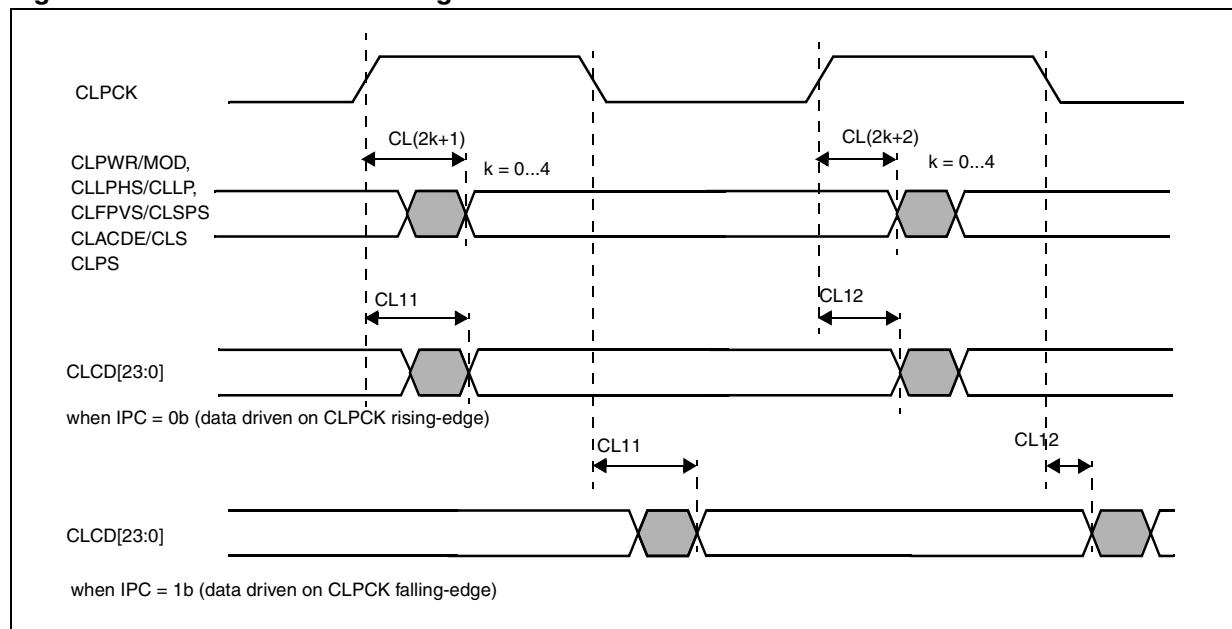
Note: All the switching timing characteristics are relative to the CLPCK signal.

Table 157. Switching characteristics for CLCD controller outputs

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
CL1	CLPCK high to CLPWR valid	tdv(PCKH - PWRV)	-	10	ns
CL2	CLPCK high to CLPWR invalid	td(PCKH - PWRIV)	-1	-	ns
CL3	CLPCK high to CLLPHS valid	td(PCKH - LPHSV)	-	10	ns
CL4	CLPCK high to CLLPHS invalid	td(PCKH - LPHSIV)	-1	-	ns
CL5	CLPCK high to CLFPVS valid	td(PCKH - FPVSV)	-	10	ns

Table 157. Switching characteristics for CLCD controller outputs

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
CL6	CLPCK high to CLFPVS invalid	td(PCKH - FPVSIV)	-1	-	ns
CL7	CLPCK high to CLACDE valid	td(PCKH - ACDEV)	-	10	ns
CL8	CLPCK high to CLACDE invalid	td(PCKH - ACDEIV)	-1	-	ns
CL9	CLPCK high to CLLE valid	td(PCKH - LEV)	-	10	ns
CL10	CLPCK high to CLLE invalid	td(PCKH - LEIV)	-1	-	ns
CL11	CLPCK transition to CLCD[23:0] valid	td(PCK - CDV)	-	10	ns
CL12	CLPCK transition to CLCD[23:0] invalid	td(PCK - CDIV)	0	-	ns

Figure 174. CLCD controller timings

30.11 Display interface (DIF) timings

The DIF does not deliver a clock signal to which all other signal switching timings can be referenced. However, the programmable clock output signal, CLKOUT0, is used during characterization as a clock reference, when configured to deliver the DIF clock input frequency (48 MHz, from PLL2). Thus, all the timing characteristics provided are relative to CLKOUT0. The switching characteristics are shown in [Table 158](#).

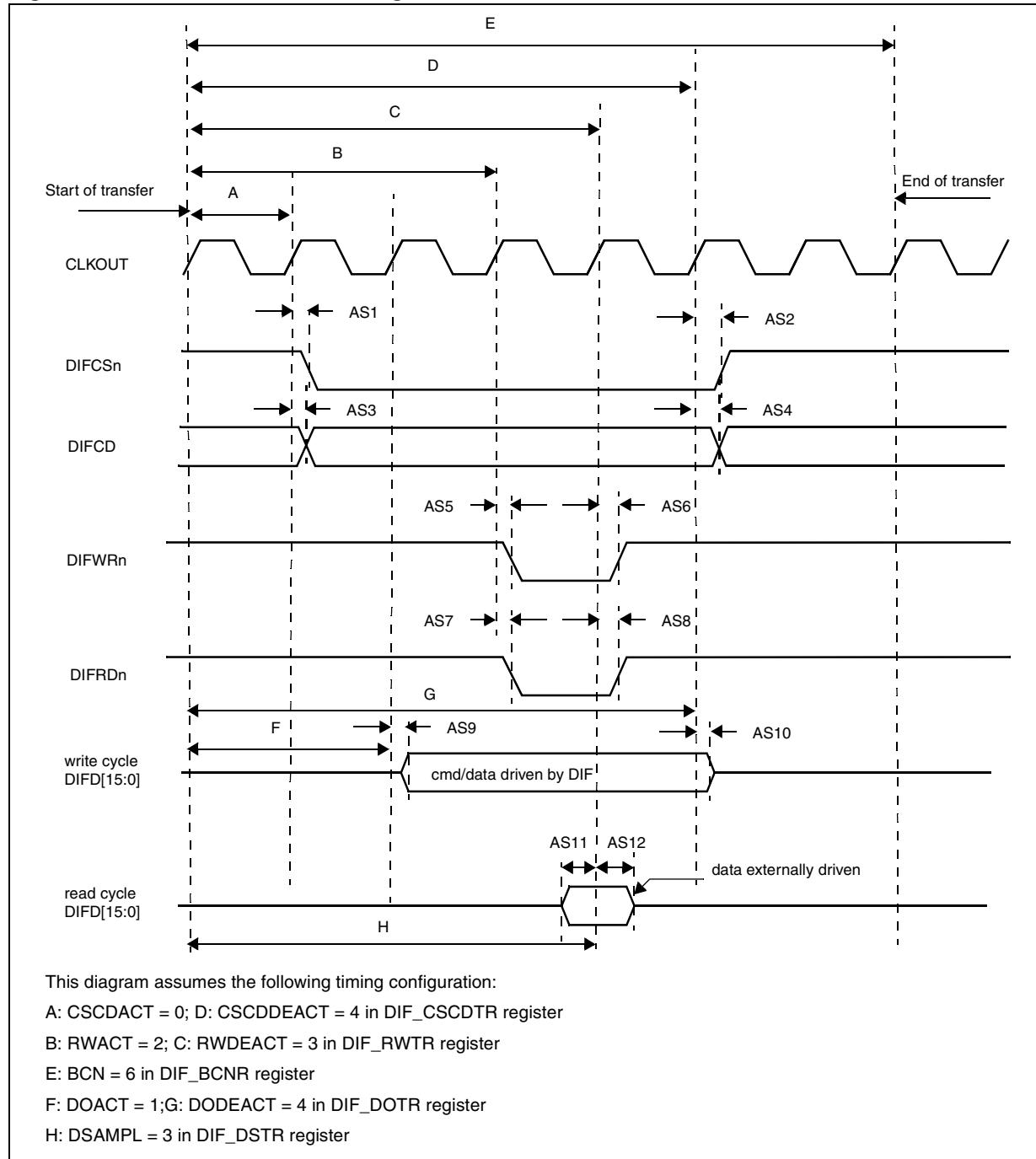
Table 158. Switching characteristics for DIF read and write cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
AS1	CLKOUT0 high to DIFCSxn valid (x = 0...1)	td(CLKH-CSnV)	-	5	ns
AS2	CLKOUT0 high to DIFCSxn invalid (x = 0...1)	td(CLKH-CSnI)	0	-	ns
AS3	CLKOUT0 high to DIFCD valid	td(CLKH-CDV)	-	5	ns
AS4	CLKOUT0 high to DIFCD invalid	td(CLKH-CDI)	0	-	ns
AS5	CLKOUT0 high to DIFWRn valid	td(CLKH-WRnV)	-	5	ns
AS6	CLKOUT0 high to DIFWRn invalid	td(CLKH-WRnI)	0	-	ns
AS7	CLKOUT0 high to DIFRDn valid	td(CLKH-RDnV)	-	5	ns
AS8	CLKOUT0 high to DIFRDn invalid	td(CLKH-RDnI)	0	-	ns
AS9	CLKOUT0 high to DIFDx valid (x = 0...15)	td(CLKH-DV)	-	5	ns
AS10	CLKOUT0 high to DIFDx invalid (x = 0...15)	td(CLKH-DI)	0	-	ns

Timing requirements for DIF read cycles are shown in [Table 159](#).

Table 159. Timing requirements for DIF read cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
AS11	DIFDx valid data before CLKOUT0 high (x = 0...15)	tsu(DV-CLKH)	5	-	ns
AS12	DIFDx valid data after CLKOUT0 high (x = 0...15)	th(CLKH-DV)	1	-	ns

Figure 175. DIF read and write timings

30.12 Serial camera interface (CSI) timings

This section provides the timing requirements for the serial camera interface. This port uses an external clock signal, CSICK, (using SubLVDS differential signaling scheme, on CSICKP/CSICKN signals) to latch incoming data bits, CSID (also using SubLVDS differential signaling scheme, on CSIDP/CSIDN signals).

Table 160. CSI timing requirements

No.	Parameter	Symbol	Timing		Unit
			min	max	
CP1	CSICKP/CSICKN period	ti(CSICK)	3.1	1000	ns
CP2	CSICKP/CSICKN high pulse width	tw(CSICKH)	1.3	-	ns
CP3	CSICKP/CSICKN low pulse width	tw(CSICKKL)	1.3	-	ns
CP4	CSIDP/CSIDN valid before CSICKP/CSICKN high	tsu(CSID - CSICK)	1	-	ns
CP5	CSIDP/CSIDN valid after CSICKP/CSICKN high	th(CSICK - CSID)	1	-	ns

Figure 176. CSI input timings, latching on CSICK rising edge

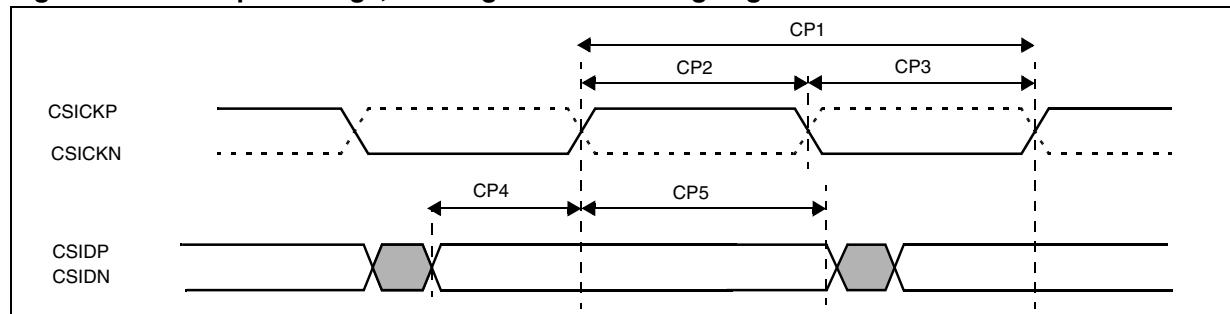
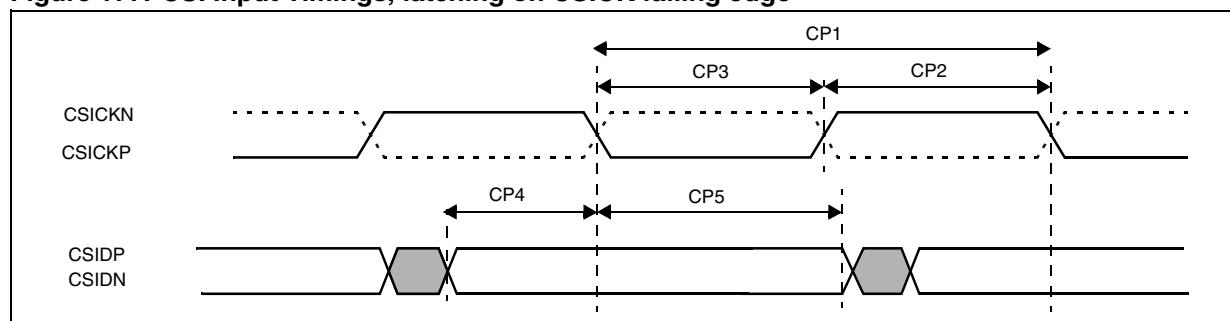


Figure 177. CSI Input Timings, latching on CSICK falling edge



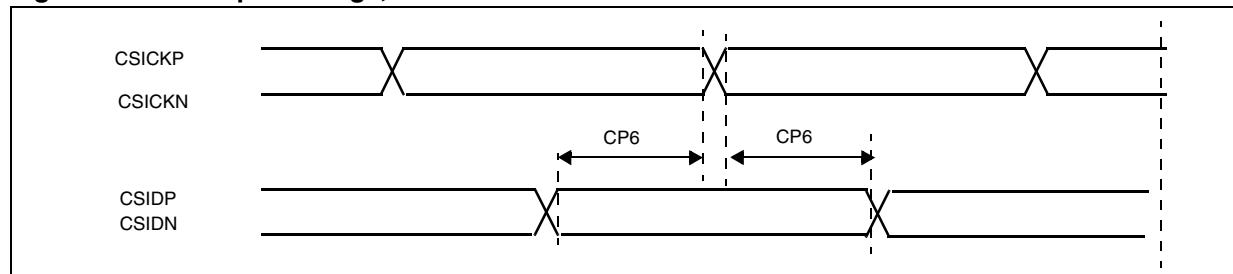
30.12.1 Timing requirements for CSI inputs in data/strobe mode

All the timing requirements are relative to CSICKP/CSICKN differential clock signal crossing point.

Table 161. CSI input timings in data/strobe mode

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
CP6	CSICKP/CSICKN (Strobe) and CSIDP/CSIDN (Data) both stable	$t_w(\text{CSICK})$	1.165	-	ns

Figure 178. CSI input timings, data/strobe mode



30.13 CCIR-656 video input port (CCIRI) timings

This section provides the timing requirements for the CCIR-656 video input port (CCIRI). This port uses an external clock signal, CCIRICLK, to latch the incoming data bus, CCIRD[7:0] and optional horizontal/vertical sync signals, CCIRIHS and CCIRIVS.

The CCIRI can be programmed to use the rising or falling edge of CCIRICLK for latching.

The data is then resynchronized in the AHB clock domain. The timing requirements for CCIR inputs are shown in [Table 162](#).

Table 162. Frequency constraints versus processing mode

Processing mode	Frequency constraint
Raw data	$f_{CCIRI} < 4/3 f_{AHB}$
YCbCr data without downsampling	$f_{CCIRI} < 4/7 f_{AHB}$
YCbCr data with downsampling	$f_{CCIRI} < 1/2 f_{AHB}$

Table 163. Timing requirements for CCIR inputs

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
CI1	CCIRICLK cycle time	t(CCIRICLK)	$t_{HCLK}^* 2$	-	ns
CI2	CCIRICLK high pulse width	tw(CCIRICLKH)	5	-	ns
CI3	CCIRICLK low pulse width	tw(CCIRICLKL)	5	-	ns
CI4	CCIRD[7:0] valid before CCIRICLK edge	tsu(CCIRDV - CCIRICLK)	4.5	-	ns
CI5	CCIRD[7:0] valid after CCIRICLK edge	thd(CCIRICLK - CCIRDV)	1	-	ns
CI6	CCIRIHS valid before CCIRICLK edge	tsu(CCIRIHSV - CCIRICLK)	4.5	-	ns
CI7	CCIRIHS valid after CCIRICLK edge	th(CCIRICLK - CCIRIHSV)	1	-	ns
CI8	CCIRIVS valid before CCIRICLK edge	tsu(CCIRIVSV - CCIRICLK)	2.5	-	ns
CI9	CCIRIVS valid after CCIRICLK edge	th(CCIRICLK - CCIRIVSV)	1	-	ns

Figure 179. CCIR-656 input timings, latching on CCIRICLK rising edge

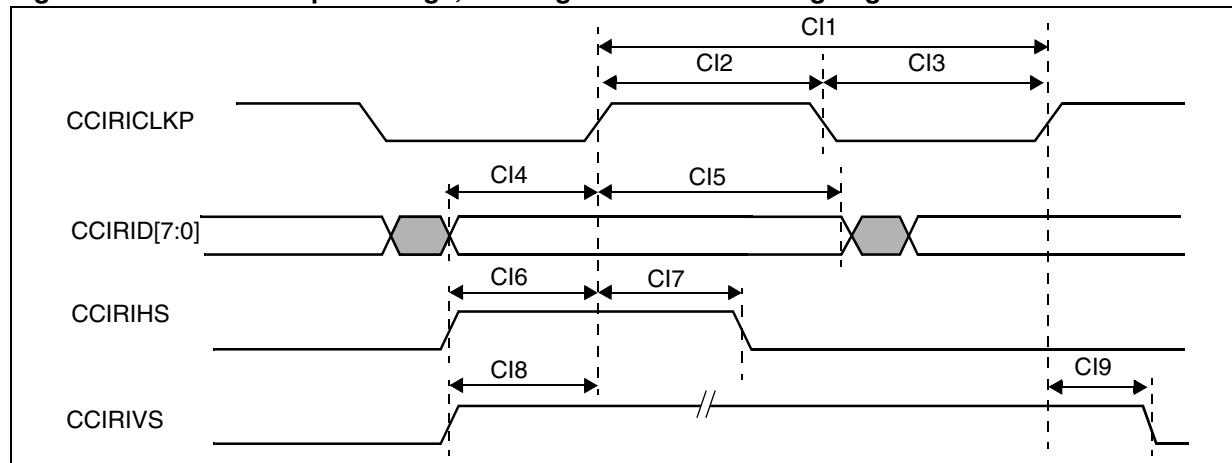
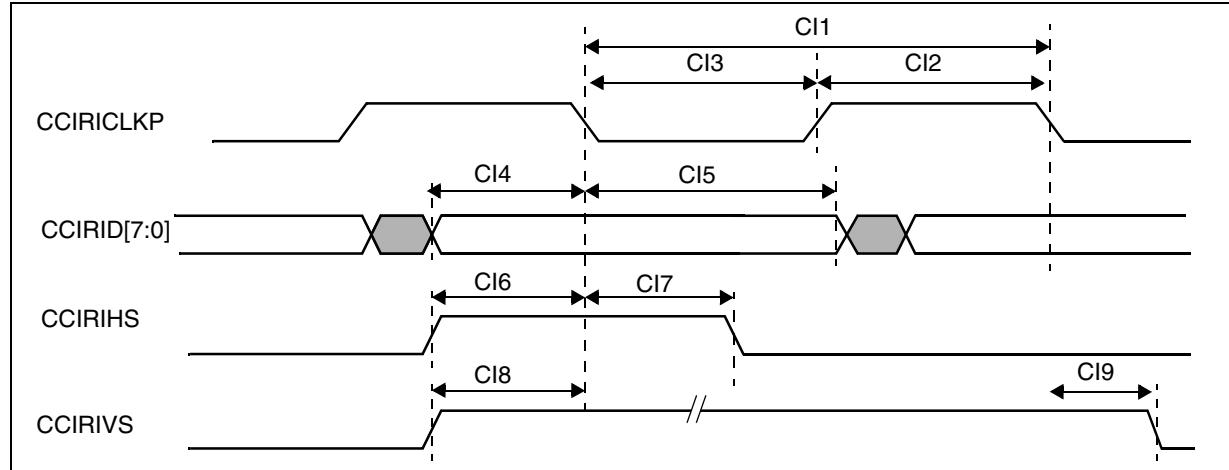


Figure 180. CCIR-656 input timings, latching on CCIRICLK falling edge

30.14 CCIR-656 video output port (CCIRO) timings

This section provides the timing requirements for the CCIR-656 video output port (CCIRO). All the timing requirements are relative to CCIROCLK clock signal.

CCIROCLK can be an output clock delivered by the STn8815A12 (CCIROCLK(out)), or an input clock signal delivered by an external device (CCIROCLK(in)). The CCIRO can be programmed to use rising or falling edges of CCIROCLK for CCIROD[7:0] latching.

The maximum CCIROCLK clock frequency is limited by the internal AHB bus clock frequency:

$$f(\text{CCIROCLK}) < f(\text{HCLK}) / 3$$

30.14.1 Switching characteristics for CCIRO outputs

Table 164. Switching requirements for video output port

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
CO1	CCIROCLK(out) cycle time	t(CCIROCLKO)	T ⁽¹⁾ -3	T ⁽¹⁾ +3	ns
CO2	CCIROCLK(out) high pulse width	tw(CCIROCLKOH)	T/2 - 2	T/2 + 2	ns
CO3	CCIROCLK(out) low pulse width	tw(CCIROCLKOL)	T/2 - 2	T/2 + 2	ns
CO4	CCIROCLK(I/O) transition to CCIROD[7:0] valid	td(CCIROCLK - CCIRODV)	-	18	ns
CO5	CCIROCLK(I/O) transition to CCIROD[7:0] invalid	td(CCIROCLK - CCIRODIV)	5	-	ns

1. $T = 1 / 27 \text{ MHz} = 37.04 \text{ ns}$ when MXTAL1 frequency is 16 MHz, 19.2 MHz or 24 MHz
 $T = 1 / 26 \text{ MHz} = 38.46 \text{ ns}$ when MXTAL1 frequency is 13 MHz or 26 MHz.

30.14.2 Timing requirements for CCIRO output

Table 165. Timing requirements for video output port

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
CO6	CCIROCLK(in) cycle time	t(CCIROCLKO)	34 ⁽¹⁾	-	ns
CO7	CCIROCLK(in) high pulse width	tw(CCIROCLKOH)	15	-	ns
CO8	CCIROCLK(in) low pulse width	tw(CCIROCLKOL)	15	-	ns

1. Typical value is $1 / 27 \text{ MHz} = 37.04 \text{ ns}$. The period must also respect the restriction regarding HCLK frequency (see above).

Figure 181. CCIR-656 output port timings, sent on CCIROCLK rising edge

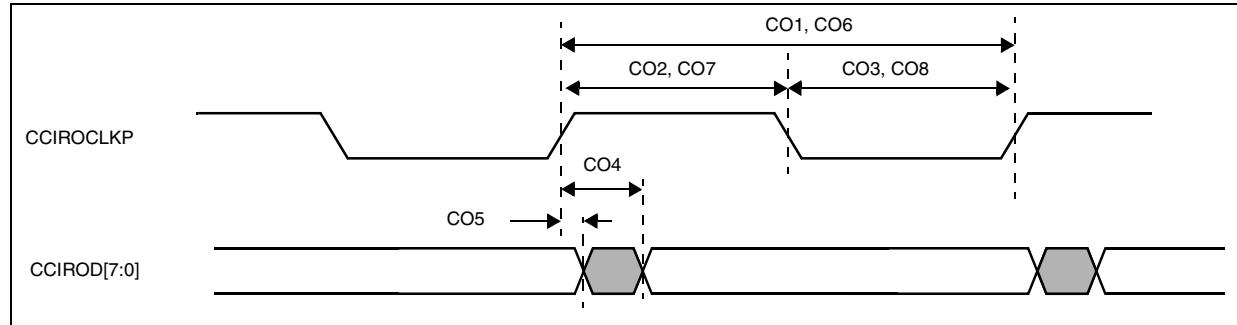
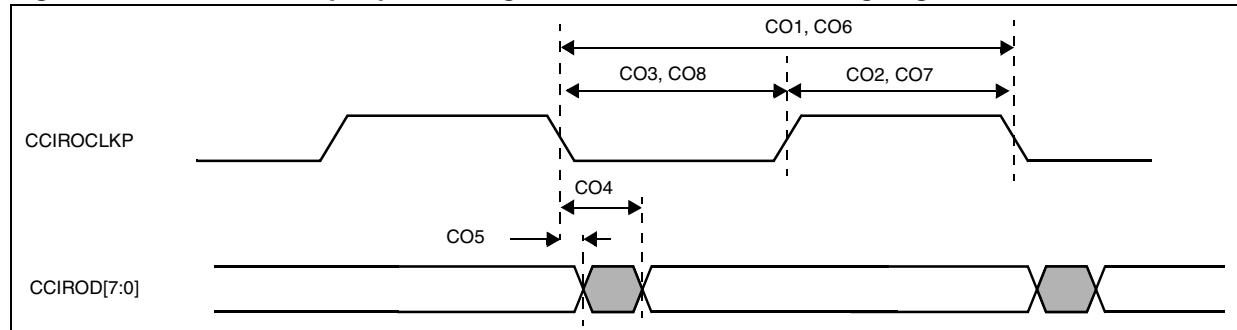


Figure 182. CCIR-656 output port timings, sent on CCIROCLK falling edge



30.15 SD-Card/MMC interface (SDI) timings

All the timing characteristics are relative to the MMCLK signal.

The switching characteristics for SDI cmd/data read and write cycles are shown in [Table 166](#).

Table 166. Switching characteristics for SDI cmd/data read and write cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MC1	MCCLK cycle time, data transfer mode	tc(MCCLKF)	20.8 ⁽¹⁾	-	ns
MC2	MCCLK cycle time, identification mode	tc(MCCLKS)	2500 ⁽²⁾	-	ns
MC3	MCCLK low to MCCMDDIR	td(MCCLKH-MCCMDDIRV)	-1	5	ns
MC4	MCCLK low to MCCMD valid	td(MCCLKH-MCCMDV)		5	ns
MC5	MCCLK low to MCCMD invalid	td(MCCLKH-MCCMDIV)	-1		ns
MC6	MCCLK low to MCDATxDIR (x = 0,2,31)	td(MCCLKH-MCDATDIRV)	-1	5	ns
MC7	MCCLK low to MCDATx valid (x = 0...7)	tdv(MCCLKH-MCDATV)		5	ns
MC8	MCCLK low to MCDATx invalid (x = 0...7)	td(MCCLKH-MCDATIV)	-1		ns

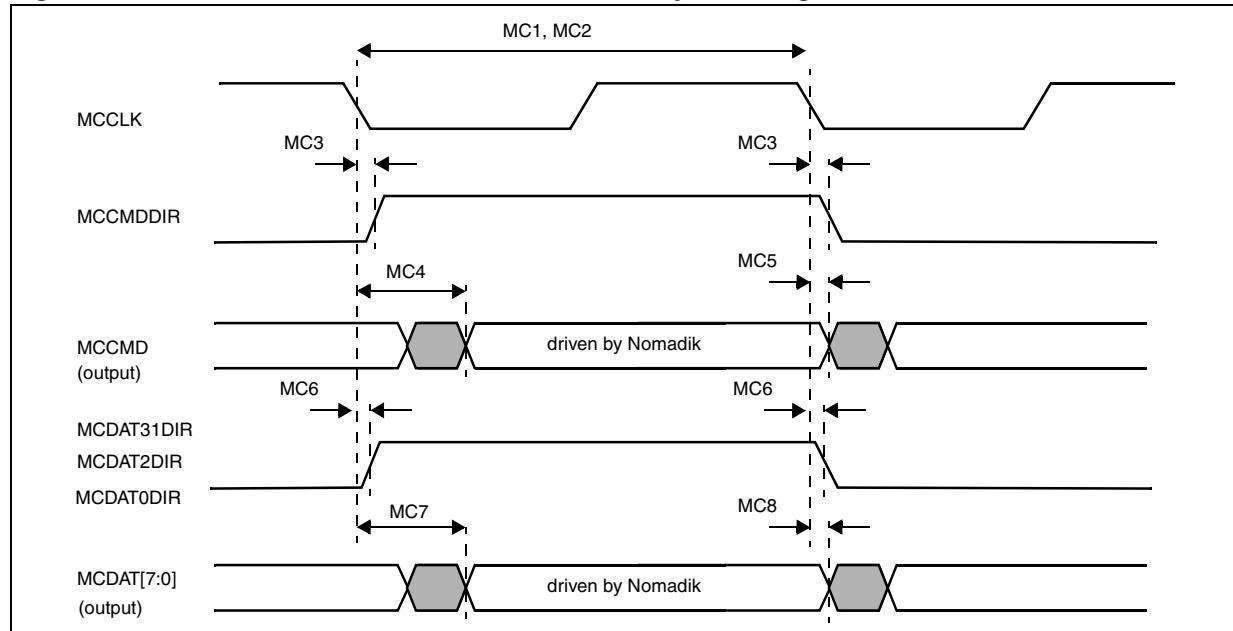
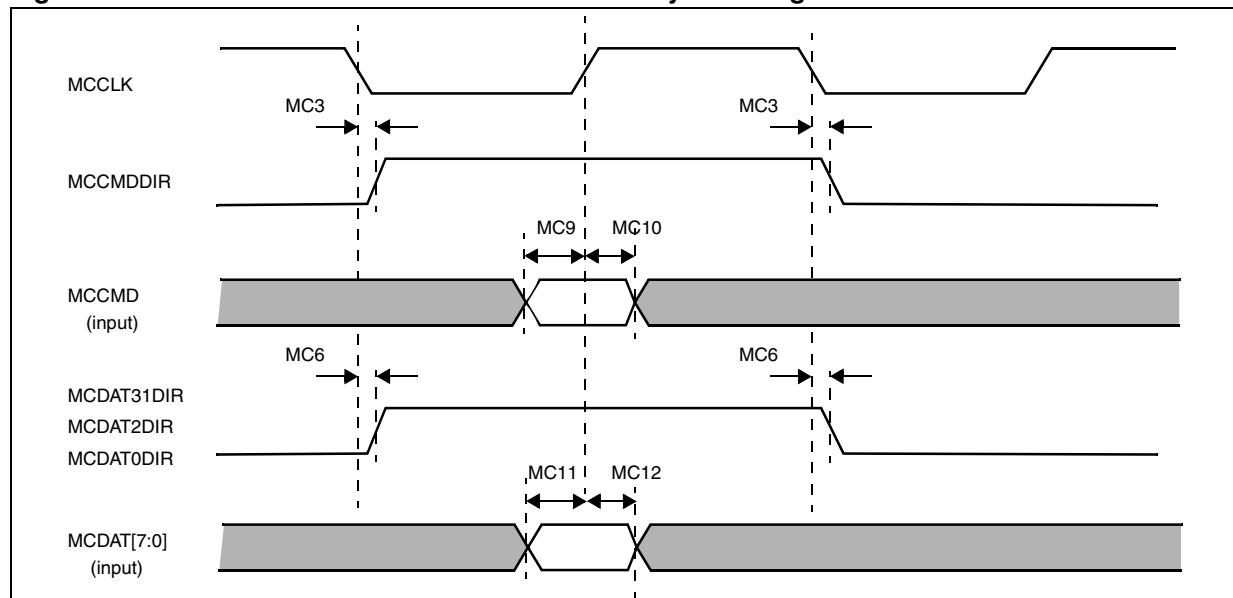
1. tc(MCCLKF)max = 1 / (48 MHz)

2. tc(MCCLKS)max = 1 / (400 kHz)

The timing requirements for SDI cmd/data read cycles are shown in [Table 167](#).

Table 167. Timing requirements for SDI cmd/data read cycles

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MC9	MCCMD valid data before MCCLK high	tsu(MCCMDV - MCCLKH)	4	-	ns
MC10	MCCMD valid data after MCCLK high	th(MCCLKH - MCCMDV)	4	-	ns
MC11	MCDAT[7:0] valid data before MCCLK high	tsu(MCDATV - MCCLKH)	4	-	ns
MC12	MCDAT[7:0] valid data after MCCLK high	th(MCCLKH - MCDATV)	2	-	ns

Figure 183. SD-Card/MMC interface cmd/data write cycle timings**Figure 184. SD-Card/MMC interface cmd/data read cycle timings**

30.16 Synchronous serial port (SSP)

All timings below are given when SSPTXD transition occurs on SSPCLK rising edge, and SSPRXD is latched on SSPCLK falling edge. For other configurations in SPI mode, the timing reference edges of SSPCLK signal are inverted.

30.16.1 Switching characteristics as master or slave

Table 168. SSP switching characteristics as master or slave

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
SS1	SSPCLK (int) cycle time (Master)	tc(CLKint)	T ⁽¹⁾	-	ns
SS2	SSPCLK (int) high pulse width (Master)	tw(CLKHint)	2*T-2	2*T+2	ns
SS3	SSPCLK (int) low pulse width (Master)	tw(CLKLint)	2*T-2	2*T-2	ns
SS4	SSPFRM (int) low to SSPCLK (int) high ⁽²⁾	td(FRMLint-CLKLint)	T	1.5*T+2	ns
SS5	SSPCLK (int) high to SSPFRM (int) high ⁽³⁾	td(CLKLint-FRMHint)	0	0.5*T+2	ns
SS6	SSPCLK (int) high to SSPTXD valid	td(CLKHint-TXDV)	-	4	ns
SS7	SSPFRM (int) valid to SSPTXD driven	td(FRMVint-TXDLZ)	0	6	ns
SS8	SSPFRM (int) invalid to SSPTXD HiZ	td(FRMHint-TXDLZ)	0	6	ns

1. $T = (1 / 48 \text{ MHz}) * \text{CPSDRV} * (\text{SCR} + 1)$ where CPSDRV is an even value from 2 to 254 and SCR is a value from 0 to 255. Thus, T min = 1 / 24 MHz = 41.66 ns

2. Valid for SPI mode with SPO = 0b (else the timing reference is inverted), valid for MicroWire mode and Unidirectional mode. Does not apply for TI mode.

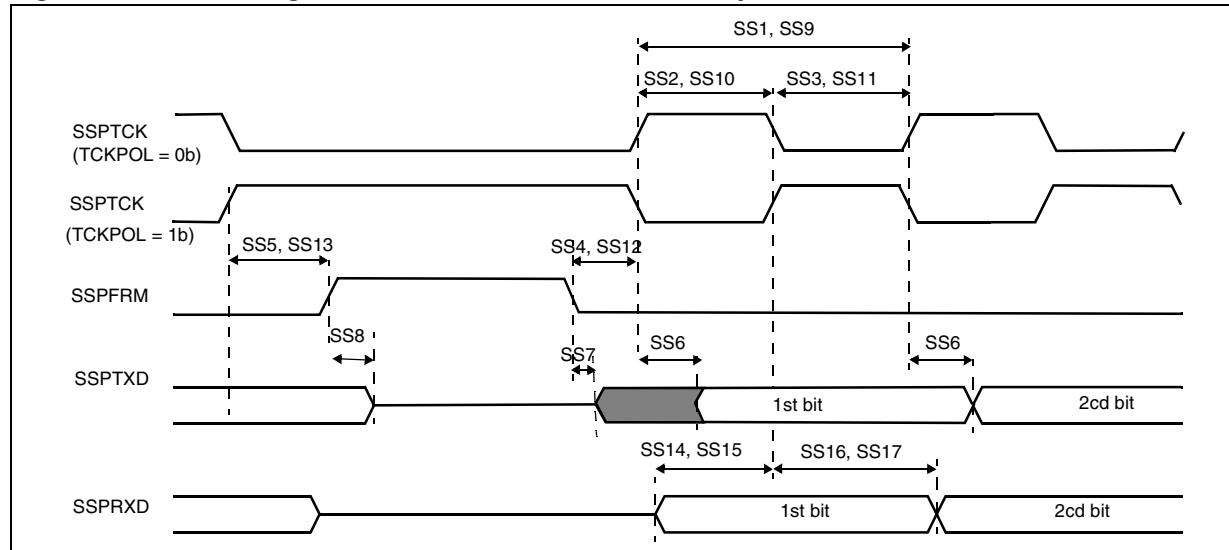
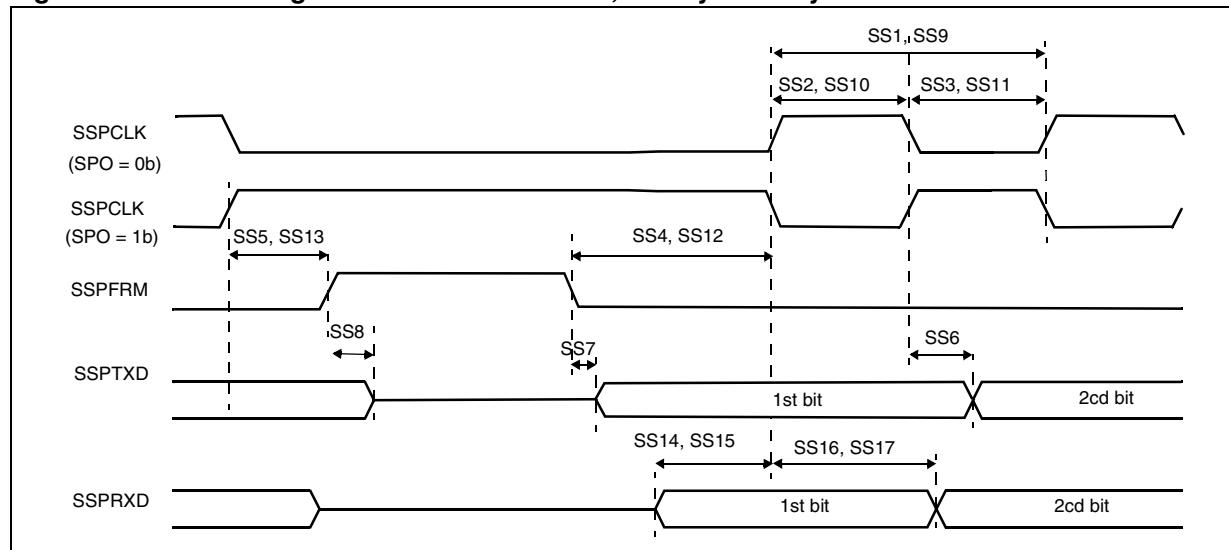
3. Valid for TI mode only.

30.16.2 Timing requirements as master or slave

Table 169. SSP timing requirements as master or slave

No.	Parameter	Symbol	Timing		Unit
			Min	Max	
SS9	SSPCLK (ext) cycle time (Slave)	tc(CLKext)	250	-	ns
SS10	SSPCLK (ext) high ⁽¹⁾ pulse width (Slave)	tw(CLKHext)	123	-	ns
SS11	SSPCLK (ext) low ⁽¹⁾ pulse width (Slave)	tw(CLKLext)	123	-	ns
SS12	SSPFRM (ext) low before SSPCLK (ext) high ⁽¹⁾	tsu(TFSLext-CLKLext)	10	-	ns
SS13	SSPFRM (ext) low after SSPCLK (ext) high	th(CLKLext-TFSLext)	10	-	ns
SS14	SSPRXD valid before SSPCLK (int) low	tsu(TRXDV-CLKLint)	10	-	ns
SS15	SSPRXD valid before SSPCLK (ext) low	tsu(TRXDV-CLKLext)	10	-	ns
SS16	SSPRXD valid after SSPCLK (int) low	th(CLKLint-TRXDV)	10	-	ns
SS17	SSPRXD valid after SSPCLK (ext) low	th(CLKLext-TRXDV)	10	-	ns

1. Valid for SPI mode with SPO = 0b (else the timing reference is inverted), valid for MicroWire mode and Unidirectional mode. Does not apply for TI mode

Figure 185. SSP timings as SPI master or slave, zero delay SPI mode**Figure 186. MSP timings as SPI master or slave, half-cycle delay SPI mode**

30.17 Multichannel serial port (MSP) timings

All timings below are given for signal polarity high. If the polarity of any of the signals is inverted, then the timing reference edges of that signal are also inverted.

30.17.1 Transmitter switching characteristics (in non-SPI mode)

Table 170. Transmitter switching characteristics (in non-SPI mode)

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MT1	MSPTCK (int) cycle time	tc(TCKint)	T ⁽¹⁾	-	ns
MT2	MSPTCK (int) high pulse width	tw(TCKHint)	Gh-2 ⁽²⁾	Gh+2 ⁽²⁾	ns
MT3	MSPTCK (int) low pulse width	tw(TCKLint)	Gl-2 ⁽³⁾	Gl+2 ⁽³⁾	ns
MT4	MSPTCK (int) high to MSPTFS (int) valid ⁽⁴⁾	td(TCKHint-TFSVint)	-4	4	ns
MT5	MSPTCK (ext) high to MSPTFS (int) valid ⁽⁴⁾	td(TCKHext-TFSVint)	4	8	ns
MT6	MSPTCK (int) high to MSPTXD valid ⁽⁴⁾⁽⁵⁾	td(TCKHint-TXDV)	-	4 ⁽⁵⁾	ns
MT7	MSPTCK (ext) high to MSPTXD valid ⁽⁴⁾⁽⁵⁾	td(TCKHext-TXDV)	-	8 ⁽⁵⁾	ns
MT8	MSPTCK (int) high to MSPTXD driven ⁽⁴⁾⁽⁵⁾	td(TCKHint-TXDLZ)	0 ⁽⁴⁾	-	ns
MT9	MSPTCK (ext) high to MSPTXD driven ⁽⁴⁾⁽⁵⁾	td(TCKHext-TXDLZ)	4 ⁽⁴⁾	-	ns
MT10	MSPTCK (int) high to MSPTXD HiZ ⁽⁴⁾	td(TCKHint-TXDHZ)	-	8	ns
MT11	MSPTCK (ext) high to MSPTXD HiZ ⁽⁴⁾	td(TCKHext-TXDHZ)	-	16	ns
MT12	MSPTFS (int) high to MSPTXD valid ⁽⁴⁾⁽⁶⁾	td(TFSHint-TXDV)	-	2	ns
MT13	MSPTFS (ext) high to MSPTXD valid ⁽⁴⁾⁽⁶⁾	td(TFSHext-TXDV)	-	8	ns
MT14	MSPTFS (int) high to MSPTXD driven ⁽⁴⁾⁽⁶⁾	td(TFSHint-TXDLZ)	0	-	ns
MT15	MSPTFS (ext) high to MSPTXD driven ⁽⁴⁾⁽⁶⁾	td(TFSHext-TXDLZ)	4	-	ns

1. $T = tc(MSPCLK)$ when $SCKSEL = 0\text{X}b$; $MSPCKL = 48 \text{ MHz}$, thus $T = 20.83 \text{ ns}$
 $T = tc(MSPSCK)$ when $SCKSEL = 1\text{X}b$.
2. $Gh = (1 + SCKDIV) * T$ when $SCKDIV$ is odd or zero; else $Gh = (2 + SCKDIV) * T$.
3. $Gl = (1 + SCKDIV) * T$ when $SCKDIV$ is odd or zero; else $Gl = SCKDIV * T$.
4. $TCKPOL = 0\text{b}$, $TFSPOL = 0\text{b}$, else the timing references are inverted.
5. Applies to all bits except the first bit transmitted when Transmit Data Delay is zero ($TDDLY = 00\text{b}$). Applies to first bit transmitted when $TDDLY = 01\text{b}$ or $1\text{X}b$ AND $TXDDL = 0\text{b}$; add $tc(MSPTCK)$ for first bit transmitted when $TDDLY = 01\text{b}$ or $1\text{X}b$ AND $TXDDL = 1\text{b}$.
6. Only applies to first bit transmitted when $TDDLY = 00\text{b}$ and $TXDDL = 0\text{b}$; add $tc(MSPTCK)$ for first bit transmitted when $TDDLY = 00\text{b}$ AND $TXDDL = 1\text{b}$.

30.17.2 Transmitter timing requirements (in non-SPI mode)

Table 171. Transmitter timing requirements (in non-SPI mode)

Ref	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MT16	MSPTCK (ext) cycle time	tc(TCKext)	T ⁽¹⁾	-	ns
MT17	MSPTCK (ext) high pulse width	tw(TCKHext)	T/2 -2	-	ns
MT18	MSPTCK (ext) low pulse width	tlx(TCKLext)	T/2-2	-	ns
MT19	MSPTCK (ext) rise time	tr(TCKext)	-	6	ns
MT20	MSPTCK (ext) fall time	tf(TCKext)	-	6	ns
MT21	MSPTFS (ext) valid before MSPTCK (int) low ⁽²⁾	tsu(TFSVext-TCKLint)	10	-	ns
MT22	MSPTFS (ext) valid before MSPTCK (ext) low ⁽²⁾	tsu(TFSVext-TCKLext)	4	-	ns
MT23	MSPTFS (ext) valid after MSPTCK (int) low ⁽²⁾	th(TCKLint-TFSVext)	0	-	ns
MT24	MSPTFS (ext) valid after MSPTCK (ext) low ⁽²⁾	th(TCKLext-TFSVext)	4	-	ns
MT25	MSPTCK (ext) rise time to MSPTFS (ext) valid	td(TCKHext-TFSVext)	5	-	ns

1. T = tc(MSPCLK) when SCKSEL = 0Xb; MSPCKL = 48 MHz, thus T = 20.83 ns
T = tc(MSPSCK) when SCKSEL = 1Xb.

2. TCKPOL = 0b, else the timing references are inverted.

Figure 187. MSP transmitter timings: start of element transmission

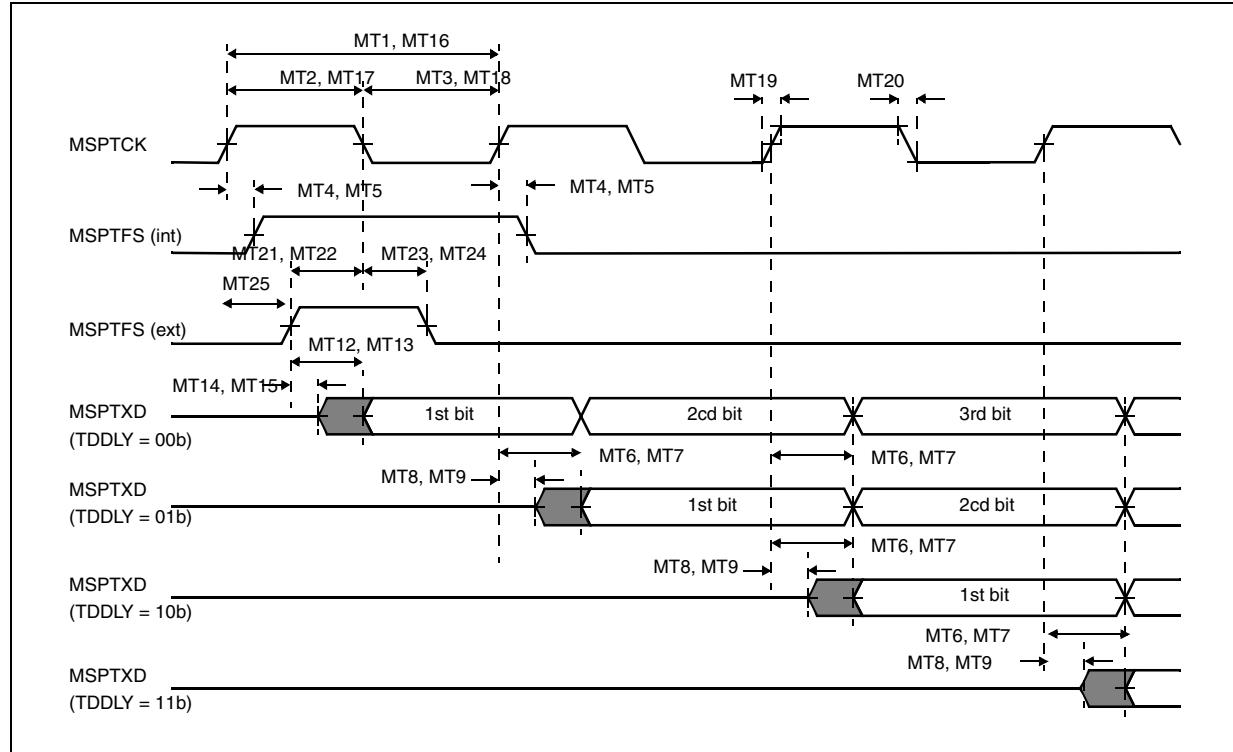
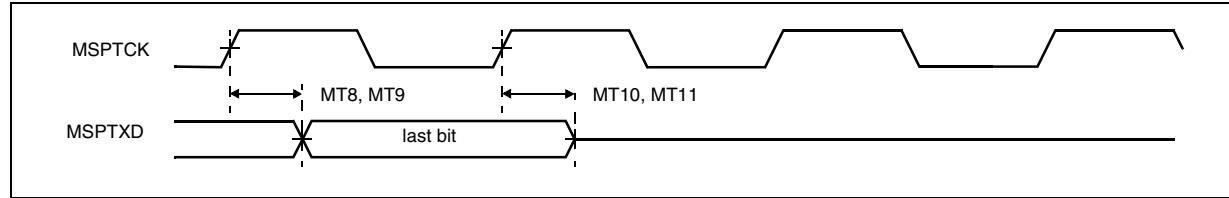


Figure 188. MSP transmitter timings: end of element transmission

30.17.3 Receiver switching characteristics (in non-SPI mode)

Table 172. Receiver switching characteristics (in non-SPI mode)

Ref	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MR1	MSPRCK (int) cycle time	tc(RCKint)	T ⁽¹⁾	-	ns
MR2	MSPRCK (int) high pulse width	tw(RCKHint)	G-2 ⁽²⁾	G+2 ⁽²⁾	ns
MR3	MSPRCK (int) low pulse width	tl(RCKLint)	GI-2 ⁽³⁾	GI+2 ⁽³⁾	ns
MR4	MSPRCK (int) high to MSPRFS (int) valid ⁽⁴⁾	td(RCKHint-RFSVint)	-4	4	ns
MR5	MSPRCK (ext) high to MSPRFS (int) valid ⁽⁴⁾	td(RCKHext-RFSVint)	4	8	ns

1. $T = tc(MSPCLK)$ when $SCKSEL = 0Xb$; $MSPCLK = 48$ MHz, thus $T = 20.83$ ns
 $T = tc(MSPSCK)$ when $SCKSEL = 1Xb$.

2. $Gh = (1 + SCKDIV) * T$ when $SCKDIV$ is odd or zero; else $Gh = (2 + SCKDIV) * T$.

3. $GI = (1 + SCKDIV) * T$ when $SCKDIV$ is odd or zero; else $GI = SCKDIV * T$.

4. $RCKPOL = 0b$, else the timing references are inverted.

30.17.4 Receiver timing requirements (in non-SPI mode)

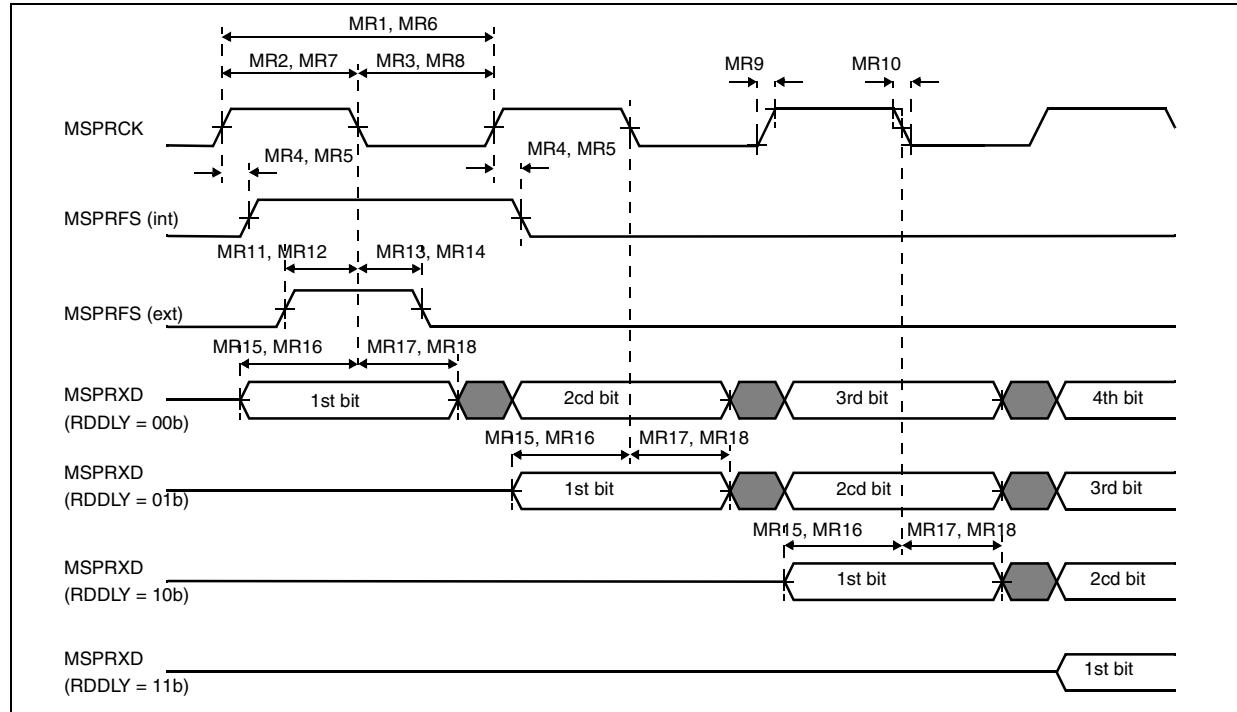
Table 173. Receiver timing requirements (in non-SPI mode)

Ref	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MR6	MSPRCK (ext) cycle time	tc(RCKext)	T ⁽¹⁾	-	ns
MR7	MSPRCK (ext) high pulse width	tw(RCKHext)	T/2 -2	-	ns
MR8	MSPRCK (ext) low pulse width	tw(RCKLext)	T/2-2	-	ns
MR9	MSPRCK (ext) rise time	tr(RCKext)	-	6	ns
MR10	MSPRCK (ext) fall time	tf(RCKext)	-	6	ns
MR11	MSPRFS (ext) high before MSPRCK (int) low	tsu(RCKLint-RFS)	10	-	ns
MR12	MSPRFS (ext) high before MSPRCK (ext) low	tsu(RCKLext-RFSHext)	4	-	ns
MR13	MSPRFS (ext) high after MSPRCK (int) low	th(RCKLint-RFSHext)	0	-	ns
MR14	MSPRFS (ext) high after MSPRCK (ext) low	th(RCKLext-RFSHext)	4	-	ns
MR15	MSPRXD valid before MSPRCK (int) low	tsu(RXDV-RCKLint)	8	-	
MR16	MSPRXD valid before MSPRCK (ext) low	tsu(RXDV-RCKLext)	4	-	

Table 173. Receiver timing requirements (in non-SPI mode)

Ref	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MR17	MSPRXD valid after MSPRCK (int) low	th(RCKLint-RXDV)	0	-	
MR18	MSPRXD valid after MSPRCK (ext) low	th(RCKLext-RXDV)	4	-	

1. $T = 2 * tc(MSPCLK)$ when $SCKSEL = 0Xb$; $MSPCLK = 48$ MHz, thus $T = 41.67$ ns
 $T = tc(MSPSCK)$ when $SCKSEL = 1Xb$.

Figure 189. MSP receiver timings

30.17.5 Switching characteristics as SPI master or slave

Table 174. Switching characteristics as SPI master or slave

Ref	Parameter	Symbol	Timing		Unit
			min.	max.	
MS1	MSPTCK (int) cycle time (SPI Master)	tc(TCKint)	T ⁽¹⁾	-	ns
MS2	MSPTCK (int) high ⁽⁴⁾ pulse width (SPI Master)	tw(TCKHint)	Gh-2 ⁽²⁾	Gh+2 ⁽²⁾	ns
MS3	MSPTCK (int) low ⁽⁴⁾ pulse width (SPI Master)	tw(TCLLInt)	Gl-2 ⁽³⁾	Gl+2 ⁽³⁾	ns
MS4	MSPTFS (int) low to MSPTCK (int) high ⁽⁴⁾	td(TFSLInt-TCKLint)	T - 3	1.5*T+2	ns
MS5	MSPTCK (int) low to MSPTFS (int) high ⁽⁴⁾	td(TCKLint-TFSHint)	0	0.5*T+2	ns
MS6	MSPTCK (int) high ⁽⁴⁾ to MSPTXD valid	td(TCKHint-TXDV)	-	3	ns
MS7	MSPTFS (int) high ⁽⁴⁾ to MSPTXD driven	td(TFSHint-TXDLZ)	0	10	ns
MS8	MSPTFS (int) high ⁽⁴⁾ to MSPTXD Hz	td(TFSHint-TXDLZ)	0	10	ns

1. T = tc(MSPCLK) when SCKSEL = 0Xb; MSPCKL = 48 MHz, thus T = 20.83 ns
T = tc(MSPSCK) when SCKSEL = 1Xb.

2. Gh = (1 + SCKDIV) * T when SCKDIV is odd or zero; else Gh = (2 + SCKDIV) * T.

3. Gl = (1 + SCKDIV) * T when SCKDIV is odd or zero; else Gl = SCKDIV * T.

4. TCKPOL = 0b, else the timing references are inverted.

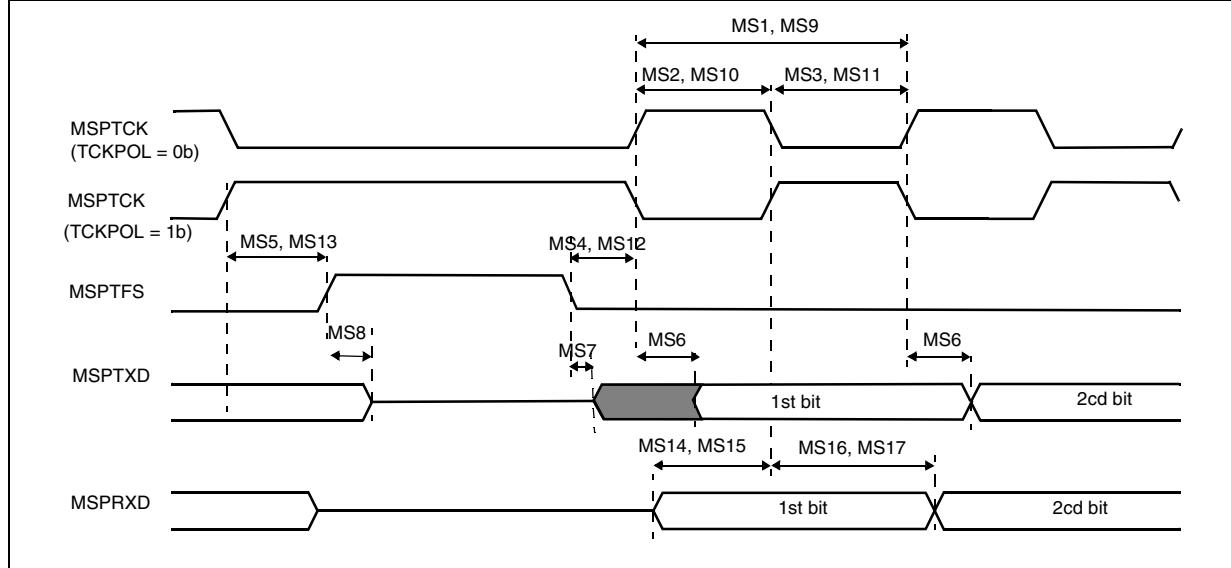
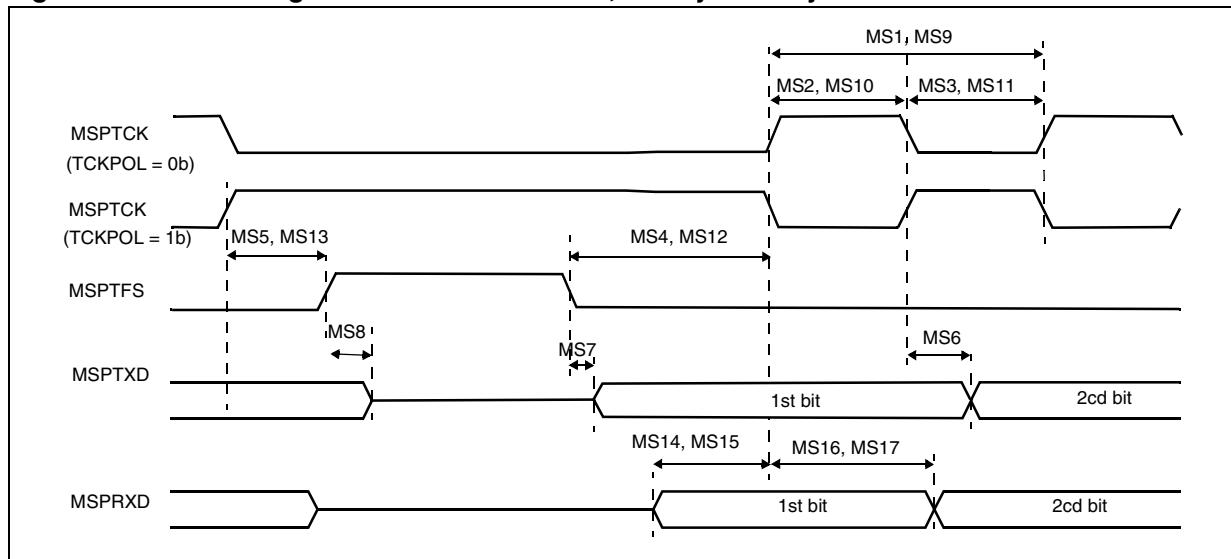
30.17.6 Timing requirements as SPI master or slave

Table 175. Timing requirements as SPI master or slave

Ref	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MS9	MSPTCK (ext) cycle time (SPI Slave)	tc(TCKext)	8 * T ⁽¹⁾	-	ns
MS10	MSPTCK (ext) high ⁽²⁾ pulse width (SPI Slave)	tw(TCKHext)	4 * T - 2	-	ns
MS11	MSPTCK (ext) low ⁽²⁾ pulse width (SPI Slave)	tw(TCKLext)	4 * T - 2	-	ns
MS12	MSPTFS (ext) low before MSPTCK (ext) high ⁽²⁾	tsu(TFSLext-TCKLext)	10	-	ns
MS13	MSPTFS (ext) low after MSPTCK (ext) high ⁽²⁾	th(TCKLext-TFSLext)	10	-	ns
MS14	MSPRXD valid before MSPTCK (int) high ⁽²⁾ , SPICLKDLY=1	tsu(TRXDV-TCKLint)	6	-	ns
	MSPRXD valid before MSPTCK (int) high ⁽²⁾ , SPICLKDLY=0	tsu(TRXDV-TCKLint)	2	-	ns
MS15	MSPRXD valid before MSPTCK (ext) high ⁽²⁾	tsu(TRXDV-TCKLext)	0	-	ns
MS16	MSPRXD valid after MSPTCK (int) high ⁽²⁾	th(TCKLint-TRXDV)	3.5	-	ns
MS17	MSPRXD valid after MSPTCK (ext) high ⁽²⁾	th(TCKLext-TRXDV)	12	-	ns

1. T = tc(MSPCLK) when SCKSEL = 0Xb; MSPCKL = 48 MHz, thus T = 20.83 ns
T = tc(MSPSCK) when SCKSEL = 1Xb.

2. TCKPOL = 0b, else the timing references are inverted.

Figure 190. MSP timings as SPI master or slave, zero delay SPI mode**Figure 191. MSP timings as SPI master or slave, half-cycle delay SPI mode**

30.17.7 Baud rate generator timing requirements (all modes)

The timings below apply to MSP0 and MSP1 only.

Table 176. Baud-rate generator timings

Ref	Parameter	Symbol	Timing		Unit
			Min.	Max.	
MB1	MSPSCK cycle time	tc(SCK)	20	-	ns
MB2	MSPSCK high pulse width	tw(SCKH)	8	-	ns
MB3	MSPSCK low pulse width	ttx(SCKL)	8	-	ns
MB4	MSPSCK rise time	tr(SCK)	-	6	ns
MB5	MSPSCK fall time	tf(SCK)	-	6	ns

30.18 I²C timings

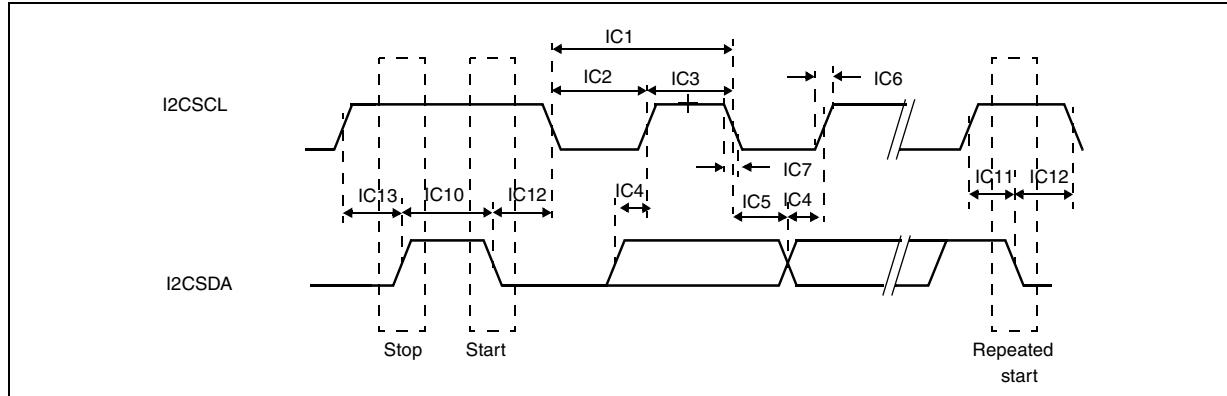
The following timings are provided for the two interfaces, I²C0 and I²C1. The I²C signals switching characteristics are shown in *Table 177*.

Table 177. I²C timing characteristics

No.	Parameter	Symbol	Standard mode		Fast mode		High speed mode ⁽¹⁾		Unit
			min.	max.	min.	max.	min.	max.	
IC1	I ² CSCL cycle time	tc(SCL)	10	-	2.5	-	0.295	-	μs
IC2	I ² CSCL low pulse width	tw(SCLL)	4.7	-	1.3	-	0.160	-	μs
IC3	I ² CSCL high pulse width	tw(SCLH)	4.0	-	0.6	-	0.06	-	μs
IC4	I ² CSDA valid before I ² CSCL high	tsu(SDAV-SCLH)	250	-	100	-	10	-	ns
IC5	I ² CSDA valid after I ² CSCL low	th(SDAV-SCLL)	0 ⁽²⁾	-	0 ⁽²⁾	0.9 ⁽³⁾	0 ⁽²⁾	0.07 ⁽⁴⁾	μs
IC6	I ² CSCL rise time	tr(SCL)	-	1000	-	300	-	-	ns
IC7	I ² CSCL fall time	tf(SCL)	-	300	-	300	-	-	ns
IC8	I ² CSDA rise time	tr(SDA)	-	1000	-	300	-	-	ns
IC9	I ² CSDA fall time	tf(SDA)	-	300	-	300	-	-	ns
IC10	I ² CSDA high pulse width between STOP and START	tw(SDAH)	4.7	-	1.3	-	-	-	μs
IC11	I ² CSCL high before I ² CSDA low for a repeated START condition	tsu(SCLH-SDAL)	4.7	-	1.3	-	0.160	-	μs
IC12	I ² CSCL low after I ² CSDA low	th(SCLL-SDAL)	4.0	-	1.3	-	0.160	-	μs
IC13	I ² CSCL high before I ² CSDA high for a STOP condition	tsu(SCLH-SDAH)	4.0	-	0.6	-	0.160	-	μs

1. High-speed values are given for a maximum load on the bus of 100pF.
2. A device must internally provide a hold time of at least 300ns for the I²CSDA signal (referred to the V_{IHmin} of the I²CSCL signal) to bridge the undefined region of the falling edge of the I²CSCL signal.
3. The maximum th(SDAV-SCLL) has only to be met if the device does not stretch the LOW period (tw(SCLL) of the I²CSCL signal).
4. The maximum th(SDAV-SCLL) has only to be met if the device does not stretch the LOW period (tw(SCLL) of the I²CSCL signal).

Note: The maximum total capacitance allowed on one bus line is 150 pF.

Figure 192. I²C bus timings

30.19 1-Wire Master interface (OWM) timings

The following timings are provided for the 1-Wire Master interface, OWM.

Table 178. 1-Wire master switching characteristics

No.	Parameter	Symbol	TSLOT parameter value				Unit
			00	01	10	11	
OW1	Time Slot width	tw(SLOT)	11	73	223	223	μs
OW2	OWMDQ low pulse width, write 0	tw(OWMDQL0)	8	63	127	127	μs
OW3	OWMDQ low pulse width, write 1	tw(OWMDQL1)	6	6	6	6	μs
OW4	OWMDQ low pulse width, read start	tw(OWMDQLR)	1	1	1	1	μs
OW5	OWMDQ low to read sampling point	td(OWMDQL-RDSP)	2	15	79	79	μs
OW6	OWMDQ low pulse width, reset	tw(OWMDQLRST)	61	488	488	223	μs
OW7	OWMDQ released pulse width, reset	tw(OWMDQHiZRST)	50	500	500	50	μs
OW8	OWMDQ released to OWMDQ sampling point, presence detect during reset	td(OWMDQHiZ-RSTSP)	3	30	30	30	μs

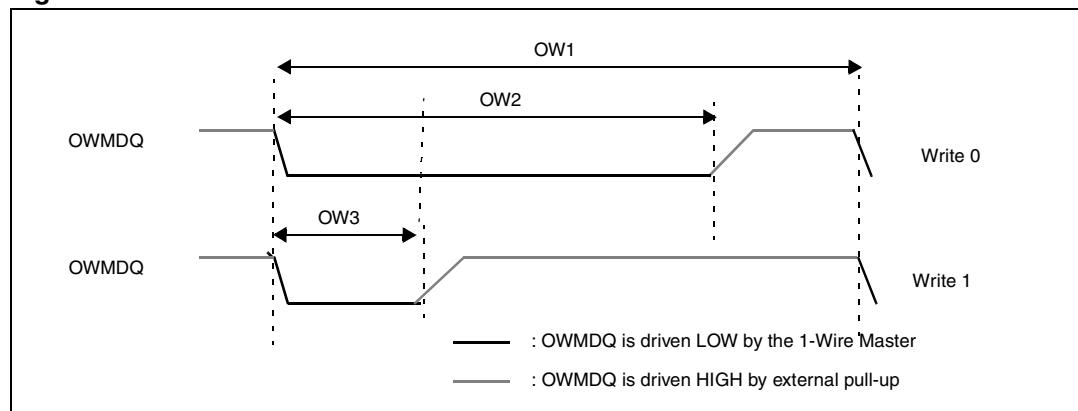
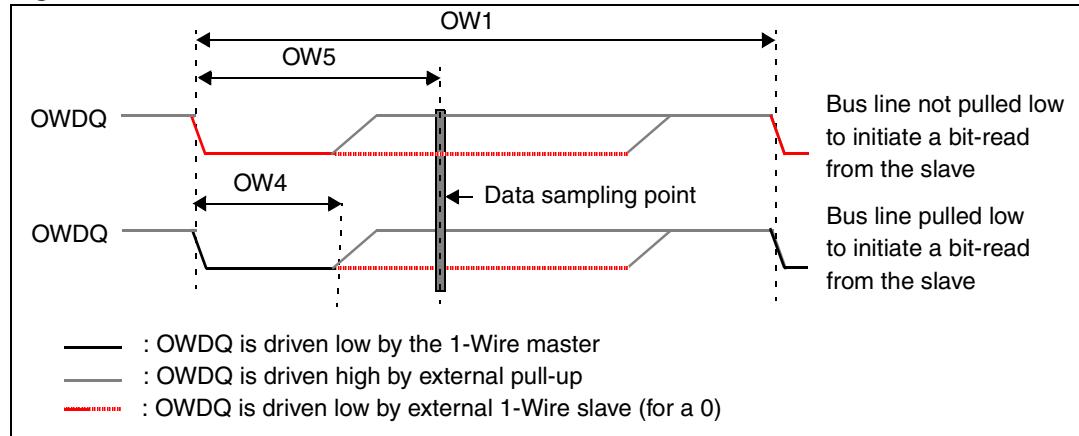
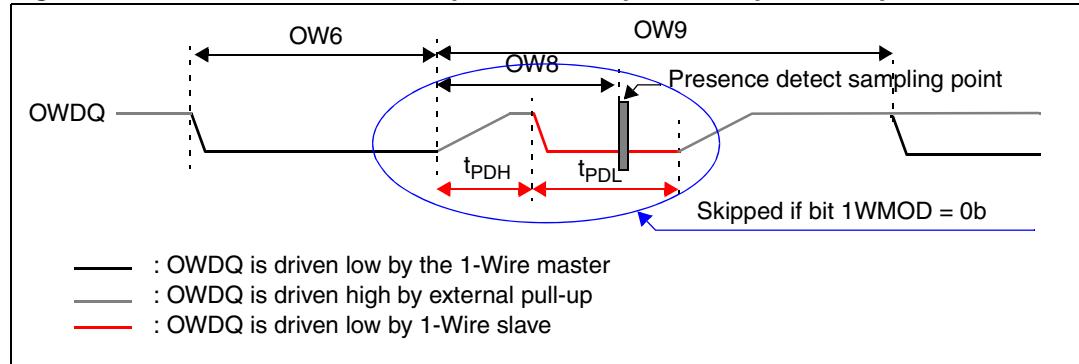
Figure 193. Write time slots

Figure 194. Read time slots**Figure 195. 1-Wire initialization sequence: reset pulse and presence pulse**

30.20 USB timing

The following timings are provided for the USB UTMI low pin count interface (ULPI). These timing characteristics are defined by the ULPI specification v1.1.

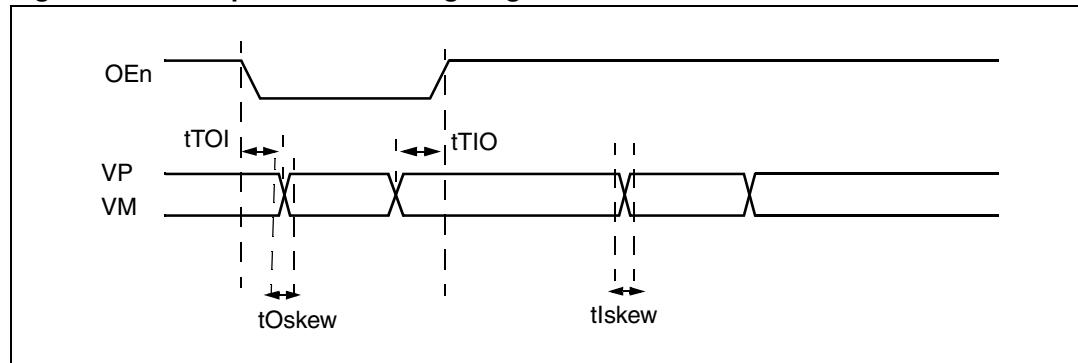
30.20.1 Full speed mode timings

Given below are the switching characteristics of the USB interface. Rising and falling edges are measured based on the 10/90% levels.

Table 179. Full speed mode timing parameters

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
FC	Clock frequency +/- 500ppm	tc(CLK)	11.97	12.03	MHz
DCYC	Duty cycle +/- 500ppm	Dsteady	49.975	50.025	%
TJ	Clock jitter	tj(XCLK)		300	ps
TRISE	Clock rise/fall time	tr/f(XCLK)		3	ns
TOI	Output bus turnaround time	tTOI	tc -3ns		ns
TIO	Input bus turnaround time	tTIO	tc -3ns		ns
Iskew	Input skew on VP/VM	tIskew	0	7	ns
Oskew	Output skew on VP/VM	tOskew	0	0.6	ns

Figure 196. Full speed mode timing diagram

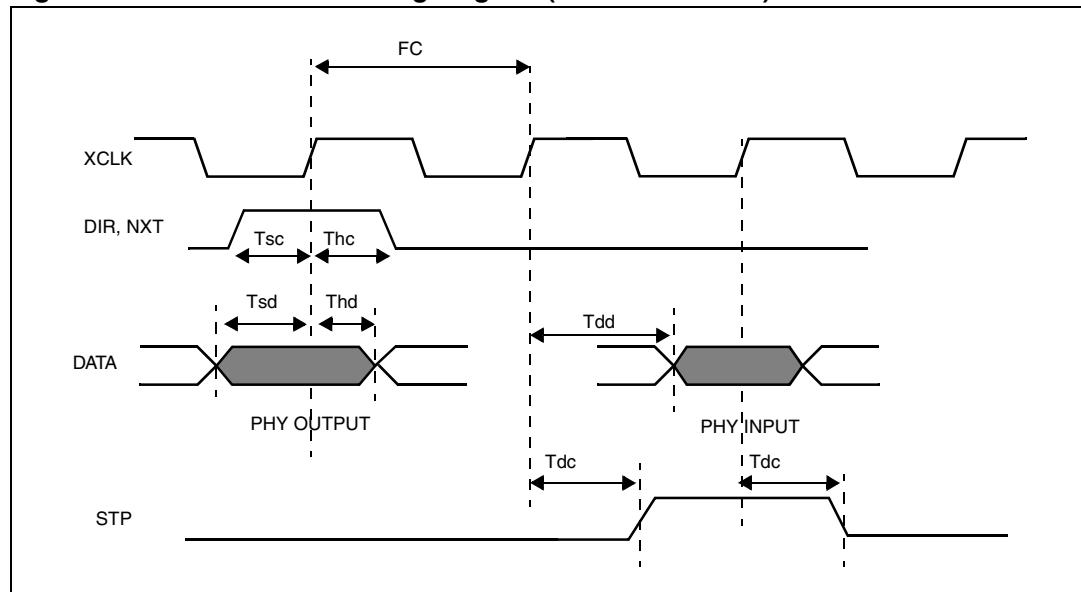


30.20.2 ULPI interface timings (single data rate)

Table 180. ULPI interface timing parameters (single data rate)

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
FC	Clock frequency +/- 500ppm	tc(CLK)	59.97	60.03	MHz
DCYC	Duty cycle +/- 500ppm	Dsteady	49.975	50.025	%
TJ	Clock jitter	tj(XCLK)		300	ps
TRISE	Clock Rise/fall time	tr/f(XCLK)			ns
	IN Data setup time	Tsc, Tsd	7.5		ns
	IN Data hold time	Thc, Thd	0		ns
	OUT Data delay	Tdc, Tdd	0	10	ns

Figure 197. ULPI interface timing diagram (normal data rate)



30.21 Embedded Trace Module (ETM) timings

30.21.1 Switching characteristics for ETM outputs

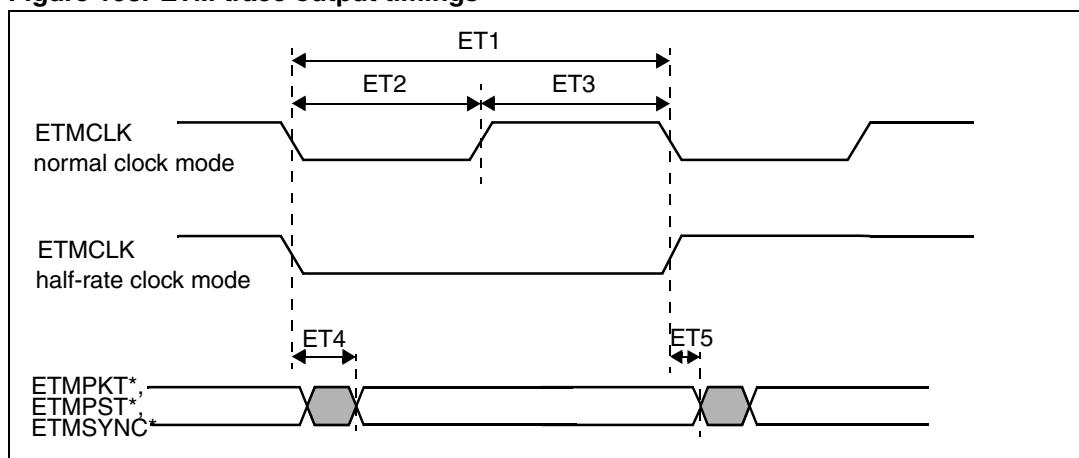
All the switching timing characteristics are relative to ETMCLK signal.

Table 181. Switching characteristics for ETM outputs

No.	Parameter	Symbol	Timing		Unit
			Min.	Max.	
ET1	ETMCLK cycle time	tc(ETMCLK)	3.78	-	ns
ET2	ETMCLK low pulse width	tw(ETMCLKL)	2	-	ns
ET3	ETMCLK high pulse width	tw(ETMCLKW)	2	-	ns
ET4	ETMCLK edge to ETMOUT ⁽¹⁾ valid	tdv(ETMCLK-ETMOUTV)	-	2	ns
ET5	ETMCLK edge to ETMOUT ⁽¹⁾ invalid	tdiv(ETMCLK-ETMOUTIV)	0	-	ns

1. ETMOUT stands here for ETMPKTA[3:0], ETMPKTA4B0, ETMPKTA5B1, ETMPKTA6B2, ETMPKTA7B3, ETMSYNCA, and ETMPSTA[2:0] signals in normal trace mode, and for ETMPKTA[3:0], ETMPKTA4B0, ETMPKTA5B1, ETMPKTA6B2, ETMPKTA7B3, ETMSYNCA, ETMSYNCB, ETMPSTA[2:0], and ETMPSTB[2:0] signals in demultiplexed trace mode.

Figure 198. ETM trace output timings



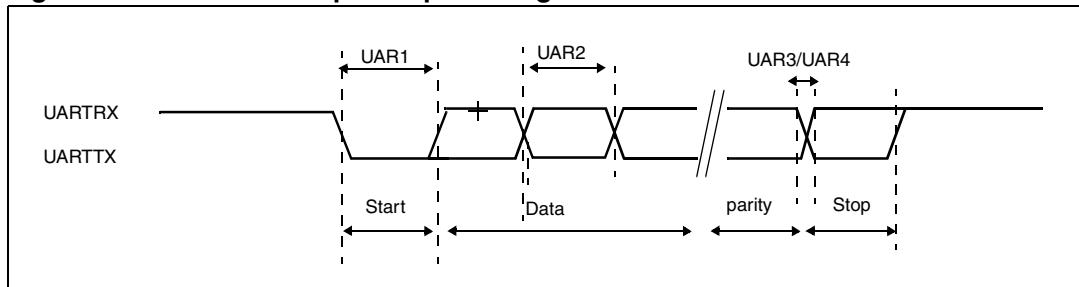
30.22 Asynchronous serial interface (UART) timings

30.22.1 Switching characteristics for UART input/output

All the switching timing characteristics are relative to a signal transition on UARTRX or UARTTX.

Table 182. UART input/output switching characteristics

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
UAR1	UARTCLK cycle time, nominal bit (3 Mbps)	tc(UARTCLK)	333.33	-	ns
UAR2	Duration time low level or high level UARTRX/TX	td(UARTRX/TX)	tc - 10	tc + 10	ns
UAR3	Rise time UARTRX/TX	tr	-	5	ns
UAR4	Fall time UARTRX/TX	tf	-	5	ns

Figure 199. UART trace input/output timings

30.23 IRDA serial interface (IRDA) timings

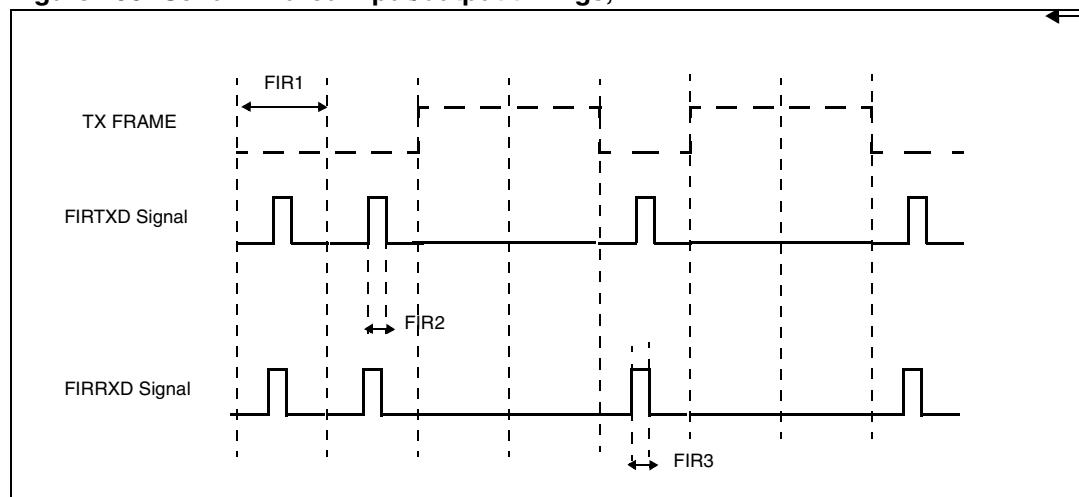
30.23.1 Switching characteristics for serial infrared (SIR), medium infrared (MIR) mode, fast infrared (FIR) mode

All the switching timing characteristics are relative to a signal transition on FIRRXD or FIRTXD.

Table 183. SIR, MIR and FIR mode switching characteristics

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
FIR1 (SIR)	FIRDACLK cycle time, (nominal bit 115 Kbits/s)		8.68	-	μs
FIR1 (MIR)	FIRDACLK cycle time, (nominal bit 1.152 Mbits/s)		868	-	ns
FIR1 (FIR)	FIRDACLK cycle time, (nominal bit 4 Mbits/s)		250	-	ns
FIR2 (SIR)	Duration time at high level (nominal 1.736μs)		1.4	2.2	μs
FIR2 (MIR)	Duration time at high level (nominal 217ns)		150	260	ns
FIR2 (FIR)	Duration time at high level (nominal 125 ns)		115	135	ns
FIR3 (SIR)	Duration time at high level (nominal 1.736μs)		1.4	2.2	μs
FIR3 (MIR)	Duration time at high level (nominal 217ns)		150	260	ns
FIR3 (FIR)	Duration time at high level (nominal 125 ns)		115	135	ns
FIR4	Rise time FIRDATA	tr	-	5	ns
FIR5	Fall time FIRDATA	tf	-	5	ns

Figure 200. Serial infrared input/output timings,



30.24 JTAG timings

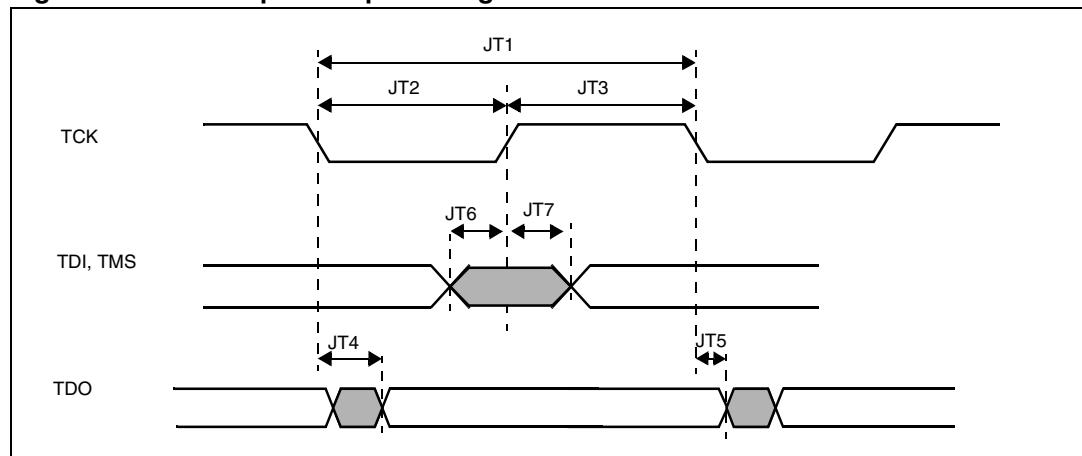
30.24.1 Switching characteristics for JTAG inputs/outputs

All the switching timing characteristics are relative to TCK signal.

Table 184. JTAG input/output switching characteristics

No.	Parameter	Symbol	Timing		Unit
			min.	max.	
JT1	TCK cycle time	tc(TCK)	250	-	ns
JT2	TCK low pulse width	twl(TCK)	110	-	ns
JT3	TCK high pulse width	twh(TCK)	110	-	ns
JT4	TCK falling edge to valid	tdov(TCK-TDO)	-	40	ns
JT5	TCK falling edge to invalid	tdoinv(TCK-TDO)	1	-	ns
JT6	Valid data before TCK rising edge	tdiv(TDI-TCK)	20	-	ns
JT7	Invalid data after TCK rising edge	tdiinv(TDI-TCK)	2	-	ns

Figure 201. JTAG inputs/output timings



31 Package mechanical data

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: www.st.com.

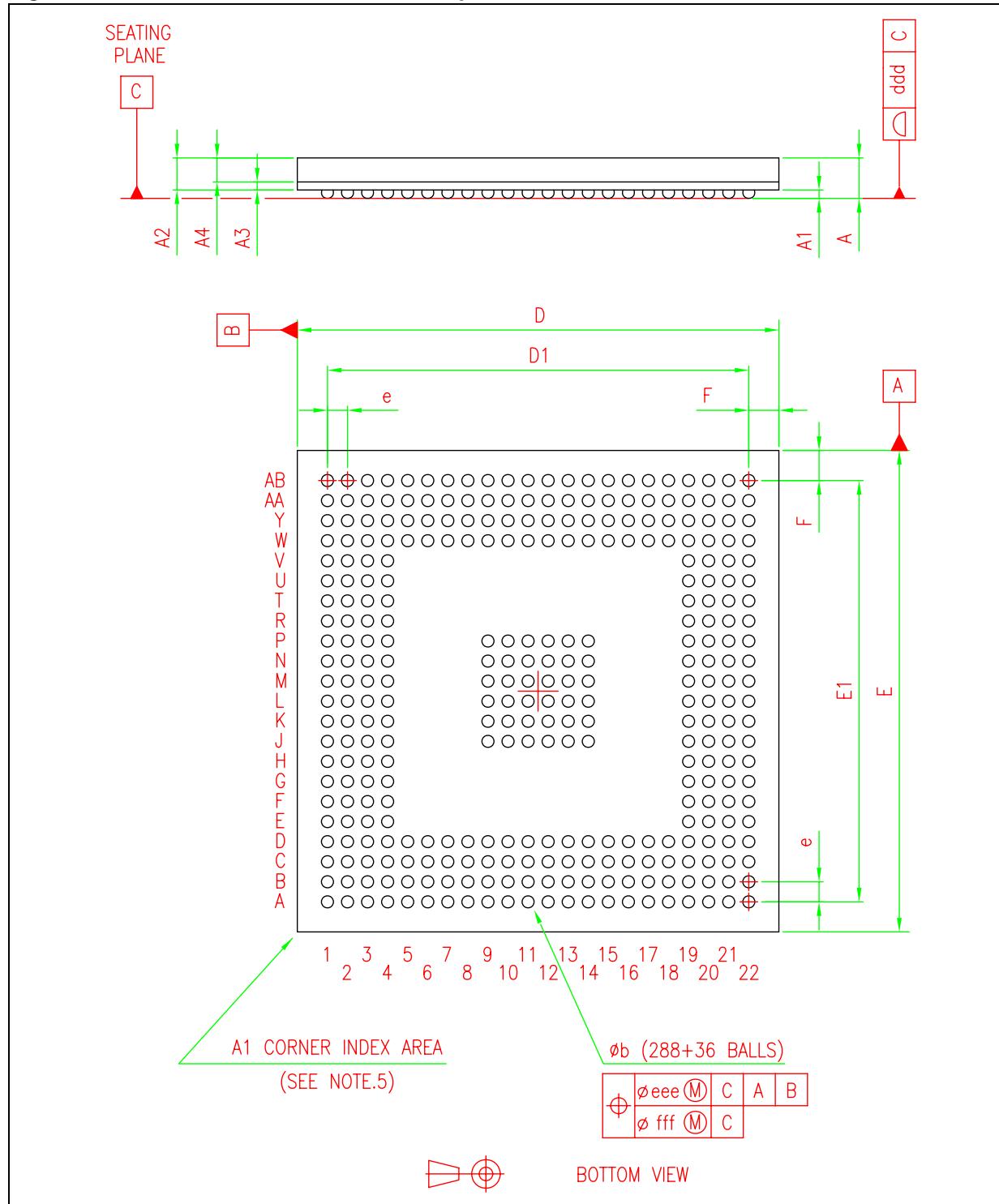
Table 185. Package dimensions

Ref.	Databook (mm)			Drawing (mm)		
	Min	Typ	Max	Min	Typ	Max
A		-	1.20	-	-	1.06 ⁽¹⁾
A1	0.15	-	-	0.16	0.21	0.26
A2	-	0.785	-	0.73	0.785	0.84
A3		0.20		0.16	0.20	0.24
A4			0.60	0.57	0.585	0.60
b	0.25	0.30	0.35	0.25	0.30	0.35 ⁽²⁾
D	11.85	12.00	12.15	11.90	12.00	12.10
D1		10.50			10.50	
E	11.85	12.00	12.15	11.90	12.00	12.10
E1		10.50			10.50	
e		0.50			0.50	
F		0.75			0.75	
ddd			0.08			0.08
eee			0.15			0.15 ⁽³⁾
fff			0.05			0.05 ⁽⁴⁾

1. TFBGA stands for **Thin profile Fine pitch Ball Grid Array**.
 - Low profile: the total profile height (dim A) is measured from the seating plane to the top of the component
 - The maximum total package height is calculated by the following methodology:

$$\text{A2 Typ} + \text{A1 Typ} \pm (\text{A1}^2 + \text{A3}^2 + \text{A4}^2 \text{ tolerance values})$$
 - Thin profile: 1.00mm < A <= 1.20mm/Fine pitch: e < 1.00mm pitch
2. The typical ball diameter before mounting is 0.30mm.
3. The tolerance of position that controls the location of the pattern of balls with respect to datums A and B. For each ball there is a cylindrical tolerance zone eee perpendicular to datum C and located on true position with respect to datums A and B as defined by e. The axis perpendicular to datum C of each ball must lie within this tolerance zone.
4. The tolerance of position that controls the location of the balls within the matrix with respect to each other. For each ball there is a cylindrical tolerance zone fff perpendicular to datum C and located on true position as defined by e. The axis perpendicular to datum C of each ball must lie within this tolerance zone. Each tolerance zone fff in the array is contained entirely in the respective zone eee above. The axis of each ball must lie simultaneously in both tolerance zones.

Figure 202. TFBGA 12mm x 12mm x 1.2mm pitch 0.5 mm ball 0.3



Note: The terminal A1 corner must be identified on the top surface by using a corner chamfer, ink or metallized markings, or other feature of package body or integral heatslug. A distinguishing feature is allowable on the bottom surface of the package to identify the terminal A1 corner. Exact shape of each corner is optional.

32 Ordering information

Table 186. Ordering information

Order codes	Package	Packing
STN8815DxA12H11E ⁽¹⁾	TFBGA 12 mm x 12 mm x 1.2 mm , 0.5 mm pitch	Tray
STN8815DxA12H11T ⁽¹⁾	TFBGA 12 mm x 12 mm x 1.2 mm , 0.5 mm pitch	Tape and reel

1. Please contact ST-Ericsson marketing and sales organization for details on the product order codes.

33 Revision history

Table 187. Document revision history

Date	Revision	Changes
21-Aug-2009	1	Initial release

Please Read Carefully:

The contents of this document are subject to change without prior notice. ST-Ericsson makes no representation or warranty of any nature whatsoever (neither expressed nor implied) with respect to the matters addressed in this document, including but not limited to warranties of merchantability or fitness for a particular purpose, interpretability or interoperability or, against infringement of third party intellectual property rights, and in no event shall ST-Ericsson be liable to any party for any direct, indirect, incidental and or consequential damages and or loss whatsoever (including but not limited to monetary losses or loss of data), that might arise from the use of this document or the information in it.

ST-Ericsson and the ST-Ericsson logo are trademarks of the ST-Ericsson group of companies or used under a license from STMicroelectronics NV or Telefonaktiebolaget LM Ericsson.

All other names are the property of their respective owners.

© ST-Ericsson, 2009 - All rights reserved

Contact information at www.stericsson.com under Contacts

www.stericsson.com