UM10314

LPC3130/31 User manual

Rev. 1 — 4 March 2009 User manual

Document information

Info	Content
Keywords	LPC3130, LPC3131, ARM9, USB
Abstract	LPC3130/31 User manual

Revision history

Rev	Date	Description
1	20090304	LPC3130/31 User manual

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

UM10314

Chapter 1: LPC3130/31 Introductory information

Rev. 1 — 4 March 2009 **User manual**

1. Introduction

The NXP LPC3130/31 combine an 180 MHz ARM926EJ-S CPU core, High Speed USB 2.0 OTG, up to 192 KB SRAM, NAND Flash Controller, flexible external bus interface, four channel 10-bit A/D, and a myriad of serial and parallel interfaces in a single chip targeted at consumer, industrial, medical, and communication markets. To optimize system power consumption, the LPC3130/31 has multiple power domains and a very flexible Clock Generation Unit (CGU) that provides dynamic clock gating and scaling.

2. Features

2.1 Key features

- CPU platform
  - 180 MHz, 32-bit ARM926EJ-S
  - 16 kB D-cache and 16 kB I-cache
  - Memory Management Unit (MMU)
- Internal memory
  - 96 kB (LPC3130) or 192 kB (LPC3131) embedded SRAM
- External memory interface
  - NAND flash controller with 8-bit ECC
  - 8/16-bit Multi-Port Memory Controller (MPMC): SDRAM and SRAM
- Communication and connectivity
  - High-speed USB 2.0 (OTG, Host, Device) with on-chip PHY
  - Two I^2S interfaces
  - Integrated master/slave SPI
  - Two master/slave I^2C-bus interfaces
  - Fast UART
  - Memory Card Interface (MCI): MMC/SD/SDIO/CE-ATA
  - Four-channel 10-bit ADC
  - Integrated 4/8/16-bit 6800/8080 compatible LCD interface
- System functions
  - Dynamic clock gating and scaling
  - Multiple power domains
  - Selectable boot-up: SPI flash, NAND flash, SD/MMC cards, UART, or USB
  - DMA controller
  - Four 32-bit timers

- Watchdog timer
- PWM module
- Random Number Generator (RNG)
- General Purpose I/O pins (GPIO)
- Flexible and versatile interrupt structure
- JTAG interface with boundary scan and ARM debug access

- Operating voltage and temperature
  - Core voltage: 1.2 V
  - I/O voltage: 1.8 V, 2.8 V, 3.3 V
  - Temperature: −40 °C to +85 °C

3. Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC3130	TFBGA180	Plastic thin fine pitch ball grid array package, 180 balls, body 12 x 12 x 0.8 mm	SOT570-2
LPC3131	TFBGA180	Plastic thin fine pitch ball grid array package, 180 balls, body 12 x 12 x 0.8 mm	SOT570-2

Table 2. Ordering options for LPC3130/31

Type number	Core/bus frequency	Total SRAM	High-speed USB	10-bit ADC channels	I^2S/ I^2C -bus	MCI SDHC/ SDIO/ CE_ATA	Temperature range
LPC3130	180 MHz/ 90 MHz	96 kB	Device/ Host/OTG	4	2 each	yes	−40 °C to +85 °C
LPC3131	180 MHz/ 90 MHz	192 kB	Device/ Host/OTG	4	2 each	yes	−40 °C to +85 °C

4. Block diagram

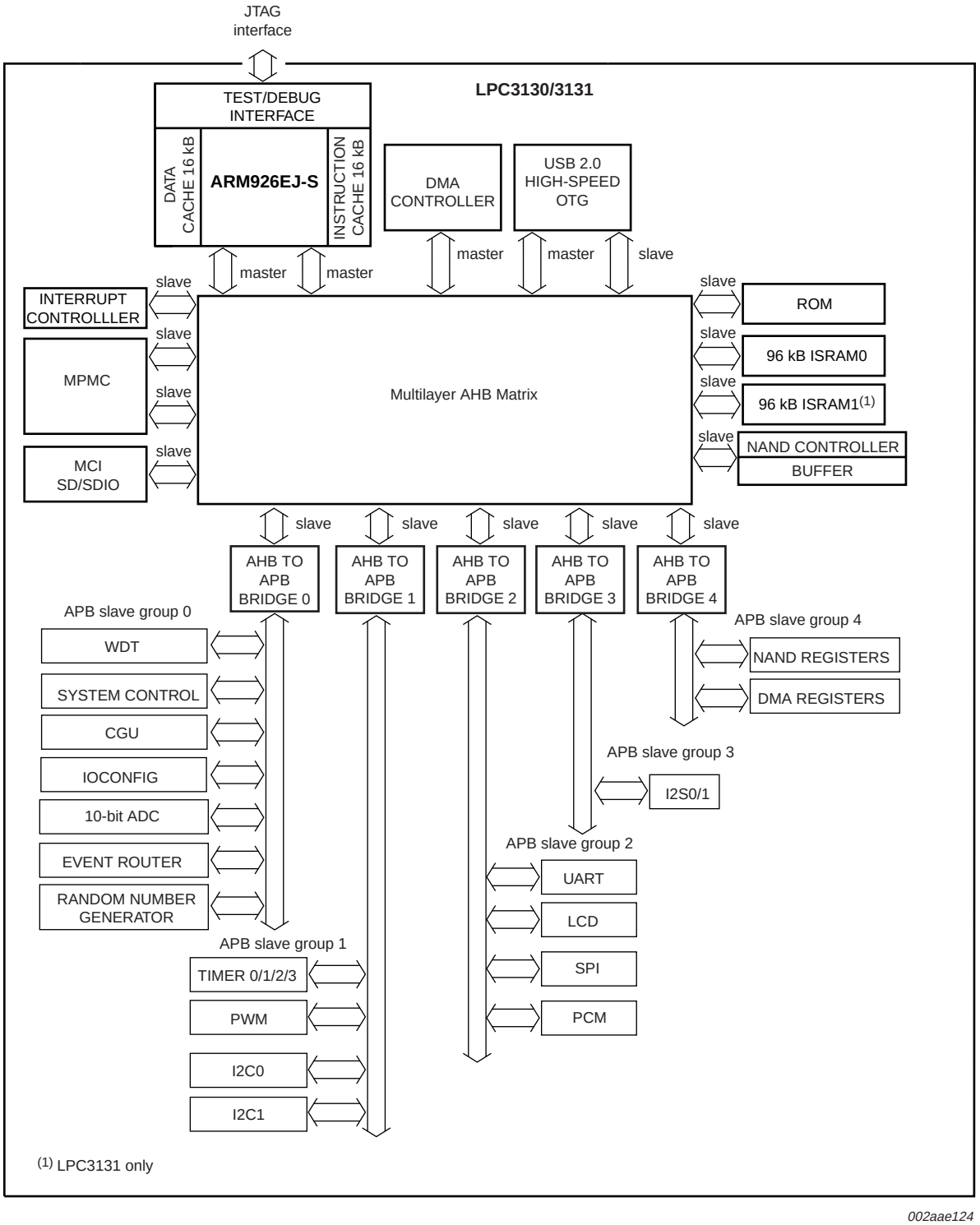


Fig 1. LPC3130/31 block diagram

5. Architectural overview

5.1 ARM926EJ-S

The processor embedded in the LPC3130/31 is the ARM926EJ-S. It is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S is intended for multi-tasking applications where full memory management, high performance, and low power are important.

This module has the following features:

- ARM926EJ-S processor core which uses a five-stage pipeline consisting of fetch, decode, execute, memory, and write stages. The processor supports both the 32-bit ARM and 16-bit Thumb instruction sets, which allows a trade off between high performance and high code density. The ARM926EJ-S also executes an extended ARMv5TE instruction set which includes support for Java byte code execution.
- Contains an AMBA BIU for both data accesses and instruction fetches.
- Memory Management Unit (MMU).
- 16 kB instruction and 16 kB data separate cache memories with an 8 word line length. The caches are organized using Harvard architecture.
- Little Endian is supported.
- The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debugging.
- Supports dynamic clock gating for power reduction.
- The processor core clock can be set equal to the AHB bus clock or to an integer number times the AHB bus clock. The processor can be switched dynamically between these settings.
- ARM stall support.

5.2 Internal ROM Memory

The internal ROM memory is used to store the boot code of the LPC3130/31. After a reset, the ARM processor will start its code execution from this memory.

The LPC3130/31 ROM memory has the following features:

- Supports booting from SPI flash, NAND flash, SD/SDHC/MMC cards, UART, and USB (DFU class) interfaces.
- Supports option to perform CRC32 checking on the boot image.
- Supports booting from managed NAND devices such as moviNAND, iNAND, eMMC-NAND and eSD-NAND using SD/MMC boot mode.
- Contains pre-defined MMU table (16 kB) for simple systems.

5.3 Internal RAM memory

The ISRAM (Internal Static Memory Controller) module is used as controller between the AHB bus and the internal RAM memory. The internal RAM memory can be used as working memory for the ARM processor and as temporary storage to execute the code that is loaded by boot ROM from external devices such as SPI-flash, NAND flash, and SD/MMC cards.

This module has the following features:

- Capacity of 96 kB (LPC3130) or 192 kB (LPC3131).
- On LPC3131 implemented as two independent 96 kB memory banks.

1. Introduction

The NAND flash controller is used to transfer data between the LPC3130/3131 and external NAND flash devices.

1.1 Features

- AHB/APB interface
  - AHB slave interface.
  - AHB interface supports 0,1 and 2 wait states.
  - 2 SRAMs of 132 words, 32 bits per word used in a double buffering accessible via the AHB bus.
  - Programming by CPU via APB interface using zero wait states.
  - Little and big endian support.
  - Automatic flow control with the DMA controller, using ext_en/ext_ack signals.
- NAND flash support
  - Dedicated interface to NAND flash devices.
  - Hardware controlled read and write data transfers.
  - Software controlled command and address transfers to support a wide range of NAND flash devices.
  - GPIO mode.
  - Software control mode where the ARM is directly master of the NAND flash device.
  - Support for 8bit & 16bit NAND flash devices.
  - Support for 528 byte, 2K and 4K page NAND flash devices.
  - Programmable NAND timing parameters.
  - Support for up to 4 NAND flash device dies in parallel with dedicated chip select and ready/busy pin per device.
  - Programmable default state of output signals.
  - Erased page detection.
  - EBI compatible.
- Error correction
  - Two Reed-Solomon error correction codes, one offering 5 symbol error correction and the other 8 symbol error correction capability. 5 symbol correcting code is of length 469, dimension 459, and minimum distance 11 over GF(2^9). 8 symbol correcting code has length 475, dimension 459 and minimum distance 17 over GF(2^9).
  - Two parity generators.
  - Wear leveling and other extra information can be integrated into protected data.
  - Interrupts generated after completion of error correction task with 3 interrupt registers.

- Error correction statistics distributed to ARM using interrupt scheme.
- Error correction can be turned on and off.

Remark: The wear-leveling algorithm ensures that data is stored in different flash pages across the flash media. This not only extends its lifetime, but also ensures reliable operation.

2. General description

2.1 Clock signals

The CGU provides different clocks to the NAND flash controller, see Table 2–3.

Table 3. NAND flash controller clock overview

Clock name	Clock acronym	I/O	Source/ Destination	Description
NANDFLASH_S0_CLK	ahb_clk	I	CGU	AHB port clock of the module
NANDFLASH_PCLK	PCLK	I	CGU	APB port clock of the module
NANDFLASH_NAND_CLK	nand_clk	I	CGU	Main clock for the module
NANDFLASH_ECC_CLK	ecc_clk	I	CGU	Main clock for ECC part in the module. This clock should be programmed to run synchronously at half the NANDFLASH_NAND_CLK in CGU block.

2.2 Reset signals

The CGU provides the following resets to the NAND flash controller (see Section 13–4.2.2).

1. AHB0_RESERT: Low-active, synchronous reset. Resets the logic in the ahb_clk domain.
2. APB4_RESETN: Low-active, synchronous reset. Resets the logic in the PCLK domain.
3. NANDFLASH_CTRL_NAND_RESET_N: High-active, synchronous reset. Resets the logic in the main NAND flash controller nand_clk domain.
4. NANFLASH_CTRL_ECC_RESET_N: High-active, synchronous reset. Resets the logic in the ecc_clk domain.

2.3 Interrupt request

The NAND flash controller generates one interrupt request towards the CPU. The interrupt sources are controlled by two sets of registers: NandIRQStatus1, NandIRQMask1, NandIRQStatusRaw1 and NandIRQStatus2, NandIRQMask2, NandIRQStatusRaw2. See Table 2–6 to Table 2–8 and Table 2–22 to Table 2–24 for a description of interrupt sources.

2.4 DMA transfers

The NAND flash controller has DMA support by means of external enabling. The transfer size is 128 words. DMA auto-flow control is supported only by DMA channel 4.

2.5 External pin connections

Table 2–4 gives an overview of the external connections to and from the NAND flash controller.

Table 4. NAND flash controller external pin overview

Pin name	Interface	Acronym	Type (Func.)	Reset Value	Description
EBI_D_[15:0]	EBI	-	I	-	16 bits data from NAND flash device
EBI_D_[15:0]		-	O	all 0	16 bits data to NAND flash device
NAND_NCS_0		CS1_n	O	1	Low-active Chip Enable 0
NAND_NCS_1		CS2_n	O	1	Low-active Chip Enable 1
NAND_NCS_2		CS3_n	O	1	Low-active Chip Enable 2
NAND_NCS_3		CS4_n	O	1	Low-active Chip Enable 3
EBI_NWE	EBI	WE_n	O	1	Low-active Write Enable
EBI_DQM_0_NOE	EBI	RE_n	O	1	Low-active Read Enable
EBI_A_0_ALE	EBI	ALE	O	0	High-active Address Latch Enable
EBI_A_1_CLE	EBI	CLE	O	0	High-active Command Latch Enable
mNAND_RYBN0		RnB0	I	-	Ready not Busy 0
mNAND_RYBN1		RnB1	I	-	Ready not Busy 1
mNAND_RYBN2		RnB2	I	-	Ready not Busy 2
mNAND_RYBN3		RnB3	I	-	Ready not Busy 3

3. Register overview

Table 2–5 indicates which registers reside in the NAND flash controller.

Table 5. Register overview: NAND flash controller (register base address: 0x1700 0800)

Name	Access	Offset	Description
NandIRQStatus1	R/W	0x00	Status register of first 32 bits interrupt register
NandIRQMask1	R/W	0x04	Mask register for first 32 bits interrupt register
NandIRQStatusRaw1	R/W	0x08	Unmasked status register of first 32 bits interrupt register
NandConfig	R/W	0x0C	NAND flash controller configuration register
NandIOConfig	R/W	0x10	Register which holds the default value settings for IO signals
NandTiming1	R/W	0x14	First NAND flash controller timing register
NandTiming2	R/W	0x18	Second NAND flash controller timing register

Table 5. Register overview: NAND flash controller (register base address: 0x1700 0800)

Name	Access	Offset	Description
NandSetCmd	R/W	0x20	Register to send specific command towards NAND flash device.
NandSetAddr	R/W	0x24	Register to send specific address towards NAND flash device
NandWriteData	R/W	0x28	Register to send specific data towards NAND flash device
NandSetCE	R/W	0x2C	Register to set all CE signals and WP_n signal
NandReadData	R	0x30	Register to check read data from NAND flash device
NandCheckSTS	R	0x34	Check status of 8 predefined interrupts
NandControlFlow	W	0x38	Register which holds command to read and write pages
NandGPIO1	R/W	0x40	Register to program IO pins, which can be used as GPIO
NandGPIO2	R	0x44	Register to program IO pins, which can be used as GPIO
NandIRQStatus2	R/W	0x48	Status register of second 32 bits interrupt register
NandIRQMask2	R/W	0x4C	Mask register for second 32 bits interrupt register
NandIRQStatusRaw2	R/W	0x50	Unmasked status register of second 32 bits interrupt register
NandECCErrStatus	R	0x78	ECC error status register in 8-symbol ECC mode

4. Register description

4.1 NandIRQStatus1

In this register the status of the different interrupt sources can be checked. All interrupts can be masked by the corresponding bit in the NandIRQMask register. A bit which has been set can only be cleared by writing a '1' to this bit in this register. Table 2–6 gives a description of this register.

Table 6. NandIRQStatus1 register description (NandIRQStatus1, address 0x1700 0800)

Bit	Symbol	Access	Reset value	Description
31	INT31S	R/W	0x0	mNAND_RYBN3 positive edge. Asserted after a positive edge of the mNAND_RYBN3 signal.
30	INT30S	R/W	0x0	mNAND_RYBN2 positive edge. Asserted after a positive edge of the mNAND_RYBN2 signal.
29	INT29S	R/W	0x0	mNAND_RYBN1 positive edge. Asserted after a positive edge of the mNAND_RYBN1 signal.
28	INT28S	R/W	0x0	mNAND_RYBN0 positive edge. Asserted after a positive edge of the mNAND_RYBN0 signal.
27	INT27S	R/W	0x0	RAM 1 erased. Whenever an erased page is read from flash (all 0xFF) this bit is asserted together with read page1 done.
26	INT26S	R/W	0x0	RAM 0 erased. Whenever an erased page is read from flash (all 0xFF) this bit is asserted together with read page0 done.
25	INT25S	R/W	0x0	Write page 1 done. Asserted when SRAM1 contents has been written to the flash.

Table 6. NandIRQStatus1 register description (NandIRQStatus1, address 0x1700 0800)

Bit	Symbol	Access	Reset value	Description
24	INT24S	R/W	0x0	Write page 0 done. Asserted when SRAM0 contents has been written to the flash.
23	INT23S	R/W	0x0	Read page 1 done. Asserted when SRAM1 contents has been read from flash and stored in SRAM1 (not error corrected yet).
22	INT22S	R/W	0x0	Read page 0 done. Asserted when SRAM0 contents has been read from flash and stored in SRAM0 (not error corrected yet).
21	INT21S	R/W	0x0	RAM 0 decoded. Asserted when the contents of SRAM0 has been decoded. Each time bit21 or bit19 are activated, one other bit will be activated too from the group Bit17-4 that indicates how many errors were detected in the current code word.
20	INT20S	R/W	0x0	RAM 0 encoded. Asserted when the contents of SRAM0 has been encoded.
19	INT19S	R/W	0x0	RAM 1 decoded. Asserted when the contents of SRAM1 has been decoded. Each time bit21 or bit19 are activated, one other bit will be activated too from the group Bit17-4 that indicates how many errors were detected in the current code word.
18	INT18S	R/W	0x0	RAM 1 encoded. Asserted when the contents of SRAM1 has been encoded.
17	INT17S	R/W	0x0	RAM 0 decoded with 0 errors
16	INT16S	R/W	0x0	In 5bit ECC mode, this interrupt bit is set when a codeword with one error is detected. In 8bit ECC mode, this interrupt bit is set when a codeword with at least one correctable error is detected. The number of errors can then be extracted from the NandEccErrStatus(0x78) register.
15	INT15S	R/W	0x0	RAM 0 decoded with 2 error
14	INT14S	R/W	0x0	RAM 0 decoded with 3 error
13	INT13S	R/W	0x0	RAM 0 decoded with 4 error
12	INT12S	R/W	0x0	RAM 0 decoded with 5 error
11	INT11S	R/W	0x0	RAM 0 uncorrectable
10	INT10S	R/W	0x0	RAM 1 decoded with 0 errors
9	INT9S	R/W	0x0	In 5bit ECC mode, this interrupt bit is set when a codeword with one error is detected. In 8bit ECC mode, this interrupt bit is set when a codeword with at least one correctable error is detected. The number of errors can then be extracted from the NandEccErrStatus(0x78) register.
8	INT8S	R/W	0x0	RAM 1 decoded with 2 error
7	INT7S	R/W	0x0	RAM 1 decoded with 3 error
6	INT6S	R/W	0x0	RAM 1 decoded with 4 error
5	INT5S	R/W	0x0	RAM 1 decoded with 5 error
4	INT4S	R/W	0x0	RAM 1 uncorrectable
3:0	-	-	-	Reserved

4.2 NandIRQMask1

Each bit in this register field masks the corresponding interrupt bit in the NandIRQStatus register. Table 2–6 gives a description of this register.

Table 7. NandIRQMask1 register description (NandIRQMask1, address 0x1700 0804)

Bit	Symbol	Access	Reset Value	Description
31	INT31M	R/W	0x1	mNAND_RYBN3 positive edge mask
30	INT30M	R/W	0x1	mNAND_RYBN2 positive edge mask
29	INT29M	R/W	0x1	mNAND_RYBN1 positive edge mask
28	INT28M	R/W	0x1	mNAND_RYBN0 positive edge mask
27	INT27M	R/W	0x1	RAM 1 erased mask
26	INT26M	R/W	0x1	RAM 0 erased mask
25	INT25M	R/W	0x1	Write page 1 done mask
24	INT24M	R/W	0x1	Write page 0 done mask
23	INT23M	R/W	0x1	Read page 1 done mask
22	INT22M	R/W	0x1	Read page 0 done mask
21	INT21M	R/W	0x1	RAM 0 decoded mask
20	INT20M	R/W	0x1	RAM 0 encoded mask
19	INT19M	R/W	0x1	RAM 1 decoded mask
18	INT18M	R/W	0x1	RAM 1 encoded mask
17	INT17M	R/W	0x1	RAM 0 decoded with 0 errors mask
16	INT16M	R/W	0x1	RAM 0 decoded with 1 error mask
15	INT15M	R/W	0x1	RAM 0 decoded with 2 error mask
14	INT14M	R/W	0x1	RAM 0 decoded with 3 error mask
13	INT13M	R/W	0x1	RAM 0 decoded with 4 error mask
12	INT12M	R/W	0x1	RAM 0 decoded with 5 error mask
11	INT11M	R/W	0x1	RAM 0 uncorrectable mask
10	INT10M	R/W	0x1	RAM 1 decoded with 0 errors mask
9	INT9M	R/W	0x1	RAM 1 decoded with 1 error mask
8	INT8M	R/W	0x1	RAM 1 decoded with 2 error mask
7	INT7M	R/W	0x1	RAM 1 decoded with 3 error mask
6	INT6M	R/W	0x1	RAM 1 decoded with 4 error mask
5	INT5M	R/W	0x1	RAM 1 decoded with 5 error mask
4	INT4M	R/W	0x1	RAM 1 uncorrectable mask
3:0	-	-	-	Reserved

4.3 NandIRQStatusRaw1

In this register the status of the different interrupt sources can be checked without masking. A bit which has been set can only be cleared by writing a '1' to this bit in this register. Table 2–8 gives a description of this register.

Table 8. NandIRQStatusRaw1 register description (NandIRQStatusRaw1, address 0x1700 0808)

Bit	Symbol	Access	Reset value	Description
31	INT31R	R/W	0x0	mNAND_RYBN3 positive edge raw value
30	INT30R	R/W	0x0	mNAND_RYBN2 positive edge raw value
29	INT29R	R/W	0x0	mNAND_RYBN1 positive edge raw value
28	INT28R	R/W	0x0	mNAND_RYBN0 positive edge raw value
27	INT27R	R/W	0x0	RAM 1 erased raw value
26	INT26R	R/W	0x0	RAM 0 erased raw value
25	INT25R	R/W	0x0	Write page 1 done raw value
24	INT24R	R/W	0x0	Write page 0 done raw value
23	INT23R	R/W	0x0	Read page 1 done raw value
22	INT22R	R/W	0x0	Read page 0 done raw value
21	INT21R	R/W	0x0	RAM 0 decoded raw value
20	INT20R	R/W	0x0	RAM 0 encoded raw value
19	INT19R	R/W	0x0	RAM 1 decoded raw value
18	INT18R	R/W	0x0	RAM 1 encoded raw value
17	INT17R	R/W	0x0	RAM 0 decoded with 0 errors raw value
16	INT16R	R/W	0x0	RAM 0 decoded with 1 error raw value
15	INT15R	R/W	0x0	RAM 0 decoded with 2 error raw value
14	INT14R	R/W	0x0	RAM 0 decoded with 3 error raw value
13	INT13R	R/W	0x0	RAM 0 decoded with 4 error raw value
12	INT12R	R/W	0x0	RAM 0 decoded with 5 error raw value
11	INT11R	R/W	0x0	RAM 0 uncorrectable raw value
10	INT10R	R/W	0x0	RAM 1 decoded with 0 errors raw value
9	INT9R	R/W	0x0	RAM 1 decoded with 1 error raw value
8	INT8R	R/W	0x0	RAM 1 decoded with 2 error raw value
7	INT7R	R/W	0x0	RAM 1 decoded with 3 error raw value
6	INT6R	R/W	0x0	RAM 1 decoded with 4 error raw value
5	INT5R	R/W	0x0	RAM 1 decoded with 5 error raw value
4	INT4R	R/W	0x0	RAM 1 uncorrectable raw value
3:0	-	-	-	Reserved

4.4 NandConfig

This register is used to configure the NAND flash controller. Table 2–9 gives a description of this register.

Table 9. NandConfig register description (NandConfig, address 0x1700 080C)

Bit	Symbol	Access	Reset value	Description
31:16	-	-	-	reserved
15	PEC	R/W	0x0	Power off ECC clock. 1 = ECC clock is off, 0 = ECC clock is on.
14	-	-	-	reserved
13	ECGC	R/W	0x0	Enable ECC clock gating
12	ECC_MODE	R/W	0x0	ECC mode 0: 5 bit ECC mode selected 1: 8 bit ECC mode selected
11:10	TL	R/W	0x0	Transfer limit, determines the number of bytes written/read to the NAND flash in one step. 00/11: 528 bytes 01: 516 bytes 10: 512 bytes
9	-	-	-	reserved
8	DC	R/W	0x1	Deactivate CE enable 0: When the nand_flash is forced of the ebi bus by a backoff signal, the CE is not deactivated 1: When the nand_flash is forced of the ebi bus by a backoff signal, the CE is deactivated
7	M	R/W	0x0	512 mode 0 : The ECC encoding is started automatically after programming byte 512 in the SRAM. To be used when byte 513-516 do not need to be written to the flash (previous data will be written in this field) 1 : The ECC encoding is started automatically after programming byte 516 in the SRAM. To be used when byte 513-516 need to be written to the flash (previous data will be written in this field)
6:5	LC	R/W	0x0	Latency Configuration 0x0 : zero wait state 0x1 : one wait state 0x2 : two wait state
4	ES	R/W	0x0	Endianess setting 0 : little endian 1 : big endian
3	DE	R/W	0x0	DMA external enable 0: disables the automatic flow control with DMA 1: enables the automatic flow control with DMA

Table 9. NandConfig register description (NandConfig, address 0x1700 080C)

Bit	Symbol	Access	Reset value	Description
2	-	-	0x0	reserved
1	WD	R/W	0x0	Wide device 0 : 8 bit NAND device mode 1 : 16 bit NAND device mode
0	EC	R/W	0x0	ECC on 0 : error correction off 1 : error correction on

[1] Software should not change the value of this register, only hardware knows if an EBI module is available.

4.5 NandIOConfig

This register defines the default values of the outputs to the NAND flash device. Default values are put on the outputs when the NAND flash controller is in idle state. Table 2–10 gives a description of this register.

Table 10. NandIOConfig register description (NandIOConfig, address 0x1700 0810)

Bit	Symbol	Access	Reset	Description
31:25	-	-	-	Reserved
24	NI	R/W	0x0	Nand IO drive default 0 : IO pad is in input mode 1 : IO pad is in output mode, data is driven on the pads.
23:8	DN	R/W	0x0	Data to nand default data_to_nand[15:0] value
7:6	CD	R/W	0x0	CLE default “00” : ‘0’ other values : ‘1’
5:4	AD	R/W	0x0	ALE default “00” : ‘0’ other values : ‘1’
3:2	WD	R/W	0x1	WE_n default “00” : ‘0’ other values : ‘1’
1:0	RD	R/W	0x1	RE_n defaul“00” : ‘0’ other values : ‘1’

4.6 NandTiming1

In this register the first set of NAND interface timing characteristics can be programmed. Each timing parameter can be set from 7 nand_clk (NANDFLASH_NAND_CLK) clock cycles to 1 nand_clk clock cycle. (A programmed zero value is treated as a one). Table 2–11 gives a description of this register.

Using tSRD and tDRD the data input circuitry can be tuned for optimal performance. Using the lower bit of these parameters one can select between clocking in on the positive edge of nand_clk or on the negative edge. The remaining bit(s) add extra nand_clk delay cycles to the data clock-in moment.

Table 11. NandTiming1 register description (NandTiming1, address 0x1700 0814)

Bit	Symbol	Access	Reset value	Description
31:22	-	-	-	Reserved
21:30	TSRD	R/W	0x0	Single data input delay The number of clock cycles between the rising edge of the RE signal and the cycle that the data is clocked in by the controller in case of software controlled single read access
19	-	-	-	Reserved
18:16	TALS	R/W	0x0	Address setup time The number of clock cycles between the rising edge of ALE and the falling edge of WE during a command transfer
15	-	-	-	Reserved
14:12	TALH	R/W	0x0	Address hold time The number of clock cycles that ALE remains asserted after the rising edge of WE
11:7	-	-	-	Reserved
6:4	TCLS	R/W	0x0	Command setup time The number of clock cycles between the rising edge of CLE and the falling edge of WE during a command transfer
3	-	-	-	Reserved
2:0	TCLH	R/W	0x0	Command hold time The number of clock cycles that CLE remains asserted after the rising edge of WE

4.7 NandTiming 2

In this register the second set of NAND interface timing characteristics can be programmed. Each timing parameter can be set from 7 nand_clk clock cycles to 1nand_clk clock cycle. (A programmed zero value is treated as a one). Table 2–12 gives a description of this register.

Table 12. NandTiming2 register description (NandTiming2, address 0x1700 0818)

Bit	Symbol	Access	Reset value	Description
31	-	-	-	Reserved
20:28	TDRD	R/W	0x0	Data input delay The number of clock cycles between the rising edge of the RE signal and the cycle that the data is clocked in by the controller in case of hardware controlled burst read access
27	-	-	-	Reserved

Table 12. NandTiming2 register description (NandTiming2, address 0x1700 0818)

Bit	Symbol	Access	Reset value	Description
26:24	TEBIDEL	R/W	0x0	EBI delay time The number of clock cycles between the rising edge of CS and the falling edge of ebireq when backing off from the EBI. OR The number of clock cycles between the rising edge of ebignt and the falling edge of CS when going on the EBI.
23	-	-	-	Reserved
22:20	TCH	R/W	0x0	Chip select hold time The number of clock cycles between the last active signal to the NAND flash and the rising edge of CS
19	-	-	-	Reserved
18:16	TCS	R/W	0x0	Chip select setup time The number of clock cycles between the falling edge of CS and the first active signal to the NAND flash
15	-	-	-	Reserved
14:12	TREH	R/W	0x0	Read enable high hold The minimum number of clock cycles that the RE pulse is held
11	-	-	-	Reserved
10:8	TRP	R/W	0x0	Read enable pulse width The number of clock cycles that the RE pulse is de-asserted
7	-	-	-	Reserved
6:4	TWH	R/W	0x0	Write enable high hold The minimum number of clock cycles that the WE pulse is held high before a next falling edge
3	-	-	-	Reserved
2:0	TWP	R/W	0x0	Write enable pulse width The number of clock cycles that the WE pulse is de-asserted. This value also covers the tDS, (data setup time) since the data is set up on the I/O line at the same moment as the falling edge of the WE pulse

4.8 NandSetCmd

This register is used to transfer a command towards the NAND flash device. Table 2–13 gives a description of this register.

Table 13. NandSetCmd register description (NandSetCmd, address 0x1700 0820)

Bit	Symbol	Access	Reset	Description
31:16	-	-	-	Reserved
15:0	CV	W	0x0	Command value Writing to this register results in a CLE-WE combined sequence that transfers the programmed command value to the NAND flash using the required timings.

4.9 NandSetAddr

This register is used to transfer an address towards the NAND flash device. Table 2–14 gives a description of this register.

Table 14. NandSetAddr register description (NandSetAddr, address 0x1700 0824)

Bit	Symbol	Access	Reset	Description
31:16	-	-	-	Reserved
15:0	AV	W	0x0	Address valueWriting to this register results in a ALE-WE combined sequence that transfers the programmed address value to the NAND flash using the required timings..

4.10 NandWriteData

This register is used to write data towards the NAND flash device. Table 2–15 gives a description of this register.

Table 15. NandWriteData register description (NandWriteData, address 0x1700 0828)

Bit	Symbol	Access	Reset	Description
31:16	-	-	-	Reserved
15:0	WV	W	0x0	Writing a value WV to this register results in a WE sequence that transfers the programmed write value to the NAND flash using the required timings.

4.11 NandSetCE

This register is used to set the values of WP_n and NAND_NCS_0 to NAND_NCS_3. Table 2–16 gives a description of this register.

Table 16. NandSetCE register description (NandSetCE, address 0x1700 082C)

Bit	Symbol	Access	Reset	Description
31:5	-	-	-	Reserved
4	WP	W	0x0	WP_n pin value Sets WP_n pin value
3:0	CV	W	0x0	Chip select value One bit per chip select output. The chip select is only made active when the controller is not in idle state and the ebi bus is granted to the nand controller.

4.12 NandReadData

This register is used to read data from the NAND flash device. Table 2–17 gives a description of this register.

Table 17. NandReadData register description (NandReadData, address 0x1700 0830)

Bit	Symbol	Access	Reset	Description
31:16	-	-	-	Reserved
15:0	RV	W	0x0	Read value Reading this register results in a RE sequence that after the necessary wait states puts the retrieved value from the NAND IO port into the register..

4.13 NandCheckSTS

This register is used to read out the status of the NAND flash controller, w.r.t. the values on the incoming RnB signals. Next to that the busy state of the APB can be checked. Table 2–18 gives a description of this register.

Table 18. NandCheckSTS register description (NandCheckSTS, address 0x1700 0834)

Bit	Symbol	Access	Reset	Description
31:9	-	-	-	Reserved
8	R3R	R	0x0	mNAND_RYBN3 rising edge. 1: Rising edge on the mNAND_RYBN3 signal has been detected. Bit is reset to 0 upon read.
7	R2R	R	0x0	mNAND_RYBN2 rising edge. 1: Rising edge on the mNAND_RYBN2 signal has been detected. Bit is reset to 0 upon read.
6	R1R	R	0x0	mNAND_RYBN1 rising edge. 1: Rising edge on the mNAND_RYBN1 signal has been detected. Bit is reset to 0 upon read.
5	R0R	R	0x0	mNAND_RYBN0 rising edge. 1: Rising edge on the mNAND_RYBN0 signal has been detected. Bit is reset to 0 upon read.
4	R3	R	0x0	mNAND_RYBN3 value. The sample value of the mNAND_RYBN3 signal from the flash.
3	R2	R	0x0	mNAND_RYBN2 value. The sample value of the mNAND_RYBN2 signal from the flash.

Table 18. NandCheckSTS register description (NandCheckSTS, address 0x1700 0834)

Bit	Symbol	Access	Reset	Description
2	R1	R	0x0	mNAND_RYBN1 value. The sample value of the mNAND_RYBN1 signal from the flash.
1	R0	R	0x0	mNAND_RYBN0 value. The sample value of the mNAND_RYBN0 signal from the flash.
0	VB	R	0x0	APB busy 1: flash access over the APB bus is busy 0: no flash access over APB bus at this moment

4.14 NandControlFlow

This register is used to start the sequences for read page and write page operation.Table 2–19 gives a description of this register.

Table 19. NandControlFlow register description (NandControlFlow, address 0x1700 0838)

Bit	Symbol	Access	Reset	Description
31:6	-	-	-	Reserved
5	W1	W	0x0	Writing a ‘1’ to this property starts up the sequence to write the contents of SRAM1 to the NAND flash (if the contents has already been protected by the necessary parity symbols)
4	W0	W	0x0	Writing a ‘1’ to this property starts up the sequence to write the contents of SRAM0 to the NAND flash (if the contents has already been protected by the necessary parity symbols)
3:2	-	-	-	Reserved
1	R1	W	0x0	Writing a ‘1’ to this property starts up the sequence to read a defined number of bytes from the NAND flash and store them in SRAM1
0	R0	W	0x0	Writing a ‘1’ to this property starts up the sequence to read a defined number of bytes from the NAND flash and store them in SRAM0

4.15 NandGPIO1

This register is used to program the IO pins in GPIO mode. Table 2–20 gives a description of this register.

Table 20. NandGPIO1 register description (NandGPIO1, address 0x1700 0840)

Bit	Symbol	Access	Reset	Description
31:27	-	-	-	Reserved
26	nand_gpio_conf	R/W	0x0	'0' : the module is in normal functional mode '1' : GPIO mode, the value of the outputs to the NAND flash can be controlled via NAND_GPIO1.
25	WP_n	R/W	0x0	Program value on WP_n
24	CLE	R/W	0x0	Program value on CLE
23	ALE	R/W	0x0	Program value on ALE
22	RE_n	R/W	0x1	Program value on RE_n
21	WE_n	R/W	0x1	Program value on WE_n
20	CE4_n	R/W	0x1	Program value on NAND_NCS_3
19	CE3_n	R/W	0x1	Program value on NAND_NCS_2
18	CE2_n	R/W	0x1	Program value on NAND_NCS_1
17	CE1_n	R/W	0x1	Program value on NAND_NCS_0
16	Nand io drive	R/W	0x0	Program value on Nand io drive
15:0	Data to nand IO	R/W	0x0	Program value on data to Nand IO

4.16 NandGPIO2

In this register the value of the input signals from NAND can be monitored on read-out. Table 2–21 gives a description of this register

Table 21. NandGPIO2 register description (NandGPIO2, address 0x1700 0844)

Bit	Symbol	Access	Reset	Description
31:20	-	-	-	Reserved
19	RnB3	R	0x0	Read value from mNAND_RYBN3
18	RnB2	R	0x0	Read value from mNAND_RYBN2
17	RnB1	R	0x0	Read value from mNAND_RYBN1
16	RnB0	R	0x0	Read value from mNAND_RYBN0
15:0	Data from NAND	R	0x0	Read data from NAND IO

4.17 NandIRQStatus2

In this register the status of the different interrupt sources can be checked. All interrupts can be masked by the corresponding bit in the NandIRQMask2 register. A bit which has been set can only be cleared by writing a '1' to this bit in this register. Table 2–22 gives a description of this register.

Table 22. NandIRQStatus2 register description (NandIRQStatus2, address 0x1700 0848)

Bit	Symbol	Access	Reset	Description
31:5	-	-	-	Reserved
4	INT36S	R/W	0x0	Page access while APB access.
3	INT35S	R/W	0x0	APB access while page access.

Table 22. NandIRQStatus2 register description (NandIRQStatus2, address 0x1700 0848)

Bit	Symbol	Access	Reset	Description
2	INT34S	R/W	0x0	Flash access while busy.
1	INT33S	R/W	0x0	RAM1 access while busy.
0	INT32S	R/W	0x0	RAM0 access while busy.

4.18 NandIRQMask2

Each bit in this register field masks the corresponding interrupt bit in the NandIRQStatus2 register. Table 2–23 gives a description of this register.

Table 23. NandIRQMask2 register description (NandIRQMask2, address 0x1700 084C)

Bit	Symbol	Access	Reset	Description
31:5	-	-	-	Reserved
4	INT36M	R/W	0x1	Page access while APB access masks
3	INT35M	R/W	0x1	APB access while page access mask
2	INT34M	R/W	0x1	Flash access while busy mask
1	INT33M	R/W	0x1	RAM1 access while busy mask
0	INT32M	R/W	0x1	RAM0 access while busy mask

4.19 NandIRQStatusRaw2

In this register the status of the different interrupt sources can be checked without masking. A bit which has been set can only be cleared by writing a '1' to this bit in this register. Table 2–24 gives a description of this register.

Table 24. NandIRQStatusRaw2 register description (NandIRQStatusRaw2, address 0x1700 0850)

Bit	Symbol	Access	Reset	Description
31:5	-	-	-	Reserved
4	INT36R	R/W	0x0	Page access while APB access raw value.
3	INT35R	R/W	0x0	APB access while page access raw value.
2	INT34R	R/W	0x0	Flash access while busy raw value.
1	INT33R	R/W	0x0	RAM1 access while busy raw value.
0	INT32R	R/W	0x0	RAM0 access while busy raw value.

4.20 NandECCErrStatus

This register is used to report error statistics of code words in 8 bit ECC mode. If at least one correctable error is detected in 8 bit ECC mode, the “RAMx decoded with one error” bit from register NandIRQ_STATUS1 is set. If this bit is set, the ARM can read out the NandECCErrStatus register to know exactly how many errors were detected. The register is updated whenever a codeword with more than one correctable error is detected.

Table 25. NandECCErrStatus register description (NandECCErrStatus, address 0x1700 0878)

Bit	Symbol	Access	Reset	Description
31:8	-	-	-	Reserved
7:4	N_ERR_1	R	0000	Number of errors in RAM1
3:0	N_ERR_0	R	0000	Number of errors in RAM0

5. Functional description

In Figure 2–2 the architecture of the NAND flash controller is displayed. The access to the AHB bus is done via the NAND-AHB interface module which resides inside of the NAND flash controller module. Two 528 bytes (132-words x 32-bits) SRAMs which are placed inside of the NAND flash controller module, are connected to the internal NAND controller in parallel and the access to these SRAMs is shared with the control module. All data path modules (codec, error corrector, syndrome generator, parity generator, nand interface) are controlled by the main control module. The configuration registers are kept in a separate sub-module which is connected to the APB interface. These registers run on the NANDFLASH_PCLK. In write mode the data is retrieved out of the SRAM by the NAND flash controller and written to the NAND flash device after being protected with parity symbols. In read mode the data is read from the NAND flash device and temporarily stored in one of the SRAMs to have it corrected by the error corrector. When these operations are done, the data can be randomly accessed from the SRAMs over the AHB bus using zero wait-states AHB access. The AHB bus is only burdened with data transfers for a very limited time. (the time to upload or download the contents from the SRAM using zero wait states). In decode mode, once the command, address and data have been sent, everything is taken over by the NAND flash controller and the AHB bus is free. In encode mode, the command and data are sent and the AHB bus is again freed up until the moment that the data is available in the SRAM.

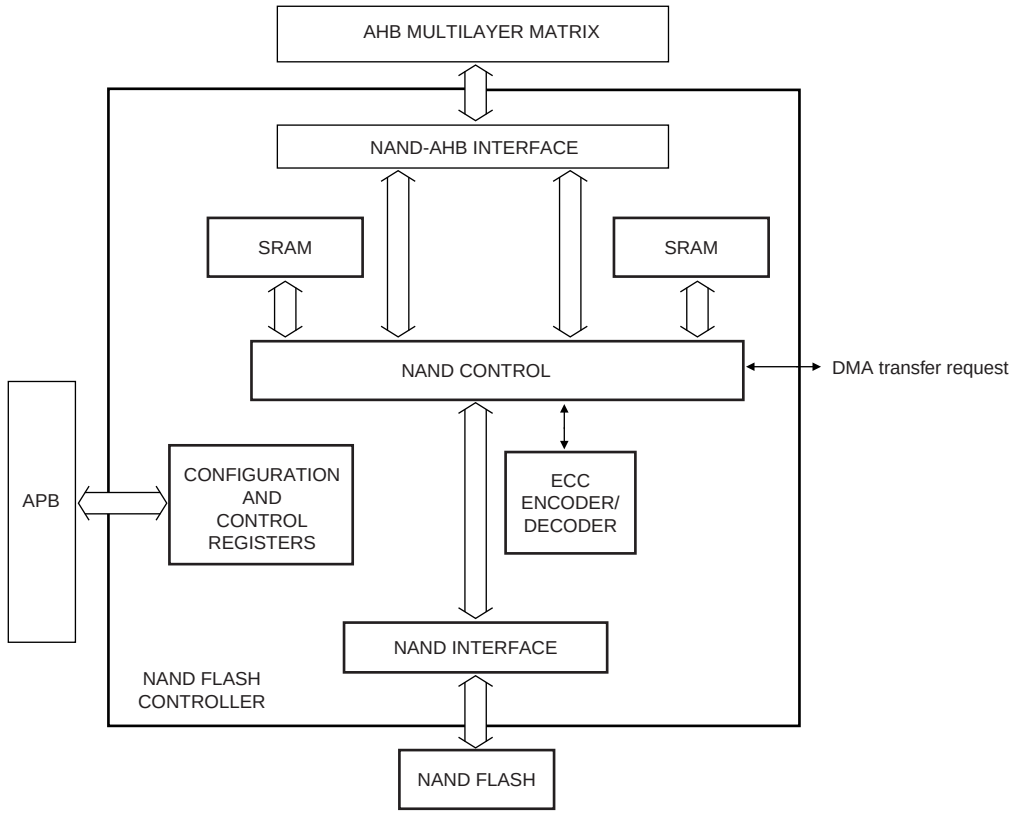


Fig 2. NAND flash controller internal architecture

5.1 NAND timing diagrams

Table 2–26 shows the timing diagram for the timing parameters in registers NandFlashTiming1 and NandFlashTiming2.

Table 26. NAND flash timing parameters

Symbol	Parameter	Description
t_{WP}	$\overline{WE}$ pulse width	This value also covers the t_{DS}, (data setup time) since the data is set up on the I/O line at the same moment as the falling edge of the WE pulse.
t_{WH}	$\overline{WE}$ HIGH hold time	The minimum number of clock cycles that the WE pulse is held high before a next falling edge.
t_{RP}	$\overline{RE}$ pulse width	The number of clock cycles that the RE pulse is de-asserted.
t_{REH}	$\overline{RE}$ HIGH hold time	The minimum number of clock cycles that the RE pulse is held high before a next falling edge.
t_{CLH}	CLE hold time	The number of clock cycles that CLE remains asserted after the rising edge of WE.
t_{CLS}	CLE set-up time	The number of clock cycles between the rising edge of CLE and the falling edge of WE during a command transfer.
t_{ALH}	ALE hold time	The number of clock cycles that ALE remains asserted after the rising edge of WE.

Table 26. NAND flash timing parameters

Symbol	Parameter	Description
t_{ALS}	ALS set-up time	The number of clock cycles between the rising edge of ALE and the falling edge of WE during a command transfer.
t_{CS}	$\overline{CE}$ set-up time	The number of clock cycles between the falling edge of CS and the first active signal to the nand flash.
t_{CH}	$\overline{CE}$ hold time	The number of clock cycles between the last active signal to the nand flash and the rising edge of CS.
t_{DRD}	data input delay time	The number of clock cycles between the rising edge of the RE signal and the cycle that the data is clocked in by the controller in case of hardware controlled burst read access.
t_{SRD}	single data input delay time	The number of clock cycles between the rising edge of the RE signal and the cycle that the data is clocked in by the controller in case of software controlled single read access.
t_{EBIDEL}	EBI delay time	The number of clock cycles between the rising edge of CS and the falling edge of ebireq (request) when backing off from the EBI. OR The number of clock cycles between the rising edge of ebignt (grant) and the falling edge of CS when going on the EBI.

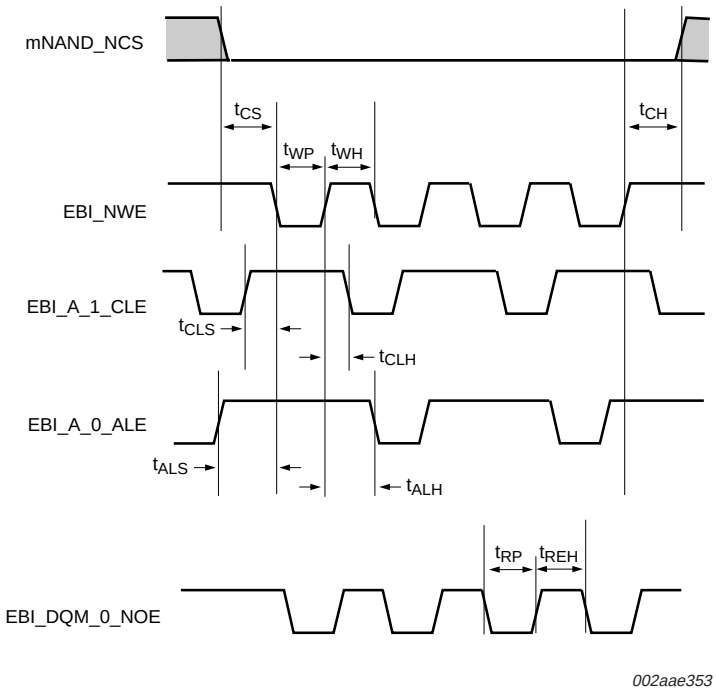


Fig 3. NAND flash WE, RE, CLE, ALE, CE timing

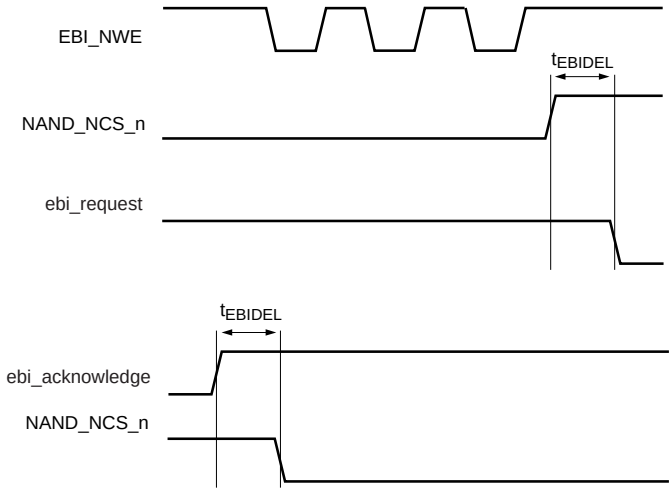


Fig 4. NAND flash EBI request/acknowledge timing

5.2 Error correction

5.2.1 Reed-Solomon code definition

The error correction code used is Reed-Solomon over GF(2^9). The primitive polynomial g(x) over GF(2) is:

$g(x) = x^9 + x^4 + 1$

The code is a Reed-Solomon code of length 469, dimension 459, and minimum distance 11. In each codeword the 10 parity symbols are defined by the remainder polynomial R(x) to form the code RS(469,459,11).

$R(x) = M(x) \times x^{10} \bmod P(x)$

where M(x) is the information and P(x) the generator polynomial for the RS code:

(1)

$$M(x) = \sum_{j=0}^{458} B_j x^j$$

(2)

$$P(x) = \prod_{k=0}^{9} (x + \alpha^k)$$

and a has the hexadecimal 9-bit representation 0x002. a is a root of the primitive polynomial $g(x) = x^9 + x^4 + 1$.

5.2.2 Mapping of the code onto flash pages

A flash page consists of 512 bytes + 16 redundant bytes or a multiple of this. Currently 2048 byte pages + 64 redundant bytes are widely used. The concept is to subdivide every page into groups of 512 information bytes and 16 redundant bytes.

A 2K flash will be subdivided as shown in

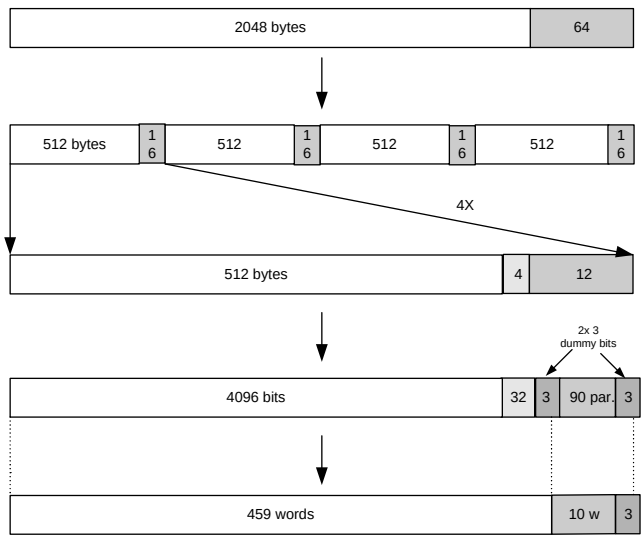


Fig 5. Structure from flash page to ECC code words

The 16 redundant bytes are subdivided into:

- 4 bytes free for purposes like wear-leveling, building tables. (an ECC layer can also be applied also over these bytes).
- 12 remaining bytes that consist of 10 parity symbols (90 bits) + 6 dummy bits.

In the end the 459 data words consist of

- 512 data bytes
- 4 extra bytes
- 3 dummy bytes at the end

The 10 remaining parity words consist of

- 10 parity symbols
- 3 dummy bits at the end

5.2.3 Error correction flow implementation

The error correction flow starting from a codeword C(x) is shown in Figure 2–6 and follows these steps:

1. Calculate syndromes out of the received codeword.
2. Solve key equation via the Euclidean algorithm.

3. The result of this is the error locator polynomial Ë(x) and the error evaluator polynomial L(x).
4. Search for zeros of error locator polynomial using the chien search & Forney algorithm.
5. Evaluate Ω(x) at zeros of L(x).
6. Send out error locations and values.

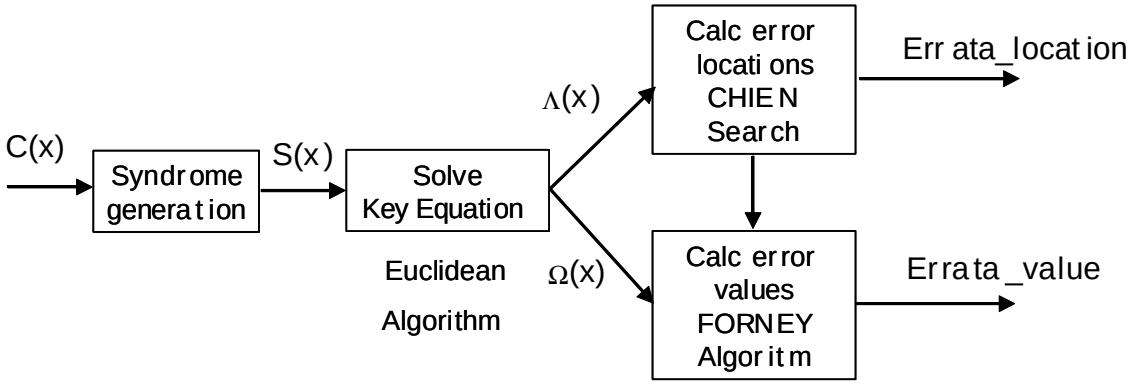


Fig 6. Reed-Solomon error correction

5.3 EBI operation

To support pin sharing with other memory controllers, NAND flash controller accesses the nand flash through the EBI module. For every access to the nand flash, the NAND flash controller will first request access through the EBI before initiating the access. When the access is done, NAND flash withdraws itself from the EBI bus.

Short access can not be interrupted via the ebibackoff signal. These accesses are:

- single byte read
- single byte write
- command write
- address write

A burst data access can be interrupted by the ebibackoff signal. This is done by going off the EBI bus after first deactivating the chip select signal. To be able to use this function the NAND flash needs to be a “CEn don’t care” device as the chip select signal to nand flash will be deactivated before going off the EBI bus. When the EBI bus is free again, chip select is again activated and the burst data access is continued.

Currently, the majority of nand flash devices support “CEn don’t care”. With “CEn don’t care”, it is possible to interrupt a sequential read/write by de-asserting CEn high. When CEn is high, the NAND flash device will ignore the values on the NAND flash control signals. This makes it possible to interrupt a sequential read/write, and pin-share the nand flash control signals with control signals of other memory controllers. To resume the sequential read/write, CEn is asserted low again.

The NAND flash controller also has an option to disable this CEn deactivation during the ebi back off procedure.

6. Power optimization

Several mechanisms can be used to save power in the NAND flash controller.

- Internal clock gating is inserted during synthesis
- The presence of variable clock scaling will switch the clocks to a lower frequency
- Software is able to enable or disable every clock to save power when certain parts are not used
  - Software is able to switch clocks in the CGU module, which is the source of the NAND flash controller clocks.

7. Programming guide

The NAND flash controller can be controlled fully by software, or partly by software and partly by hardware. Both options are described in the next paragraphs.

7.1 Software controlled access

The software has basic control over the NAND flash device by accessing registers in the NAND flash controller over the APB bus. The NAND flash controller will then make sure that the IO signals react in the corresponding way. This is implemented in the form of a number of independent actions. These are summed up in Table 2–27

Table 27. NAND flash controller software control

Command	Resulting action
Write NandSetCmd register for NAND flash controller command.	Hardware pulls
	CS down
	CLE up & puts command value on the I/O lines
	WE down
	WE up
	CLE down
	CE up

Table 27. NAND flash controller software control

Command	Resulting action
Write NandSetAddr register for NAND flash controller address.	Hardware pulls CS down ALE up & puts command value on the I/O lines WE down WE up ALE down CE up
Write NandWriteData register for NAND flash controller write data.	Hardware pulls CS down WE down & puts data on the I/O lines WE up CS up
Read NandReadData register for NAND flash controller read data.	Hardware pulls CS down RE down RE up & clocks in data from I/O lines CS up

7.2 Hardware controlled access

In this mode the hardware directly performs read and write operations using RE_n/WE_n pulses to the NAND flash device. The ARM processor makes sure that the necessary commands and addresses are supplied to the NAND flash device. To do this it programs registers over the APB bus in the NAND flash controller. A read or write burst access is initiated by the NAND flash controller after receiving a command from the CPU. The NAND flash controller will always read data in chunks of 528 bytes per read page command regardless of the fact that the error corrector is turned on or off. For writing the same is valid.

7.3 Writing small page NAND flash devices

The following steps are performed when writing a page into a NAND flash device with 512byte large pages. Figure 2–7 illustrates this flow.

1. ARM or DMA writes the 512 or 516 bytes of target data into the SRAM. This triggers the ECC to start generating parity symbols. The RS encoding action is automatically started when the controller detects that byte 512 or byte 516 (can be configured inside the NAND flash controller) is written to one of the SRAMs.
2. ARM sends the data[1] for the CE,CLE and ALE sequences to the NAND flash controller.
3. ARM sends write_page command, via register NandControlFlow. This will trigger the NAND flash controller to write the contents of the SRAM to the NAND flash device as soon as possible (after filling in the parity bytes).
4. When done the NAND flash controller triggers an interrupt.

1.This is writing the command and address values to the registers in the NAND flash controller, this automatically initiates proper CLE/ALE sequence to the NAND flash device.

5. ARM writes secondary write command to command register.
6. NAND flash controller polls the busy signal, when it goes high an interrupt is triggered.
7. ARM can read the status information via command register and an RE pulse.

Note 1: This is writing the command and address values to the registers in the NAND flash controller, this automatically initiates proper CLE/ALE sequence to the NAND flash device.

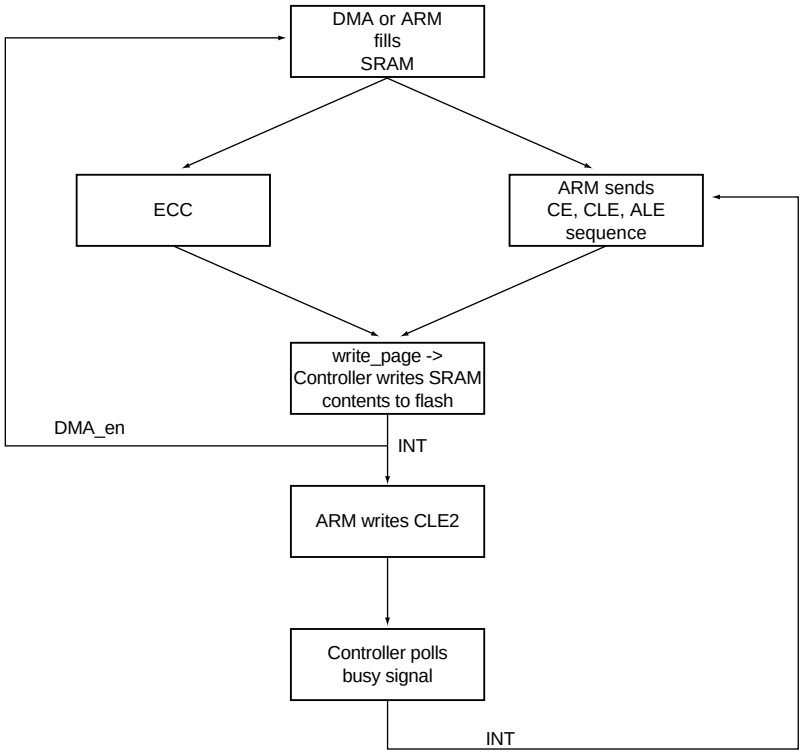


Fig 7. Encode flow of events for 0.5 kByte page NAND flash devices

7.3.1 Writing large page NAND flash devices

In the case of a NAND flash device with pages larger that 0.5kB (2kB for example), the ARM only needs to send new commands and addresses every fourth time.

7.3.2 Read small page NAND flash devices

The following steps are performed when reading a 528 byte group from the NAND flash device.

1. ARM sends the sequence and data[1] for the CE,CLE and ALE pulses
2. When the NAND flash device is ready the NAND flash controller starts reading the data from the NAND flash device using RE pulses
3. When the SRAM has been filled, the error correction is started up on the code word automatically. At the same time, the NAND flash controller triggers an interrupt to let the ARM know that it can start a new read operation

4. After ECC operations have finished, the previous decoded data can be read from the SRAM.

Note 1: This is writing the command and address values to the registers in the NAND flash controller, this automatically initiates proper CLE/ALE sequence to the NAND flash device.

7.3.3 Read large page NAND flash devices

As explained earlier for encode mode the ARM only needs to send new commands and addresses every fourth time in the case of a NAND flash device with 2kB large pages.

1. Introduction

The multi-port memory controller supports the interface to a large number of memory types, such as SDRAM, Low-power (LP) SDRAM, flash, Synchronous Micron flash and ROM.

1.1 Feature list

- AMBA 32-bit AHB compliancy.
- Dynamic-memory interface support including SDRAM, JEDEC low-power SDRAM and Micron SyncFlash.
- Asynchronous static memory device support, including RAM, ROM and flash with or without asynchronous page mode.
- Low transaction latency.
- Read and write buffers to reduce latency and to improve performance, particularly for un-cached processors.
- Two AHB-interfaces:
  - one interface for accessing external memory.
  - one separate control interface to program the MPMC. This enables the MPMC registers to be situated in memory with other system peripheral registers.
- 8-bit and 16-bit wide static memory support.
- 16-bit wide chip select SDRAM memory support.
- 16-bit wide chip select Micron SyncFlash memory support.
- Static memory features include:
  - Asynchronous page mode read
  - Programmable wait states
  - Bus turnaround delay
  - Output enable and write enable delays
  - Extended wait
- One chip select for synchronous memory devices and two chip selects for static memory devices.
- Software controllable HCLK to MPMCCLKOUT ratio.
- Power-saving modes control dynamically SDRAM clock enable EBI_CKE (pin mLCD_E_RD) and EBI_CLKOUT (pin mLCD_DB_0).
- Dynamic-memory self-refresh mode supported by either a Power Management Unit (PMU) interface or by software.
- Controller supports 2K, 4K and 8K row address synchronous-memory parts. That is typical 512 Mbit, 256 Mbit, 128 Mbit and 16 Mbit parts, with either 8 DQ bits or 16 DQ bits per device.

- Two reset domains enable dynamic-memory contents to be preserved over a soft reset.
- Locked AHB-transactions supported.
- Support for all AHB burst types.
- Little-endian and big-endian support.
- Support for the External Bus Interface (EBI) that enables the memory controller pads to be shared.

2. General description

2.1 Interface diagram

Figure 3–8 shows the interface diagram of the MPMC module with all connected modules in this IC. The bus-width on the pads reflects the number of bits that are used in this IC. This is because only 16 address and data lines are used. In addition, only one dynamic device and two static devices are supported.

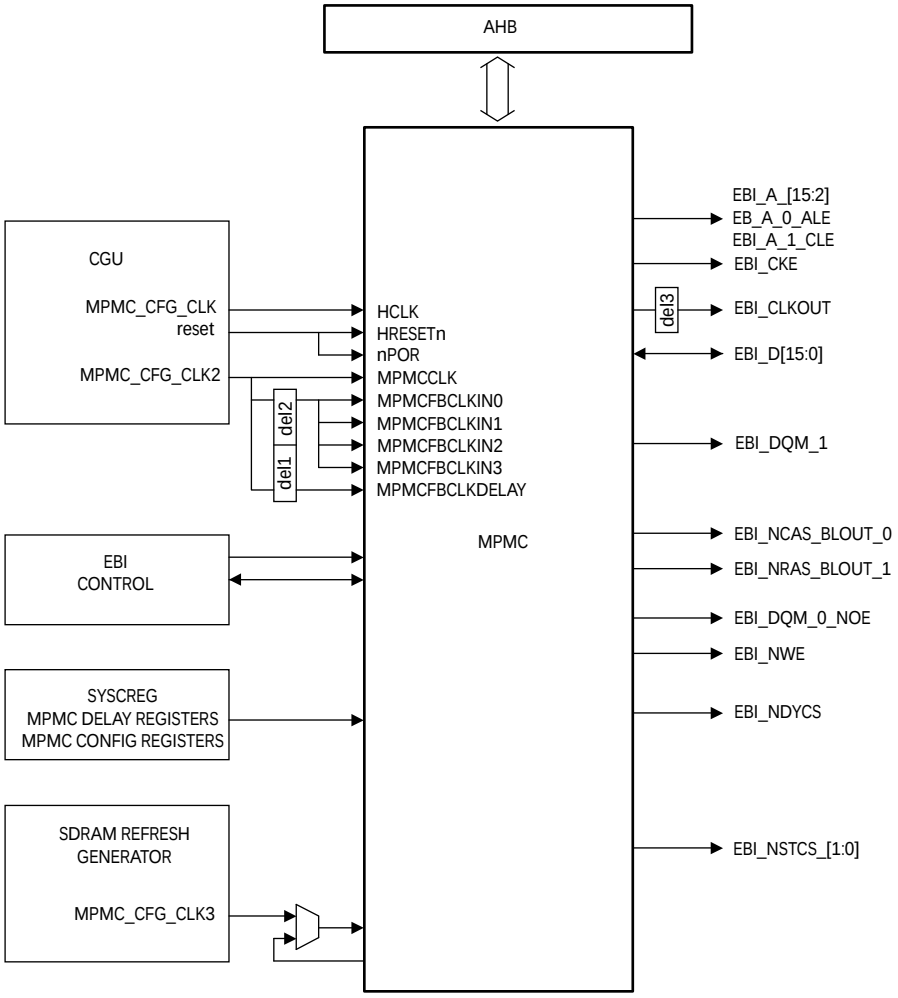


Fig 8. MPMC module interface diagram

2.2 Interface description

2.2.1 Clock signals

Table 3–28 shows an overview of all clocks that are connected to the MPMC module.

Table 28. MPMC module clock overview

Internal MPMC Clock Name	Clock name	I/O	Source / Destination	Description
HCLK	MPMC_CFG_CLK	I	CGU	Main AHB bus clock
MPMCCLK	MPMC_CFG_CLK_2	I	CGU	Clock for timing all external memory transfers. Should be synchronous to HCLK, where MPMCCLK can be twice the frequency of HCLK
MPMCCLKOUT	EBI_CLKOUT	O	MPMC	Clock towards SDRAM devices. Follows MPMCCLK

Table 28. MPMC module clock overview

Internal MPMC Clock Name	Clock name	I/O	Source / Destination	Description
CLK	MPMC_CFG_CLK3	I	CGU	clock used to generate the refresh pulses towards SDRAM - not influenced by variable clock scaling.
Feedback clocks to re-synchronize SDRAM read data from the off-chip to on-chip domains.				
MPMCFBCLKIN0	Delayed clock from MPMC_CFG_CLK_2 (see Section 26–4.6.2)	I	CGU	Feedback clock 0
MPMCFBCLKIN1		I	CGU	Feedback clock 1
MPMCFBCLKIN2		I	CGU	Feedback clock 2
MPMCFBCLKIN3		I	CGU	Feedback clock 3
MPMCCLKDELAY		I	CGU	Delayed version of MPMCCLK, used in command delayed mode

Several clocks are connected to the MPMC module. Figure 3–9 gives an overview of all connections.

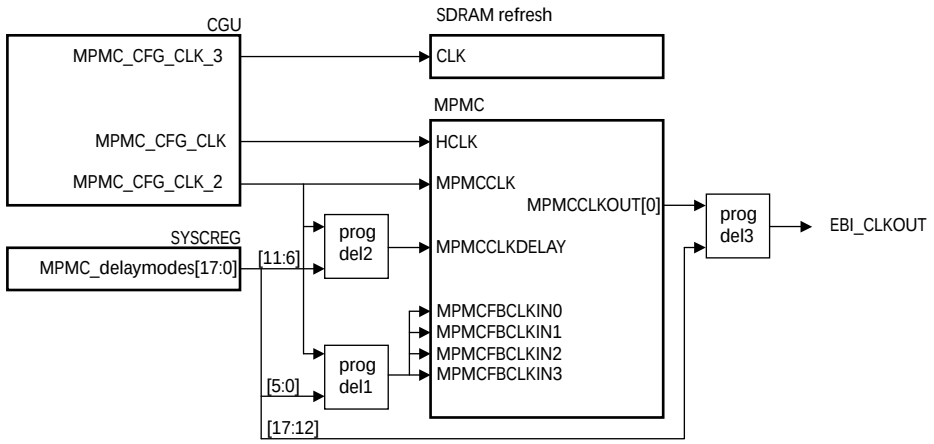


Fig 9. MPMC module clock connection overview

In total 3 delay lines are available, which are described below:

- MPMCCLKDELAY: The amount of delay for MPMCCLKDELAY w.r.t. HCLK can be programmed with register MPMC_delaymodes bits [11:6]. All outgoing signals (data, address and commands) will be delayed with respect to MPMCCLKOUT
- MPMCCLKOUT: The amount of delay for MPMCCLKOUT w.r.t. MPMCCLK can be programmed with register MPMC_delaymodes bits [17:12]. MPMC_CLKOUT/EBI_CLKOUT can get an extra delay w.r.t. outgoing data, address and commands
- MPMCFBCLKIN3..0: The amount of delay for MPMCFBCLKIN3..0, w.r.t. MPMCCLK can be programmed with register MPMC_delaymodes bits [5:0]. This delay is used to fine-tune the register moment of data that is read from external memory.

Register MPMC_delaymodes[17:0] resides in the SYSCREG module (see Table 26–537).

The MPMC_CFG_CLK_3 is a clock that is not influenced by variable clock scaling and used to generate the refresh pulses towards SDRAM.

2.2.2 Reset signals

Table 3–29 shows an overview of all resets that are connected to the MPMC module.

Table 29. MPMC module reset overview

Name	Type	Description
HRESETn	I	Active low reset for this module
nPOR	I	Active low power on reset for this module

2.2.3 External pin connections

Table 3–30 shows all external pin connection signals towards and from the MPMC module.

Table 30. MPMC module external signals

MPMC module name	Pin name/ function	Interface	Type	Description
MPMCADDROUT[15:0]	mLCD_DB_[15:2]/ EBI_A_[15:2][1] and EBI_A_O_ALE, EBI_A_1_CLE	EBI	O	Address output. Used for both static and SDRAM devices.
MPMCCKEOUT0	mLCD_E_RD/ EBI_CKE[1]	EBI	O	SDRAM clock enables. Used for SDRAM devices.
MPMCCLKOUT0	mLCD_DB_0/ EBI_CLKOUT[1]	EBI	O	SDRAM clocks. Used for SDRAM devices.
MPMCDATAIN[15:0]	EBI_D[15:0]	EBI	I	Read data from memory. Used for both static memory and dynamic memory devices.
MPMCDATAOUT[15:0]	EBI_D[15:0]	EBI	O	Data output to memory. Used for both static memory and dynamic memory devices.
MPMCDQMOUT1	mLCD_RW_WR/ EBI_DQM_1[1]	EBI	O	Data mask output to SDRAMs. Used for SDRAM devices.
MPMCDQMOUT0[2]/ nMPMCOEOUT[3]	EBI_DQM_0_NOE	EBI	O	For static memory devices this is data mask output (MPMCOEOUT). And for SDRAM devices this is MPMCDQMOUT[0].
nMPMCBLSOUT0/ nMPMCCASOUT	EBI_NCAS_BLOUT_0	EBI	O	Byte lane 0 select (active low) for Static memories(nMPMCCLSOUT0). Same signal acts as column strobe for SDRAM devices (nMPMCCASOUT)
nMPMCBLSOUT1/ nMPMCRASOUT	EBI_NRAS_BLOUT_1	EBI	O	Byte lane 1 select (active low) for Static memories(nMPMCCLSOUT1). Same signal acts as row strobe for SDRAM devices (nMPMCRASOUT).
nMPMCDYCSOUT0	mLCD_RS/EBI_NDYCS[1]	EBI	O	SDRAM chip selects. Used for SDRAM devices.
nMPMCSTSCOUT[1:0]	mLCD_CSB/EBI_NSTCS_0[1] and mLCD_DB_1/ EBI_NSTCS_1[1]	EBI	O	Static memory chip selects. Default active low. Used for static memory devices.
nMPMCWEOUT	EBI_NWE	EBI	O	Write enable. Used for SDRAM and static memories.

[1] The EBI address and control pins are multiplexed with the LCD data and control pins (see Section 26–4.8).

[2] For SDRAM devices.

[3] For static memory devices.

2.3 Functional description

The multi-port memory controller block optimizes and controls external memory transactions. The functions of the MPMC blocks are described in this chapter.

2.3.1 AHB slave register interface

The AHB slave register interface block enables the registers of the MPMC to be programmed. This module also contains most of the registers and performs the majority of the register address decoding.

2.3.2 Memory transaction endianness and transfer width towards registers

To eliminate the possibility of endianness problems, all data transfers to and from the registers of the MPMC must be 32-bits wide. when an access is attempted with a size other than a word (32-bits), it causes an ERROR response on HRESP and the transfer is terminated.

2.3.3 AHB slave memory interfaces

The AHB slave memory interfaces enable devices to access the 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 MPMC, without data having to pass over the main system bus. All AHB burst types are supported, enabling the most efficient use of memory bandwidth. The AHB interfaces do not generate SPLIT and RETRY responses.

2.3.4 Memory transaction endianness

The endianness of the data transfers to and from the external memories are determined by the Endian mode (N) bit in the MPMCConfig register. The memory controller must be idle (see the busy field of the MPMCStatus register) before endianness is changed, so that the data is transferred correctly.

2.3.5 Memory transaction size

Memory transactions can be 8-bits, 16-bits or 32-bits wide. Any access attempted with a size greater than a word (32-bits) causes an ERROR response on HRESP and the transfer is terminated.

2.3.6 Write protected memory areas

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

2.3.7 Arbiter

The arbiter arbitrates between the AHB slave memory interfaces. AHB interface 0 has the highest access priority and AHB interface 3 has the lowest priority.

2.3.8 Data buffers

The AHB interfaces use read and write buffers to improve memory bandwidth and reduce transaction latency. The MPMC contains four 16-word buffers. The buffers are not tied to a particular AHB interface and can be used as read buffers, write buffers or a combination of both. The buffers are allocated automatically. Because of the way the buffers are

designed they are always coherent for reads and writes and across AHB memory interfaces. The buffers are enabled on a memory bank basis, using the MPMCDynamicConfig or the MPMCStaticConfig registers.

Write buffers are used to:

- Merge write transactions so that the number of external transactions are minimized
- Buffer data until the MPMC can complete the write transaction improving AHB write latency
- Convert all dynamic memory write transactions into quad word bursts on the external memory interface. This enhances transfer efficiency for dynamic memory
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption

Write buffer operation:

- When the buffers are enabled, an AHB write operation writes into the Least Recently Used (LRU) buffer when empty
- When the LRU buffer is not empty the contents of the buffer are flushed to memory to make space for the AHB write data
- When a buffer contains write data it is marked as dirty and its contents are written to memory before the buffer can be reallocated

The write buffers are flushed whenever:

- The memory controller state machine is not busy performing accesses to external memory
- The memory controller state machine is not busy performing accesses to external memory and a AHB interface is writing to a different buffer

The smallest buffer flush is a quad word of data.

Read buffers are used to:

- Buffer read requests from memory. Future read requests that hit the buffer read the data from the buffer rather than memory reduce transaction latency
- Convert all read transactions into quad word bursts on the external memory interface. This enhances transfer efficiency for dynamic memory
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Read buffer operation:

- When the buffers are enabled and the read data is contained in one of the buffers the read data is provided directly from the buffer
- When the read data is not contained in a buffer, the LRU buffer is selected. when the buffer is dirty (contains write data), the write data is flushed to memory. When an empty buffer is available the read command is posted to the memory. While the memory controller is waiting for the data to be returned the memory controller can re-arbitrate to enable additional memory transactions to be processed. When the first

data item is returned from memory the read data is provided to the respective AHB port. Other AHB ports can access the data in the buffer when the read transaction has completed.

A buffer filled by performing a read from memory is marked as not-dirty (not containing write data) and its contents are not flushed back to the memory controller unless a subsequent AHB transfer performs a write that hits the buffer.

2.3.9 Memory controller state machine

The memory controller state machine comprises two functional blocks:

- A static memory controller
- A dynamic-memory controller

The memory controller state machine holds up to two requests in its internal buffer. It prioritizes and rearranges accesses to maximize memory bandwidth and minimize transaction latency. For example, when AHB interfaces 1 and 0 simultaneously request a data transfer from dynamic memory to different memory banks, and the port 0 request address is to a closed page, but port 1 address is for an already open page, the following sequence occurs:

- The ACT command is sent to open the SDRAM row specified by the AHB interface 0 address
- The AHB interface 1 access is completed
- The AHB interface 0 access is completed.

The access priority is modified to take into account the ease of getting data to complete each transfer, but the access priority is always biased to the highest priority AHB interface.

2.3.10 Pad interface

The pad interface block provides the interface to the pads. The pad interface uses feedback clocks, MPMCFBCLKIN[3:0], to re synchronize SDRAM read data from the off-chip to on-chip domains.

3. Register overview

The registers shown in Table 3–31 are part of the MPMC module. Each register is accessible via the AHB register interface. Note that some configuration registers reside in the SYSCREG module (see Section 26–4.6.1).

Table 31. Register overview: MPMC module (register base address: 0x1700 8000)

Name	R/W	Address Offset	Description
MPMCControl	R/W	0x000	Control Register
MPMCStatus	R	0x004	Status Register
MPMCConfig	R/W	0x008	Configuration register
MPMCDynamicControl	R/W	0x020	Dynamic Memory Control Register
MPMCDynamicRefresh	R/W	0x024	Dynamic Memory Refresh Timer Register
MPMCDynamicReadConfig	R/W	0x028	Dynamic Memory Read Configuration Register

Table 31. Register overview: MPMC module (register base address: 0x1700 8000) *…continued*

Name	R/W	Address Offset	Description
MPMCDynamictRP	R/W	0x030	Dynamic Memory Precharge Command Period Register
MPMCDynamictRAS	R/W	0x034	Dynamic Memory Active To Precharge Command Period Register
MPMCDynamictSREX	R/W	0x038	Dynamic Memory Self-refresh Exit Time Register
MPMCDynamictAPR	R/W	0x03C	Dynamic Memory Last Data Out To Active Time Register
MPMCDynamictDAL	R/W	0x040	Dynamic Memory Data-in To Active Command Time Register
MPMCDynamictWR	R/W	0x044	Dynamic Memory Write Recovery Time Register
MPMCDynamictRC	R/W	0x048	Dynamic Memory Active To Active Command Period Register
MPMCDynamictRFC	R/W	0x04C	Dynamic Memory Auto-refresh Period Register
MPMCDynamictXSR	R/W	0x050	Dynamic Memory Exit Self-refresh Register
MPMCDynamictRRD	R/W	0x054	Dynamic Memory Active Bank A to Active Bank B Time Register
MPMCDynamictMRD	R/W	0x058	Dynamic Memory Load Mode Register To Active Command Time Register
MPMCStaticExtendedWait	R/W	0x080	Static Memory Extended Wait Register
MPMCDynamicConfig0	R/W	0x100	Dynamic Memory Configuration Registers 0
MPMCDynamicRasCas0	R/W	0x104	Dynamic Memory RAS and CAS Delay Registers 0
-	R/W	0x120 - 0x164	reserved
MPMCStaticConfig0	R/W	0x200	Static Memory Configuration Registers 0
MPMCStaticWaitWen0	R/W	0x204	Static Memory Write Enable Delay Registers 0
MPMCStaticWaitOen0	R/W	0x208	Static Memory Output Enable Delay Registers 0
MPMCStaticWaitRd0	R/W	0x20C	Static Memory Read Delay Registers 0
MPMCStaticWaitPage0	R/W	0x210	Static Memory Page Mode Read Delay Registers 0
MPMCStaticWaitWr0	R/W	0x214	Static Memory Write Delay Registers 0
MPMCStaticWaitTurn0	R/W	0x218	Static Memory Turn Round Delay Registers 0
MPMCStaticConfig1	R/W	0x220	Static Memory Configuration Registers 1
MPMCStaticWaitWen1	R/W	0x224	Static Memory Write Enable Delay Registers 1
MPMCStaticWaitOen1	R/W	0x228	Static Memory Output Enable Delay Registers 1
MPMCStaticWaitRd1	R/W	0x22C	Static Memory Read Delay Registers 1
MPMCStaticWaitPage1	R/W	0x230	Static Memory Page Mode Read Delay Registers 1
MPMCStaticWaitWr1	R/W	0x234	Static Memory Write Delay Registers 1
MPMCStaticWaitTurn1	R/W	0x238	Static Memory Turn Round Delay Registers 1
-	R/W	0x240 - 0x278	reserved

4. Register description

The chapters that follow will give a description for each register that resides in the MPMC module.

4.1 MPMC control

The MPMCControl register is a 3-bit, read/write register that controls the memory controller operation. The control bits can be altered during normal operation. This register can be accessed with zero wait states. Table 3–32 gives a description of register MPMCControl.

Table 32. Description of the register MPMCControl (address 0x1700 8000)

Bit	Symbol	Access	Reset Value	Description
31:3	-	-	-	Reserved
2	L	R/W	0x0	Indicates normal or low-power mode: • 0 = normal mode (reset value on nPOR and HRESETn) • 1 = low-power mode Entering low-power mode reduces the power consumption of the memory controller. The Dynamic Memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit (L), or by AHB or power-on reset. This bit must only be modified when the MPMC is in the idle state. [1]
1	M	R/W	0x1	Indicates normal or reset memory map: • 0 = normal memory map • 1 = reset memory map. Static memory chip select 1 is mirrored onto chip select 0 (reset value on nPOR) On power-on reset, chip select 1 is mirrored to both chip select 0 and chip select 1 memory areas. Clearing the M bit enables chip select 0 memory to be accessed.
0	E	R/W	0x1	Indicates when the MPMC is enabled or disabled: • 0 = disabled • 1 = enabled (reset value on nPOR and HRESETn) Disabling the MPMC reduces power consumption. When the memory controller is disabled the memory is not refreshed. The memory controller is enabled by setting the enable bit or by AHB or power-on reset. This bit must only be modified when the MPMC is in the idle state. [1]

[1] The external memory cannot be accessed in either the low-power or the disabled state. When a memory access is performed, an error response is generated. The memory-controller AHB-register programming-port can be accessed normally. You can program the MPMC registers in the low-power and/or the disabled state.

4.2 MPMCStatus

The 3-bit read-only MPMCStatus register provides MPMC status information. This register can be accessed with zero wait states. Table 3–33 gives a description of register MPMCStatus.

Table 33. Description of the register MPMCStatus (address 0x1700 8004)

Bit	Symbol	Access	Reset Value	Description
31:3	-	-	-	Reserved
2	SA	R	0x1	This bit indicates the operating mode of the MPMC: 0 = normal mode 1 = self-refresh mode (reset value on nPOR).
1	S	R	0x0	Write buffer status. This bit enables the MPMC to enter cleanly either low-power mode or disabled mode: 0 = write buffers empty (reset value on nPOR) 1 = write buffers contain data.
0	B	R	0x1	This bit ensures that the memory controller enters cleanly either the low-power mode or the disabled mode by determining whether the memory controller is busy or not: 0 = MPMC is idle (reset value on HRESETn) 1 = MPMC is busy performing memory transactions, commands, auto-refresh cycles or is in self-refresh mode (reset value on nPOR and HRESETn).

4.3 MPMCConfig

The 2-bit, read/write, MPMCConfig register configures the operation of the memory controller. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode.

MPMCConfig is accessed with one wait state. Table 3–34 gives a description of register MPMCConfig.

Table 34. Description of the register MPMCConfig (address 0x1700 8008)

Bit	Symbol	Access	Reset Value	Description
31:9	-	-	-	Reserved
8	CLK	R/W	0x0	Clock ratio, CLK HCLK:MPMCCLKOUT ratio: 0 = 1:1 (reset value on nPOR) 1 = 1:2
7:1	-	-	-	Reserved
0	N	R/W	0x0	Endian mode, N Endian mode: 0 = little-endian mode 1 = big-endian mode. The MPMCBIGENDIAN signal determines the value of the endian bit on power-on reset, nPOR. Software can override this value. This field is unaffected by the AHB reset, HRESETn. [1]

[1] The value of the MPMCBIGENDIAN signal is reflected in this field. When programmed this register reflects the last value that is written into it. You must flush all data in the MPMC before switching between little-endian mode and big-endian mode.

4.4 MPMCDynamicControl

The 9-bit, read/write, MPMCDynamicControl register is used to control dynamic-memory operation. The control bits can be altered during normal operation. This register can be accessed with zero wait states. Table 3–35 gives a description of register MPMCDynamicControl.

Table 35. Description of the register MPMCDynamicControl (address 0x1700 8020)

Bit	Symbol	Access	Reset Value	Description
31:14	-	-	0x0	Reserved
13	DP	R/W	0x0	Low-power SDRAM deep-sleep mode, DP: 0 = normal operation (reset value on nPOR) 1 = enter deep power down mode
12:9	-	-	0x0	Reserved
8:7	I	R/W	0x0	SDRAM initialization, I: 00 = issue SDRAM NORMAL operation command (reset value on nPOR) 01 = issue SDRAM MODE command 10 = issue SDRAM PALL (pre charge all) command 11 = issue SDRAM NOP (no operation) command)
6	-	-	0x0	Reserved
5	MMC	R/W	0x0	Memory clock control, MMC: 0 = MPMCCLKOUT enabled (reset value on nPOR) 1 = MPMCCLKOUT disabled [1]
4:3	-	-	0x0	Reserved

Table 35. Description of the register MPMCDynamicControl (address 0x1700 8020)

Bit	Symbol	Access	Reset Value	Description
2	SR	R/W	0x0	Self-refresh request, MPMCSREFREQ, SR: 0 = normal mode 1 = enter self-refresh mode (reset value on nPOR) By writing 1 to this bit, self-refresh can be entered under software control. Writing 0 to this bit returns the MPMC to normal mode. The self-refresh acknowledge bit in the MPMCStatus register must be polled to discover the current operating mode of the MPMC. [2]
1	CS	R/W	0x0	Dynamic-memory clock control, CS: 0 = MPMCCLKOUT stops when all SDRAMs are idle and during self-refresh mode 1 = MPMCCLKOUT runs continuously (reset value on nPOR) When the clock control is LOW the output clock MPMCCLKOUT is stopped when there are no SDRAM transactions. The clock is also stopped during self-refresh mode.
0	CE	R/W	0x0	Dynamic-memory clock enable, CE: 0 = clock enable of idle devices are deserted to save power (reset value on nPOR) 1 = all clock enables are driven HIGH continuously [3]

[1] You can disable MPMCCLKOUT when there are no SDRAM memory transactions. When enabled you can use this field in conjunction with the dynamic-memory clock control (CS) field.

[2] The memory controller exits from the power-on reset with the self-refresh bit on HIGH. To enter normal functional mode, set this bit LOW. Writing to this register with a HIGH places this register into the self-refresh mode. This functionality enables data to be stored over SDRAM self-refresh when the ASIC is powered down.

[3] Clock enable must be HIGH during SDRAM initialization.

4.5 MPMCDynamicRefresh

The 11-bit, read/write, MPMCDynamicRefresh register configures dynamic-memory operation. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering during either the low-power or the disabled mode. However, these control bits can be altered during normal operation when necessary. This register is accessed with one wait state. Table 3–36 gives a description of register MPMCDynamicRefresh.

Table 36. Description of the register MPMCDynamicRefresh (address 0x1700 8024)

Bit	Symbol	Access	Reset Value	Description
31:11	-	-	-	Reserved
10:0	REFRESH	R/W	0x0	Refresh timer, REFRESH: 0x0 = refresh disabled (reset value on nPOR) 0x1 1(x16) = 16 HCLK ticks between SDRAM refresh cycles 0x8 8(x16) = 128 HCLK ticks between SDRAM refresh cycles 0x1-0x7FF n(x16) = 16n HCLK ticks between SDRAM refresh cycles

For example, for the refresh period of 16 μs and an HCLK frequency of 50 MHz, the following value must be programmed into this register:

$$16 \times 10^{-6} \times \frac{50 \times 10^{-6}}{16} = 50 = 0 \times 32$$

The refresh cycles are distributed evenly. However, there might be slight variations when the auto-refresh command is issued depending on the status of the memory controller.

Unlike other SDRAM memory timing parameters the refresh period is programmed in the HCLK domain. When variable clock scaling is used, you can program the desired value for the refresh timer in the MPMC_testmode0 register of the SYSCREG block (see Table 26–542).

4.6 MPMCDynamicReadConfig

The 2-bit, read/write, MPMCDynamicReadConfig register enables you to configure the dynamic-memory read strategy. This register must be modified only during system initialization. This register can be accessed with one wait state. Table 3–37 gives a description of register MPMCDynamicReadConfig.

Table 37. Description of the register MPMCDynamicReadConfig (address 0x1700 8028)

Bit	Symbol	Access	Reset Value	Description
31:2	-	-	-	Reserved
1:0	RD	R/W	0x0	Read data strategy, RD: 00 = clock out delayed strategy, using MPMCCLKOUT (command not delayed, clock out delayed). Reset value on nPOR 01 = command delayed strategy, using MPMCCLKDELAY (command delayed, clock out not delayed) 10 = command delayed strategy plus one clock cycle, using MPMCCLKDELAY (command delayed, clock out not delayed) 11 = command delayed strategy plus two clock cycles, using MPMCCLKDELAY (command delayed, clock out not delayed)

4.7 MPMCDynamictRP

The 4-bit, read/write, MPMCDynamictRP register enables you to program the pre charge command period, t_{RP}. This register must be modified only during system initialization. This value is found normally in SDRAM data sheets as t_{RP}. This register can be accessed with one wait state. Table 3–38 gives a description of register MPMCDynamictRP.

Table 38. Description of the register MPMCDynamictRP (address 0x1700 8030)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tRP	R/W	0xF	Precharge command period, tRP: 0x0-0xE = n + 1 clock cycles 0xF = 16 clock cycles (reset value on nPOR)

4.8 MPMCDynamictRAS

The 4-bit, read/write, MPMCDynamictRAS register enables you to program the active to pre charge command period, t_{RAS}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{RAS}. This register can be accessed with one wait state. Table 3–39 gives a description of register MPMCDynamictRAS.

Table 39. Description of the register MPMCDynamictRAS (address 0x1700 8034)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tRAS	R/W	0xF	Active to pre charge command period, tRAS: 0x0-0xE = n + 1 clock cycles 0xF = 16 clock cycles (reset value on nPOR)

4.9 MPMCDynamictSREX

The 4-bit, read/write, MPMCDynamictSREX register enables you to program the self-refresh exit time, t_{SREX}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{SREX}. For devices without this parameter you use the same value as t_{XSR}. This register can be accessed with one wait state. Table 3–40 gives a description of register MPMCDynamictSREX.

Table 40. Description of the register MPMCDynamictSREX (address 0x1700 8038)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tSREX	R/W	0xF	Self-refresh exit time, tSREX: 0x0-0xE = n + 1 clock cycles [1] 0xF = 16 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.10 MPMCDynamictAPR

The 4-bit, read/write, MPMCDynamictAPR register enables you to program the last-data-out to active command time, t_{APR}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{APR}. This register can be accessed with one wait state. Table 3–41 gives a description of register MPMCDynamictAPR.

Table 41. Description of the register MPMCDynamictAPR (address 0x1700 803C)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tAPR	R/W	0xF	Last-data-out to active command time, tAPR: 0x0-0xE = n + 1 clock cycles [1] 0xF = 16 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.11 MPMCDynamictDAL

The 4-bit, read/write, MPMCDynamictDAL register enables you to program the data-in to active command time, t_{DAL}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{DAL} or t_{APW}. This register can be accessed with one wait state. Table 3–42 gives a description of register MPMCDynamictDAL.

Table 42. Description of the register MPMCDynamictDAL (address 0x1700 0840)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tDAL	R/W	0xF	Data-in to active command, tDAL: 0x0-0xE = n clock cycles [1] 0xF = 15 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.12 MPMCDynamictWR

The 4-bit, read/write, MPMCDynamictWR register enables you to program the write recovery time, t_{WR}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{WR}, t_{DPL}, t_{RWL} or t_{RDL}. This register can be accessed with one wait state. Table 3–43 gives a description of register MPMCDynamictWR.

Table 43. Description of the register MPMCDynamictWR (address 0x1700 8044)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tWR	R/W	0xF	Write recovery time, tWR: 0x0-0xE = n + 1 clock cycles [1] 0xF = 16 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.13 MPMCDynamictRC

The 5-bit, read/write, MPMCDynamictRC register enables you to program the active to active command period, t_{RC}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{RC}. This register can be accessed with one wait state. Table 3–44 gives a description of register MPMCDynamictRC.

Table 44. Description of the register MPMCDynamictRC (address 0x1700 8048)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	-	Reserved
4:0	tRC	R/W	0x1F	Active to active command period, tRC: 0x0-0x1E = n + 1 clock cycles [1] 0x1F = 32 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.14 MPMCDynamictRFC

The 5-bit, read/write, MPMCDynamictRFC register enables you to program the auto-refresh period and auto-refresh to active command period, t_{RFC}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering either low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{RFC} or sometimes as t_{RC}. This register can be accessed with one wait state. Table 3–45 gives a description of register MPMCDynamictRFC.

Table 45. Description of the register MPMCDynamictRFC (address 0x1700 804C)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	-	Reserved
4:0	tRFC	R/W	0x1F	Auto-refresh period and auto-refresh to active command period, tRFC: 0x0-0x1E = n + 1 clock cycles [1] 0x1F = 32 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.15 MPMCDynamictXSR

The 5-bit, read/write, MPMCDynamictXSR register enables you to program the exit self-refresh to active command time, t_{XSR}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{XSR}. This register can be accessed with one wait state. Table 3–46 gives a description of register MPMCDynamictXSR.

Table 46. Description of the register MPMCDynamictXSR (address 0x1700 8050)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	-	Reserved
4:0	tXSR	R/W	0x1F	Exit self-refresh to active command time, tXSR: 0x0-0x1E = n + 1 clock cycles [1] 0x1F = 32 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.16 MPMCDynamictRRD

The 4-bit, read/write, MPMCDynamictRRD register enables you to program the active bank A to active bank B latency, t_{RRD}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{RRD}. This register can be accessed with one wait state. Table 3–47 gives a description of register MPMCDynamictRRD.

Table 47. Description of the register MPMCDynamictRRD (address 0x1700 8054)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tRRD	R/W	0xF	Active bank A to active bank B latency, tRRD: 0x0-0xE = n + 1 clock cycles [1] 0xF = 16 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.17 MPMCDynamictMRD

The 4-bit, read/write, MPMCDynamictMRD register enables you to program the load mode register to active command time, t_{MRD}. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. This value is found normally in SDRAM data sheets as t_{MRD} or t_{RSA}. This register can be accessed with one wait state. Table 3–48 gives a description of register MPMCDynamictMRD.

Table 48. Description of the register MPMCDynamictMRD (0x1700 8058)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	tMRD	R/W	0xF	Load mode register to active command time, tMRD: 0x0-0xE = n + 1 clock cycles [1] 0xF = 16 clock cycles (reset value on nPOR)

[1] The delay is in MPMCCLK cycles.

4.18 MPMCStaticExtendedWait

The 10-bit, read/write, MPMCStaticExtendedWait register is used to time long static memory read and write transfers (that are longer than can be supported by the MPMCStaticWaitRd[n] or MPMCStaticWaitWr[n] registers) when the EW bit of the MPMCStaticConfig registers is enabled. There is only a single MPMCStaticExtendedWait register. This is used by the relevant static memory chip select when the appropriate ExtendedWait (EW) bit in the MPMCStaticConfig register is set. It is recommended that this register is modified either during system initialization or when there are no current or outstanding transactions. However, when necessary, these control bits can be altered during normal operation. This register can be accessed with one wait state. Table 3–49 gives a description of register MPMCStaticExtendedWait.

Table 49. Description of the register MPMCStaticExtendedWait (address 0x1700 8080)

Bit	Symbol	Access	Reset Value	Description
31:10	-	-	-	Reserved
9:0	EXTENDEDWAIT	R/W	0x0	External wait time out, EXTENDEDWAIT: 0x0 = 16 clock cycles [1] (reset value on nPOR) 0x1-0x3FF = (n+1) x16 clock cycles

[1] The delay is in MPMCCLK cycles.

For example, for a static memory read/write transfer time of 16us and an MPMCCLK frequency of 50 MHz, the following value must be programmed into this register:

$$\left[16x10^{-6}x\left[\frac{50x10^{6}}{16}\right]\right] - 1 = 49 = 0x31$$

4.19 MPMCDynamicConfig0

The 11-bit, read/write, MPMCDynamicConfig0 registers enable you to program the configuration information for the relevant dynamic-memory chip select. This register is modified normally only during system initialization. These registers can be accessed with one wait state. Table 3–50 gives a description of register MPMCDynamicConfig0.

Table 50. Description of the register MPMCDynamicConfig0 (address 0x1700 8100)

Bit	Symbol	Access	Reset Value	Description
31:21	-	-	-	Reserved
20	P	R/W	0x0	Write protect, P: 0 = writes not protected (reset value on nPOR) 1 = write protected.
19	B	R/W	0x0	Buffer enable, B: 0 = buffer disabled for accesses to this chip select (reset value on nPOR) 1 = buffer enabled for accesses to this chip select [1]
18:15	-	-	-	Reserved
14	AM	R/W	0x0	Address mapping, AM. 0 = reset value on nPOR
13	-	-	-	Reserved
12:7	AM	R/W	0x0	Address mapping, AM. 00000000 = reset value on nPOR [2]
6:5	-	-	-	Reserved
4:3	MD	R/W	0x0	Memory device, MD: 00 = SDRAM (reset value on nPOR) 01 = low-power SDRAM 10 = Micron SyncFlash 11 = reserved
2:0	-	-	-	Reserved

[1] The buffers must be disabled during SDRAM and SyncFlash initialization. They must also be disabled when performing SyncFlash commands. The buffers must be enabled during normal operation.

[2] The SDRAM column and row width and number of banks are computed automatically from the address mapping.

Address mappings that are not shown in Table 3–51 are reserved.

Table 51. Address mapping

[14]	[12]	[11:9]	[8:7]	Description
16-bit external bus high-performance address mapping (Row, Bank, Column)				
0	0	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	0	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	0	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	0	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	0	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	0	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	0	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	0	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	0	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	0	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
16-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
0	1	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	1	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	1	001	00	64Mb (8Mx8), 2 banks, row length = 12, column length = 9
0	1	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	1	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	1	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	1	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	1	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	1	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	1	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10

The LPC31xx always interface with external SDRAM devices using a 16-bit wide data bus. The SDRAM chip select can be connected to a single 16-bit memory device; in this case x16 mapping values (MPMCDynamicConfig0[8:7] = 01) should be used. When the SDRAM chip select is connected to two 8-bit memory devices to form a 16-bit external system memory, x8 mapping values (MPMCDynamicConfig0[8:7] = 00) should be used. For linear AHB address to SDRAM bank mapping the bank select signals (BAx) of the SDRAM device should be connected as shown in the following tables.

Table 52. 16-bit wide data bus address mapping, SDRAM (RBC)

16-bit wide device 16M SDRAM (1M × 16, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	9/BA	-	-	-	20	19	18	17	16	15	14	13	12	11	10
AHB address to column address	9/BA	-	-	-	AP	-	-	8	7	6	5	4	3	2	1
Memory device connections	BA	-	-	-	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 16M SDRAM (2M × 8, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	-	10/BA	-	-	21	20	19	18	17	16	15	14	13	12	11
AHB address to column address	-	10/BA	-	-	AP	-	9	8	7	6	5	4	3	2	1
Memory device connections	-	BA	-	-	10/AP	9	8	7	6	5	4	3	2	1	0

Table 52. 16-bit wide data bus address mapping, SDRAM (RBC) *…continued*

16-bit wide device 64M SDRAM (4M × 16, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	9/BA0	10/BA1	-	22	21	20	19	18	17	16	15	14	13	12	11
AHB address to column address	9/BA0	10/BA1	-	-	AP	-	-	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 64M SDRAM (8M × 8, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	11/BA1	10/BA0	-	23	22	21	20	19	18	17	16	15	14	13	12
AHB address to column address	11/BA1	10/BA0	-	-	AP	-	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
16-bit wide device 128M SDRAM (8M × 16, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	11/BA1	10/BA0	-	23	22	21	20	19	18	17	16	15	14	13	12
AHB address to column address	11/BA1	10/BA0	-	-	AP	-	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 128M SDRAM (16M × 8, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	11/BA0	12/BA1	-	24	23	22	21	20	19	18	17	16	15	14	13
AHB address to column address	11/BA0	12/BA1	-	-	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
16-bit wide device 256M SDRAM (16M × 16, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	11/BA1	10/BA0	24	23	22	21	20	19	18	17	16	15	14	13	12
AHB address to column address	11/BA1	10/BA0	-	-	AP	-	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	12	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 256M SDRAM (32M × 8, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	11/BA0	12/BA1	25	24	23	22	21	20	19	18	17	16	15	14	13
AHB address to column address	11/BA0	12/BA1	-	-	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	12	11	10/AP	9	8	7	6	5	4	3	2	1	0
16-bit wide device 512M SDRAM (32M × 16, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	11/BA0	12/BA1	25	24	23	22	21	20	19	18	17	16	15	14	13
AHB address to column address	11/BA0	12/BA1	-	-	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	12	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 512M SDRAM (64M × 8, RBC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	13/BA1	12/BA0	26	25	24	23	22	21	20	19	18	17	16	15	14
AHB address to column address	13/BA1	12/BA0	-	11	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	12	11	10/AP	9	8	7	6	5	4	3	2	1	0

Table 53. 16-bit wide data bus address mapping, SDRAM (BRC)

16-bit wide device 16M SDRAM (1M × 16, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	-	20/BA0	-	-	19	18	17	16	15	14	13	12	11	10	9
AHB address to column address	-	20	-	-	AP	-	-	8	7	6	5	4	3	2	1
Memory device connections	-	BA	-	-	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 16M SDRAM (2M × 8, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	21/BA	-	-	-	20	19	18	17	16	15	14	13	12	11	10
AHB address to column address	21/BA	-	-	-	AP	–	9	8	7	6	5	4	3	2	1
Memory device connections	BA	-	-	-	10/AP	9	8	7	6	5	4	3	2	1	0
16-bit wide device 64M SDRAM (4M × 16, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	21/BA0	22/BA1	-	20	19	18	17	16	15	14	13	12	11	10	9
AHB address to column address	21/BA0	22/BA1	-	-	AP	-	-	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 64M SDRAM (8M × 8, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	23/BA1	22/BA0	-	21	20	19	18	17	16	15	14	13	12	11	10
AHB address to column address	23/BA1	22/BA0	-	-	AP	-	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
16-bit wide device 128M SDRAM (8M × 16, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	23/BA1	22/BA0	—	21	20	19	18	17	16	15	14	13	12	11	10
AHB address to column address	23/BA1	22/BA0	—	—	AP	—	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 128M SDRAM (16M × 8, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	23/BA0	24/BA1	—	22	21	20	19	18	17	16	15	14	13	12	11
AHB address to column address	23/BA0	24/BA1	—	—	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	-	11	10/AP	9	8	7	6	5	4	3	2	1	0
16-bit wide device 256M SDRAM (16M × 16, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	23/BA0	24/BA1	22	21	20	19	18	17	16	15	14	13	12	11	10
AHB address to column address	23/BA0	24/BA1	—	—	AP	—	9	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	12	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 256M SDRAM (32M × 8, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	25/BA1	24/BA0	23	22	21	20	19	18	17	16	15	14	13	12	11
AHB address to column address	25/BA1	24/BA0	—	—	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	12	11	10/AP	9	8	7	6	5	4	3	2	1	0

Table 53. 16-bit wide data bus address mapping, SDRAM (BRC) *…continued*

16-bit wide device 512M SDRAM (32M × 16, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	25/BA1	24/BA0	23	22	21	20	19	18	17	16	15	14	13	12	11
AHB address to column address	25/BA1	24/BA0	—	—	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA1	BA0	12	11	10/AP	9	8	7	6	5	4	3	2	1	0
Two 8-bit wide devices 512M SDRAM (64M × 8, BRC)															
External address pin, EBI_A[14:0]	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB address to row address	25/BA0	26/BA1	24	23	22	21	20	19	18	17	16	15	14	13	12
AHB address to column address	25/BA0	26/BA1	—	11	AP	10	9	8	7	6	5	4	3	2	1
Memory device connections	BA0	BA1	12	11	10/AP	9	8	7	6	5	4	3	2	1	0

4.20 MPMCDynamicRasCas0

The 4-bit, read/write, MPMCDynamicRasCas0 registers enable you to program the RAS and CAS latencies for the relevant dynamic memory. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. The MPMCDynamicRasCas0 registers are accessed with one wait state. The values programmed into these registers must be consistent with the values used to initialize the SDRAM memory device. Table 3–54 gives a description of register MPMCDynamicRasCas0.

Table 54. Description of the register MPMCDynamicRasCas0 (address 0x1700 8104)

Bit	Symbol	Access	Reset Value	Description
31:10	-	-	-	Reserved
9:8	CAS	R/W	0x3	CAS latency, CAS: 00 = reserved 01 = one clock cycle 10 = two clock cycles 11 = three clock cycles (reset value on nPOR)
7:2	-	-	-	Reserved
1:0	RAS	R/W	0x3	RAS latency (active to read/write delay), RAS: 00 = reserved 01 = one clock cycle [1] 10 = two clock cycles 11 = three clock cycles (reset value on nPOR)

[1] The RAS to CAS latency (RAS) and the CAS latency (CAS) are each defined in MPMCCLK cycles.

4.21 MPMCStaticConfig0/1

The 8-bit, read/write, MPMCStaticConfig0/1 registers are used to configure the static memory configuration. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. The MPMCStaticConfig0/1 registers are accessed with one wait state. Synchronous burst mode memory devices are not supported. Table 3–55 gives a description of register MPMCStaticConfig0/1.

Table 55. Description of the register MPMCStaticConfig (MPMCStaticConfig0, address 0x1700 8120 and MPMCStaticConfig1 0x1700 8220)

Bit	Symbol	Access	Reset Value	Description
31:21	-	-	0x0	reserved
20	WP	R/W	0x0	WP R/W 0x0 Write protect. 0 = writes not protected (reset value on nPOR) 1 = write protected.
19	B	R/W	0x0	Buffer enable. 0 = write buffer disabled (reset value on nPOR) 1 = write buffer enabled.
18:9	-	W	0x0	Read undefined. Write as 0.
8	EW	R/WQ	0x0	Extended wait (EW) uses the MPMCStaticExtendedWait Register to time both the read and write transfers rather than the MPMCStaticWaitRd and MPMCStaticWaitWr Registers. This enables much longer transactions. 0 = Extended wait disabled (reset value on nPOR) 1 = Extended wait enabled.
7	BLS	R/W	0x0	This bit affects the behavior of the EBI_NCAS_BLOUT_0, EBI_NRAS_BLOUT_1 and EBI_nWE signals on the External Memory Interface. When the BLS bit is set to 1, the nBLOUT[1:0] signals are byte lane enable strobes and will be low for both static memory read and write access, and signal EBI_nWE will be low for writes. This is used when interfacing to a static memory with multiple byte lane strobe pins and a separate write strobe pin. When the BLS bit is set to 0, the nBLOUT[1:0] signals become byte lane write strobes and will only be low during static memory writes. The EBI_nWE signal never goes active when BLS is 0. Writes: 1 = The active bits in nBLOUT[1:0] are LOW; EBI_nWE is active 0 = The active bits in nBLOUT[1:0] are LOW; EBI_nWE is NOT active Reads: 1 = The active bits in nBLOUT[1:0] are LOW 0 = All the bits in nBLOUT[1:0] are HIGH
6	PC	R/W	0x0	Chip select polarity, PC: 0 = active LOW chip select 1 = active HIGH chip select The relevant MPMCSTCSxPOL signal determines the value of the chip select polarity on power-on reset, nPOR. Software can override this value. This field is unaffected by AHB reset, HRESETn. [3]
5:4	-	-	-	Reserved

Table 55. Description of the register MPMCStaticConfig (MPMCStaticConfig0, address 0x1700 8120 and MPMCStaticConfig1 0x1700 8220) *…continued*

Bit	Symbol	Access	Reset Value	Description
3	PM	R/W	0x0	Page mode, PM: 0 = disabled (reset value on nPOR) 1 = Async page mode enabled (page length four) In page mode the MPMC can burst up to four external accesses. Therefore, devices with asynchronous page mode burst four or higher devices are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.
2	-	-	-	Reserved
1:0	MW	R/W	0x0	Memory width, MW: 00 = 8-bit (reset value for chip select 0, 2 and 3 on nPOR) 01 = 16-bit 10 = reserved 11 = reserved The MPMCSTCS1MW[1:0] signal determines the value of the chip select 1 memory width field on power-on reset, nPOR. Software can override this value. This field is unaffected by AHB reset, HRESETn. [4]

[1] Extended wait and page mode cannot be selected simultaneously.

[2] For chip select 1, the value of the MPMCSTCS1PB signal is reflected in this field. When programmed this register reflects the last value that is written into it.

[3] The value of the relevant MPMCSTCSxPOL signal is reflected in this field. When programmed this register reflects the last value that is written into it.

[4] For chip select 1, the value of the MPMCSTCS1MW[1:0] signal is reflected in this field. When programmed this register reflects the last value that is written into it. MPMCSTCS1MW[1:0] value can be set in SYSCREG_WIRE_EBI_MSIZE_INIT (address 0x1300 2874).

4.22 MPMCStaticWaitWen0/1

The 4-bit, read/write, MPMCStaticWaitWen0/1 registers enable you to program the delay from the chip select to the write enable. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. The MPMCStaticWaitWen0/1 registers are accessed with one wait state. Table 3–56 gives a description of register MPMCStaticWaitWen0/1.

Table 56. Description of the register MPMCStaticWaitWen (MPMCStaticWaitWen0, address 0x1700 8204 and MPMCStaticWaitWen1, address 0x1700 8224)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	WAITWEN	R/W	0x0	Wait write enable, WAITWEN Delay from chip select assertion to write enable: 0000 = one HCLK cycle delay between assertion of chip select and write enable (reset value on nPOR) 0001-1111 = (n + 1) HCLK cycle delay [1]

[1] The delay is (WAITWEN +1) x tHCLK.

4.23 MPMCStaticWaitOen0/1

The 4-bit, read/write, MPMCStaticWaitOen0/1 registers enable you to program the delay from the chip select or address change, whichever is later, to the output enable. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. The MPMCStaticWaitOen0/1 registers are accessed with one wait state. Table 3–57 gives a description of register MPMCStaticWaitOen0/1.

Table 57. Description of the register MPMCStaticWaitOen (MPMCStaticWaitOen0, address 0x1700 8208 and MPMCStaticWaitOen1, address 0x1700 8228)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	WAITOEN	R/W	0x0	Wait output enable, WAITOEN Delay from chip select assertion to output enable: 0000 = No delay (reset value on nPOR) 0001-1111= n cycle delay [1]

[1] The delay is WAITOEN x tHCLK.

4.24 MPMCStaticWaitRd0/1

The 5-bit, read/write, MPMCStaticWaitRd0/1 registers enable you to program the delay from the chip select to the read access. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. It is not used when the extended wait bit is enabled in the MPMCStaticConfig0/1 registers. The MPMCStaticWaitRd0/1 registers are accessed with one wait state. Table 3–58 gives a description of register MPMCStaticWaitRd0/1.

Table 58. Description of the register MPMCStaticWaitRd (MPMCStaticWaitRd0, address ox1700 820C and MPMCStaticWaitRd1, address 0x1700 8022C)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	-	Reserved
4:0	WAITRD	R/W	0x1F	Non-page mode read wait states or asynchronous page mode read first access wait state, WAITRD Non-page mode read or asynchronous page mode read, first read only: 00000-11110 = (n + 1) HCLK cycles for read accesses [1] 11111 = 32 HCLK cycles for read accesses (reset value on nPOR)

[1] For non-sequential reads, the wait state time is (WAITRD + 1) x tHCLK.

4.25 MPMCStaticWaitPage0/1

The 5-bit, read/write, MPMCStaticWaitPage0/1 registers enable you to program the delay for asynchronous page mode sequential accesses. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. MPMCStaticWaitPage0/1 is accessed with one wait state. Table 3–59 gives a description of register MPMCStaticWaitPage0/1.

Table 59. Description of the register MPMCStaticWaitPage (MPMCStaticWaitPage0, address 0x1700 8210 and MPMCStaticWaitPage1, address 0x1700 8230)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	-	Reserved
4:0	WAITPAGE	R/W	0x1F	Asynchronous page mode read after the first read wait states, WAITPAGE Number of wait states for asynchronous page mode read accesses after the first read: 00000-11110 = (n+ 1) HCLK cycle read access time [1] 11111 = 32 HCLK cycle read access time (reset value on nPOR)

[1] For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE + 1) x tHCLK.

4.26 MPMCStaticWaitWr0/1

The 5-bit, read/write, MPMCStaticWaitWr0/1 registers enable you to program the delay from the chip select to the write access. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. These registers are not used when the extended wait (EW) bit is enabled in the MPMCStaticConfig register. The MPMCStaticWaitWr0/1 registers are accessed with one wait state. Table 3–60 gives a description of register MPMCStaticWaitWr0/1.

Table 60. Description of the register MPMCStaticWaitWr (MPMCStaticWaitWr0,address 0x1700 8214 and MPMCStaticWaitWr1, address 0x1700 8234)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	-	Reserved
4:0	WAITWR	R/W	0x1F	Write wait states, WAITWR SRAM wait state time for write accesses after the first read: 00000-1110 = (n + 2) HCLK cycle write access time [1] 11111 = 33 HCLK cycle write access time (reset value on nPOR)

[1] The wait state time for write accesses after the first read is WAITWR (n + 2) x tHCLK.

4.27 MPMCStaticWaitTurn0/1

The 4-bit, read/write, MPMCStaticWaitTurn0/1 registers enable you to program the number of bus turnaround cycles. It is recommended that these registers are modified either during system initialization or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. The MPMCStaticWaitTurn0/1 registers are accessed with one wait state. To prevent bus contention on the external memory data bus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses. The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses. Table 3–61 gives a description of register MPMCStaticWaitTurn0/1.

Table 61. Description of the register MPMCStaticWaitTurn (MPMCStaticWaitTurn0, address 0x1700 8218 and MPMCStaticWaitTurn1, address 0x1700 8238)

Bit	Symbol	Access	Reset Value	Description
31:4	-	-	-	Reserved
3:0	WAITTURN	R/W	0xF	Bus turnaround cycles, WAITTURN: 0000-1110 = (n + 1) HCLK turnaround cycles [1] 1111 = 16 HCLK turnaround cycles (reset value on nPOR)

[1] Bus turnaround time is (WAITTURN + 1) x tHCLK.

5. Power optimization

The MPMC is partly low-level clock gated. This feature is built in hardware and software does not have any control over this feature. Low-level clock gating involves grouping a number (in this case 32) of d-type flip-flops together and only enabling a clock to them when required. This is done for reducing power consumption.

6. Programming guide

6.1 SDRAM initialization

This chapter describes the initialization of both high performance and low performance SDRAM. And describes in more detail the initialization of the Mode Register and/or Extended Mode Register that is part of the initialization of the SDRAM.

6.2 Initialization of high performance SDRAM (RBC)

Although the initialization for different kind of SDRAMs is almost the same, it is advised to check the SDRAM data sheet for the start up procedure. For extensive examples consult the *ARM Technical Reference Manual of the MPMC (PI172)*. An example is given below:

1. Disable buffers during SDRAM initialization. Dynamic-memory clock enable (CE) must be HIGH during SDRAM initialization. Enable MPMC and use normal address map in control register.
2. Wait 100ms after the power is applied and the clocks are stable.
3. Set SDRAM Initialization value to NOP in Dynamic Control register, to perform a NOP command to SDRAM.
4. Set SDRAM Initialization value to PALL in Dynamic Control register, to perform a pre-charge all command to SDRAM.
5. Program minimum refresh value (0x1 = 16 HCLKS) in MPMCDynamicRefresh register..
6. Wait until several (Micron recommends minimum of 2 refresh cycles for their SDRAMs) SDRAM refresh cycles have occurred.
7. Program the operational value in the refresh register.
8. Program the operational value in the latency register.
9. Program the operational value in the configuration register.

10. Set SDRAM Initialization value to MODE in Dynamic Control register, to perform a MODE command to SDRAM.
11. Read from SDRAM to program mode register.
12. Set SDRAM Initialization value to NORMAL. Dynamic-memory clock enable (CE) can be made LOW, than idle devices are de-asserted to save power.
13. Enable buffers.

It does not matter what type of SDRAM is used for choosing high performance or low power address mapping. Meaning to say it is possible to use a high performance SDRAM with low power address mapping and the other way around, but it can have an influence on the power dissipation.

Example:

```
void sdram_init(void)
{
pSYSCREG_REGS syscregs = (pSYSCREG_REGS)SYSCREG_BASE;
pvhMpmcPl172Regs pl172Regs =
pvhMpmcPl172Regs)AHB_MPMC_PL172_CFG_BASE;
UInt32 *ptr = NULL;
int i = 0;
volatile int j,addr;
syscregs->mpmp_delaymodes=0x0;
pl172Regs->MpmcControl = 0x01;
pl172Regs->MpmcConfig = 0x000;
pl172Regs->MpmcDyCntl = VH_MPMC_DYCNTL_REG_POR_VAL;
// Ensure that the following values have the following value:
pl172Regs->MpmcDyCntl = 0x7; // Clock enable must be high during SDRAM initialisation
pl172Regs->MpmcDynamic[SDRAM_SEL].Config = 0x0; // During SDRAM initialisation buffers
// disabled.
/* Clock out delay methodology */
pl172Regs->MpmcDyRdCfg = 0x00;
/* 0. Clear M-bit in MPMCControl */
pl172Regs->MpmcControl = 0x01;
/* 1. Wait 100ms after the power is applied and the clocks are stabilized
*/
for (i=500;i>0;i--) {}
/* 2. Set SDRAM Initialization (I) value to NOP. This issues a NOP to the SDRAM */
pl172Regs->MpmcDyCntl = 0x183; //issue NOP to SDRAM
/* wait */
for (i=4;i>0;i--) {}
/* 3. Set SDRAM Initialization (I) value to PALL (PRE-ALL). This issues a pre charge
all instruction to the SDRAM */
pl172Regs->MpmcDyCntl = 0x103; //issue pre charge all instruction to the SDRAM
memories
/* Wait for tRP approx */
for (i=4;i>0;i--) {}
/* 4. Perform a number of refresh cycles */
pl172Regs->MpmcDyRef = 0x2;
/* Wait for tRP approx */
for (i=4;i>0;i--) {}
```

```
/* 5. Wait until two SDRAM refresh cycles have occurred */
for(i=10;i>0;i--) {}
/* 6. Program the operational value in the refresh register.
MpmcDyRef = (tref / #rows) * HCLK / 16.
For tref = 64000 ms, #rows = 4096, HCLK = 16 MHz à MpmcDyRef =11,7 => 0xB */
pl172Regs->MpmcDyRef = 0xB;
/* Wait for tRP approx */
for (i=20;i>0;i--) {}
/* 7. Program the operational value in the latency register. */
pl172Regs->MpmcDynamic[SDRAM_SEL].RasCas = 0x0202;
/* Wait for tRP approx */
for (i=20;i>0;i--) {}
/* 8. Program the operational value in the configuration register. */
pl172Regs->MpmcDynamic[SDRAM_SEL].Config = 0x00280; // RBC address mapping
/* Wait for tRP approx */
for (i=20;i>0;i--) {}
/* 9. Set SDRAM Initialization (I) value to MODE.*/
pl172Regs->MpmcDyCntl = 0x083;
addr = SDRAM0 + 0x11800; /* */
ptr = (UInt32 *) addr;
for (i =0; i < 1; i++) {
/* 10. Read from SDRAM to program mode register. Write 0x22 to add pins 0f SDRAM.*/
j = ptr[i];
// The following 'if' statement doesn't have any functional value. This is done to
avoid that the // statement j = ptr[i] is not compiled away.
if (j!= (i | (1 << 9) | (2 << 18) | (3 << 27))){}}
/* Wait some time */
for (i=400;i>0;i--) {}
/* 11. Initialisation of SDRAM to SDRAM NORMAL.*/
pl172Regs->MpmcDyCntl = 0x000; //initialisation to SDRAM NORMAL
/* 12. Buffers are enabled.*/
pl172Regs->MpmcDynamic[SDRAM_SEL].Config = 0x80283; // RBC address mapping
pl172Regs->MpmcControl = 0x01;}
```

6.3 Initialization of low power SDRAM (BRC)

The procedure shown in Section 3–6.2 can be used also for the initialization of a low power SDRAM. In that case only the address mapping has to be changed for low power and the address for programming the Mode Register has to be changed.

For a specific low power initialization sequence, you can consult the *ARM Technical Reference Manual of the MPMC*.

6.4 Initialization Mode Register or Extended Mode Register of SDRAM

Dependent of the size of the SDRAM, RBC address mapping or BRC address mapping you have to read from different addresses to program the Mode register or Extended Mode Register of the (mobile) SDRAM.

6.5 High performance SDRAM (RBC)

Writing 0x23 to the Mode Register implies the following:

- Burst length = 8
- Burst type = Sequential
- CAS latency = 2
- Operating mode = Standard Operation
- Write burst mode = Programmed Burst Length.

Offset mode register settings are given in Table 3–62:

Table 62. High performance SDRAM address mapping (RBC)

SDRAM size (Mbit)	Total SDRAM size	Bit places shifted (bit)	Offset mode register address [1]
16 (1Mx16)	2 MB	10	0x08C00
16 (2Mx8)	4 MB	11	0x11800
64 (4Mx16)	8 MB	11	0x11800
64 (8Mx8)	16 MB	12	0x23000
128 (8Mx16)	16 MB	12	0x23000
128 (16Mx8)	32 MB	13	0x46000
256 (16Mx16)	32 MB	12	0x23000
256 (32Mx8)	64 MB	13	0x46000
512 (32Mx16)	64 MB	13	0x46000
512 (64Mx8)	128 MB	14	0x8C000

[1] Base address of the SDRAM is 0x3000 0000 for programming mode register.

Example: Initialization of the mode register of a high performance SDRAM 64 Mbit (4M x 16) with RBC address mapping.

```
#define SDRAM0_BASE 0x30000000
#define SDRAM0 SDRAM0_BASE
int addr;
UInt32 *ptr = NULL;
addr = SDRAM0 + 0x11800; //see Table above
ptr = (UInt32 *) addr;
```

6.6 Low power SDRAM (BRC)

Writing 0x23 to the Mode Register implies the following:

- Burst length = 8
- Burst type = Sequential
- CAS latency = 2
- Operating mode = Standard Operation
- Write burst mode = Programmed Burst Length.

Writing 0x00 to the Extended Mode Register, implies the following:

- Partial array self-refresh = All Banks
- Temperature compensated self-refresh = 70°C.

Offset mode register settings are given in Table 3–63.

Table 63. Low power SDRAM address mapping (BRC)

SDRAM size (Mbit)	Total SDRAM size	Bit places shifted (bit)	Offset mode register address [1]	Offset Extended mode register address [2]
16 (1Mx16)	2 MB	9	0x04600	0x0100000
16 (2Mx8)	4 MB	10	0x08C00	0x0200000
64 (4Mx16)	8 MB	9	0x04600	0x0200000
64 (8Mx8)	16 MB	10	0x08C00	0x0800000
128 (8Mx16)	16 MB	10	0x08C00	0x0800000
128 (16Mx8)	32 MB	11	0x11800	0x0800000
256 (16Mx16)	32 MB	10	0x08C00	0x0800000
256 (32Mx8)	64 MB	11	0x11800	0x2000000
512 (32Mx16)	64 MB	11	0x11800	0x2000000
512 (64Mx8)	128 MB	12	0x23000	0x4000000

[1] Base address of the SDRAM is 0x30000000 for programming mode register.

[2] Base address of the SDRAM is 0x30000000 for programming extended mode register.

Example:
Initialization of the mode register of a low power SDRAM 256 Mbit (16M x 16) with BRC address mapping:

```
#define SDRAM0_BASE 0x30000000
#define SDRAM0 SDRAM0_BASE
int addr;
UInt32 *ptr = NULL;
addr = SDRAM0 + 0x8C00; //see Table above
ptr = (UInt32 *) addr;
```

6.7 MPMC_testmode1 register configuration by measurement

MPMC_testmode1 should be set to 0x20 (when base frequency is 24 MHz).

When the MCPC is configured for a certain SDRAM, the duration of the SDRAM refresh period can be measured using the following method:

1. Enable the clock-gating of the SDRAM (bits 0 and 1 of the MpmcDyCntl then should be '0')
2. Perform the SDRAM initialization, without any clock-initialization
3. Stop the system
4. Now measure the MPMC_clkout pin using an oscilloscope and trigger it do a one-shot measurement
5. Count the amount of clock-cycles used by the SDRAM refresh

This needs to be measured once only for a specific type of SDRAM:

- Check MPMC setting with sdram configuration used.

Now: mpmc_testmode1= <amount of measured clocks> * fracdiv_setting_highspeed.

Remark: The fractional divider_setting_highspeed determines how much faster the base runs than the AHB clock. The 'amount of measured clocks' are AHB cycles.

1. Introduction

The EBI module acts as multiplexer with arbitration between the NAND flash and the SDRAM/SRAM memory modules connected externally through the MPMC.

The main purpose of using the EBI module is to save external pins. However only data and address pins are multiplexed. Control signals towards and from the external memory devices are not multiplexed.

1.1 Feature list

- Multiplexing of 16 bit data and 16 bit address signals.
- Two ports of three ports are connected to support MPMC and NAND flash.
- Request, Grant, and Back off mechanism is used for arbitration.
- In case of equal priority and simultaneous requests, a round-robin scheme is used.
- Priority can be set per port via software.

2. General description

2.1 Block diagram

Figure 4–10, shows the block diagram of the EBI module with all connected modules.

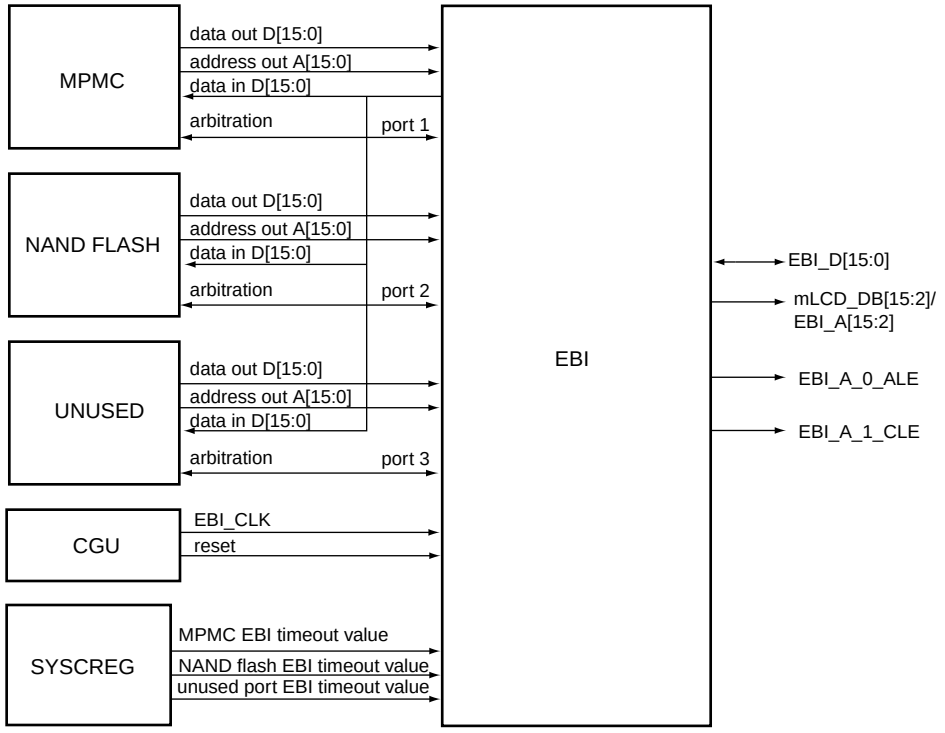


Fig 10. EBI block diagram

2.2 Interface description

2.2.1 Clock Signals

Table 4–64 gives detailed information about the clock that is connected to the EBI module.

Table 64. EBI Module Clock Overview

Clock Name	I/O	Source/ Destination	Description
EBI_CLK	I	CGU	Main Clock off the module. This clock should run on the same clock as the highest clock on which the external memory controllers are running.

When external memory controllers run on different clock frequencies following restriction should be applied:

- All of the clocks must be synchronous and an integer multiple of each other.
- The fastest clock should also be connected to the EBI_CLK.

2.2.2 Reset Signals

The CGU creates an asynchronous low-active reset signal (nPOR) that resets the logic in the EBI_CLK domain.

2.2.3 External memory controller interface signals

In total, three external memory controller interface ports are available on the EBI module. On the LPC3130/31 two ports are used: port 1 for the MPMC and port 2 for the NAND flash controller. Port 3 is not connected.

2.2.4 External Pin Connections

Table 4–65 shows all external pin connections towards and from external memory devices.

Table 65. EBI external pin connections

Name	Type	Reset Value	Description
mLCD_A[15:2]/ EBI_A[15:2]	O	-	16 bit address output towards all connected external memories.
EBI_A_0_ALE	O	-	
EBI_A_1_CLE	O	-	
EBI_D[15:0]	I/O	-	16 bit data towards and from all connected external memories.

3. Register overview

The EBITIMEOUTVALUE signals for ports one to three signals are connected to software registers that reside in the SysCReg module. Section 4–4.2 will describe in more detail about the usage of the EBITIMEOUTVALUE registers/ports.

4. Functional description

4.1 Arbitration

The arbitration inside of the EBI module is explained in the example that follows.

An external memory controller 1 can indicate via its EBIREQ1 signal that it needs external bus access. The EBI module will wait until the currently granted external memory controller 2 is finished, by looking to its EBIREQ2 signal to go low. After that the external memory controller 1, which requested external bus access will be granted access to the external bus via its EBIGNT1 signal. In case the requesting external memory controller 1 has a higher priority then the already granted external memory controller 2, the back off mechanism is used. An EBIBACKOFF2 signal is sent to external memory controller 2, which indicates that external memory controller 2 should end its current external bus access as soon as possible, by making EBIREQ2 low. In that case, EBIGNT2 can go low, and EBIGNT1 can go high.

See Figure 4–11.

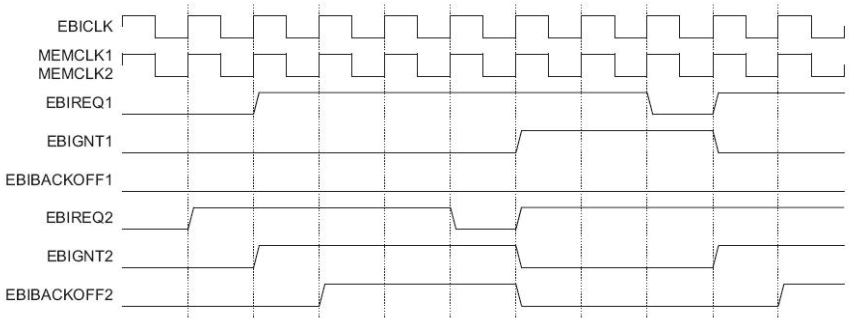


Fig 11. EBI Arbitration Diagram

4.2 Priority

The EBITIMEOUTVALUE is used to set the priority. When an external memory controller 1 is producing a request, its EBITIMEOUTVALUE is loaded in a register. When the counter reaches zero, the BACKOFF signal is sent to the external memory controller that currently occupies the external bus. When more requests are sent but the EBITIMEOUTVALUE settings are different, the counter that first reaches zero is given the highest priority so gets its EBIGNT signal set to access the bus. When priorities are equal, a round-robin mechanism is used.

4.3 Clock restrictions

All clocks shall be synchronous, and an integer multiple of each other. The fastest clock should also be connected to the EBICLK. The example below indicates that EBICLK is equal to MEMCLK1. MEMCLK2 is equal to half of the EBICLK frequency illustrates what is described above.

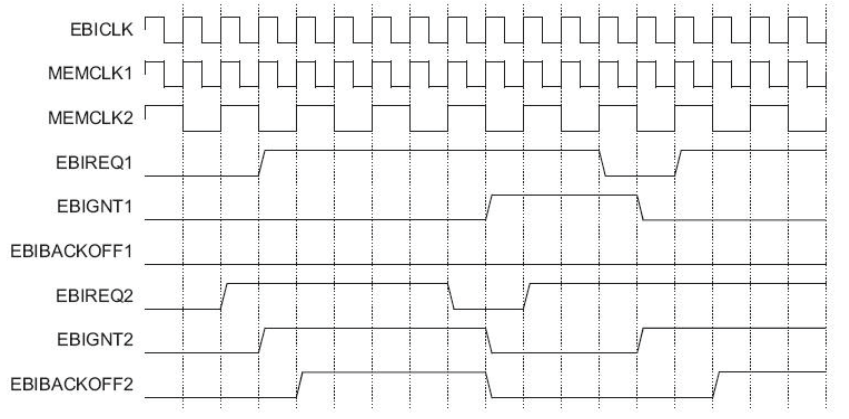


Fig 12. Timing Restrictions

5. Power optimization

The EBI module has clock gating inserted via synthesis.

6. Programming guide

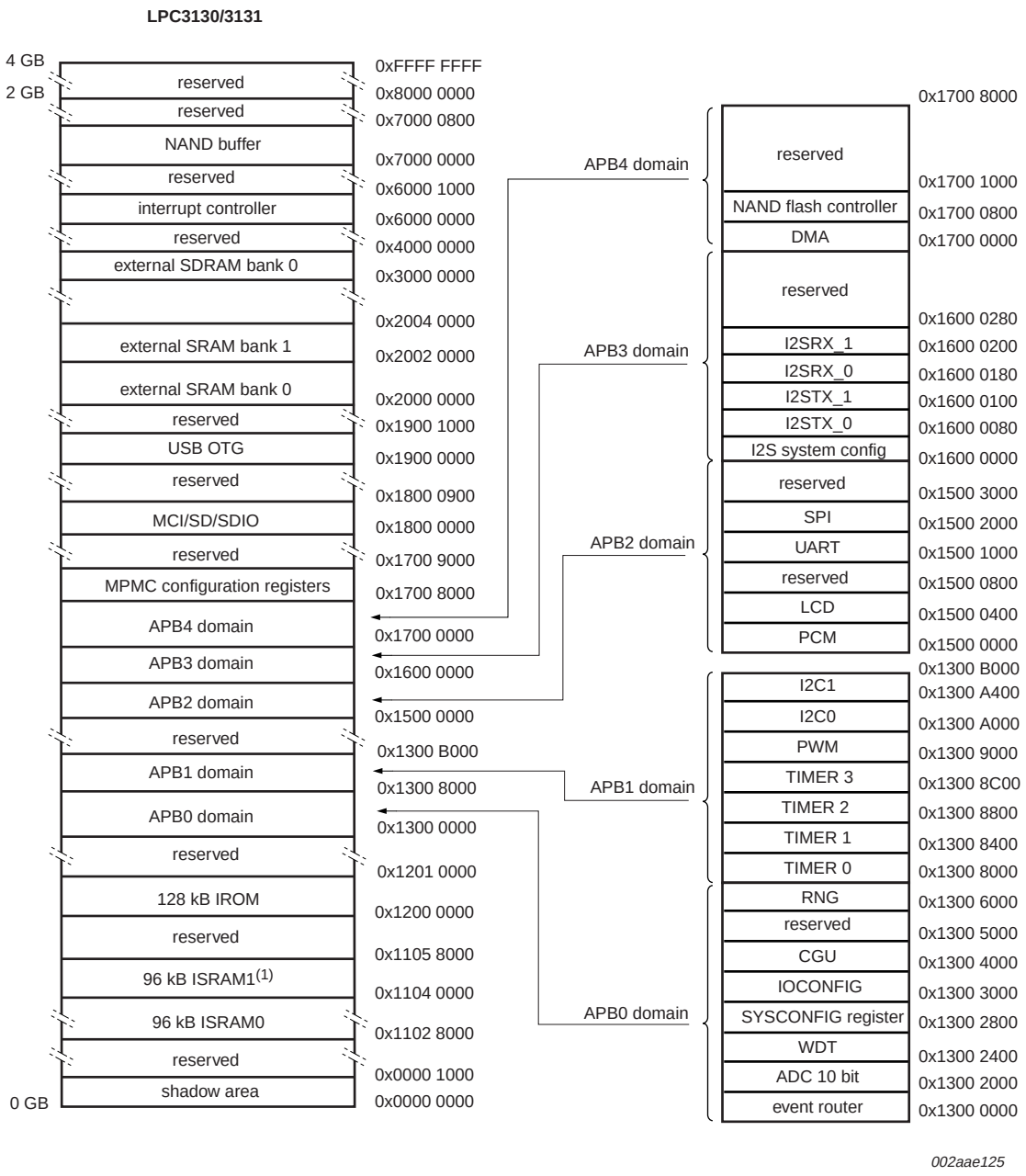
The EBI module has an internal arbitration mechanism, which does not need any programming. The only thing which can be programmed, is the priority of the different ports. This can be done by programming values in the EBITIMEOUTVALUE1..3 registers, which reside in the SYSCREG module. The lower the programmed value is, the higher the priority of that port is. By default in the LPC3130/31 port 1 (MPMC) has the highest priority. Port 2 (NAND flash) and 3 (not used) have equal priority.

1. Introduction

The memory map provides information about the memory address space of all internal registers and memory definitions for both internal and external memories. For more detailed information use the module chapters of this document.

Table 66. General address map

Module	Max Address Space		Data Width	Device Size	Remark
Shadow Space	0x0000 0000	0x0000 0FFF		4 kByte	
Internal SRAM 0	0x1102 8000	0x1103 FFFF	32 bit	96 kByte	
Internal SRAM 1 (LPC3131 only)	0x1104 0000	0x1105 7FFF	32 bit	96 kByte	
Internal SROM 0 EROM 0	0x1200 0000	0x1200 FFFF	32 bit	128 kByte	
NANDFLASH Ctrl	0x7000 0000	0x7000 07FF	32 bit	2 kByte	
External SRAM 0	0x2000 0000	0x2000 FFFF	8 bit	64 kByte	When MPMCSTCS0 is configured for 8bit device.
-	0x2000 0000	0x2001 FFFF	16 bit	128 kByte	When MPMCSTCS0 is configured for 16bit device.
External SRAM 1	0x2002 0000	0x2002 FFFF	8 bit	64 kByte	When MPMCSTCS1 is configured for 8bit device.
-	0x2002 0000	0x2003 FFFF	16 bit	128 kByte	When MPMCSTCS1 is configured for 16bit device.
External SDRAM 0	0x3000 0000	0x37FF FFFF	16 bit	128 MByte	
Peripherals	0x1300 0000	0x1300 7FFF	32 bits	32 kByte	APB0
	0x1300 8000	0x1300 BFFF	32 bits	16 kByte	APB1
	0x1500 0000	0x1500 3FFF	32 bits	16 kByte	APB2
	0x1600 0000	0x1600 03FF	32 bits	1 kByte	APB3
	0x1700 0000	0x1700 0FFF	32 bits	4 kByte	APB4
	0x1700 8000	0x1700 8FFF	32 bits	4 kByte	MPMC cfg
	0x1800 0000	0x1800 03FF	32 bits	1 kByte	MCI
	0x1900 0000	0x1900 0FFF	32 bits	4 kByte	USB OTG
	0x6000 0000	0x6000 0FFF	32 bits	4 kByte	Interrupt controller



(1) LPC3131 only.

Fig 13. LPC3130/31 memory map

1. Introduction

The internal ROM memory is used to store the boot code of the LPC3130/3131. After a reset, the ARM processor will start its code execution from this memory.

1.1 Feature list

The LPC3130/31 has the following features:

- Supports booting from SPI flash, NAND flash, SD/SDHC/MMC cards, UART, and USB (DFU class) interfaces.
- Supports option to perform CRC32 checking on the boot image
- Supports booting from managed NAND devices such as moviNAND, iNAND, eMMC-NAND and eSD-NAND using SD/MMC boot mode.
- Contains pre-defined MMU table (16 kB) for simple systems.
- Full implementation of AHB protocol compliant to AMBA specification (Rev 2.0).
- Configurable latency (0, 1, 2 AHB wait states) through SYSCREG_ISROM_LATENCY_CFG (address 0x1300 2860) register (see Table 26–535).
- ROM capacity of 128 kB.

2. General description

2.1 Interface description

2.1.1 Clock signals

The CGU will provides the clock for the ISROM module (see Table 6–67).

Table 67. ISROM module clock overview.

Clock name	I/O	Source/ destination	Max Freq	Description
ISROM_CLK	I	CGU	75MHz.	Main clock of the module - runs all internal logic.

2.1.2 Reset signals

The CGU provides an asynchronous low-active reset (AHB_RST_N) which resets the logic in the ISROM_CLK clock domain.

2.1.3 DMA Transfers

The ISROM module does not make use of flow control but is able to make use of DMA via the DMA module.

3. Register overview

The ISROM latency configuration registers reside in the SysCReg module (see Section 26–4.5).

4. Functional description

All of the ARM cores are configured to start executing the code, upon reset, with the program counter being set to the value 0x00000000. The design of LPC3130/31 is such that the first 4 KB page of the ROM (starting at 0x1200 0000) is shadowed upon reset over the first 4 KB page of the address space of the processor. This ensures that the first code executed in the system is the boot code of the ROM. The boot code starts with position independent set of instructions that branches the execution of the code to the address space occupied by the ROM, thus removing the limitation of 4 KB for the code size.

The boot ROM determines the boot mode based on reset state of GPIO0, GPIO1, and GPIO2 pins. Table 8 shows the various boot modes supported on the LPC3130/3131.

Table 68. LPC3130/31 boot modes

Boot mode	GPIO0	GPIO1	GPIO2	Description
NAND	0	0	0	Boots from NAND flash. If proper image is not found, boot ROM will switch to DFU boot mode.
SPI	0	0	1	Boot from SPI NOR flash connected to SPI_CS_OUT0. If proper image is not found, boot ROM will switch to DFU boot mode.
DFU	0	1	0	Device boots via USB using DFU class specification.
SD/MMC	0	1	1	Boot ROM searches all the partitions on the SD/MMC/SDHC/MMC+/eMMC/eSD card for boot image. If partition table is missing, it will start searching from sector 0. A valid image is said to be found if a valid image header is found, followed by a valid image. If a proper image is not found, boot ROM will switch to DFU boot mode.
Reserved 0	1	0	0	Reserved for testing.
NOR flash	1	0	1	Boot from parallel NOR flash connected to EBI_NSTCS_1.
UART	1	1	0	Boot ROM tries to download boot image from UART ((115200 – 8 – n -1) assuming 12MHz FFAST clock).
Test	1	1	1	Boot ROM is testing ISRAM using memory pattern test. After test switches to UART boot mode.

4.1 Boot process

LPC3130/31 top level boot process is illustrated in Figure 6–14. As shown in the picture the boot ROM determines the boot mode based on the reset state of the pins GPIO0, GPIO1 and GPIO2. The boot ROM indicates any error during boot process by toggling GPIO2 pin hence it is advised to connect this pin to a LED to get visual feedback. Boot ROM copies/downloads the image to internal SRAM at location 0x1102 9000 and jumps to that location (sets ARM's program counter register to 0x1102 9000) after image verification. Hence the images for LPC3130/31 should be compiled with entry point at 0x1102 9000.

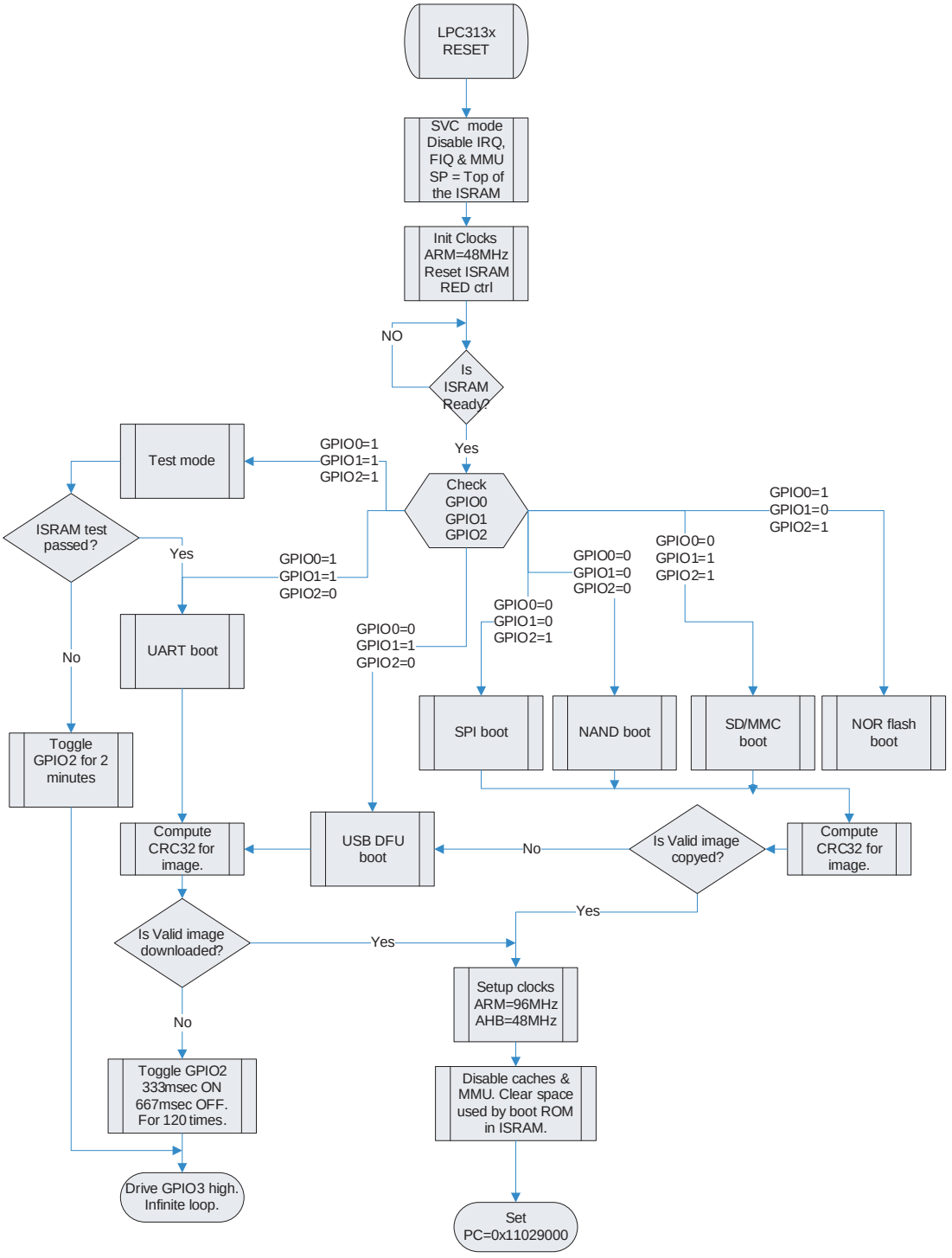


Fig 14. LPC3130/31 boot process flow chart

4.2 Boot image format

LPC3130/31 boot ROM expects the boot image be compiled with entry point at 0x1102 9000 and has the layout described in the Table 6–69 (except for “Parallel NOR flash” boot mode).

Table 69. Image format

Field Name	Offset	Size in bytes	Description
Image Header			
vector	0x00	4	Valid ARM instruction. Usually this will be a branch instruction to entry point of the image.
magic	0x04	4	This field is used by boot ROM to detect a valid image header. This field should always be set to 0x41676d69.
execution_crc32	0x08	4	CRC32 value of execution part of the image. If the ‘image_type’ is set to ‘0xA’, this field is ignored by boot ROM.
Reserved0	0x0C	16	Should be zero.
imageType	0x1C	4	Specifies whether CRC check should be done on the image or not. 0xA – No CRC check required. 0xB – Do CRC32 check on both header and execution part of the image.
imageLength	0x20	4	Total image length including header.
releaseID	0x24	4	Release or version number of the image. Note, this field is not used by boot ROM but is provided to track the image versions.
buildTime	0x28	4	Time (expressed in EPOC time format) at which image is built. Note, this field is not used by boot ROM but is provided to track the image versions.
sbzBootParameter	0x2C	4	Should be zero.
cust_reserved	0x68	60	Reserved for customer use.
header_crc32	0x6C	4	CRC32 value of the header (bytes 0x00 to 0x6C of the image). If the ‘image_type’ is set to ‘0xA’, this field is ignored by boot ROM.
Reserved1	0x70	16	Should be zero.
Execution Part			
Program code	0x80	Max. 128 kB	Program code. The maximum size of the image allowed by boot ROM is 128 kB (including header). For LPC3130 restrict the size to 80 kB.

4.3 NAND boot mode

Figure “NAND boot flow” details the boot-flow steps of the NAND boot mode. As already mentioned, the execution of this mode happens only if the mode pins had proper value on reset (see Table 6–68).

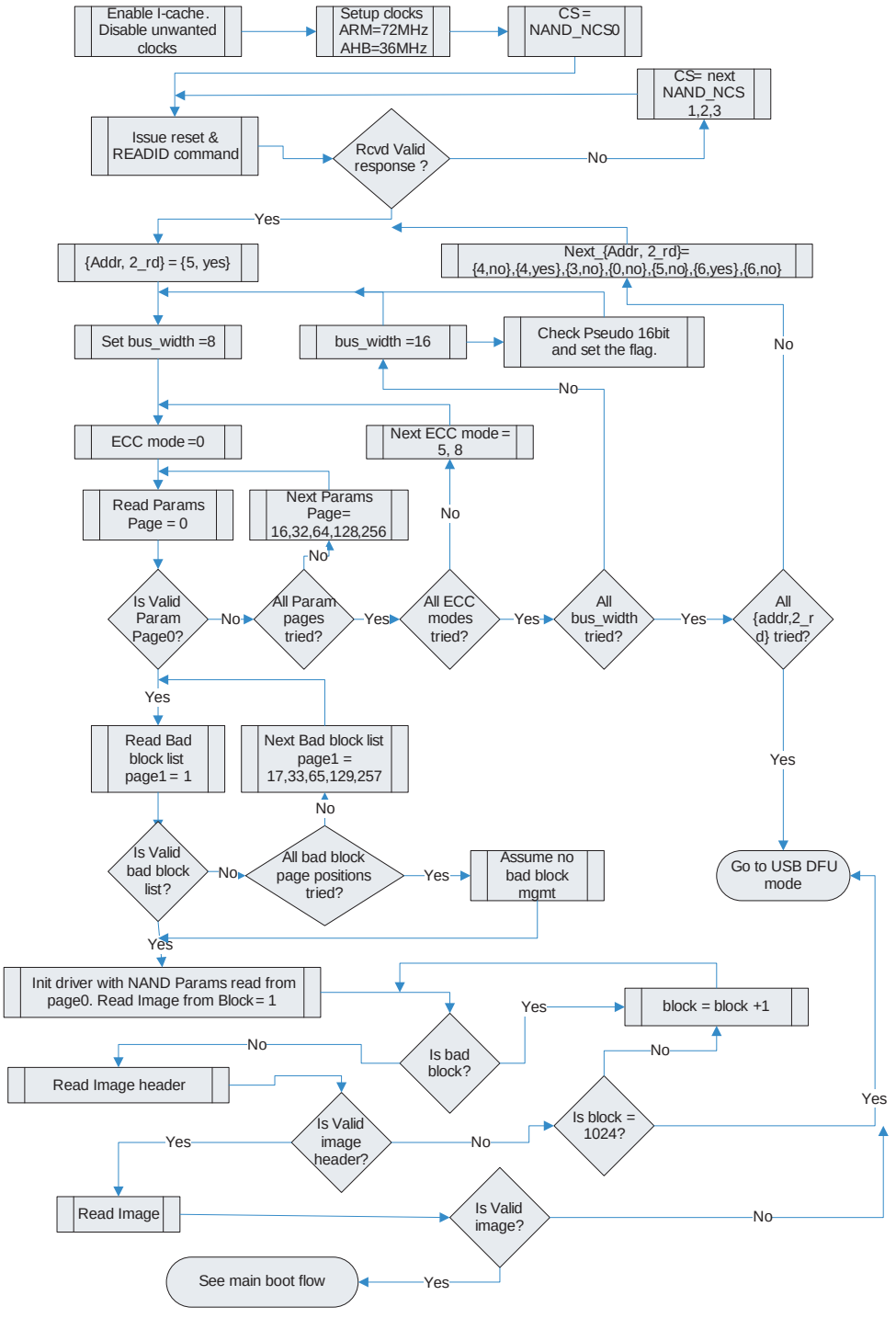


Fig 15. NAND boot flow

4.3.1 NAND parameters

The boot ROM expects the NAND flash device settings and bad block list information is written in block zero. The LPC3130/31 boot ROM defines NAND flash parameters page as below:

Table 70. NAND flash parameters

Field Name	Offset	Size in bytes	Description
tag	0x00	8	Parameter page marker. This field should always be set as ASCII string "NANDflsh".
interface_width	0x08	1	This field should be set to 0x10 when 16 bit NAND device is connected. Boot ROM assumes 8-bit NAND device if any other value is set in this field.
reserved	0x09	1	Should be zero.
page_size_in_bytes	0x0A	2	Page size in bytes. For small page NAND flash set to 512.
page_size_in_32bit_words	0x0C	2	Page size in number of 32 bit words. For small page NAND flash set to 128.
pages_per_block	0x0E	2	Number of pages per block.
nbr_of_blocks	0x10	4	Number of block present on device.
amount_of_address_bytes	0x14	1	Number of page address cycles expected by device during read/program operations.
amount_of_erase_address_bytes	0x15	1	Number of address cycles expected by device during erase operation.
support_read_terminate	0x16	1	Set non-zero value for devices which require 2nd read command (30h) cycle.
page_increment_byte_nr	0x17	1	Number of dummy address cycles. Boot ROM sends 0 address for these many address cycles before sending page address cycles.
device_name	0x18	40	User defined ASCII string.
timing1	0x40	4	Value to be set in NandTiming1 register.
timing2	0x44	4	Value to be set in NandTiming2 register.
ecc_mode	0x48	1	Set 0 for no hardware ECC. Set 5 to use 5 bit ECC corrector. Set 8 to use 8 bit ECC corrector. Other values are ignored.
Reserved	0x49	3	Should be zero.
User_def	0x4C	176	User defined values.
CRC32	0xFC	4	CRC32 value of the above defined structure.

The LPC3130/31 boot ROM defines NAND flash bad block list page as below:

Table 71. Bad block list page

Field Name	Offset	Size in bytes	Description
Page 1			
Bad_block_list_size	0x00	4	Size of bad block list
bad block number	0x04	4	First bad block number
bad block number	0x08	4	Second bad block number

Table 71. Bad block list page *…continued*

Field Name	Offset	Size in bytes	Description
…			
Marker	0x1F8	3	Bad page marker. This field should always be set as ASCII string "BAD".
Bad_page_nr	0x1FB	1	Page number. Value = 1.
CRC32	0x1FC	4	CRC32 value of the current page excluding the current word. I.e. CRC32 of bytes 0 to 508 of this page.
Page 2			
bad block number	0x00	4	126th bad block number
bad block number	0x04	4	127th bad block number
…			
Marker	0x1F8	3	Bad page marker. This field should always be set as ASCII string "BAD".
Bad_page_nr	0x1FB	1	Page number. Value = 2.
CRC32	0x1FC	4	CRC32 value of the current page excluding the current word. I.e. CRC32 of bytes 0 to 508 of this page.
Last bad block Page n			
bad block number	0x00	4	(125 + ((n-1) * 125) + 1) bad block number
bad block number	0x04	4	next bad block number
Marker	0x08	3	Bad page marker. This field should always be set as ASCII string "BAD".
Bad_page_nr	0x0B	1	Page number. Value = n.
CRC32	0x0C	4	CRC32 value of the current page excluding the current word. I.e. CRC32 of bytes 0 to 12 of this page.

In the above table, assumption is made that the last bad block page contained only 2 bad block numbers. If the number of bad blocks is less than 125 then the page 1 structure will be as shown below:

Table 72. Bad block list page (page 1)

Field Name	Offset	Size in bytes	Description
Page 1			
Bad_block_list_size	0x00	4	Size of bad block list
bad block number	0x04	4	First bad block number
bad block number	0x08	4	Second bad block number
…			

Table 72. Bad block list page (page 1) *…continued*

Field Name	Offset	Size in bytes	Description
Marker	0xYY (where 0xYY is less than 0x1F8)	3	Bad page marker. This field should always be set as ASCII string "BAD".
Bad_page_nr	0xYY + 3	1	Page number. Value = 1.
CRC32	0xYY + 4	4	CRC32 value of the current page excluding the current word. I.e. CRC32 of bytes 0 to (0xYY+4) of this page.

Since boot ROM needs to know the number of address cycles, chip select, timings, device bus width, page size etc. to read the parameter page it employs the following pseudo-code algorithm to auto detect and read the NAND parameter page. Once it reads the parameter page it uses the values configured in that page to access the NAND device then onwards.

1. For each chip select (NAND_NCS_0 to NAND_NCS_3) repeat steps 2 to 12.
2. Determine if it is a pseudo 16-bit setup. A pseudo 16 bit setup is defined as connecting two identical 8-bit wide NAND devices (same manufacturer & same product type) in parallel to LPC3130/31 to create a 16-bit wide device. Boot ROM sends READID command on both upper & lower byte of 16 bit NAND bus and checks the response received on both upper byte and lower byte of the bus are identical.
3. If the device is pseudo 16 bit now onwards repeat command and address bytes on both upper and lower byte of the bus. As far as data is concerned treat the device as a 16-bit wide device.
4. If no response is received check the presence of device on next chip select.
5. Initially assume 8 bit wide device and do the following steps.
6. Access the NAND device assuming number of address cycles & requires 2nd read command (30h) cycle in the following order: {address cycles, Command 30h required} {5,yes},{4,no},{4,yes},{3,no},{0,no},{5,no},{6,yes},{6,no}.
7. Issue reset to NAND device every time the access method is changed.
8. Read page with hardware ECC check disabled.
9. Check whether the page has parameter information. Verification involves checking the presence of parameter marker (ASCII string "NANDflsh") at offset 0x00 and also the CRC32 of the parameter page matches the value set at offset 0xFC.
10. Repeat read page (steps 7 & 8) with hardware ECC set to 5 bit mode and then 8 bit mode.
11. Repeat steps 7 to 9 until a valid parameter page is found using following page indexes: 0, 16, 32, 64, 128, 256.
12. If no valid parameter page is found repeat steps 5 to 10 assuming 16 bit device.

Once the boot ROM reads the parameter page it employs the following algorithm to read bad block list.

1. Repeat the following steps 2 to 3 until valid bad block list is found using current page index as: 1, 17, 33, 65 and 257.
2. Check the page has valid bad block “page 1” information. See “Bad block list pages” table for more information on page structure.
3. Read next pages until all bad block numbers are read. Note, the first page contains the “Bad_block_list_size” field which tells boot ROM how many blocks are marked bad on this device.

4.3.2 Search for the valid NAND flash executable image

The first step in the execution of the image from the NAND flash is the search for it. The search for the valid image starts at the block 1 of the NAND flash. Block 0 of the NAND flash is filled in with the information about initial bad blocks and information about the geometry of the NAND flash itself, used during the initialization of the ROM based NAND flash driver. This can be seen in Figure 6–16.

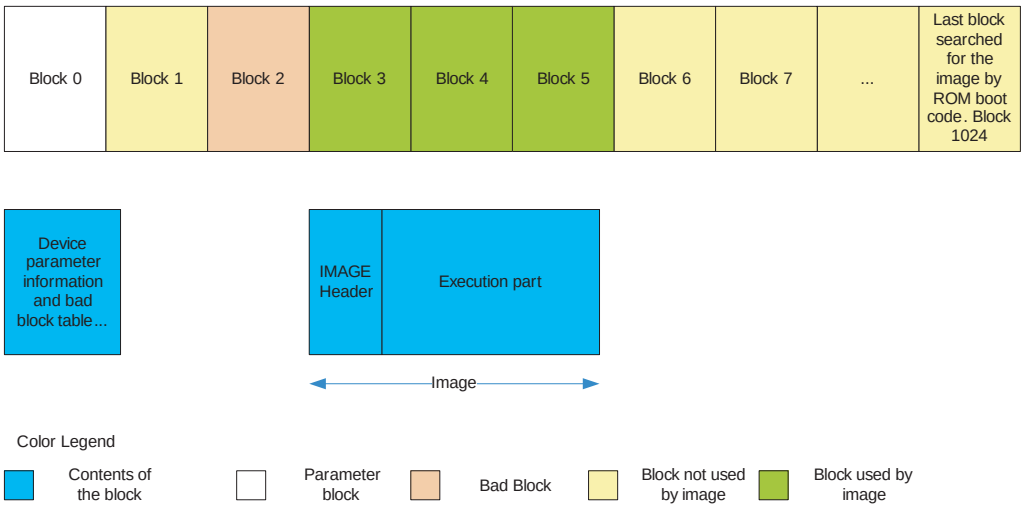


Fig 16. NAND search

It is recommended that the application which stored the image on the NAND flash, should make sure that it occupies a contiguous set of good physical blocks, so there is no need for the bad block management scheme to be implemented in the ROM boot. Boot ROM skips bad blocks as long as the bad block is listed in the bad block list page. In the example in Figure 6–16 the image occupies 3 blocks, and the first free set of the contiguous 3 blocks are blocks 3 to 5. The search process tries to find appropriate header in block 1 and fails. The header check is started by loading 128 bytes from the beginning of the current block (block 1 in this case) and checking the magic word first. There is a high probability that the block that does not contain header has an incorrect magic word value, so no more time is spent on this block and next block can be searched. If the magic word value was correct, the challenging of the header continues by calculating the CRC32 over header scope and comparing it against the one stored in the header. The probability of an accidental match is quite small. Further CRC32 check is done on the execution part of the image, that the calculated CRC32 matches the one that was stored in the header.

In the example illustrated in Figure 6–16, the normal course of action would be the failed search starting from the block 1, failed search starting from the block 2 and successful search starting from the block 3.

The LPC3130/31 boot ROM searches for valid image starting from Block 1 to block 1024 (if present on the device). If a bad block is present in between the image blocks and the block is listed in bad block list page, the boot ROM skips that block and assumes the consecutive blocks have the rest of the image.

4.4 SPI NOR-flash boot mode

Figure 6–17 details the boot-flow steps of the SPI NOR-flash boot mode. As already mentioned, the execution of this mode happens only if the mode pins had proper value on reset (see Boot modes Table 6–68).

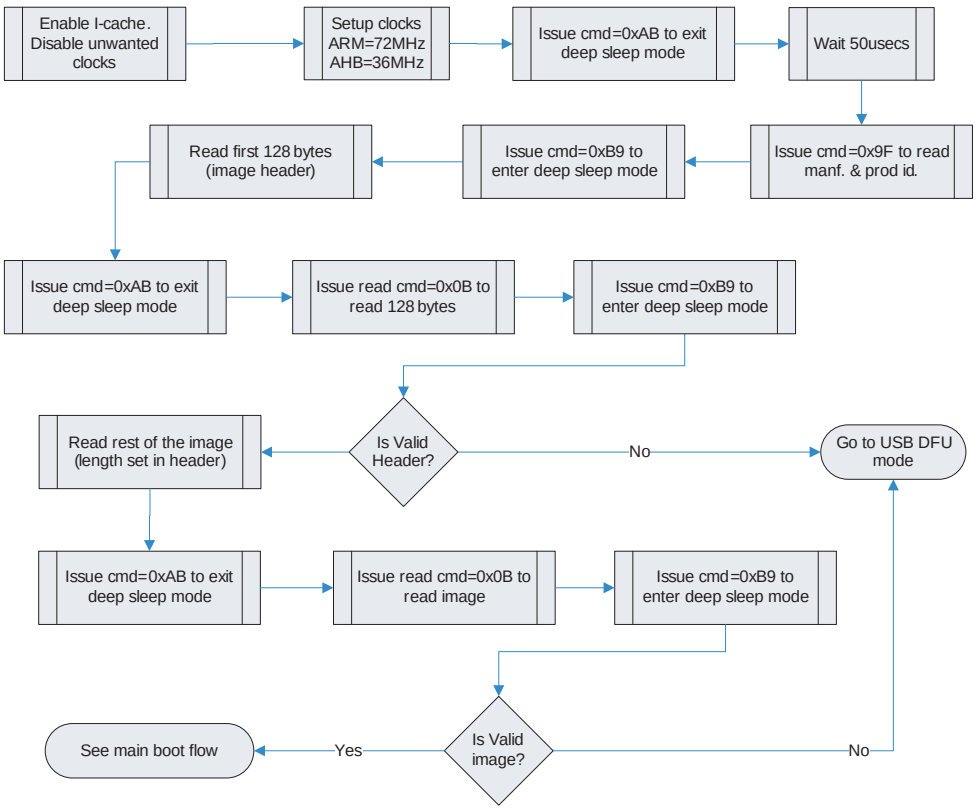


Fig 17. SPI boot flow

As illustrated in the figure Figure 6–17, for LPC3130/31 boot ROM to support a SPI NOR-flash boot, the device should support “High frequency continuous array read” (command 0x0B). Since boot ROM doesn’t rely on response for commands 0xAB, 0xB9 and 0x9F, as long as the SPI devices ignore or respond correctly to these command LPC3130/31 should be able to boot from them.

4.5 DFU boot mode

Device Firmware Upgrade (DFU) is a USB class specification defined by USB.org. LPC3130/31 boot ROM uses this class specification to implement USB boot mode. Figure Figure 6–18 details the boot-flow steps of the USB boot mode.

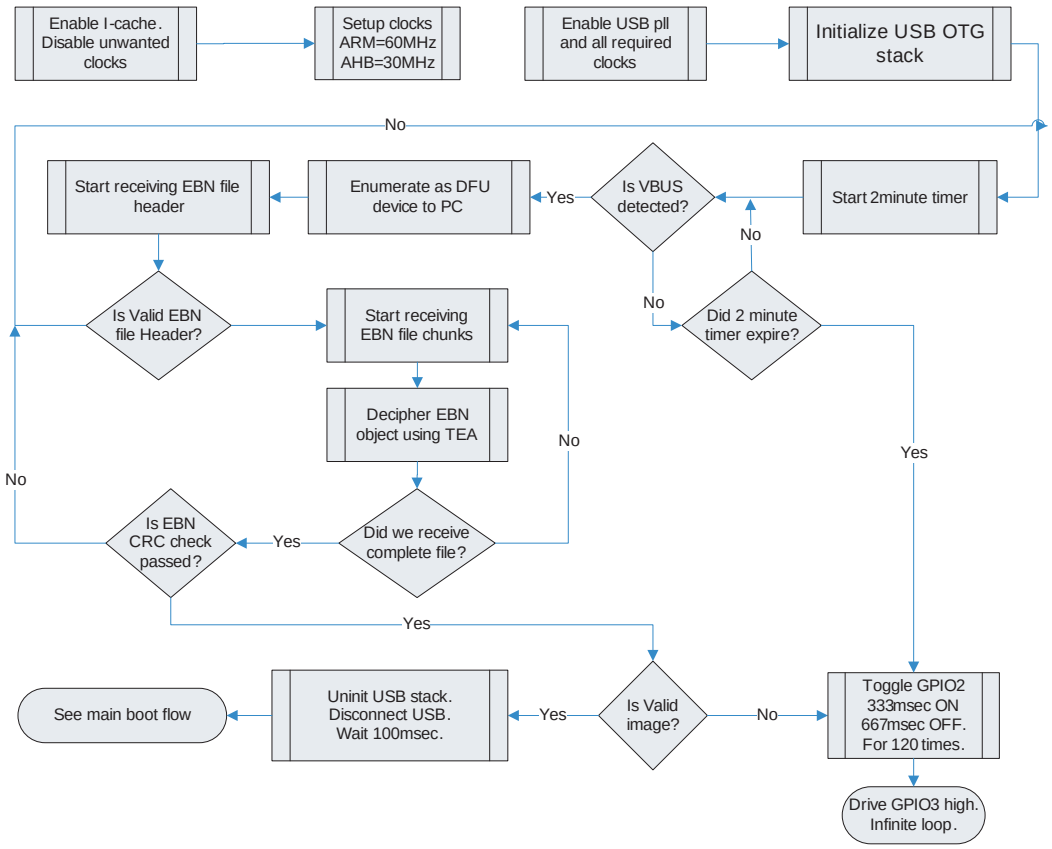


Fig 18. USB DFU boot mode

The boot flows in following steps through the DFU mode:

1. Setup the clocks for DFU mode.
2. Initialize USB OTG stack that implements DFU class.
3. Enable and configure any additional necessary hardware (i.e. supply voltage must be guaranteed to be greater than 3.1 V).
4. Wait for the detection of the connection of the USB host. This is implemented as a polling loop that blocks the execution until condition USB connected is read. When the VBUS of the USB is not detected within 2 minute, the boot ROM indicates error by toggling GPIO2 for 2 minutes and then driving GPIO3 high.
5. Start downloading the first header of the EBN file. The EBN file is a collection of objects. Each object consists of the header and the TEA (Tiny Encryption Algorithm) encrypted image. Apart from the size of the image, the header indicates the TEA key offset and the 32b CRC of the contained image. If the EBN header indicates the size of the object is larger than 128 KB, an error is signaled to the USB host and the execution is returned to the top, waiting for the USB connection.

6. Download the rest of the EBN image and perform TEA decryption using the key indicated by the key offset in the EBN header.
7. Calculate CRC32 checksum value of the TEA decrypted image and compare it against the value stored in the EBN header. When the hash values do not match, the error is signaled to the host and the execution is returned to the top, waiting for the USB connection.
8. Validate the header of the execution image, using header CRC32 checksum value (only done if image_type is set to 0xB), checking the magic values, size and image type indicator.
9. Calculate the CRC32 checksum value for execution part of the image (only done if image_type is set to 0xB) and compare it against the value stored in execution image header. If the calculated and stored checksum values do not match an error is signaled to the USB host and the execution is returned to the top, waiting for the USB connection.

4.6 SD/MMC boot mode

Figure “SD/MMC boot flow” details the boot-flow steps of the SD/MMC boot mode. As already mentioned, the execution of this mode happens only if the mode pins had proper value on reset (see Boot modes Table 6–68).

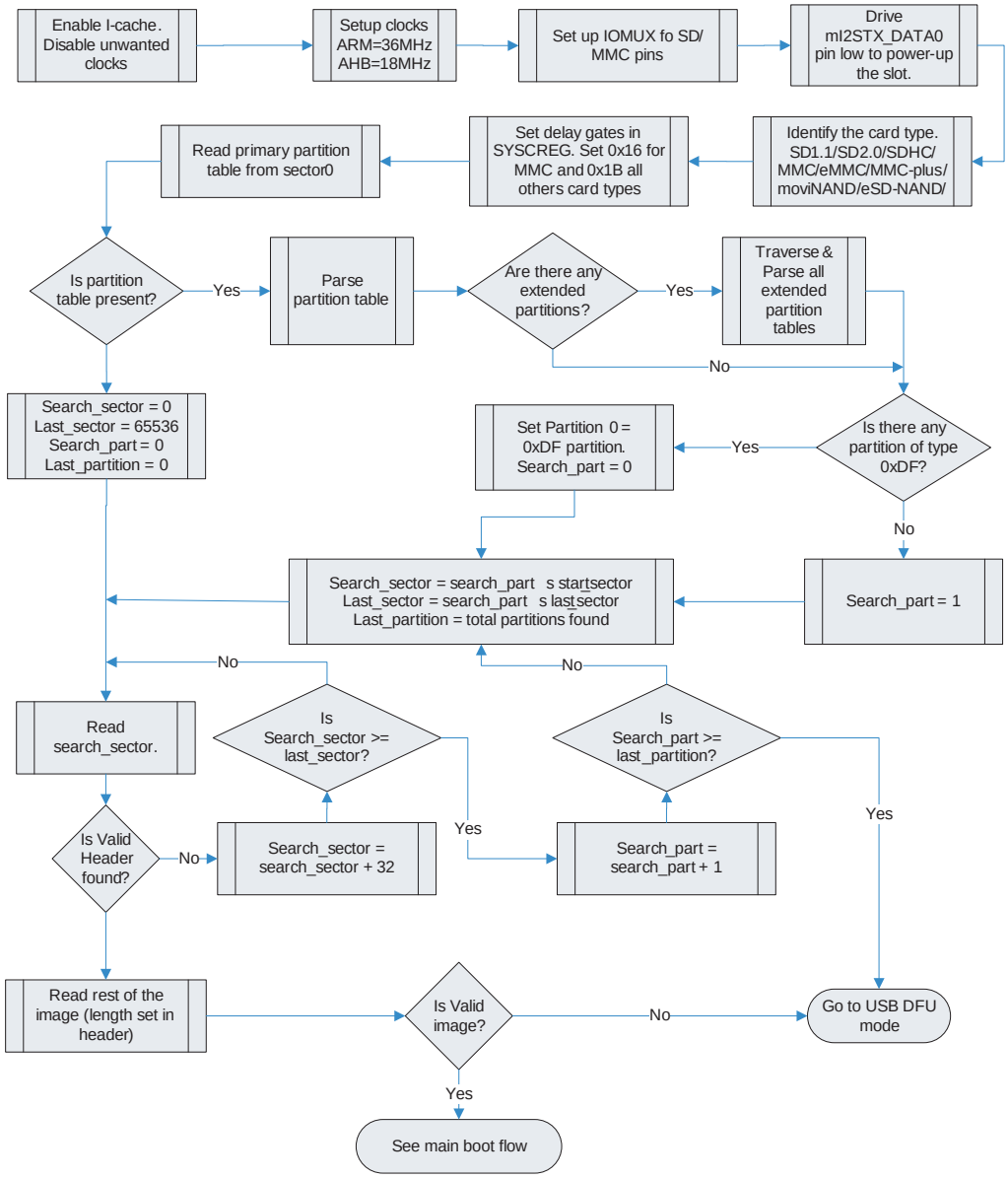


Fig 19. SD/MMC boot flow

As illustrated in Figure 6–19, boot ROM supports parsing of partition tables on the card. The boot ROM doesn't have any knowledge of file system. Hence to boot from SD/MMC cards the user has to create "0xDF" partition and copy the boot image directly to the raw sectors of that partition.

As shown creation of partition type "0xDF" is not a compulsory requirement but, it speeds up the search process. A user could create any other partition type and dump the boot image to that partition.

LPC3130/31 boot ROM deploys a comprehensive card detection process to detect MMC, eMMC, SD1.1, SD2.0, SDHC, eSD, managed-NAND and moviNAND devices.

LPC3130/31 boot ROM interacts with memory cards in 1-bit bus mode only.

4.7 UART boot mode

Figure 6–20 details the boot-flow steps of the UART boot mode. As already mentioned, the execution of this mode happens only if the mode pins had proper value on reset (see Boot modes Table 6–68).

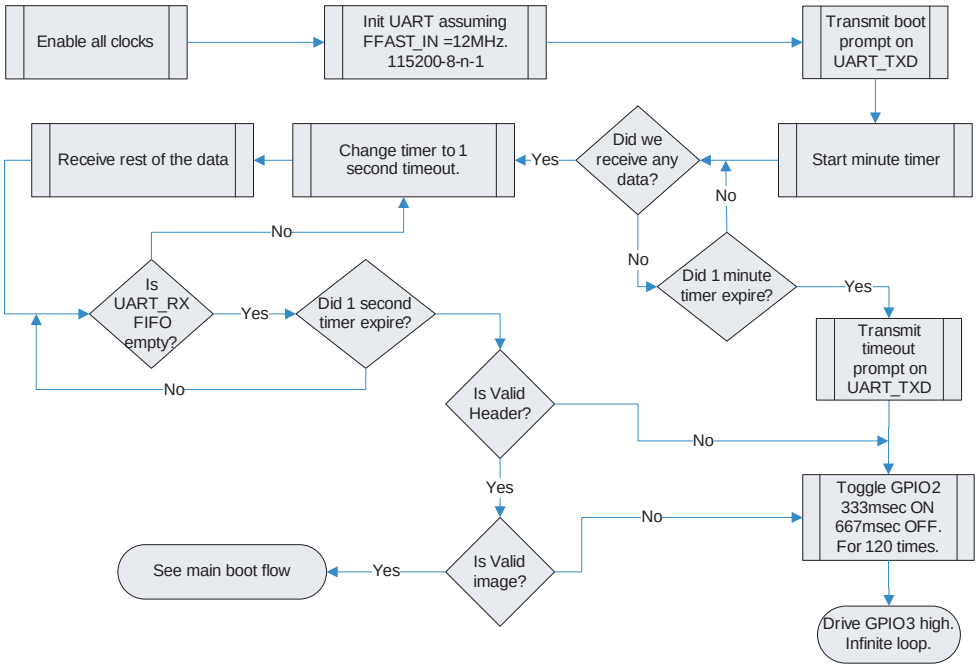


Fig 20. UART boot mode

As illustrated in Figure 6–20 configure UART with following settings:

- Baud rate = 115200 (UART divisor registers are programmed 12MHz crystal frequency)
- Data bits = 8
- Parity = None
- Stop bits = 1

The boot ROM doesn't implement any flow control or any handshake mechanisms during file transfer. Hence it is recommended to create CRC check images (image_type set to 0xB) for UART boot.

4.8 Parallel NOR-flash boot mode

Unlike other boot modes "NOR-flash" boot mode uses simplified image header. When this boot mode is selected boot ROM reads EBI_NSTCS_1 chip select area in 16 bit mode. The wait states for the interface are set to default values. See MPMC chapter for details. In this boot mode boot ROM copies the image from NOR-flash to ISRAM0 (location 0x1102 9000) and jumps to that location (sets the program counter of ARM to 0x1102 9000). Also no CRC check is done to validate the image.

LPC3130/31 boot ROM expects the boot image be compiled with entry point at 0x1102 9000 and has the layout described in the table "NOR Image format".

Table 73. NOR image format

Field Name	Offset	Size in bytes	Description
Image Header			
vector	0x00	4	Valid ARM instruction. Usually this will be a branch instruction to entry point of the image.
magic	0x04	4	This field is used by boot ROM to detect a valid image header. This field should always be set to 0x3150F2E5.
imageLength	0x08	4	Total image length including header. Maximum allowed value for LPC3131 is 128 Kbytes and for LPC3130 is 80 Kbytes.

4.9 Test mode

LPC3130/31 boot ROM does memory pattern tests on ISRAM0 and ISRAM1. During the test if any errors are found the boot ROM will toggle GPIO2 pin continuously for 2 minutes and then drives GPIO3 pin high. See Figure 6–14 for details. If the memory test passes, the boot ROM changes to UART boot mode.

4.10 ISROM latency

The CPU can read boot-code from the ROM via the ISROM module. The CPU will address the ISROM module, which will translate the incoming AHB address in a ROM address. Based on this address the ROM will provide the ISROM module with data, which is stored on the given address. Then the ISROM module will transport the data read from the ROM to the CPU. By changing the latency through the memory controller less or more pipeline stages will be added. The more pipeline stages are used, the higher the frequency is which can be used, but the bigger the latency through the ISROM module is.

4.11 Built-in MMU table

ARM926EJS core requires memory management unit (MMU) to be initialized to make use of Data-cache and other memory protection functionality. For initializing MMU a translation table is required which defines section entries (Virtual to physical address mapping, cache enable, buffer enable, domain permission etc.). See ARM926EJS TRM for more details. The translation table has up to 4096 x 32-bit entries (total 16Kbytes of memory), each describing 1MB of virtual memory. This enables up to 4GB of virtual memory to be addressed. For systems which don't use any external memory (or systems which have tight memory requirement), LPC3130/31 boot ROM provides a pre-defined MMU translation table in its ISROM. This translation table is available at location 0x1201 C000 in ISROM.

Table 74. MMU translation table

Virtual Address Range	Size in bytes	Entry Type	Physical Address	C	B	Other settings
0x00000000	1M	Section entry	0x00000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x00100000	1M	COARSE TABLE	0x11057C00	-	-	Memory mapping defined by Coarse page table at 0x11057C00. Belongs to domain 0.
0x00200000	1M	COARSE TABLE	0x11057800	-	-	Memory mapping defined by Coarse page table at 0x11057800. Belongs to domain 0

Table 74. MMU translation table *…continued*

Virtual Address Range	Size in bytes	Entry Type	Physical Address	C	B	Other settings
0x00300000	1M	COARSE TABLE	0x11057400	-	-	Memory mapping defined by Coarse page table at 0x11057400. Belongs to domain 0
0x00400000	1M	COARSE TABLE	0x11057000	-	-	Memory mapping defined by Coarse page table at 0x11057000. Belongs to domain 0
0x00500000	1M	COARSE TABLE	0x11056C00	-	-	Memory mapping defined by Coarse page table at 0x11057C00. Belongs to domain 0
0x00600000	1M	COARSE TABLE	0x11056800	-	-	Memory mapping defined by Coarse page table at 0x11057800. Belongs to domain 0
0x00700000	1M	COARSE TABLE	0x11056400	-	-	Memory mapping defined by Coarse page table at 0x11057400. Belongs to domain 0
0x00800000	1M	COARSE TABLE	0x11056000	-	-	Memory mapping defined by Coarse page table at 0x11057000. Belongs to domain 0
0x00900000	1M	COARSE TABLE	0x11055C00	-	-	Memory mapping defined by Coarse page table at 0x11057C00. Belongs to domain 0
0x00A00000	1M	COARSE TABLE	0x11055800	-	-	Memory mapping defined by Coarse page table at 0x11057800. Belongs to domain 0
0x00B00000	1M	COARSE TABLE	0x11055400	-	-	Memory mapping defined by Coarse page table at 0x11057400. Belongs to domain 0
0x00C00000	1M	COARSE TABLE	0x11055000	-	-	Memory mapping defined by Coarse page table at 0x11057000. Belongs to domain 0
0x00D00000	1M	COARSE TABLE	0x11054C00	-	-	Memory mapping defined by Coarse page table at 0x11057C00. Belongs to domain 0
0x00E00000	1M	COARSE TABLE	0x11054800	-	-	Memory mapping defined by Coarse page table at 0x11057800. Belongs to domain 0
0x00F00000	1M	COARSE TABLE	0x11054400	-	-	Memory mapping defined by Coarse page table at 0x11057400. Belongs to domain 0
0x01000000	1M	COARSE TABLE	0x11054000	-	-	Memory mapping defined by Coarse page table at 0x11054000. Belongs to domain 0
0x01100000	1M	COARSE TABLE	0x11053C00	-	-	Memory mapping defined by Coarse page table at 0x11053C00. Belongs to domain 0.
0x01200000	1M	COARSE TABLE	0x11053800	-	-	Memory mapping defined by Coarse page table at 0x11053800. Belongs to domain 0
0x01300000	1M	COARSE TABLE	0x11053400	-	-	Memory mapping defined by Coarse page table at 0x11053400. Belongs to domain 0
0x01400000	1M	COARSE TABLE	0x11053000	-	-	Memory mapping defined by Coarse page table at 0x11053000. Belongs to domain 0
0x01500000	1M	COARSE TABLE	0x11052C00	-	-	Memory mapping defined by Coarse page table at 0x11052C00. Belongs to domain 0
0x01600000	1M	COARSE TABLE	0x11052800	-	-	Memory mapping defined by Coarse page table at 0x11052800. Belongs to domain 0
0x01700000	1M	COARSE TABLE	0x11052400	-	-	Memory mapping defined by Coarse page table at 0x11052400. Belongs to domain 0
0x01800000	1M	COARSE TABLE	0x11052000	-	-	Memory mapping defined by Coarse page table at 0x11052000. Belongs to domain 0

Table 74. MMU translation table *…continued*

Virtual Address Range	Size in bytes	Entry Type	Physical Address	C	B	Other settings
0x01900000	1M	COARSE TABLE	0x11051C00	-	-	Memory mapping defined by Coarse page table at 0x11051C00. Belongs to domain 0
0x01A00000	1M	COARSE TABLE	0x11051800	-	-	Memory mapping defined by Coarse page table at 0x11051800. Belongs to domain 0
0x01B00000	1M	COARSE TABLE	0x11051400	-	-	Memory mapping defined by Coarse page table at 0x11051400. Belongs to domain 0
0x01C00000	1M	COARSE TABLE	0x11051000	-	-	Memory mapping defined by Coarse page table at 0x11051000. Belongs to domain 0
0x01D00000	1M	COARSE TABLE	0x11050C00	-	-	Memory mapping defined by Coarse page table at 0x11050C00. Belongs to domain 0
0x01E00000	1M	COARSE TABLE	0x11050800	-	-	Memory mapping defined by Coarse page table at 0x11050800. Belongs to domain 0
0x01F00000	1M	COARSE TABLE	0x11050400	-	-	Memory mapping defined by Coarse page table at 0x11050400. Belongs to domain 0
0x02000000	1M	COARSE TABLE	0x11050000	-	-	Memory mapping defined by Coarse page table at 0x11050000. Belongs to domain 0
0x02100000	1M	FINE TABLE	0x11057000	-	-	Memory mapping defined by Fine page table at 0x11057000. Belongs to domain 0.
0x02200000	1M	FINE TABLE	0x11056000	-	-	Memory mapping defined by Fine page table at 0x11056000. Belongs to domain 0.
0x02300000	1M	FINE TABLE	0x11055000	-	-	Memory mapping defined by Fine page table at 0x11055000. Belongs to domain 0.
0x02400000	1M	FINE TABLE	0x11054000	-	-	Memory mapping defined by Fine page table at 0x11054000. Belongs to domain 0.
0x02500000	1M	FINE TABLE	0x11053000	-	-	Memory mapping defined by Fine page table at 0x11053000. Belongs to domain 0.
0x02600000	1M	FINE TABLE	0x11052000	-	-	Memory mapping defined by Fine page table at 0x11052000. Belongs to domain 0.
0x02700000	1M	FINE TABLE	0x11051000	-	-	Memory mapping defined by Fine page table at 0x11051000. Belongs to domain 0.
0x02800000	1M	FINE TABLE	0x11050000	-	-	Memory mapping defined by Fine page table at 0x11050000. Belongs to domain 0.
0x02900000 to 0x10FFFFFF	231M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x11000000	1M	Section entry	0x11000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x11100000	1M	Section entry	0x11000000	1	1	Belongs to domain 0x0, cache enabled, write buffer enabled, and permission - User R/W, Supervisor R/W.
0x11200000 to 0x11FFFFFF	14M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.

Table 74. MMU translation table *…continued*

Virtual Address Range	Size in bytes	Entry Type	Physical Address	C	B	Other settings
0x12000000	1M	Section entry	0x12000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x12100000	1M	Section entry	0x12000000	1	1	Belongs to domain 0x0, cache enabled, write buffer enabled, and permission - User R/W, Supervisor R/W.
0x12200000 to 0x12FFFFFF	14M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x13000000	1M	Section entry	0x13000000	0	0	Belongs to domain 0x0, cache enabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x13100000 to 0x14FFFFFF	31M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x15000000	1M	Section entry	0x15000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x15100000 to 0x15FFFFFF	15M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x16000000	1M	Section entry	0x16000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x16100000 to 0x16FFFFFF	15M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x17000000	1M	Section entry	0x17000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x17100000 to 0x17FFFFFF	15M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x18000000	1M	Section entry	0x18000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x18100000 to 0x18FFFFFF	15M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x19000000	1M	Section entry	0x19000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x19100000 to 0x1FFFFFFF	111M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.

Table 74. MMU translation table *…continued*

Virtual Address Range	Size in bytes	Entry Type	Physical Address	C	B	Other settings
0x20000000	1M	Section entry	0x20000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x20100000	1M	Section entry	0x20000000	1	1	Belongs to domain 0x0, cache enabled, write buffer enabled, and permission - User R/W, Supervisor R/W.
0x20200000 to 0x2FFFFFFF	254M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x30000000 to 0x3FFFFFFF	256M	Section entries	0x30000000 to 0x3FFFFFFF	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x40000000 to 0x4FFFFFFF	256M	Section entries	0x30000000 to 0x3FFFFFFF	1	1	Belongs to domain 0x0, cache enabled, write buffer enabled, and permission - User R/W, Supervisor R/W.
0x50000000 to 0x5FFFFFFF	256M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x60000000	1M	Section entry	0x60000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x60100000 to 0x6FFFFFFF	255M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.
0x70000000	1M	Section entry	0x70000000	0	0	Belongs to domain 0x0, cache disabled, write buffer disabled, and permission - User R/W, Supervisor R/W.
0x70100000 to 0xFFFFFFFF	2303M	Section entries	0xFFF00000	0	0	Belongs to domain 0xF, cache disabled, buffer disabled, and permission - User no access, Supervisor R/W.

In the above table when,

C = 0, Cache is disabled for that section of virtual memory space.

C = 1, Cache is enabled for that section of virtual memory space.

B = 0, Write buffer is disabled for that section of virtual memory space.

B = 1, Write buffer is enabled for that section of virtual memory space.

5. Programming guide

5.1 Creating LPC3130/31 bootable partition on SD/MMC cards using the ‘fdisk’ utility

This section gives the step-by-step instructions in creating LPC3130/31 bootable partition on SD/MMC cards using “fdisk” utility available on Linux PC.

1. Invoke fdisk on the device node associated with SD card. Use ‘dmesg’ command to figure out “/dev/sdxx” device Linux used for the current USB card reader. The “/dev/sdxx” log entries appear at the very end of the dmesg output.

```
$ sudo fdisk /dev/sde
[sudo] password for xxx_user:
```

2. Print the current partition table entries.

```
Command (m for help): p
Disk /dev/sde: 32 MB, 32112640 bytes
1 heads, 62 sectors/track, 1011 cylinders
Units = cylinders of 62 * 512 = 31744 bytes
Disk identifier: 0xde283a86
 Device Boot      Start         End      Blocks   Id  System
/dev/sde1               2         899       27838    6  FAT16
/dev/sde2             900        1011        3472   df  BootIt
Command (m for help):
```

3. Note, always create "bootit" (partition type 0xDF) partition as second partition. So that when the card is plugged back into a Windows PC it doesn't format “bootit” partition. Windows will not complain as long as the first partition is either FAT or NTFS partition.
4. You could use 'm' command under “fdisk” to get help on other “fdisk” commands.
5. Delete all existing partitions on the card one at a time

```
Command (m for help): d
Partition number (1-4): 1
Command (m for help): d
Partition number (1-4): 2
```

6. Now create new partitions. To specify the amount of space you need to specify start block and end block for each partition. This is usually the cylinders numbers. Since they vary from card to card it is little confusing what to specify. So we create the second partition first with +1M (1 MB size).

```
Command (m for help): n
Command action
e   extended
 p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (1-1011, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1011, default 1011): +1M
Command (m for help): t
Selected partition 2
Hex code (type L to list codes): df
Changed system type of partition 2 to df (BootIt)
Command (m for help):
```

7. Now create first partition of type FAT16 or FAT32. The card used in illustration is 32MB only so we will create FAT16 in this example.

```
Command (m for help): n
Command action
e   extended
 p   primary partition (1-4)
```

```
p
Partition number (1-4): 1
First cylinder (34-1011, default 34):
Using default value 34
Last cylinder or +size or +sizeM or +sizeK (34-1011, default 1011):
Using default value 1011
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 6
Changed system type of partition 1 to 6 (FAT16)
Command (m for help): p
Disk /dev/sde: 32 MB, 32112640 bytes
1 heads, 62 sectors/track, 1011 cylinders
Units = cylinders of 62 * 512 = 31744 bytes
Disk identifier: 0xde283a86
 Device Boot      Start         End      Blocks   Id  System
/dev/sde1              34        1011       30318    6  FAT16
/dev/sde2               1          33        1022+  df  BootIt
Partition table entries are not in disk order
Command (m for help):
```

8. Now write the table and exit from fdisk

```
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
$
```

9. Now dump the boot image to /dev/sde2 partition using "dd" command as follows. If you are using latest LPX313x CDL, the bin files generated by make system can be written directly to the card. If not then you need to create the image in the format described in Section 6–4.2.

```
$ sudo dd if=./image.bin of=/dev/sde2 bs=512
[sudo] password for xxxuser:
102+1 records in
102+1 records out
52528 bytes (53 kB) copied, 0.186911 s, 281 kB/s
$
```

10. Now the card is ready for booting. Don't forget to "sync" the card before ejecting. Also don't forget to put LPC3130/31 in SD/MMC boot mode.

UM10314

Chapter 7: LPC3130/31 Internal Static RAM (ISRAM) memory controller

Rev. 1 — 4 March 2009 User manual

1. Introduction

The ISRAM controller is used as a memory controller between the AHB bus and the internal RAM memory. The internal RAM memory can be used as working memory for the ARM processor and as temporary storage to execute the code that is loaded by boot ROM from external devices such as SPI-flash, NAND flash, parallel NOR-flash and SD/MMC cards.

1.1 Feature list

- Full implementation of AHB protocol compliant to AMBA specification (Rev 2.0).
- Configurable latency (0, 1 or 2 AHB wait states) through SYSCREG_ISRAM0_LATENCY_CFG (address 0x1300 2858, see Table 26–533) and SYSCREG_ISRAM1_LATENCY_CFG (address 0x1300 285C, see Table 26–534) registers in SYSCREG.
- Support bus endianess configuration through ARM926 coprocessor register setting.
- Single AHB slave interface towards multiple memory instances (ROM or SRAM).
- OR-bus compliant outputs.
- RAM capacity of 96 kB (LPC3130) or 128 kB (LPC3131).
- Implemented as two memories of 96 kB on the LPC3131. ISRAM0 starting at address 0x1102 8000 and ISRAM1 starting at address 0x1104 0000.

2. General description

2.1 Interface description

2.1.1 Clock signals

CGU will provides the clocks the ISRAM module (see Table 7–75).

Table 75. ISRAM module clock overview

Clock name	I/O	Source/ destination	Max. Freq.	Description
ISROM0/1_CLK	I	CGU	75 MHz.	Main clock of the module. This clock is part of the AHB bus, and runs on the same clock as the AHB main clock.

2.1.2 Reset signals

The CGU provides an asynchronous active-low reset (AHB_RST_N) which resets the logic in the ISRAM0/1_CLK clock domain.

2.1.3 DMA transfers

The ISRAM module does not make use of flow control, but it is able to make use of DMA via the DMA module.

3. Register overview

The latency configuration signal is programmed by software through the SYSCREG_ISRAM0/1_LATENCY_CFG registers in the SysCReg block (see Section 26–4.4).

4. Functional description

The CPU can read or write data from or to the RAM via the ISRAM module. The CPU will address the ISRAM module, which will translate the incoming AHB address in a RAM address. Based on this address the RAM will provide the ISRAM module with data, which is stored on that address or will store data from the ISRAM module into the RAM. Then the ISRAM module will transport the data read from the RAM to the CPU, in case of a read operation, or transport data from the CPU to the RAM in case of a write operation.

By changing the latency through the memory controller less or more pipeline stages will be added. The more pipeline stages are used, the higher the frequency is which can be used, but the bigger the latency through the ISRAM module is.

5. Power optimization

This section describes the power optimization possibilities that are included in the ISRAM module.

- Internal power consumption is minimized by extensive use of enable signals, thus limiting the switching power dissipated. The user cannot influence this process.
- SRAM is designed to support the low power features of memory. According to the low power use of memory specifications, the address, data, WEB and BSEL inputs to memory remain still when no memory transfer is under progress. The user cannot influence this process.
- When the memory is not used the addresses remain as still as much as possible.
- When memory is inactive, keep the CL input activated (CL = H) to prevent any toggling on the address, DATA and BSEL inputs, from consuming any power within the RAM.
- The CS input should be inactive (CS = L) to prevent the memory from being activated and consuming any read or write power when clock is activated (CL transition L to H).
- The Write Enable (low-active) input should be placed in the read position (WEB = H) when the RAM is not selected as this will stop any transition on the DATA and BSEL inputs from consuming any power within the RAM.
- When the memory is not used the DATA and BSEL inputs should either remain stable as much as is possible or the memory should be placed in the read mode (WEB = H) or the memory should be placed in inactive mode with CL input activated (CL = H).

6. Programming guide

In total one value can be programmed, and two values are fixed by default. The latency through the ISRAM module can be programmed by software via register ISRAM_LATENCY_CFG (see Table 26–533 and Table 26–534). Table 7–76 gives an overview of possible latency settings.

Table 76. ISRAM_latency_cfg.

Value	Description
00	No latency (default in the LPC3130/31)
01	Insert 1 wait state
11	Insert 2 wait states

Table 7–77 indicates which configuration settings can be done in the ISRAM module.

Table 77. ISRAM configuration settings.

Name	Description	Setting in this IC
bigend_a	0: set in little endianess mode 1: set in big endianess mode	Follows AHB0 big endianess setting. This signal is connected internally to CFGBIGEND signal coming out of ARM926 core. To change endianess user has to set endian bit (bit7) of control register c1 of ARM926EJ-S processor.
stall_req_a	0: normal mode 1: stall mode, operation is halted.	fixed to 0

1. Introduction

Universal Serial Bus (USB) is a standard protocol developed to connect several types of devices to each other in order to exchange data or for other purposes. Many portable devices can benefit from the ability to communicate to each other over the USB interface without intervention of a host PC. The addition of the On-The-Go functionality to USB makes this possible without losing the benefits of the standard USB protocol. Examples of USB devices are: PC, mouse, keyboard, MP3 player, digital camera, USB storage device (USB stick).

1.1 Features

- Complies with Universal Serial Bus specification 2.0.
- Complies with USB On-The-Go supplement.
- Complies with Enhanced Host Controller Interface Specification.
- Complies with AMBA specification.
- Supports auto USB 2.0 mode discovery.
- Supports all high-speed USB-compliant peripherals.
- Supports all full-speed USB-compliant peripherals.
- Supports all low-speed USB-compliant peripherals.
- Supports software HNP and SRP for OTG peripherals.
- Contains UTMI+ compliant transceiver (PHY).
- Supports power management.
- Supports four endpoints, control endpoint included.

1.2 About USB On-The-Go

The USB On-The-Go block enables usage in both device mode and in host mode. This means that you can connect to a PC to exchange data, but also to another USB device such as a digital camera or MP3 player.

The LPC3130/31 boot ROM implements the Device Firmware Upgrade (DFU) class specification to download new applications into internal SRAM.

1.3 USB acronyms and abbreviations

Table 78. USB related acronyms

Acronym	Description
ATX	Analog Transceiver
DCD	Device Controller Driver
dQH	device Endpoint Queue Head
dTD	device Transfer Descriptor

Table 78. USB related acronyms

Acronym	Description
EOP	End Of Packet
EP	End Point
FS	Full Speed
HCD	Host Controller Driver
HS	High Speed
LS	Low Speed
MPS	Maximum Packet Size
NAK	Negative Acknowledge
OTG	On-The-Go
PID	Packet Identifier
QH	Queue Head
SE0	Single Ended 0
SOF	Start Of Frame
TT	Transaction Translator
USB	Universal Serial Bus

1.4 Transmit and receive buffers

The USB OTG controller contains a Tx buffer to store data to be transmitted on the USB and an Rx buffer to store data received from the USB. The Rx buffer contains 256 words, and the Tx buffer contains 128 words for each endpoint in device mode and 512 words in host mode.

1.5 Fixed endpoint configuration

Table 8–79 shows the supported endpoint configurations. The Maximum Packet Size (MPS) (see Table 8–80) is dependent on the type of endpoint and the device configuration (low-speed, full-speed, or high-speed).

Table 79. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction
0	0	Control	Out
0	1	Control	In
1	2	Bulk/Isochronous	Out
1	3	Interrupt/Bulk/Isochronous	In
2	4	Bulk/Isochronous	Out
2	5	Interrupt/Bulk/Isochronous	In
3	6	Bulk/Isochronous	Out
3	7	Interrupt/Bulk/Isochronous	In

Table 80. USB Packet size

Endpoint type	Speed	Packet size (byte)
Control	Low-speed	8
	Full-speed	8, 16, 32, or 64
	High-speed	64
Isochronous	Low-speed	n/a
	Full-speed	up to 1023
	High-speed	up to 1024
Interrupt	Low-speed	up to 8
	Full-speed	up to 64
	High-speed	up to 1024
Bulk	Low-speed	n/a
	Full-speed	8, 16, 32, or 64
	High-speed	8, 16, 32, 64 or 512

2. General description

2.1 Block diagram

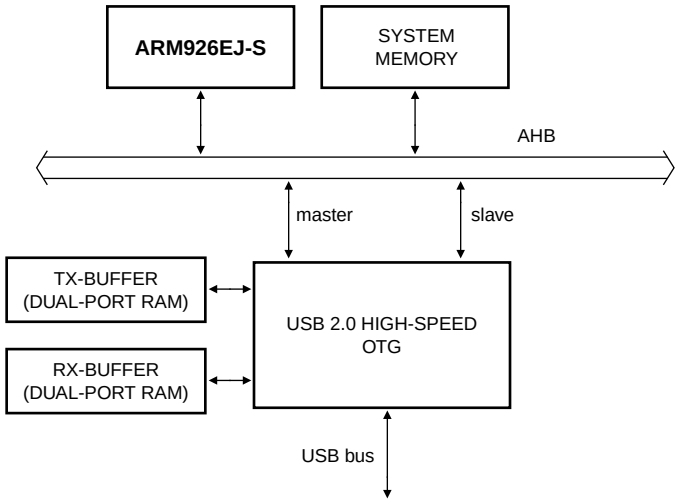


Fig 21. High-speed USB OTG block diagram

2.2 Interface description

2.2.1 Clock signals

Table 81. Clock signals of the USB-OTG

Clock name	I/O	Source/ destination	Description
USB_OTG_AHB_CLK	I	CGU	AHB bus clock. Minimum frequency is 53 MHz in order to meet the turn-around times.
USB_OTG_CLK	I	USB PLL	480 MHz main USB clock. This clock is generated by dedicated USB PLL present on chip. This PLL can be programmed through the SYSCREG block, see Section 26–4.3.1.

2.2.2 Pin connections

Table 82. USB-OTG pin configuration

Name	Type	Description
USB_DP	IO	Positive USB line
USB_DM	IO	Negative USB line
USB_ID	I	Indicates to the USB transceiver whether in device (USB_ID HIGH) or host (USB_ID LOW) mode
USB_VBUS	I	USB power
USB_RREF	IO	Connected to external resistor for reference current
Power and ground pins		
USB_VDDA33_DRV		Analog power supply for driver
USB_VDDA33		Analog power supply (3.3 V) for PHY
USB_VSSA_TERM		Analog termination ground
USB_GNDA		Analog ground
USB_VSSA_REF		Analog reference ground

2.2.3 Interrupt requests

The USB controller has one configurable USB interrupt request line.

Pins USB_VBUS and USB_ID are external pins connected to the event router. In addition the following USB signals are connected to the event router: usb_otg_vbus_pwr_en, usb_atx_pll_lock, usb_otg_ahb_needclk (see Table 17–339).

2.2.4 Reset signals

The CGU provides one AHB domain reset signal to the USB register block.

3. Register overview

Table 83. Register access abbreviations

Abbreviation	Description
R/W	Read/Write
R/WC	Read/Write one to Clear

Table 83. Register access abbreviations

Abbreviation	Description
R/WO	Read/Write Once
RO	Read Only
WO	Write Only

Table 84. Register overview: USB OTG controller (register base address 0x1900 0000)

Name	Access	Address offset	Description
-	-	0x000 - 0x0FF	Reserved
Device/host capability registers			
CAPLENGTH	RO	0x100	Capability register length
HCIVERSION	RO	0x102	Host interface version number
HCSPARAMS	RO	0x104	Host controller structural parameters
HCCPARAMS	RO	0x108	Host controller capability parameters
DCIVERSION	RO	0x120	Device interface version number
DCCPARAMS	RO	0x124	Device controller capability parameters
Device/host operational registers			
USBCMD	R/W	0x140	USB command
USBSTS	R/W	0x144	USB status
USBINTR	R/W	0x148	USB interrupt enable
FRINDEX	R/W	0x14C	USB frame index
PERIODICLISTBASE_ DEVICEADDR	R/W	0x154	Frame list base address/ USB device address
ASYNCLISTADDR_ ENDPOINTLISTADDR	R/W	0x158	Next asynchronous list address/ Address of endpoint list in memory
TTCTRL	R/W	0x15C	Asynchronous buffer status for embedded TT
BURSTSIZE	R/W	0x160	Programmable burst size
TXFILLTUNING	R/W	0x164	Host transmit pre-buffer packet tuning
BINTERVAL	R/W	0x174	Length of virtual frame
ENDPTNAK	R/W	0x178	Endpoint NAK
ENDPTNAKEN	R/W	0x17C	Endpoint NAK Enable
CONFIGFLAG	RO	0x180	Configured flag register
PORTSC1	R/W	0x184	Port status/control 1
OTGSC	R/W	0x1A4	OTG status and control
USBMODE	R/W	0x1A8	USB device mode
Device endpoint registers			
ENDPTSETUPSTAT	R/W	0x1AC	Endpoint setup status
ENDPTPRIME	R/W	0x1B0	Endpoint initialization
ENDPTFLUSH	R/W	0x1B4	Endpoint de-initialization
ENDPTSTATUS	RO	0x1B8	Endpoint status
ENDPTCOMPLETE	R/W	0x1BC	Endpoint complete
ENDPTCTRL0	R/W	0x1C0	Endpoint control 0

Table 84. Register overview: USB OTG controller (register base address 0x1900 0000)

Name	Access	Address offset	Description
ENDPTCTRL1	R/W	0x1C4	Endpoint control 1
ENDPTCTRL2	R/W	0x1C8	Endpoint control 2
ENDPTCTRL3	R/W	0x1CC	Endpoint control 3

3.1 Use of registers

The register interface has bit functions described for device mode and bit functions described for host mode. However, during OTG operations it is necessary to perform tasks independent of the controller mode.

The only way to transition the controller mode out of host or device mode is by setting the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independently of a controller reset as well as independently of the controller mode.

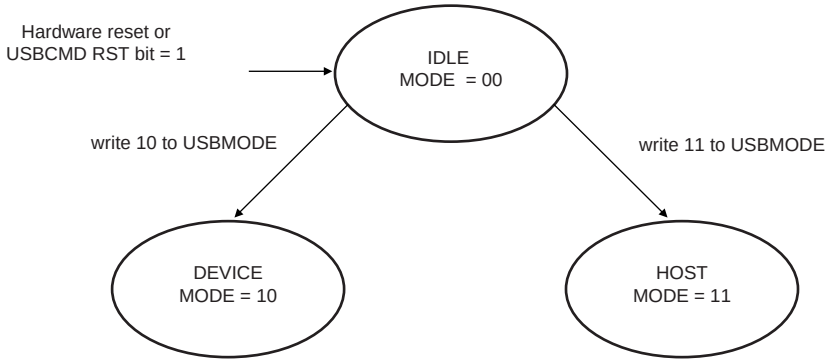


Fig 22. USB controller modes

The following registers and register bits are used for OTG operations. The values of these register bits are independent of the controller mode and are not affected by a write to the RESET bit in the USBCMD register.

- All identification registers
- All device/host capabilities registers
- All bits of the OTGSC register (Section 8–4.2.15)
- The following bits of the PORTSC register (Section 8–4.2.14):
  - PTS (parallel interface select)
  - STS (serial transceiver select)
  - PTW (parallel transceiver width)
  - PHCD (PHY low power suspend)
  - WKOC, WKDC, WKCN (wake signals)
  - PIC[1:0] (port indicators)
  - PP (port power)

4. Register description

4.1 Device/host capability registers

Table 85. CAPLENGTH (address 0x1900 0100)

Bit	Symbol	R/W	Reset value	Description
7:0	CAPLENGTH	RO	0x40	Indicates offset to add to the register base address at the beginning of the Operational Register

Table 86. HCIVERSION (address 0x1900 0102)

Bit	Symbol	R/W	Reset value	Description
15:0	HCIVERSION	RO	0x100	BCD encoding of the EHCI revision number supported by this host controller.

Table 87. HCSPARAMS (address 0x1900 0104)

Bit	Symbol	R/W	Reset value	Description
31:28	-	-	-	These bits are reserved and should be set to zero.
27:24	N_TT	RO	0x0	Number of Transaction Translators. This field indicates the number of embedded transaction translators associated with the USB2.0 host controller.
23:20	N_PTT	RO	0x0	Number of Ports per Transaction Translator. This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller.
19:17	-	-	-	These bits are reserved and should be set to zero.
16	PI	RO	0x1	Port indicators. This bit indicates whether the ports support port indicator control.
15:12	N_CC	RO	0x0	Number of Companion Controller. This field indicates the number of companion controllers associated with this USB2.0 host controller.
11:8	N_PCC	RO	0x0	Number of Ports per Companion Controller. This field indicates the number of ports supported per internal Companion Controller.
7:5	-	-	-	These bits are reserved and should be set to zero.
4	PPC	RO	0x1	Port Power Control. This field indicates whether the host controller implementation includes port power control.
3:0	N_PORTS	RO	0x1	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller.

Table 88. HCCPARAMS (address 0x1900 0108)

Bit	Symbol	R/W	Reset value	Description
31:9	-	-	-	These bits are reserved and should be set to zero.
15:8	EECP	RO	0	EHCI Extended Capabilities Pointer. This optional field indicates the existence of a capabilities list.
7:4	IST	RO	0	Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule.
2	ASP	RO	1	Asynchronous Schedule Park Capability. If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule.The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register.
1	PFL	RO	1	Programmable Frame List Flag. If set to one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-boundary. This requirement ensures that the frame list is always physically contiguous.
0	ADC	RO	0	64-bit Addressing Capability. If zero, no 64-bit addressing capability is supported.

Table 89. DCIVERSION (address 0x1900 0120)

Bit	Symbol	R/W	Reset value	Description
15:0	DCIVERSION	RO	0x1	The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

Table 90. DCCPARAMS (address 0x1900 0124)

Bit	Symbol	R/W	Reset value	Description
31:9	-	-	-	These bits are reserved and should be set to zero.
8	HC	RO	0x1	Host Capable.
7	DC	RO	0x1	Device Capable.
6:5	-	-	-	These bits are reserved and should be set to zero.
4:0	DEN	RO	0x4	Device Endpoint Number.

4.2 Device/host operational registers

4.2.1 USB Command register (USBCMD)

The host/device controller executes the command indicated in this register.

4.2.1.1 Device mode

Table 91. USB Command register (USBCMD - address 0x1900 0140) bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
0	RS		Run/Stop	R/W	0
		1	Writing a one to this bit will cause the device controller to enable a pull-up on USB_DP and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized.		
		0	Writing a 0 to this bit will cause a detach event.		
1	RST		Controller reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.	R/W	0
		1	When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial values. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.		
		0	Set to 0 by hardware when the reset process is complete.		
3:2	FS[1:0]	-	Not used in device mode.	-	0
4	PSE	-	Not used in device mode.	-	0
5	ASE	-	Not used in device mode.	-	0
6	IAA	-	Not used in device mode. Writing a one to this bit when the device mode is selected, will have undefined results.	-	-
7	-	-	Reserved. These bits should be set to 0.	-	-
9:8	ASP[1:0]	-	Not used in Device mode.	-	-
10	-		Reserved.These bits should be set to 0.	-	0
11	ASPE	-	Not used in Device mode.	-	
12	ATDTW		Add dTD trip wire This bit is used as a semaphore to ensure the to proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software during the process of adding a new dTD. See also Section 8–8. This bit shall also be cleared by hardware when its state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.	R/W	0

Table 91. USB Command register (USBCMD - address 0x1900 0140) bit description - device mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
13	SUTW		Setup trip wire During handling a setup packet, this bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (see USBMODE register) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. (See Section 8–8).	R/W	0
14	-	-	Reserved.These bits should be set to 0.		0
15	FS2		Not used in device mode.	-	-
23:16	ITC		Interrupt threshold control. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. All other values are reserved.	R/W	0x8
		0x0	Immediate (no threshold)		
		0x1	1 micro frame.		
		0x2	2 micro frames.		
		0x4	4 micro frames.		
		0x8	8 micro frames.		
		0x10	16 micro frames.		
		0x20	32 micro frames.		
		0x40	64 micro frames.		
31:24	-		Reserved		0

4.2.1.2 Host mode

Table 92. USB Command register (USBCMD - address 0x1900 0140) bit description - host mode

Bit	Symbol	Value	Description	Access	Reset value
0	RS		Run/Stop	R/W	0
		1	When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one.		
		0	When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the USBSTS register is a one).		

Table 92. USB Command register (USBCMD - address 0x1900 0140) bit description - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
1	RST		Controller reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.	R/W	0
		1	When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior.		
		0	This bit is set to zero by hardware when the reset process is complete.		
2	FS0	see Table 8–93	Bit 0 of the Frame List Size bits. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2.		0
3	FS1	see Table 8–93	Bit 1 of the Frame List Size bits.		0
4	PSE		This bit controls whether the host controller skips processing the periodic schedule.	R/W	0
		1	Use the PERIODICLISTBASE register to access the periodic schedule.		
		0	Do not process the periodic schedule.		
5	ASE		This bit controls whether the host controller skips processing the asynchronous schedule.	R/W	0
		1	Use the ASYNCLISTADDR to access the asynchronous schedule.		
		0	Do not process the asynchronous schedule.		
6	IAA		This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule.	R/W	0
		1	Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results.		
		0	The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one.		
7	-	-	Reserved		0
8:9	ASP[1:0]		Asynchronous schedule park mode	R/W	11

Table 92. USB Command register (USBCMD - address 0x1900 0140) bit description - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
		00, 01, 10, 11	Contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x1 to 0x3. **Remark:** Software must not write 00 to this bit when Park Mode Enable is one as this will result in undefined behavior.		
10	-	-	Reserved.	-	0
11	ASPE		Asynchronous Schedule Park Mode Enable	R/W	1
		1	Park mode is enabled.		
		0	Park mode is disabled.		
12	ATDTW	-	Not used in Host mode.	-	0
13	SUTW	-	Not used in Host mode.	-	
14	-	-	Reserved.	-	0
15	FS2	see Table 8–93	Bit 2 of the Frame List Size bits.	-	0
23:16	ITC		Interrupt threshold control. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. All other values are reserved.	R/W	0x8
		0x0	Immediate (no threshold)		
		0x1	1 micro frame.		
		0x2	2 micro frames.		
		0x4	4 micro frames.		
		0x8	8 micro frames.		
		0x10	16 micro frames.		
		0x20	32 micro frames.		
		0x40	64 micro frames.		
31:24	-		Reserved		0

Table 93. Frame list size values

USBCMD bit 15	USBCMD bit 3	USBCMD bit 2	Frame list size
0	0	0	1024 elements (4096 bytes) - default value
0	0	1	512 elements (2048 bytes)
0	1	0	256 elements (1024 bytes)
0	1	1	128 elements (512 bytes)
1	0	0	64 elements (256 bytes)
1	0	1	32 elements (128 bytes)
1	1	0	16 elements (64 bytes)
1	1	1	8 elements (32 bytes)

4.2.2 USB Status register (USBSTS)

This register indicates various states of the Host/Device controller and any pending interrupts. Software sets a bit to zero in this register by writing a one to it.

Remark: This register does not indicate status resulting from a transaction on the serial bus.

4.2.2.1 Device mode

Table 94. USB Status register (USBSTS - address 0x1900 0144) register bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
0	UI		USB interrupt	R/WC	0
		1	This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.		
		0	This bit is cleared by software writing a one to it.		
1	UEI		USB error interrupt	R/WC	0
		1	When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. The device controller detects resume signaling only (see Section 8–8.11.6).		
		0	This bit is cleared by software writing a one to it.		
2	PCI		Port change detect.	R/WC	0
		1	The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit (URI) and the DCSuspend bits (SLI) respectively.		
		0	This bit is cleared by software writing a one to it.		
3	FRI		Not used in Device mode.		
4	-	0	Reserved.		
5	AAI		Not used in Device mode.	-	0
6	URI		USB reset received	R/WC	0
		1	When the device controller detects a USB Reset and enters the default state, this bit will be set to a one.		
		0	This bit is cleared by software writing a one to it.		

Table 94. USB Status register (USBSTS - address 0x1900 0144) register bit description - device mode *…continued*

Bit	Symbol	Value	Description	Access	Reset value
7	SRI		SOF received	R/WC	0
		1	When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1 ms in device FS mode and every 125 μs in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp.		
		0	This bit is cleared by software writing a one to it.		
8	SLI		DCSuspend	R/WC	0
		1	When a device controller enters a suspend state from an active state, this bit will be set to a one.		
		0	The device controller clears the bit upon exiting from a suspend state. This bit is cleared by software writing a one to it.		
11:9	-	-	Reserved. Software should only write 0 to reserved bits.		
12	HCH	-	Not used in Device mode.		0
13	RCL	-	Not used in Device mode.		0
14	PS	-	Not used in Device mode.		0
15	AS	-	Not used in Device mode.		0
16	NAKI		NAK interrupt bit	RO	0
		1	It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set.		
		0	This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared.		
17	-	-	Reserved. Software should only write 0 to reserved bits.	-	0
18	UAI		Not used in Device mode.	-	0
19	UPI		Not used in Device mode.	-	0
31:20	-	-	Reserved. Software should only write 0 to reserved bits.	-	

4.2.2.2 Host mode

Table 95. USB Status register (USBSTS - address 0x1900 0144) register bit description - host mode

Bit	Symbol	Value	Description	Access	Reset value
0	UI		USB interrupt (USBINT)	R/WC	0
		1	This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.		
		0	This bit is cleared by software writing a one to it.		
1	UEI		USB error interrupt (USBERRINT)	R/WC	0
		1	When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set.		
		0	This bit is cleared by software writing a one to it.		
2	PCI		Port change detect.	R/WC	0
		1	The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port.		
		0	This bit is cleared by software writing a one to it.		
3	FRI		Frame list roll-over	R/WC	0
		1	The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FRINDEX [12] toggles (see Section 8–4.2.4).		
		0	This bit is cleared by software writing a one to it.		
4	-	0	Reserved.		
5	AAI		Interrupt on async advance	R/WC	0
		1	System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source.		
		0	This bit is cleared by software writing a one to it.		
6	URI	-	Not used by the Host controller.	R/WC	0
7	SRI		SOF received	R/WC	0
		1	In host mode, this bit will be set every 125 μs and can be used by host controller driver as a time base.		
		0	This bit is cleared by software writing a one to it.		
8	SLI	-	Not used by the Host controller.	-	-
11:9	-	-	Reserved.		

Table 95. USB Status register (USBSTS - address 0x1900 0144) register bit description - host mode *…continued*

Bit	Symbol	Value	Description	Access	Reset value
12	HCH		HCHalted	RO	1
		1	The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. because of an internal error).		
		0	The RS bit in USBCMD is set to zero. Set by the host controller.		
13	RCL		Reclamation	RO	0
		1	An empty asynchronous schedule is detected. Set by the host controller.		
		0	No empty asynchronous schedule detected.		
14	PS		Periodic schedule status This bit reports the current real status of the Periodic Schedule. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (if both are 1) or disabled (if both are 0).	RO	0
		1	The periodic schedule status is enabled.		
		0	The periodic schedule status is disabled.		
15	AS		Asynchronous schedule status This bit reports the current real status of the Asynchronous Schedule. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (if both are 1) or disabled (if both are 0).		0
		1	Asynchronous schedule status is enabled.		
		0	Asynchronous schedule status is disabled.		
16	NAKI		Not used on Host mode.	-	0
17	-		Reserved.		
18	UAI		USB host asynchronous interrupt (USBHSTASYNCINT)	R/WC	0
		1	This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set **and** the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected **and** the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes.		
		0	This bit is cleared by software writing a one to it.		
19	UPI		USB host periodic interrupt (USBHSTPERINT)	R/WC	0
		1	This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set **and** the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected **and** the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes.		
		0	This bit is cleared by software writing a one to it.		
31:20	-				

4.2.3 USB Interrupt register (USBINTR)

The software interrupts are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software. All interrupts must be acknowledged by software by clearing (that is writing a 1 to) the corresponding bit in the USBSTS register.

4.2.3.1 Device mode

Table 96. USB Interrupt register (USBINTR - address 0x1900 0148) bit description - device mode

Bit	Symbol	Description	Access	Reset value
0	UE	USB interrupt enable When this bit is one, and the USBINT bit in the USBSTS register is one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit in USBSTS.	R/W	0
1	UEE	USB error interrupt enable When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.	R/W	0
2	PCE	Port change detect enable When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit in USBSTS.	R/W	0
3	FRE	Not used by the Device controller.		
4	-	Reserved	-	0
5	AAE	Not used by the Device controller.		
6	URE	USB reset enable When this bit is a one, and the USB Reset Received bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit.	R/W	0
7	SRE	SOF received enable When this bit is a one, and the SOF Received bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit.	R/W	0
8	SLE	Sleep enable When this bit is a one, and the DCSuspend bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the DCSuspend bit.	R/W	0
15:9	-	Reserved	-	-
16	NAKE	NAK interrupt enable This bit is set by software if it wants to enable the hardware interrupt for the NAK Interrupt bit. If both this bit and the corresponding NAK Interrupt bit are set, a hardware interrupt is generated.	R/W	0
17	-	Reserved		
18	UAIE	Not used by the Device controller.		
19	UPIA	Not used by the Device controller.		
31:20	-	Reserved		

4.2.3.2 Host mode

Table 97. USB Interrupt register (USBINTR - address 0x1900 0148) bit description - host mode

Bit	Symbol	Description	Access	Reset value
0	UE	USB interrupt enable When this bit is one, and the USBINT bit in the USBSTS register is one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit in USBSTS.	R/W	0
1	UEE	USB error interrupt enable When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.	R/W	0
2	PCE	Port change detect enable When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit in USBSTS.	R/W	0
3	FRE	Frame list rollover enable When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit.		
4	-	Reserved	-	0
5	AAE	Interrupt on asynchronous advance enable When this bit is a one, and the Interrupt on Async Advance bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit.	R/W	0
6	-	Not used by the Host controller.	-	0
7	SRE	If this bit is one and the SRI bit in the USBSTS register is one, the host controller will issue an interrupt. In host mode, the SRI bit will be set every 125 μs and can be used by the host controller as a time base. The interrupt is acknowledged by software clearing the SRI bit in the USBSTS register.	-	0
8	SLE	Not used by the Host controller.	-	0
15:9	-	Reserved		
16	NAKE	Not used by the host controller.	R/W	0
17	-	Reserved		
18	UAIE	USB host asynchronous interrupt enable When this bit is a one, and the USBHSTASYNCINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTASYNCINT bit.	R/W	0
19	UPIA	USB host periodic interrupt enable When this bit is a one, and the USBHSTPERINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTPERINT bit.	R/W	0
31:20	-	Reserved		

4.2.4 Frame index register (FRINDEX)

4.2.4.1 Device mode

In Device mode this register is read only, and the device controller updates the FRINDEX[13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX[13:3] will be checked against the SOF marker. If FRINDEX[13:3] is different from the SOF marker, FRINDEX[13:3] will be set to the SOF value and FRINDEX[2:0] will be set to zero (i.e. SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX[2:0] will be incremented (i.e. SOF for 125 μs micro-frame) by hardware.

Table 98. USB frame index register (FRINDEX - address 0x1900 014C) bit description - device mode

Bit	Symbol	Description	Access	Reset value
2:0	FRINDEX[2:0]	Current micro frame number	RO	N/A
13:3	FRINDEX[13:3]	Current frame number of the last frame transmitted	RO	N/A
31:14	-	Reserved		N/A

4.2.4.2 Host mode

This register is used by the host controller to index the periodic frame list. The register updates every 125 μs (once each micro-frame). Bits[N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the USBCMD register.

This register must be written as a DWord. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit in the USBSTS register (host mode). A write to this register while the Run/Stop bit is set to a one produces undefined results. Writes to this register also affect the SOF value.

Table 99. USB frame index register (FRINDEX - address 0x1900 014C) bit description - host mode

Bit	Symbol	Description	Access	Reset value
2:0	FRINDEX[2:0]	Current micro frame number	R/W	N/A
N:3	FRINDEX[N:3]	Frame list current index	R/W	N/A
31:(N+1)	-	Reserved		N/A

Table 100. Number of bits used for the frame list index

USBCMD bit 15	USBCMD bit 3	USBCMD bit 2	Frame list size	N
0	0	0	1024 elements (4096 bytes). Default value.	12
0	0	1	512 elements (2048 bytes)	11
0	1	0	256 elements (1024 bytes)	10
0	1	1	128 elements (512 bytes)	9
1	0	0	64 elements (256 bytes)	8

Table 100. Number of bits used for the frame list index

USBCMD bit 15	USBCMD bit 3	USBCMD bit 2	Frame list size	N
1	0	1	32 elements (128 bytes)	7
1	1	0	16 elements (64 bytes)	6
1	1	1	8 elements (32 bytes)	5

4.2.5 Device address (DEVICEADDR - device) and Periodic List Base (PERIODICLISTBASE- host) registers

4.2.5.1 Device mode

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

The USBADRA bit is used to accelerate the SET_ADDRESS sequence by allowing the DCD to preset the USBADR register bits before the status phase of the SET_ADDRESS descriptor.

Table 101. USB Device Address register (DEVICEADDR - address 0x1900 0154) bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
23:0	-		reserved	-	0
24	USBADRA		Device address advance		
		1	When the user writes a one to this bit at the same time or before USBADR is written, the write to USBADR fields is staged and held in a hidden register. After an IN occurs on endpoint 0 and is acknowledged, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: • IN is ACKed to endpoint 0. USBADR is updated from the staging register. • OUT/SETUP occurs on endpoint 0. USBADR is **not** updated. • Device reset occurs. USBADR is set to 0. **Remark:** After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2 ms USB requirement.		
		0	Any write to USBADR are instantaneous.		
31:25	USBADR		USB device address	R/W	0

4.2.5.2 Host mode

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. The host controller driver (HCD) loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4 kb aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Table 102. USB Periodic List Base register (PERIODICLISTBASE - address 0x1900 0154) bit description - host mode

Bit	Symbol	Description	Access	Reset value
11:0	-	reserved	-	N/A
31:12	PERBASE[31:12]	Base Address (Low) These bits correspond to the memory address signals[31:12].	R/W	N/A

4.2.6 Endpoint List Address register (ENDPOINTLISTADDR - device) and Asynchronous List Address (ASYNCLISTADDR - host) registers

4.2.6.1 Device mode

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits[10:0] of this register cannot be modified by the system software and will always return a zero when read.The memory structure referenced by this physical memory pointer is assumed 64 byte aligned.

Table 103. USB Endpoint List Address register (ENDPOINTLISTADDR - address 0x1900 0158) bit description - device mode

Bit	Symbol	Description	Access	Reset value
10:0	-	reserved	-	0
31:11	EPBASE[31:11]	Endpoint list pointer (low) These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 4 Queue Heads (QH). (i.e. one queue head per endpoint and direction.)	R/W	N/A

4.2.6.2 Host mode

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Table 104. USB Asynchronous List Address register (ASYNCLISTADDR- address 0x1900 0158) bit description - host mode

Bit	Symbol	Description	Access	Reset value
4:0	-	Reserved	-	0
31:5	ASYBASE[31:5]	Link pointer (Low) LPL These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH).	R/W	N/A

4.2.7 TT Control register (TTCTRL)

4.2.7.1 Device mode

This register is not used in device mode.

4.2.7.2 Host mode

This register contains parameters needed for internal TT operations. This register is used by the host controller only. Writes must be in Dwords.

Table 105. USB TT Control register (TTCTRL - address 0x1900 015C) bit description - host mode

Bit	Symbol	Description	Access	Reset value
23:0	-	Reserved.	-	0
30:24	TTHA	Hub address when FS or LS device are connected directly.	R/W	N/A
31	-	Reserved.		0

4.2.8 Burst Size register (BURSTSIZE)

This register is used to control and dynamically change the burst size used during data movement on the master interface of the USB DMA controller. Writes must be in Dwords.

The default for the length of a burst of 32-bit words for RX and TX DMA data transfers is 16 words each.

Table 106. USB burst size register (BURSTSIZE - address 0x1900 0160) bit description - device/host mode

Bit	Symbol	Description	Access	Reset value
7:0	RXPBURST	Programmable RX burst length This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.	R/W	0x10
15:8	TXPBURST	Programmable TX burst length This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.	R/W	0x10
31:16	-	reserved	-	-

4.2.9 Transfer buffer Fill Tuning register (TXFILLTUNING)

4.2.9.1 Device controller

This register is not used in device mode.

4.2.9.2 Host controller

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level

T_s = Total packet flight time (send-only) packet; $T_s = T_0 + T_1$

T_p = Total packet time (fetch and send) packet; $T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the (micro) frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a “backoff”

event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Backoffs can be minimized with use of the TSCHHEALTH (T_{ff}) described below.

Table 107. USB Transfer buffer Fill Tuning register (TXFIFOFILLTUNING - address 0x1900 0164) bit description - host mode

Bit	Symbol	Description	Access	Reset value
7:0	TXSCHOH	FIFO burst threshold This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set. .	R/W	0x2
12:8	TXSCHEATLTH	Scheduler health counter This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame . This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter. The maximum value is 31.	R/W	0x0
15:13	-	reserved	-	-
21:16	TXFIFOTHRES	Scheduler overhead This register adds an additional fixed offset to the schedule time estimator described above as T_{ff}. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 μs when a device is connected in High-Speed Mode for OTG and SPH. The time unit represented in this register is 6.333 μs when a device is connected in Low/Full Speed Mode for OTG and SPH.	R/W	0x0
31:22	-	reserved		

4.2.10 BINTERVAL register

This register defines the bInterval value which determines the length of the virtual frame (see Section 8–5.7).

Table 108. USB BINTERVAL register (BINTERVAL - address 0x1900 0174) bit description - device/host mode

Bit	Symbol	Description	Access	Reset value
3:0	BINT	bInterval value (see Section 8–5.7)	R/W	0x00
31:4	-	reserved	-	-

4.2.11 USB Endpoint NAK register (ENDPTNAK)

4.2.11.1 Device mode

This register indicates when the device sends a NAK handshake on an endpoint. Each Tx and Rx endpoint has a bit in the EPTN and EPRN field respectively.

A bit in this register is cleared by writing a 1 to it.

Table 109. USB endpoint NAK register (ENDPTNAK - address 0x1900 0178) bit description - device mode

Bit	Symbol	Description	Access	Reset value
3:0	EPRN[3:0]	Rx endpoint NAK Each RX endpoint has one bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. Bit 3 corresponds to endpoint 3. ... Bit 1 corresponds to endpoint 1. Bit 0 corresponds to endpoint 0.	R/WC	0x00
15:4	-	reserved	-	-
19:16	EPTN[3:0]	Tx endpoint NAK Each TX endpoint has one bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. Bit 3 corresponds to endpoint 3. ... Bit 1 corresponds to endpoint 1. Bit 0 corresponds to endpoint 0.	R/WC	0x00
31:20	-	reserved	-	-

4.2.11.2 Host mode

This register is not used in host mode.

4.2.12 USB Endpoint NAK Enable register (ENDPTNAKEN)

4.2.12.1 Device mode

Each bit in this register enables the corresponding bit in the ENDPTNAK register. Each Tx and Rx endpoint has a bit in the EPTNE and EPRNE field respectively.

Table 110. USB Endpoint NAK Enable register (ENDPTNAKEN - address 0x1900 017C) bit description - device mode

Bit	Symbol	Description	Access	Reset value
3:0	EPRNE[3:0]	Rx endpoint NAK enable Each bit enables the corresponding RX NAK bit. If this bit is set and the corresponding RX endpoint NAK bit is set, the NAK interrupt bit is set. Bit 3 corresponds to endpoint 3. ... Bit 1 corresponds to endpoint 1. Bit 0 corresponds to endpoint 0.	R/W	0x00
15:4	-	reserved	-	-
19:16	EPTNE[3:0]	Tx endpoint NAK Each bit enables the corresponding TX NAK bit. If this bit is set and the corresponding TX endpoint NAK bit is set, the NAK interrupt bit is set. Bit 3 corresponds to endpoint 3. ... Bit 1 corresponds to endpoint 1. Bit 0 corresponds to endpoint 0.	R/W	0x00
31:20	-	reserved	-	-

4.2.12.2 Host mode

This register is not used in host mode.

4.2.13 CONFIGFLAG register

This register is not used on the LPC3130/31.

Table 111. CONFIGFLAG (address 0x1900 0180) bit description

Bit	Symbol	R/W	Reset value	Description
31:0	CONFIGFLAG	R	0x1	Not used in this implementation

4.2.14 Port Status and Control register (PORTSC1)

4.2.14.1 Device mode

The device controller implements one port register, and it does not support power control. Port control in device mode is used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling. This register allows software to put the PHY into low-power Suspend mode and disable the PHY clock.

Table 112. Port Status and Control register (PRTSC1 - address 0x1900 0184) bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
0	CCS		Current connect status	RO	0
		1	Device attached. A one indicates that the device successfully attached and is operating in either high-speed mode or full-speed mode as indicated by the High Speed Port bit in this register.		
		0	Device not attached A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.		
1	CSC	-	Not used in device mode	-	0
2	PE	1	Port enable. This bit is always 1. The device port is always enabled.	RO	1
3	PEC	0	Port enable/disable change This bit is always 0. The device port is always enabled.	RO	0
5:4	-	-	Reserved	RO	0
6	FPR		Force port resume After the device has been in Suspend State for 5 ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. When this bit transitions to a one because a J-to-K transition detected, the Port Change Detect bit in the USBSTS register is set to one as well.	R/W	0
		1	Resume detected/driven on port.		
		0	No resume (K-state) detected/driven on port.		
7	SUSP		Suspend In device mode, this is a read-only status bit .	RO	0
		1	Port in suspend state		
		0	Port not in suspend state		
8	PR		Port reset In device mode, this is a read-only status bit. A device reset from the USB bus is also indicated in the USBSTS register.	RO	0
		1	Port is in the reset state.		
		0	Port is not in the reset state.		
9	HSP		High-speed status **Remark:** This bit is redundant with bits [27:26] (PSPD) in this register. It is implemented for compatibility reasons.	RO	0
		1	Host/device connected to the port is in High-speed mode.		
		0	Host/device connected to the port is not in High-speed mode.		
11:10	LS	-	Not used in device mode.		
12	PP	-	Not used in device mode.		
13	-	-	Reserved	-	-

Table 112. Port Status and Control register (PRTSC1 - address 0x1900 0184) bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
15:14	PIC[1:0]		Port indicator control Writing to this field effects the value of the USB_OTG_PORT_IND_CTL (address 0x1300 2838) register (see Table 26–526).	R/W	00
		00	Port indicators are off.		
		01	amber		
		10	green		
		11	undefined		
19:16	PTC[3:0]		Port test control Any value other than 0000 indicates that the port is operating in test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_HS/FS/LS values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point.	R/W	0000
		0000	TEST_MODE_DISABLE		
		0001	J_STATE		
		0010	K_STATE		
		0011	SE0 (host)/NAK (device)		
		0100	Packet		
		0101	FORCE_ENABLE_HS		
		0110	FORCE_ENABLE_FS		
		0111	invalid in device mode.		
		1000 to 1111	Reserved		
20	WKCN	-	Not used in device mode. This bit is always 0 in device mode.	-	0
21	WKDC	-	Not used in device mode. This bit is always 0 in device mode.	-	0
22	WKOC		Not used in device mode. This bit is always 0 in device mode.	-	0
23	PHCD		PHY low power suspend - clock disable (PLPSCD) In device mode, The PHY can be put into Low Power Suspend – Clock Disable when the device is not running (USBCMD Run/Stop = 0) or the host has signaled suspend (PORTSC SUSPEND = 1). Low power suspend will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the device controller driver must clear this bit.	R/W	0
		1	Writing a 1 disables the PHY clock. Reading a 1 indicates the status of the PHY clock (disabled).		
		0	Writing a 0 enables the PHY clock. Reading a 0 indicates the status of the PHY clock (enabled).		

Table 112. Port Status and Control register (PRTSC1 - address 0x1900 0184) bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
24	PFSC		Port force full speed connect	R/W	0
		1	Writing this bit to a 1 will force the port to only connect at full speed. It disables the chirp sequence that allows the port to identify itself as High-speed. This is useful for testing FS configurations with a HS host, hub or device.		
		0	Port connects at any speed.		
25	-	-	reserved		
27:26	PSPD		Port speed This register field indicates the speed at which the port is operating.	RO	0
		00	Full-speed		
		01	invalid in device mode		
		10	High-speed		
31:28	-	-	Reserved	-	-

4.2.14.2 Host mode

The host controller uses one port. The register is only reset when power is initially applied or in response to a controller reset. The initial conditions of the port are:

- No device connected
- Port disabled

If the port has power control, this state remains until software applies power to the port by setting port power to one in the PORTSC register.

Table 113. Port Status and Control register (PRTSC1 - address 0x1900 0184) - host mode

Bit	Symbol	Value	Description	Access	Reset value
0	CCS		Current connect status This value reflects the current state of the port and may not correspond directly to the event that caused the CSC bit to be set. This bit is 0 if PP (Port Power bit) is 0. Software clears this bit by writing a 1 to it.	R/WC	0
		1	Device is present on the port.		
		0	No device is present.		
1	CSC		Connect status change Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This bit is 0 if PP (Port Power bit) is 0	R/WC	0
		1	Change in current status.		
		0	No change in current status.		

Table 113. Port Status and Control register (PRTSC1 - address 0x1900 0184) - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
2	PE	1	Port enable. Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled. downstream propagation of data is blocked except for reset. This bit is 0 if PP (Port Power bit) is 0.	R/W	0
		1	Port enabled.		
		0	Port disabled.		
3	PEC	0	Port disable/enable change For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See *Chapter 11 of the USB Specification*). Software clears this by writing a one to it. This bit is 0 if PP (Port Power bit) is 0,	R/WC	0
		1	Port enabled/disabled status has changed.		
		0	No change.		
4	OCA		Over-current active This bit will automatically transition from 1 to 0 when the over-current condition is removed. This bit gets set when the usb_otg_vbus_pwr_fault (bit 3) in Table 493. USB_OTG_CFG (address 0x1300 2834) is set (see Table 26–525). Software should monitor OC condition on an unused GPIO pin and set USB_OTG_CFG register, so that the standard EHCI driver can use this bit.	RO	0
		1	The port has currently an over-current condition.		
		0	The port does not have an over-current condition.		
5	OCC		Over-current change This bit gets set to one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position.	R/WC	0

Table 113. Port Status and Control register (PRTSC1 - address 0x1900 0184) - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
6	FPR		Force port resume Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no affect because the port controller will time the resume operation clear the bit the port control state switches to HS or FS idle. This bit is 0 if PP (Port Power bit) is 0.	R/W	0
		1	Resume detected/driven on port.		
		0	No resume (K-state) detected/driven on port.		
7	SUSP		Suspend Together with the PE (Port enabled bit), this bit describes the port states, see Table 8–114 "Port states as described by the PE and SUSP bits in the PORTSC1 register". The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This bit is 0 if PP (Port Power bit) is 0.	R/W	0
		1	Port in suspend state When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.		
		0	Port not in suspend state		
8	PR		Port reset When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. This bit is 0 if PP (Port Power bit) is 0.	R/W	0
		1	Port is in the reset state.		
		0	Port is not in the reset state.		

Table 113. Port Status and Control register (PRTSC1 - address 0x1900 0184) - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
9	HSP		High-speed status	RO	0
		1	Host/device connected to the port is in High-speed mode.		
		0	Host/device connected to the port is not in High-speed mode.		
11:10	LS		Line status These bits reflect the current logical levels of the USB_DP and USB_DM signal lines. USB_DP corresponds to bit 11 and USB_DM to bit 10. In host mode, the use of linestate by the host controller driver is not necessary for this controller (unlike EHCI) because the controller hardware manages the connection of LS and FS.	RO	0x3
		00	SE0 (USB_DP and USB_DM LOW)		
		10	J-state (USB_DP HIGH and USB_DM LOW)		
		01	K-state (USB_DP LOW and USB_DM HIGH)		
		11	Undefined		
12	PP	-	Port power control Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e. PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port).	R/W	0
		1	Port power on.		
		0	Port power off.		
13	-	-	Reserved	-	0
15:14	PIC[1:0]		Port indicator control Writing to this field effects the value of the USB_OTG_PORT_IND_CTL (address 0x1300 2838) register (see Table 26–526).	R/W	00
		00	Port indicators are off.		
		01	Amber		
		10	Green		
		11	Undefined		

Table 113. Port Status and Control register (PRTSC1 - address 0x1900 0184) - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
19:16	PTC[3:0]		Port test control Any value other than 0000 indicates that the port is operating in test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point.	R/W	0000
		0000	TEST_MODE_DISABLE		
		0001	J_STATE		
		0010	K_STATE		
		0011	SE0 (host)/NAK (device)		
		0100	Packet		
		0101	FORCE_ENABLE_HS		
		0110	FORCE_ENABLE_FS		
		0111	FORCE_ENABLE_LS		
		1000 to 1111	reserved		
20	WKCN		Wake on connect enable (WKCNNT_E) This bit is 0 if PP (Port Power bit) is 0	R/W	0
		1	Writing this bit to a one enables the port to be sensitive to device connects as wake-up events.		
		0	Disables the port to wake up on device connects.		
21	WKDC		Wake on disconnect enable (WKDSCNNT_E) This bit is 0 if PP (Port Power bit) is 0.	R/W	0
		1	Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events.		
		0	Disables the port to wake up on device disconnects.		
22	WKOC		Wake on over-current enable (WKOC_E)	R/W	0
		1	Writing a one to this bit enabled the port to be sensitive to over-current conditions as wake-up events.		
		0	Disables the port to wake up on over-current events.		
23	PHCD		PHY low power suspend - clock disable (PLPSCD) In host mode, the PHY can be put into Low Power Suspend – Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software.	R/W	0
		1	Writing a 1 disables the PHY clock. Reading a 1 indicates the status of the PHY clock (disabled).		
		0	Writing a 0 enables the PHY clock. Reading a 0 indicates the status of the PHY clock (enabled).		

Table 113. Port Status and Control register (PRTSC1 - address 0x1900 0184) - host mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
24	PFSC		Port force full speed connect	R/W	0
		1	Writing this bit to a 1 will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device.		
		0	Port connects at any speed.		
25	-	-	reserved		
27:26	PSPD		Port speed This register field indicates the speed at which the port is operating. For HS mode operation in the host controller and HS/FS operation in the device controller the port routing steers data to the Protocol engine. For FS and LS mode operation in the host controller, the port routing steers data to the Protocol Engine w/ Embedded Transaction Translator.	RO	0
		00	Full-speed		
		01	Low-speed		
		10	High-speed		
31:28		-	Reserved	-	-

Table 114. Port states as described by the PE and SUSP bits in the PORTSC1 register

PE bit	SUSP bit	Port state
0	0 or 1	disabled
1	0	enabled
1	1	suspend

4.2.15 OTG Status and Control register (OTGSC)

The OTG register has four sections:

- OTG interrupt enables (R/W)
- OTG Interrupt status (R/WC)
- OTG status inputs (RO)
- OTG controls (R/W)

The status inputs are debounced using a 1 msec time constant. Values on the status inputs that do not persist for more than 1 msec will not cause an update of the status input register or cause an OTG interrupt.

Table 115. OTG Status and Control register (OTGSC - address 0x1900 01A4) bit description

Bit	Symbol	Value	Description	Access	Reset value
OTG controls					
0	VD		VBUS_Discharge Setting this bit to 1 causes VBUS to discharge through a resistor.	R/W	0
1	VC		VBUS_Charge Setting this bit to 1 causes the VBUS line to be charged. This is used for VBUS pulsing during SRP.	R/W	0

Table 115. OTG Status and Control register (OTGSC - address 0x1900 01A4) bit description *…continued*

Bit	Symbol	Value	Description	Access	Reset value
2	HAAR		Hardware assist auto_reset	R/W	0
		1	Enable automatic reset after connect on host port.		
		0	Disabled		
3	OT		OTG termination This bit must be set to 1 when the OTG controller is in device mode. This controls the pull-down on USB_DM.	R/W	0
4	DP		Data pulsing Setting this bit to 1 causes the pull-up on USB_DP to be asserted for data pulsing during SRP.	R/W	0
5	IDPU		ID pull-up. This bit provides control over the pull-up resistor.	R/W	1
		1	Pull-up on.		
		0	Pull-up off. The ID bit will not be sampled.		
6	HADP		Hardware assist data pulse Write a 1 to start data pulse sequence.	R/W	0
7	HABA		Hardware assist B-disconnect to A-connect	R/W	0
		1	Enable automatic B-disconnect to A-connect sequence.		
		0	Disabled.		
OTG status inputs					
8	ID		USB ID	RO	0
		1	B-device		
		0	A-device		
9	AVV		A-VBUS valid Reading 1 indicates that VBUS is above the A-VBUS valid threshold.	RO	0
10	ASV		A-session valid Reading 1 indicates that VBUS is above the A-session valid threshold.	RO	0
11	BSV		B-session valid Reading 1 indicates that VBUS is above the B-session valid threshold.	RO	0
12	BSE		B-session end Reading 1 indicates that VBUS is below the B-session end threshold.	RO	0
13	1mST		1 millisecond timer toggle This bit toggles once per millisecond.	RO	0
14	DPS		Data bus pulsing status Reading a 1 indicates that data bus pulsing is detected on the port.	RO	0
15	-	-	reserved		0
OTG interrupt status					
16	IDIS		USB ID interrupt status This bit is set when a change on the ID input has been detected. Software must write a 1 to this bit to clear it.	R/WC	0

Table 115. OTG Status and Control register (OTGSC - address 0x1900 01A4) bit description *…continued*

Bit	Symbol	Value	Description	Access	Reset value
17	AVVIS		A-VBUS valid interrupt status This bit is set then VBUS has either risen above or fallen below the A-VBUS valid threshold (4.4 V on an A-device). Software must write a 1 to this bit to clear it.	R/WC	0
18	ASVIS		A-Session valid interrupt status This bit is set then VBUS has either risen above or fallen below the A-session valid threshold (0.8 V). Software must write a 1 to this bit to clear it.	R/WC	0
19	BSVIS		B-Session valid interrupt status This bit is set then VBUS has either risen above or fallen below the B-session valid threshold (0.8 V). Software must write a 1 to this bit to clear it.	R/WC	0
20	BSEIS		B-Session end interrupt status This bit is set then VBUS has fallen below the B-session end threshold. Software must write a 1 to this bit to clear it.	R/WC	0
21	1msS		1 millisecond timer interrupt status This bit is set once every millisecond. Software must write a 1 to this bit to clear it.	R/WC	0
22	DPIS		Data pulse interrupt status This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when the CM bit in USBMODE = Host (11) and the PortPower bit in PORTSC = Off (0). Software must write a 1 to this bit to clear it.	R/WC	0
23	-	-	reserved		0
OTG interrupt enable					
24	IDIE		USB ID interrupt enable Setting this bit enables the interrupt. Writing a 0 disables the interrupt.	R/W	0
25	AVVIE		A-VBUS valid interrupt enable Setting this bit enables the A-VBUS valid interrupt. Writing a 0 disables the interrupt.	R/W	0
26	ASVIE		A-session valid interrupt enable Setting this bit enables the A-session valid interrupt. Writing a 0 disables the interrupt	R/W	0
27	BSVIE		B-session valid interrupt enable Setting this bit enables the B-session valid interrupt. Writing a 0 disables the interrupt.	R/W	0
28	BSEIE		B-session end interrupt enable Setting this bit enables the B-session end interrupt. Writing a 0 disables the interrupt.	R/W	0

Table 115. OTG Status and Control register (OTGSC - address 0x1900 01A4) bit description *…continued*

Bit	Symbol	Value	Description	Access	Reset value
29	1msE		1 millisecond timer interrupt enable Setting this bit enables the 1 millisecond timer interrupt. Writing a 0 disables the interrupt.	R/W	0
30	DPIE		Data pulse interrupt enable Setting this bit enables the data pulse interrupt. Writing a 0 disables the interrupt	R/W	0
31	-	-	Reserved	-	0

4.2.16 USB Mode register (USBMODE)

The USBMODE register sets the USB mode for the OTG controller. The possible modes are Device, Host, and Idle mode for OTG operations.

4.2.16.1 Device mode

Table 116. USB Mode register (USBMODE - address 0x1900 01A8) bit description - device mode

Bit	Symbol	Value	Description	Access	Reset value
1:0	CM[1:0]		Controller mode The controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register.	R/ WO	00
		00	Idle		
		01	Reserved		
		10	Device controller		
		11	Host controller		
2	ES		Endian select This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words.	R/W	0
		1	Big endian: first byte referenced in most significant byte of 32-bit word.		
		0	Little endian: first byte referenced in least significant byte of 32-bit word.		
3	SLOM		Setup Lockout mode In device mode, this bit controls behavior of the setup lock mechanism. See Section 8–8.8.	R/W	0
		1	Setup Lockouts Off (DCD requires the use of Setup Buffer Tripwire in USBCMD)		
		0	Setup Lockouts on		

Table 116. USB Mode register (USBMODE - address 0x1900 01A8) bit description - device mode *...continued*

Bit	Symbol	Value	Description	Access	Reset value
4	SDIS		Stream disable mode **Remark:** The use of this feature substantially limits the overall USB performance that can be achieved.	R/W	0
		1	Disabled. Setting this bit to one disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active.		
		0	Not disabled		
5	VBPS		Not used in device mode.	-	0
31:6	-	-	reserved		

4.2.16.2 Host mode

Table 117. USB Mode register (USBMODE - address 0x1900 01A8) bit description - host mode

Bit	Symbol	Value	Description	Access	Reset value
1:0	CM[1:0]		Controller mode The controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register.	R/ WO	00
		00	Idle		
		01	Reserved		
		10	Device controller		
		11	Host controller		
2	ES		Endian select This bit can change the byte ordering of the transfer buffers. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words.	R/W	0
		1	Big endian: first byte referenced in most significant byte of 32-bit word.		
		0	Little endian: first byte referenced in least significant byte of 32-bit word.		
3	SLOM		Not used in host mode	-	0

Table 117. USB Mode register (USBMODE - address 0x1900 01A8) bit description - host mode *…continued*

Bit	Symbol	Value	Description	Access	Reset value
4	SDIS		Stream disable mode **Remark:** The use of this feature substantially limits the overall USB performance that can be achieved.	R/W	0
		1	Disabled. Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the the TX latency is filled to capacity before the packet is launched onto the USB. Note: Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TXFILLTUNING to characterize the adjustments needed for the scheduler when using this feature.		
		0	Not disabled		
5	VBPS		VBUS power select This bit sets the vbus_pwr_select output LOW or HIGH to select between an on-chip VBUS power source and an off-chip power source. Setting this bit will trigger the usb_otg_vbus_pwr_en event in the event router (see Section 8–2.2.3).	R/WO	0
		0	vbus_pwr_select is set LOW.		
		1	vbus_pwr_select is set HIGH		
31:6	-	-	reserved		

4.3 Device endpoint registers

4.3.1 USB Endpoint Setup Status register (ENDPSETUPSTAT)

Table 118. USB Endpoint Setup Status register (ENDPTSETUPSTAT - address 0x1900 01AC) bit description

Bit	Symbol	Description	Access	Reset value
3:0	ENDPT SETUP STAT[3:0]	Setup endpoint status for logical endpoints 0 to 3. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lockout mechanism is engaged.	R/WC	0
31:3	-	reserved		

4.3.2 USB Endpoint Prime register (ENDPTPRIME)

For each endpoint, software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.

Remark: These bits will be momentarily set by hardware during hardware endpoint re-priming operations when a dTD is retired and the dQH is updated.

Table 119. USB Endpoint Prime register (ENDPTPRIME - address 0x1900 01B0) bit description

Bit	Symbol	Description	Access	Reset value
3:0	PERB[3:0]	Prime endpoint receive buffer for physical OUT endpoints 3 to 0. For each OUT endpoint, a corresponding bit is set to 1 by software to request a buffer be prepared for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. PERB0 = endpoint 0 ... PERB3 = endpoint 3	R/WS	0
15:4	-	reserved		
19:16	PETB[3:0]	Prime endpoint transmit buffer for physical IN endpoints 3 to 0. For each IN endpoint a corresponding bit is set to one by software to request a buffer be prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. PETB0 = endpoint 0 ... PETB3 = endpoint 3	R/WS	0
31:20	-	reserved		

4.3.3 USB Endpoint Flush register (ENDPTFLUSH)

Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful.

Table 120. USB Endpoint Flush register (address 0x1900 01B4) bit description

Bit	Symbol	Description	Access	Reset value
3:0	FERB[3:0]	Flush endpoint receive buffer for physical OUT endpoints 3 to 0. Writing a one to a bit(s) will clear any primed buffers. FERB0 = endpoint 0 ... FERB3 = endpoint 3	R/WS	0

Table 120. USB Endpoint Flush register (address 0x1900 01B4) bit description

Bit	Symbol	Description	Access	Reset value
15:4	-	reserved		
19:16	FETB[3:0]	Flush endpoint transmit buffer for physical IN endpoints 3 to 0. Writing a one to a bit(s) will clear any primed buffers. FETB0 = endpoint 0 ... FETB3 = endpoint 3	R/WS	0
31:20	-	reserved		

4.3.4 USB Endpoint Status register (ENDPSTAT)

One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set by hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.

Remark: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired and the dQH is updated.

Table 121. USB Endpoint Status register (address 0x1900 01B8) bit description

Bit	Symbol	Description	Access	Reset value
3:0	ERBR[3:0]	Endpoint receive buffer ready for physical OUT endpoints 3 to 0. This bit is set to 1 by hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. ERBR0 = endpoint 0 ... ERBR3 = endpoint 3	RO	0
15:4	-	reserved		
19:16	ETBR[3:0]	Endpoint transmit buffer ready for physical IN endpoints 3 to 0. This bit is set to 1 by hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. ETBR0 = endpoint 0 ... ETBR3 = endpoint 3	RO	0
31:20	-	reserved		

4.3.5 USB Endpoint Complete register (ENDPTCOMPLETE)

Each bit in this register indicates that a received/transmit event occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT.

Writing a one will clear the corresponding bit in this register.

Table 122. USB Endpoint Complete register (address 0x1900 01BC) bit description

Bit	Symbol	Description	Access	Reset value
3:0	ERCE[3:0]	Endpoint receive complete event for physical OUT endpoints 3 to 0. This bit is set to 1 by hardware when receive event (OUT/SETUP) occurred. ERCE0 = endpoint 0 ... ERCE3 = endpoint 3	R/WC	0
15:4	-	reserved		
19:16	ETCE[3:0]	Endpoint transmit complete event for physical IN endpoints 3 to 0. This bit is set to 1 by hardware when a transmit event (IN/INTERRUPT) occurred. ETCE0 = endpoint 0 ... ETCE3 = endpoint 3	R/WC	0
31:20	-	reserved		

4.3.6 USB Endpoint 0 Control register (ENDPTCTRL0)

This register initializes endpoint 0 for control transfer. Endpoint 0 is always a control endpoint.

Table 123. USB Endpoint 0 Control register (ENDPTCTRL0 - address 0xFFE0 C1C0) bit description

Bit	Symbol	Value	Description	Access	Reset value
0	RXS		Rx endpoint stall	R/W	0
		1	Endpoint stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software, or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDSETUPSTAT bit is cleared.[1]		
		0	Endpoint ok.		
1	-	-	reserved		
3:2	RXT[1:0]	00	Endpoint type Endpoint 0 is always a control endpoint.	R/W	00
6:4	-	-	reserved		
7	RXE	1	Rx endpoint enable Endpoint enabled. Control endpoint 0 is always enabled. This bit is always 1.	RO	1
15:8	-	-	reserved		

Table 123. USB Endpoint 0 Control register (ENDPTCTRL0 - address 0xFFE0 C1C0) bit description *...continued*

Bit	Symbol	Value	Description	Access	Reset value
16	TXS		Tx endpoint stall	R/W	
		1	Endpoint stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software, or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDSETUPSTAT bit is cleared.[1]		
		0	Endpoint ok.		
17	-	-	reserved		
19:18	TXT[1:0]	00	Endpoint type Endpoint 0 is always a control endpoint.	RO	00
22:20	-	-	reserved		
23	TXE	1	Tx endpoint enable Endpoint enabled. Control endpoint 0 is always enabled. This bit is always 1.	RO	1
31:24	-	-	reserved		

[1] There is a slight delay (50 clocks max) between the ENPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely that the DCD software will observe this delay. However, should the DCD notice that the stall bit is not set after writing a one to it, software should continually write this stall bit until it is set or until a new setup has been received by checking the associated ENDPTSETUPSTAT bit.

4.3.7 Endpoint 1 to 3 control registers (ENDPTCTRL1 to ENDPTCTRL3)

Each endpoint that is not a control endpoint has its own register to set the endpoint type and enable or disable the endpoint.

Remark: The reset value for all endpoint types is the control endpoint. If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled, then the endpoint type of the unused direction must be changed from the control type to any other type (e.g. bulk). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint.

Table 124. USB Endpoint 1 to 3 control registers (ENDPTCTRL1 to ENDPPTCTRL3 - address 0x1900 01C4 to 0x1900 01CC) bit description

Bit	Symbol	Value	Description	Access	Reset value
0	RXS		Rx endpoint stall	R/W	0
		1	Endpoint stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software, or it will automatically be cleared upon receipt of a new SETUP request.[1]		
		0	Endpoint ok. This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.		

Table 124. USB Endpoint 1 to 3 control registers (ENDPTCTRL1 to ENDPPTCTRL3 - address 0x1900 01C4 to 0x1900 01CC) bit description *…continued*

Bit	Symbol	Value	Description	Access	Reset value
1	-		Reserved	R/W	0
3:2	RXT[1:0]		Endpoint type	R/W	00
		00	Control		
		01	Isochronous		
		10	Bulk		
		11	Reserved		
4	-	-	Reserved		
5	RXI		Rx data toggle inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packets regardless of their data PID.	R/W	0
		1	Enabled		
		0	Disabled		
6	RXR		Rx data toggle reset Write 1 to reset the PID sequence. Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device.	WS	0
7	RXE		Rx endpoint enable **Remark:** An endpoint should be enabled only after it has been configured.	R/W	0
		1	Endpoint enabled.		
		0	Endpoint disabled.		
15:8	-	-	reserved		
16	TXS		Tx endpoint stall	R/W	0
		1	Endpoint stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software, or it will automatically be cleared upon receipt of a new SETUP request.[1]		
		0	Endpoint ok. This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint, and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.		
17	-	-	Reserved	-	0
19:18	TXT[1:0]		Tx endpoint type	R/W	00
		00	Control		
		01	Isochronous		
		10	Bulk		
		11	Interrupt		
20	-	-	reserved		

Table 124. USB Endpoint 1 to 3 control registers (ENDPTCTRL1 to ENDPPTCTRL3 - address 0x1900 01C4 to 0x1900 01CC) bit description *…continued*

Bit	Symbol	Value	Description	Access	Reset value
21	TXI		Tx data toggle inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packets regardless of their data PID.	R/W	0
		1	Disabled		
		0	Enabled		
22	TXR		Tx data toggle reset Write 1 to reset the PID sequence. Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.	WS	1
23	TXE		Tx endpoint enable **Remark:** An endpoint should be enabled only after it has been configured	R/W	0
		1	Endpoint enabled.		
		0	Endpoint disabled.		
31:24	-	-	reserved		0

[1] For control endpoints only: There is a slight delay (50 clocks max) between the ENPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely that the DCD software will observe this delay. However, should the DCD notice that the stall bit is not set after writing a one to it, software should continually write this stall bit until it is set or until a new setup has been received by checking the associated ENDPTSETUPSTAT bit.

5. Functional description

5.1 OTG core

The OTG core forms the main digital part of the USB-OTG. See the *USB EHCI specification* for details about this core.

5.2 Host data structures

See *Chapter 4 of Enhanced Host Controller Interface Specification for Universal Serial Bus 1.0.*

5.3 Host operational model

See *Chapter 3 of Enhanced Host Controller Interface Specification for Universal Serial Bus 1.0*.

5.4 ATX_RGEN module

There are a number of requirements for the reset signal towards the ATX transceiver, these are as follows:

- it requires the clocks to be running for a reset to occur correctly.

- it must see a rising edge of reset to correctly reset the clock generation module.
- the reset must be a minimum of 133 ns (4 × 30 MHz clock cycles) in duration to reset all logic correctly.

The ATX_RGEN module generates a reset signal towards the ATX fulfilling above 3 requirements, no matter how the AHB reset looks like.

5.5 ATX transceiver

The USB-OTG has a USB transceiver with UTMI+ interface. It contains the required transceiver OTG functionality; this includes:

- VBUS sensing for producing the session-valid and VBUS-valid signals.
- sampling of the USB_ID input for detection of A-device or B-device connection.
- charging and discharging of VBUS for starting and ending a session as B-device.

5.6 Modes of operation

In general, the USB-OTG can be operating either in host mode or in device mode. Software must put the core in the appropriate mode by setting the USBMODE.CM field (‘11’ for host mode, ‘10’ for device mode).

The USBMODE.CM field can also be equal to ‘00’, which means that the core is in idle mode (neither host nor device mode). This will happen after the following:

- a hardware reset.
- a software reset via the USBCMD.RST bit; e.g. when switching from host mode to device mode as part of the HNP protocol (or vice versa), software must issue a software reset by which the core will be to the idle state; this will happen in a time frame dependent on the software.

5.7 SOF/VF indicator

The USB-OTG generates a SOF/VF indicator signal, which can be used by user specific external logic.

In FS mode, the SOF/VF indicator signal has a frequency equal to the frame frequency, which is about 1 kHz. The signal is high for half of the frame period and low for the other half of the frame period. The positive edge is aligned with the start of a frame (= SOF).

In HS mode, the SOF/VF indicator signal has a frequency equal to the virtual frame frequency. The signal is high for half of the virtual frame period and low for the other half of the virtual frame period. The positive edge is aligned with the start of a virtual frame (= VF).

The length of the virtual frame is defined as: VF = microframe × $2^{bInterval}$;

bInterval is specified in the 4-bit programmable BINTERVAL.BINT register field. The minimum value of bInterval is 0, the maximum value is 15.

In suspend mode the SOF/VF indicator signal is turned off (= remains low).

5.8 Hardware assist

The hardware assist provides automated response and sequencing that may not be possible in software if there are significant interrupt latency response times. The use of this additional circuitry is optional and can be used to assist the following three state transitions by setting the appropriate bits in the OTGSC register:

- Auto reset (set bit HAAR).
- Data pulse (set bit HADP).
- B-disconnect to A-connect (set bit HABA).

5.8.1 Auto reset

When the HAAR in the OTGSC register is set to one, the host will automatically start a reset after a connect event. This shortcuts the normal process where software is notified of the connect event and starts the reset. Software will still receive notification of the connect event (CCS bit in the PORTSC register) but should not write the reset bit in the USBCMD register when the HAAR is set. Software will be notified again after the reset is complete via the enable change bit in the PORTSC register which causes a port change interrupt.

This assist will ensure the OTG parameter TB_ACON_BSE0_MAX = 1 ms is met (see *OTG specification* for an explanation of the OTG timing requirements).

5.8.2 Data pulse

Writing a one to HADP in the OTGSC register will start a data pulse of approximately 7 ms in duration and then automatically cease the data pulsing. During the data pulse, the DP bit will be set and then cleared. This automation relieves software from accurately controlling the data-pulse duration. During the data pulse, the HCD can poll to see that the HADP and DP bit have returned low to recognize the completion, or the HCD can simply launch the data pulse and wait to see if a VBUS Valid interrupt occurs when the A-side supplies bus power.

This assist will ensure data pulsing meets the OTG requirement of > 5 ms and < 10 ms.

5.8.3 B-disconnect to A-connect (Transition to the A-peripheral state)

During HNP, the B-disconnect occurs from the OTG A_suspend state, and within 3 ms, the A-device must enable the pullup on the DP leg in the A-peripheral state. For the hardware assist to begin the following conditions must be met:

- HABA is set.
- Host controller is in suspend mode.
- Device is disconnecting.

The hardware assist consists of the following steps:

1. Hardware resets the OTG controller (writes 1 to the RST bit in USBCMD).
2. Hardware selects the device mode (writes 10 to bits CM[1:0] in USBMODE).
3. Hardware sets the RS bit in USBCMD and enables the necessary interrupts:
   - USB reset enable (URE) - enables interrupt on USB bus reset to device.
   - Sleep enable (SLE) - enables interrupt on device suspend.

– Port change detect enable (PCE) - enables interrupt on device connect.

When software has enabled this hardware assist, it must not interfere during the transition and should not write any register in the OTG core until it gets an interrupt from the device controller signifying that a reset interrupt has occurred or until it has verified that the core has entered device mode. HCD/DCD must not activate the core soft reset at any time since this action is performed by hardware. During the transition, the software may see an interrupt from the disconnect and/or other spurious interrupts (i.e. SOF/etc.) that may or may not cascade and my be cleared by the soft reset depending on the software response time.

After the core has entered device mode with help of the hardware assist, the DCD must ensure that the ENDPTLISTADDR is programmed properly before the host sends a setup packet. Since the end of the reset duration, which may be initiated quickly (a few microseconds) after connect, will require at a minimum 50 ms, this is the time for which the DCD must be ready to accept setup packets after having received notification that the reset has been detected or simply that the OTG is in device mode which ever occurs first.

If the A-peripheral fails to see a reset after the controller enters device mode and engages the D+-pullup, the device controller interrupts the DCD signifying that a suspend has occurred. This assist will ensure the parameter TA_BDIS_ACON_MAX = 3ms is met.

6. Deviations from EHCI standard

For the purposes of a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification. Device operation and On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is specific to the LPC31xx. The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator – Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- On-The-Go Operation - This design includes an On-The-Go controller.

6.1 Embedded Transaction Translator function

The USB-HS OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator. Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

6.1.1 Capability registers

The following items have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT bits added to HCSPARAMS – Host Control Structural Parameters (see Table 8–87).
- N_PTT added to HCSPARAMS – Host Control Structural Parameters (see Table 8–87).

6.1.2 Operational registers

The following items have been added to the operational registers to support the embedded TT:

- New register TTCTRL (see Section 8–4.2.7).
- Two-bit Port Speed (PSPD) bits added to the PORTSC1 register (see Section 8–4.2.14).

6.1.3 Discovery

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation. The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (i.e. Chirp completes successfully). Since this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in PORTSC1 (see Section 8–4.2.14).

Table 125. Handling of directly connected full-speed and low-speed devices

Standard EHCI model	EHCI with embedded Transaction Translator
After the port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After the port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSC1.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSC1.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X, where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (i.e. Split target hub).	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached, where HubAddr = TTHA (TTHA is programmable and defaults to 0) and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (i.e. Split target hub is the root hub).

6.1.4 Data structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator. Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) – Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - Hub Address = TTHA (default TTHA = 0)
   - Transactions to direct attached device/hub: QH.EPS = Port Speed
   - Transactions to a device downstream from direct attached FS hub: QH.EPS = Downstream Device Speed

     Remark: When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

     Maximum Packet Size must be less than or equal 64 or undefined behavior may result.
2. siTD (for direct attach FS) – Periodic (ISO Endpoint)

   all FS ISO transactions:

   Hub Address = (default TTHA = 0)

   siTD.EPS = 00 (full speed)

   Maximum Packet Size must less than or equal to 1023 or undefined behavior may result.

6.1.5 Operational model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification) and for the EHCI controller moving packets between system memory and a USB-HS hub. Since the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

6.1.5.1 Micro-frame pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators. Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

6.1.6 Split state machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator. Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 126. Split state machine properties

	Condition	Emulate TT response
Start-split	All asynchronous buffers full.	NAK
	All periodic buffers full.	ERR
	Success for start of Async. Transaction.	ACK
	Start Periodic Transaction.	No Handshake (Ok)
Complete-split	Failed to find transaction in queue.	Bus Time Out
	Transaction in Queue is Busy.	NYET
	Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

6.1.7 Asynchronous Transaction scheduling and buffer management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

1. *USB 2.0 specification, section 11.17.3*: Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.
2. *USB 2.0 specification, section 11.17.4*: Transaction tracking for 2 data pipes.
3. *USB 2.0 specification, section 11.17.5*: Clear_TT_Buffer capability provided though the use of the TTCTRL register.

6.1.8 Periodic Transaction scheduling and buffer management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

1. *USB 2.0 specs, section 11.18.6.[1-2]*:
   - Abort of pending start-splits:

     EOF (and not started in micro-frames 6)

     Idle for more than 4 micro-frames
   - Abort of pending complete-splits:

     EOF

     Idle for more than 4 micro-frames
2. *USB 2.0 specs, section 11.18.6.[7-8]*:
   - Transaction tracking for up to 16 data pipes:

Some applications may not require transaction tracking up to a maximum of 16 periodic data pipes. The option to limit the tracking to only 4 periodic data pipes exists in the by changing the configuration constant VUSB_HS_TT_PERIODIC_CONTEXTS to 4. The result is a significant gate count savings to the core given the limitations implied.

Remark: Limiting the number of tracking pipes in the EMBedded TT to four (4) will impose the restriction that no more than 4 periodic transactions (INTERRUPT/ISOCHRONOUS) can be scheduled through the embedded TT per frame. The number 16 was chosen in the USB specification because it is sufficient to ensure that the high-speed to full- speed periodic pipeline can remain full. keeping the pipeline full puts no constraint on the number of periodic transactions that can be scheduled in a frame and the only limit becomes the flight time of the packets on the bus.

– Complete-split transaction searching:

  There is no data schedule mechanism for these transactions other than micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

6.1.9 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the HCSPARAMS – Host Control Structural Parameters register.

6.2 Device operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

6.2.1 USBMODE register

Given that the dual-role controller is initialized in neither host nor device mode, the USBMODE register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

6.2.2 Non-Zero Fields the register file

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode.

6.2.3 SOF interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode. EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USBSTS (Section 8–4.2.2) and USBINTR (Section 8–4.2.3) registers.

6.3 Miscellaneous variations from EHCI

6.3.1 Discovery

6.3.1.1 Port reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSCx register for a duration of 10 ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a ‘1’ to the reset the device.
- Software shall write a ‘0’ to the reset the device after 10 ms.

  This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a ‘0’ to the reset bit while a reset is in progress the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

6.3.1.2 Port speed detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed – This information is redundant with the 2-bit Port Speed indicator above.

7. Device data structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this chapter support a 32-bit memory buffer address space.

Remark: The Software must ensure that no interface data structure reachable by the Device controller crosses a 4k-page boundary

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

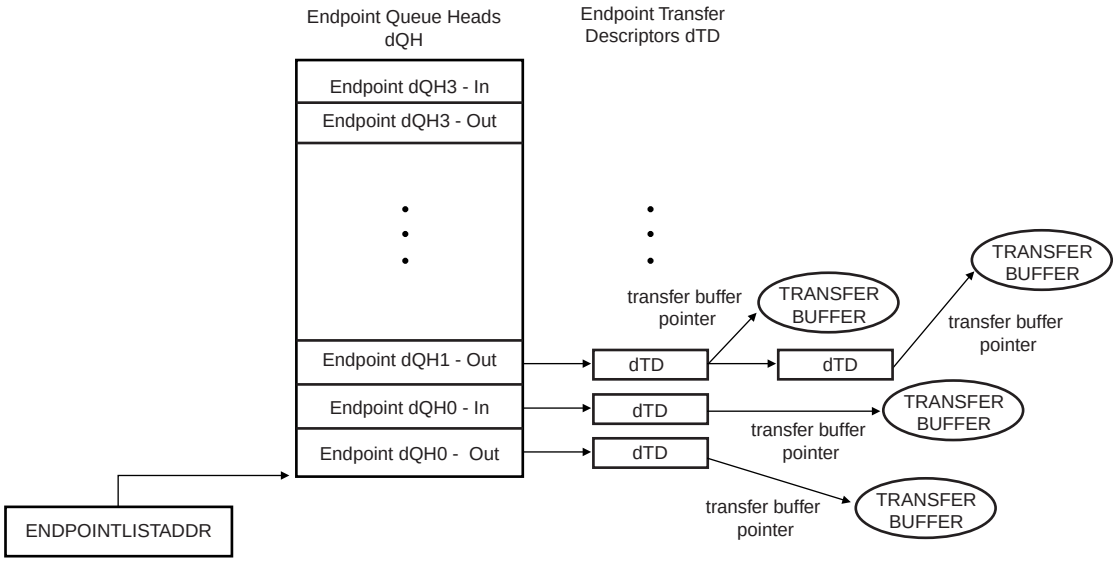


Fig 23. Endpoint queue head organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the ENDPOINTLISTADDR pointer. The even –numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

Remark: The Endpoint Queue Head List must be aligned to a 2k boundary.

7.1 Endpoint queue head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the

dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

7.1.1 Endpoint capabilities and characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 127. Endpoint capabilities and characteristics

Access	Bit	Name	Description
RO	31:30	MULT	Number of packets executed per transaction descriptor 00 - Execute N transactions as demonstrated by the USB variable length protocol where N is computed using Max_packet_length and the Total_bytes field in the dTD. 01 - Execute one transaction 10 - Execute two transactions 11 - Execute three transactions **Remark:** Non-isochronous endpoints must set MULT = 00. **Remark:** Isochronous endpoints must set MULT = 01, 10, or 11 as needed.
RO	29	ZLT	Zero length termination select This bit is used for non-isochronous endpoints to indicate when a zero-length packet is received to terminate transfers in case the total transfer length is “multiple”. 0 - Enable zero-length packet to terminate transfers equal to a multiple of Max_packet_length (default). 1 - Disable zero-length packet on transfers that are equal in length to a multiple Max_packet_length.
RO	28:27	-	reserved
RO	26:16	Max_packet_length	Maximum packet size of the associated endpoint (< 1024)
RO	15	IOS	Interrupt on setup This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
RO	14:0	-	reserved

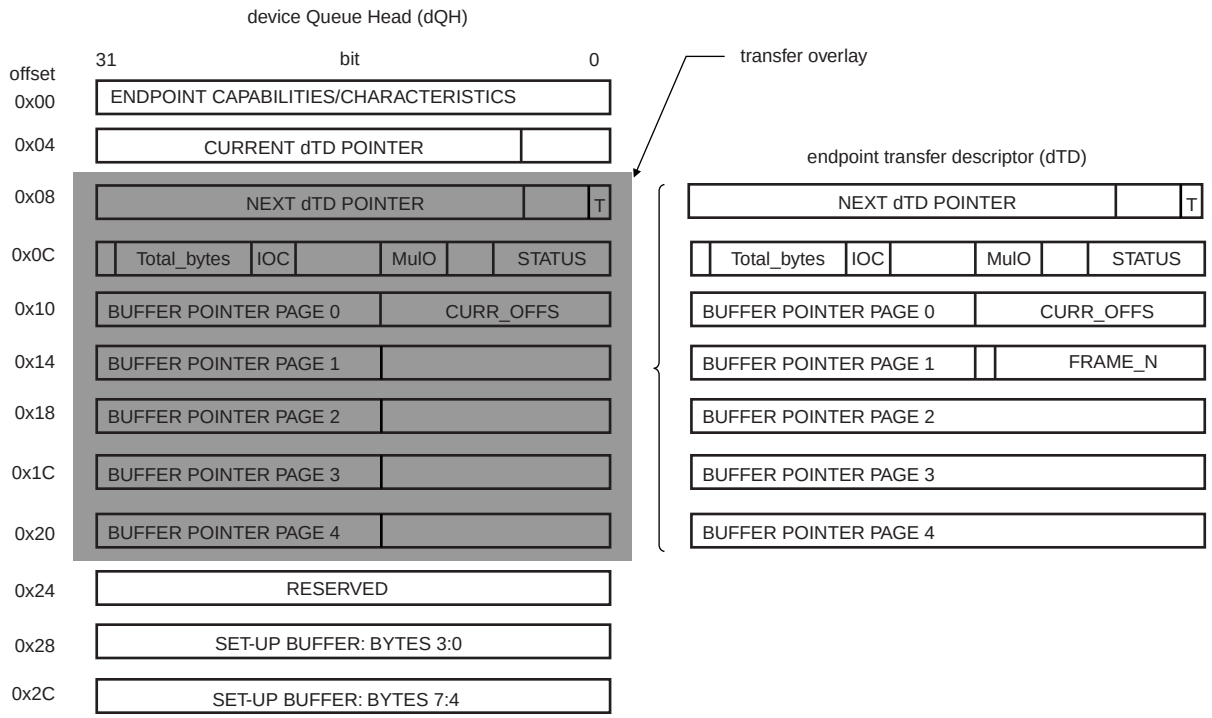


Fig 24. Endpoint queue head data structure

7.1.2 Transfer overlay

The seven DWords in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue. See dTD for a description of the overlay fields.

7.1.3 Current dTD pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

Table 128. Current dTD pointer

Access	Bit	Name	Description
R/W (hardware only)	31:5	Current_TD_pointer	Current dTD pointer This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the device controller to the next dTD pointer during endpoint priming or queue advance.
-	4:0	-	reserved

7.1.4 Set-up buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

Remark: Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

Table 129. Set-up buffer

Dword	Access	Bit	Name	Description
1	R/W	31:0	BUF0	Setup buffer 0 This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	R/W	31:0	BUF1	Setup buffer 1 This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

7.2 Endpoint transfer descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in Section 8–8.11.

Table 130. Next dTD pointer

Access	Bit	Name	Description
RO	31:5	Next_link_pointer	Next link pointer This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
	4:1	-	reserved
	0	T	Terminate This bit indicates to the device controller when there are no more valid entries in the queue. 1 - pointer is invalid 0 - Pointer is valid, i.e. pointer points to a valid transfer element descriptor.

Table 131. dTD token

Access	Bit	Name	Description
-	31	-	reserved
R/W	30:16	Total_bytes	Total bytes This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and it is decremented only when the transaction has been completed successfully. The maximum value software can write into this field is 0x5000 (5 x 4 kB) for the maximum number of bytes five page pointers can access. Although it is possible to create a transfer up to 20 kB this assumes that the first offset into the first page is zero. When the offset cannot be predetermined, crossing past the fifth page can be guaranteed by limiting the total bytes to 16 kB. Therefore, the maximum recommended Total-Bytes = 16 kB (0x4000). If Total_bytes = 0 when the host controller fetches this transfer descriptor and the active bit is set in the Status field of this dTD, the device controller executes a zero-length transaction and retires the dTD. **Remark:** For IN transfers, it is not a requirement that Total_bytes is an even multiple of Max_packet_length. If software builds such a dTD, the last transaction will always be less than Max_packet_length.
RO	15	IOC	Interrupt on complete This bit is used to indicate if USBINT will be set when the device controller is finished with this dTD. 1 - USBINT set. 0 - USBINT not set.
-	14:12	-	reserved

Table 131. dTD token *…continued*

Access	Bit	Name	Description
RO	11:10	MultO	Multiplier Override (see Section 8–7.2.1 for an example) This field can be used for transmit ISOs to override the MULT field in the dQH. This field must be zero for all packet types that are not transmit-ISO. 00 - Execute N transactions as demonstrated by the USB variable length protocol where N is computed using Max_packet_length and the Total_bytes field in the dTD. 01 - Execute one transaction 10 - Execute two transactions 11 - Execute three transactions **Remark:** Non-ISO and Non-TX endpoints must set MultO="00".
	9:8	-	reserved
R/W	7:0	Status	Status This field is used by the device controller to communicate individual execution states back to the software. This field contains the status of the last transaction performed on this dTD. Bit 7 = 1 - status: Active Bit 6 = 1 - status: Halted Bit 5 = 1 - status: Buffer Error Bit 4 - reserved Bit 3 = 1 - status: Transaction Error Bit 2 - reserved Bit 1 - reserved Bit 0 - reserved

Table 132. dTD buffer page pointer list

Access	Bit	Name	Description
RO	31:12	BUFF_P	Selects the page offset in memory for the packet buffer. Non-virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
	page 0: 11:0	CURR_OFFS	Offset into the 4 kB buffer where the packet is to begin.
	page 1: 10:0	FRAME_N	Written by the device controller to indicate the frame number in which a packet finishes. This is typically used to correlate relative completion times of packets on an isochronous endpoint.

7.2.1 Determining the number of packets for Isochronous IN endpoints

The following examples show how the MULT field in the dQH and the MultO in the dTD are used to control the number of packets sent in an In-transaction for an isochronous endpoint:

Example 1

MULT = 3; Max_packet_size = 8; Total_bytes = 15; MultO = 0 (default)

In this case three packets are sent: Data2 (8 bytes), Data1 (7 bytes), Data0 (0 bytes).

Example 2

MULT = 3; Max_packet_size = 8; Total_bytes = 15; MultO = 2

In this case two packets are sent: Data1 (8 bytes), Data0 (7 bytes).

To optimize efficiency for IN transfers, software should compute MultO = greatest integer of (Total_bytes/Max_packet_size). If Total_bytes = 0, then MultO should be 1.

8. Device operational model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

8.1 Device controller initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB_DM is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.

   Remark: Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

2. Allocate and Initialize device queue heads in system memory (see Section 8–7).

   Minimum: Initialize device queue heads 0 Tx & 0 Rx.

   Remark: All device queue heads associated with control endpoints must be initialized before the control endpoint is enabled. Non-Control device queue heads must be initialized before the endpoint is used and not necessarily before the endpoint is enabled.

3. Configure ENDPOINTLISTADDR Pointer (see Section 8–4.2.6).

4. Enable the microprocessor interrupt associated with the USB-HS core.

   Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend (see Table 8–96).

5. Set Run/Stop bit to Run Mode.

   After the Run bit is set, a device reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.

Remark: Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework command set (see *USB Specification Rev. 2.0*, *chapter 9*).

8.2 Port state and control

From a chip or system reset, the device controller enters the powered state. A transition from the powered state to the attach state occurs when the Run/Stop bit is set to a '1'. After receiving a reset on the bus, the port will enter the defaultFS or defaultHS state in accordance with the reset protocol described in *Appendix C.2 of the USB Specification Rev. 2.0*. The following state diagram depicts the state of a USB 2.0 device.

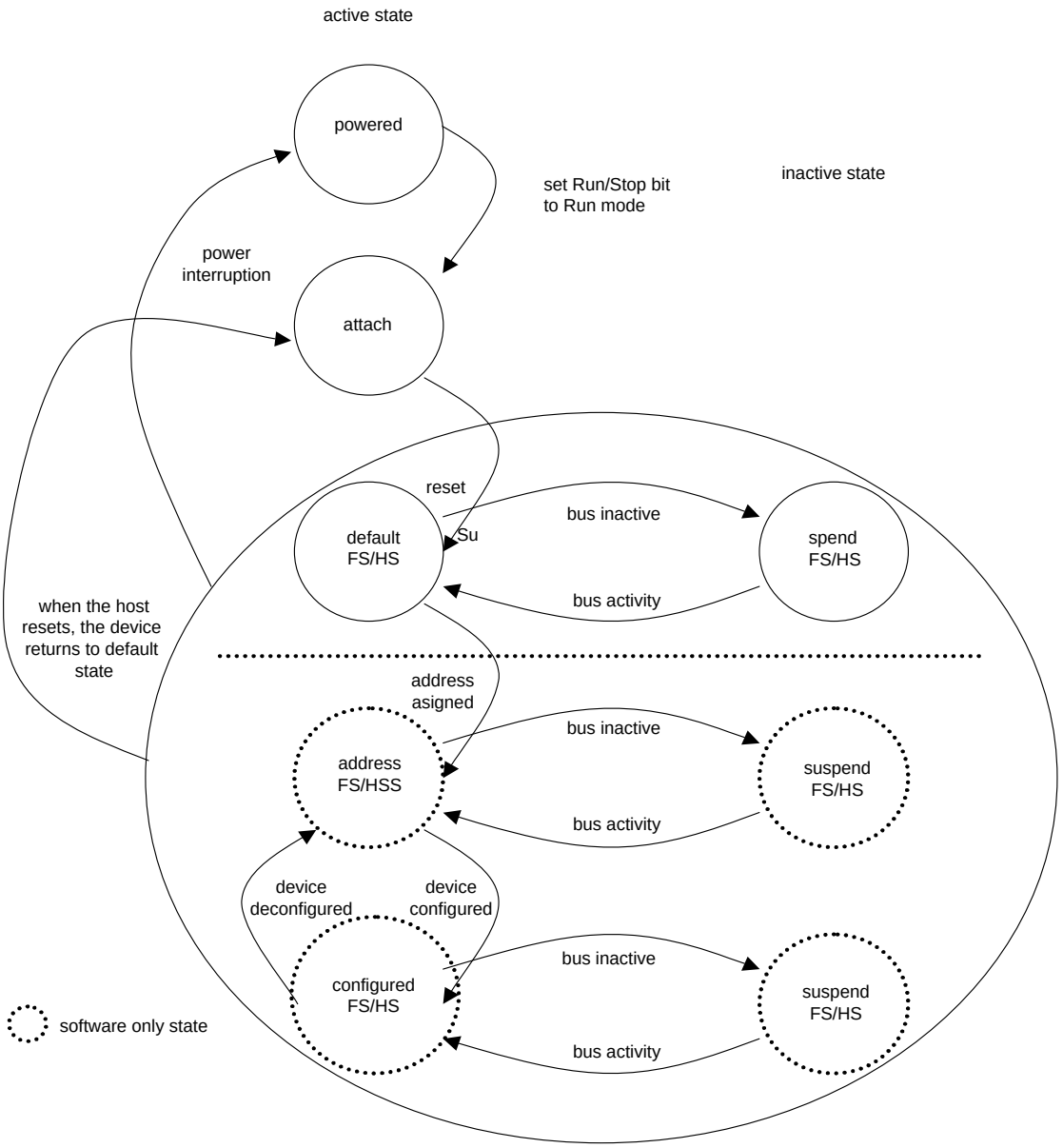


Fig 25. Device state diagram

The states powered, attach, default FS/HS, suspend FS/HS are implemented in the device controller and are communicated to the DCD using the following status bits:

- DCSuspend - see Table 8–94.
- USB reset received - see Table 8–94.
- Port change detect - see Table 8–94.
- High-speed port - see Table 8–112.

It is the responsibility of the DCD to maintain a state variable to differentiate between the DefaultFS/HS state and the Address/Configured states. Change of state from Default to Address and the configured states is part of the enumeration process described in the *device framework section of the USB 2.0 Specification*.

As a result of entering the Address state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the Configured indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

8.3 Bus reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

- Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.
- Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.
- Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFFFFFF to ENDPTFLUSH.
- Read the reset bit in the PORTSCx register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare).

  Remark: A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.
- Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated. At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.
- After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSCx to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the *USB2.0 specification Chapter 9 - Device Framework*.

  Remark: The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.

8.4 Suspend/resume

8.4.1 Suspend

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

8.4.1.1 Operational model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming DC Suspend Interrupt is enabled). When the DCSuspend bit in the PORTSCx is set to a ‘1’, the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation. Information on the bus power limits in suspend state can be found in *USB 2.0 specification*.

8.4.2 Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a ‘1’ to the Resume bit in the in the PORTSCx while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

Remark: Before resume signaling can be used, the host must enable it by using the Set Feature command defined in *device framework (chapter 9) of the USB 2.0 Specification*.

8.5 Managing endpoints

The *USB 2.0 specification* defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a control type data channel used for device discovery and enumeration. Other types of endpoints support by USB include bulk, interrupt, and isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the *USB 2.0 specification*.

The LPC3130/31 supports up to four endpoints.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1- OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a queue head allocated in memory. If the maximum of 4 endpoint numbers, one for each endpoint direction are being used by the device controller, then 8 queue heads are required. The operation of an endpoint and use of queue heads are described later in this document.

8.5.1 Endpoint initialization

After hardware reset, all endpoints except endpoint zero are un-initialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRLx register (see Table 8–124). Each 32-bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

Table 133. Device controller endpoint initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 - control
	01 - isochronous
	10 - bulk
	11 - interrupt
Endpoint Stall	0

8.5.2 Stalling

There are two occasions where the device controller may need to return to the host a STALL:

1. The first occasion is the **functional stall**, which is a condition set by the DCD as described in the *USB 2.0 device framework (chapter 9)*. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.
2. A **protocol stal**l, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

Remark: Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (i.e. perform a read-modify-write).

Table 134. Device controller stall response matrix

USB packet	Endpoint STALL bit	Effect on STALL bit	USB response
SETUP packet received by a non-control endpoint.	N/A	None	STALL
IN/OUT/PING packet received by a non-control endpoint.	1	None	STALL
IN/OUT/PING packet received by a non-control endpoint.	0	None	ACK/NAK/NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint.	1	None	STALL
IN/OUT/PING packet received by a control endpoint.	0	None	ACK/NAK/NYET

8.5.3 Data toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer *to the USB 2.0 specification.*

8.5.3.1 Data toggle reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a ‘1’ to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

8.5.3.2 Data toggle inhibit

Remark: This feature is for test purposes only and should never be used during normal device controller operation.

Setting the data toggle Inhibit bit active (‘1’) causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state. In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

8.6 Operational model for packet transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were designed so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is designed in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as “priming” the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be designed properly to use priming. Further, note that the term “flushing” is used to describe the action of clearing a packet that was queued for execution.

8.6.1 Priming transmit endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received. After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to four endpoints.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB. Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the

transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. On the LPC3130/31, 128 x 36 bit dual port memory FIFOs are used for each IN endpoint.

8.6.2 Priming receive endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host. Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

8.7 Interrupt/bulk endpoint operational model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 135. Variable length transfer protocol example (ZLT = 0)

Bytes (dTD)	Max Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	-
512	256	3	256	256	0
512	512	2	512	0	-

Table 136. Variable length transfer protocol example (ZLT = 1)

Bytes (dTD)	Max Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	-
512	256	2	256	256	-
512	512	1	512	-	-

Remark: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

TX-dTD is complete when all packets described dTD were successfully transmitted.Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction. On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

Remark: All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

8.7.1 Interrupt/bulk endpoint bus response matrix

Table 137. Interrupt/bulk endpoint bus response matrix

Token type	STALL	Not primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	n/a	n/a
In	STALL	NAK	Transmit	BS error	n/a
Out	STALL	NAK	Receive and NYET/ACK	n/a	NAK
Ping	STALL	NAK	ACK	n/a	n/a
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

[1] BS error = Force Bit Stuff Error

[2] NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

[3] SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

8.8 Control endpoint operational model

8.8.1 Setup phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

In hardware the setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

8.8.1.1 Setup Packet Handling using setup lockout mechanism

After receiving an interrupt and inspecting USBMODE to determine that a setup packet was received on a particular pipe:

1. Duplicate contents of dQH.SsetupBuffer into local software byte array.
2. Write '1' to clear corresponding ENDPTSETUPSTAT bit and thereby disabling Setup Lockout (i.e. the Setup Lockout activates as soon as a setup arrives. By writing to the ENDPTSETUPSTAT, the device controller will accept new setup packets.).
3. Process setup packet using local software byte array copy and execute status/handshake phases.

   Remark: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.
4. Before priming for status/handshake phases ensure that ENDPTSETUPSTAT is ‘0’. The time from writing a ‘1’ to ENDPTSETUPSTAT and reading back a ‘0’ may vary according to the type of traffic on the bus up to nearly a 1ms, however the it is absolutely necessary to ensure ENDPTSETUPSTAT has transitioned to ‘0’ after step 1) and before priming for the status/handshake phases.

Remark: To limit the exposure of setup packets to the setup lockout mechanism (if used), the DCD should designate the priority of responding to setup packets above responding to other packet completions

8.8.1.2 Setup Packet Handling using trip wire mechanism

- Disable Setup Lockout by writing ‘1’ to Setup Lockout Mode (SLOM) in USBMODE. (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

  Remark: Leaving the Setup Lockout Mode As ‘0’ will result in pre-2.3 hardware behavior.
- After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:

  a. Write '1' to clear corresponding bit ENDPTSETUPSTAT.

  b. Duplicate contents of dQH.SetupBuffer into local software byte array.

  c. Write ‘1’ to Setup Tripwire (SUTW) in USBCMD register.

  d. Read Setup TripWire (SUTW) in USBCMD register. (if set - continue; if cleared - go to b).

  e. Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.

f. Process setup packet using local software byte array copy and execute status/handshake phases.

g. Before priming for status/handshake phases ensure that ENDPTSETUPSTAT is '0'.

- A poll loop should be used to wait until ENDPTSETUPSTAT transitions to '0' after step a) above and before priming for the status/handshake phases.
- In core versions 3.2 and later, the time from writing a '1' to ENDPTSETUPSTAT and reading back a '0' is very short (~1-2 us) so a poll loop in the DCD will not be harmful.
- In core versions 3.1 and earlier, the time from writing a '1' to ENDPTSETUPSTAT and reading back a '0' may vary according to the type of traffic on the bus up to nearly a 1ms, however the it is absolutely necessary to ensure ENDPTSETUPSTAT has transitioned to '0' after step a) and before priming for the status/handshake phases.

Remark: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

8.8.2 Data phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, i.e. The ENDPTPRIME bit goes to zero and the ENDPTSTATUS bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTATUS) to enforce data coherency with the setup packet.

Remark: The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

Remark: Error handling of data phase packets is the same as bulk packets described previously.

8.8.3 Status phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

Remark: The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

Remark: Error handling of data phase packets is the same as bulk packets described previously.

8.8.4 Control endpoint bus response matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

Table 138. Control endpoint bus response matrix

Token type	Endpoint sate					Setup lockout
	STALL	Not primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	n/a	SYSERR	-
In	STALL	NAK	Transmit	BS error	n/a	n/a
Out	STALL	NAK	Receive and NYET/ACK	n/a	NAK	n/a
Ping	STALL	NAK	ACK	n/a	n/a	n/a
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	ignore

[1] BS error = Force Bit Stuff Error

[2] NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

[3] SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

8.9 Isochronous endpoint operational model

Isochronous endpoints are used for real-time scheduled delivery of data, and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller is accomplished by the following:

- Exactly MULT Packets per (micro) Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro) frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro) frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro) frame. Once an ISO transaction is started in a (micro) frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition. The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data. Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the Transaction Error bit and the data is stored as usual for the application software to sort out.

TX packet retired

- MULT counter reaches zero.
- Fulfillment Error [Transaction Error bit is set].
- # Packets Occurred > 0 AND # Packets Occurred < MULT.

Remark: For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

RX packet retired

- MULT counter reaches zero.
- Non-MDATA Data PID is received.

  Remark: Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
- Overflow Error:
  - Packet received is > maximum packet length. [Buffer Error bit is set].
  - Packet received exceeds total bytes allocated in dTD. [Buffer Error bit is set].
- Fulfillment error [Transaction Error bit is set]:

  # Packets Occurred > 0 AND # Packets Occurred < MULT.
- CRC Error [Transaction Error bit is set]

Remark: For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro) frame to (micro) frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro) frames.

8.9.1 Isochronous pipe synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro) frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (micro) frame number [N], the DCD should interrupt on SOF during

frame N-1. When the FRINDEX=N–1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro) frame N–1 so that the device controller will execute delivery during (micro) frame N.

Remark: Priming an endpoint towards the end of (micro) frame N-1 will not guarantee delivery in (micro) frame N. The delivery may actually occur in (micro) frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

8.9.2 Isochronous endpoint bus response matrix

Table 139. Isochronous endpoint bus response matrix

Token type	STALL	Not primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	n/a	n/a
In	NULL packet	NULL packet	Transmit	BS error	n/a
Out	Ignore	Ignore	Receive	n/a	Drop packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

[1] BS error = Force Bit Stuff Error

[2] NULL packet = Zero length packet.

8.10 Managing queue heads

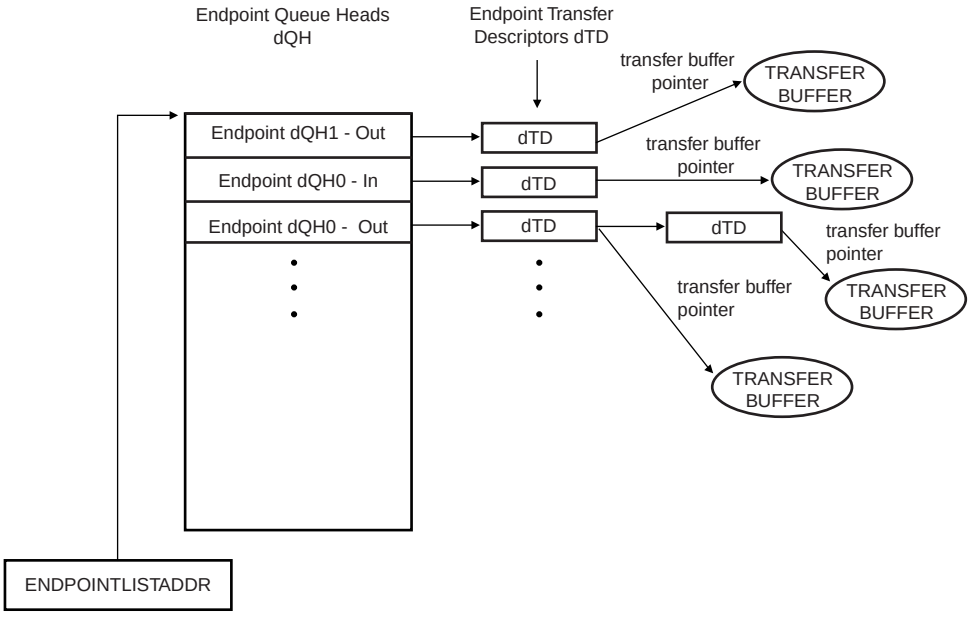


Fig 26. Endpoint queue head diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown

Figure 8–26. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see Section 8–8.11.1).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multiplier, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

8.10.1 Queue head initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the *USB Chapter 9* or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the *USB Chapter 9 protocol*. Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.
- Write the next dTD Terminate bit field to "1".
- Write the Active bit in the status field to "0".
- Write the Halt bit in the status field to "0".

Remark: The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

8.10.2 Operational model for setup transfers

As discussed in section Control Endpoint Operational Model (Section 8–8.8), setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

   Remark: The acknowledge must occur before continuing to process the setup packet.

   Remark: After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH – RX. Only the local software copy should be examined.
3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.

Remark: It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

4. Decode setup packet and prepare data phase [optional] and status phase transfer as require by the *USB Chapter 9* or application specific protocol.

8.11 Managing transfers with transfer descriptors

8.11.1 Software link pointers

It is necessary for the DCD software to maintain head and tail pointers to the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

Remark: To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

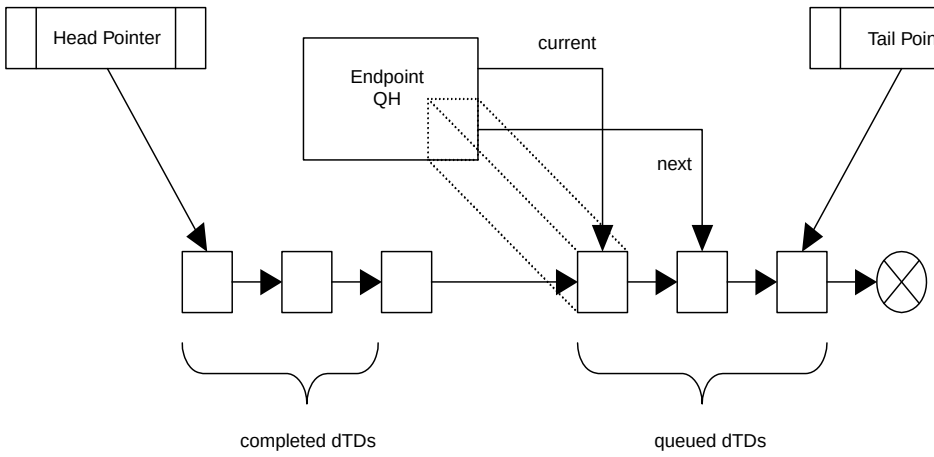


Fig 27. Software link pointers

8.11.2 Building a transfer descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs:

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to “00000”.

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to “1”.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.

5. Initialize the status field with the active bit set to “1” and all remaining status bits set to “0”.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

8.11.3 Executing a transfer descriptor

To safely add a dTD, the DCD must follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty: Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

Link list is empty

1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
2. Clear active and halt bits in dQH (in case set from a previous error).
3. Prime endpoint by writing ‘1’ to correct bit position in ENDPTPRIME.

Link list is not empty

1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
2. Read correct prime bit in ENDPTPRIME – if ‘1’ DONE.
3. Set ATDTW bit in USBCMD register to ‘1’.
4. Read correct status bit in ENDPTSTAT. (Store in temp variable for later).
5. Read ATDTW bit in USBCMD register.
   - If ‘0’ go to step 3.
   - If ‘1’ continue to step 6.
6. Write ATDTW bit in USBCMD register to ‘0’.
7. If status bit read in step 4 (ENDPSTAT reg) indicates endpoint priming is DONE (corresponding ERBRx or ETBRx is one): DONE.
8. If status bit read in step 4 is 0 then go to Linked list is empty: Step 1.

8.11.4 Transfer completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

Remark: Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

Active = 0

Halted = 0

Transaction Error = 0

Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix (see Table 8–140).

In addition to checking the status bit, the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

8.11.5 Flushing/De-priming an endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a ‘1’ to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are ‘0’.

   Remark: Software note: This operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read ENDPTSTAT to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now ‘0’. If the corresponding bits are ‘1’ after step #2 has finished, then the flush failed as described in the following:

   In very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush by repeating steps 1-3 until each endpoint is successfully flushed.

8.11.6 Device error matrix

The Table 8–140 summarizes packet errors that are not automatically handled by the Device Controller.

The following errors can occur:

Overflow: Number of bytes received exceeded max. packet size or total buffer length. This error will also set the Halt bit in the dQH, and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.

ISO packet error: CRC Error on received ISO packet. Contents not guaranteed to be correct.

ISO fulfillment error: Host failed to complete the number of packets defined in the dQH mult field within the given (micro) frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro) frame. During the “dead” (micro) frame, the Device Controller reports error on the pipe and primes for the following frame

Table 140. Device error matrix

Error	Direction	Packet type	Data buffer error bit	Transaction error bit
Overflow	Rx	Any	1	0
ISO packet error	Rx	ISO	0	1
ISO fulfillment error	Both	ISO	0	1

8.12 Servicing interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

8.12.1 High-frequency interrupts

High frequency interrupts in particular should be handed in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 141. High-frequency interrupt events

Execution order	Interrupt	Action
1a	USB interrupt: ENDPTSETUPSTATUS [1]	Copy contents of setup buffer and acknowledge setup packet (as indicated in Section 8–8.10). Process setup packet according to *USB 2.0 Chapter 9* or application specific protocol.
1b	USB interrupt: ENDPTCOMPLETE[1]	Handle completion of dTD as indicated in Section 8–8.10.
2	SOF interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

[1] It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

8.12.2 Low-frequency interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don’t occur often in comparison to the high-frequency interrupts.

Table 142. Low-frequency interrupt events

Interrupt	Action
Port change	Change software state information.
Sleep enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

8.12.3 Error interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Table 143. Error interrupt events

Interrupt	Action
USB error interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

9. USB power optimization

The USB-HS core is a fully synchronous static design. The power used by the design is dependent on the implementation technology used to fabricate the design and on the application usage of the core. Applications that transfer more data or use a greater number of packets to be sent will consume a greater amount of power.

Because the design is synchronous and static, power may be conserved by reducing the transitions of the clock net. This may be done in several ways.

1. Reduce the clock frequency to the core. The clock frequency may not be reduced below the minimum recommended operating frequency of the core without first disabling the USB operation.
2. Reduce transition on the clock net through the use of clock gating methods. (The LPC3130/31 is synthesized using this mechanism).
3. The clock may be shut off to the core entirely to conserve power. Again this may only be done after the USB operations on the bus have been disabled.

A device may suspend operations autonomously by disconnecting from the USB, or, in response to the suspend signaling, the USB has moved it into the suspend state. A host can suspend operation autonomously, or it can command portions or the entire USB to transition into the suspend state.

9.1 USB power states

The USB provides a mechanism to place segments of the USB or the entire USB into a low-power suspend state. USB bus powered devices are required to respond to a 3ms lack of activity on the USB bus by going into a suspend state. In the USB-HS core software is notified of the suspend condition via the transition in the PORTSC register. Optionally an interrupt can be generated which is controlled by the port change Detect Enable bit in the USBINTR control register. Software then has 7 ms to transition a bus powered device into the suspend state. In the suspend state, a USB device has a maximum USB bus power budget of 500μA. In general, to achieve that level of power conservation, most of the device circuits will need to be switched off, or clock at an extremely low frequency. This can be accomplished by suspending the clock.

The implementation of low power states in the USB-HS core is dependant on the use of the device role (host or peripheral), whether the device is bus powered, and the selected clock architecture of the core.

Bus powered peripheral devices are required by the USB specification to support a low power suspend state. Self powered peripheral devices and hosts set their own power management strategies based on their system level requirements. The clocking architecture selected is important to consider as it determines what portions of the design will remain active when transitioned into the low power state.

Before the system clock is suspended or set to a frequency that is below the operational frequency of the USB-HS core, the core must be moved from the operational state to a low power state. The power strategies designed into the USB-HS core allow for the most challenging case, a self powered device that is clocked entirely by the transceiver clock.

9.2 Device power states

A bus powered peripheral device must move through the power states as directed by the host. Optionally autonomously directed low power states may be implemented.

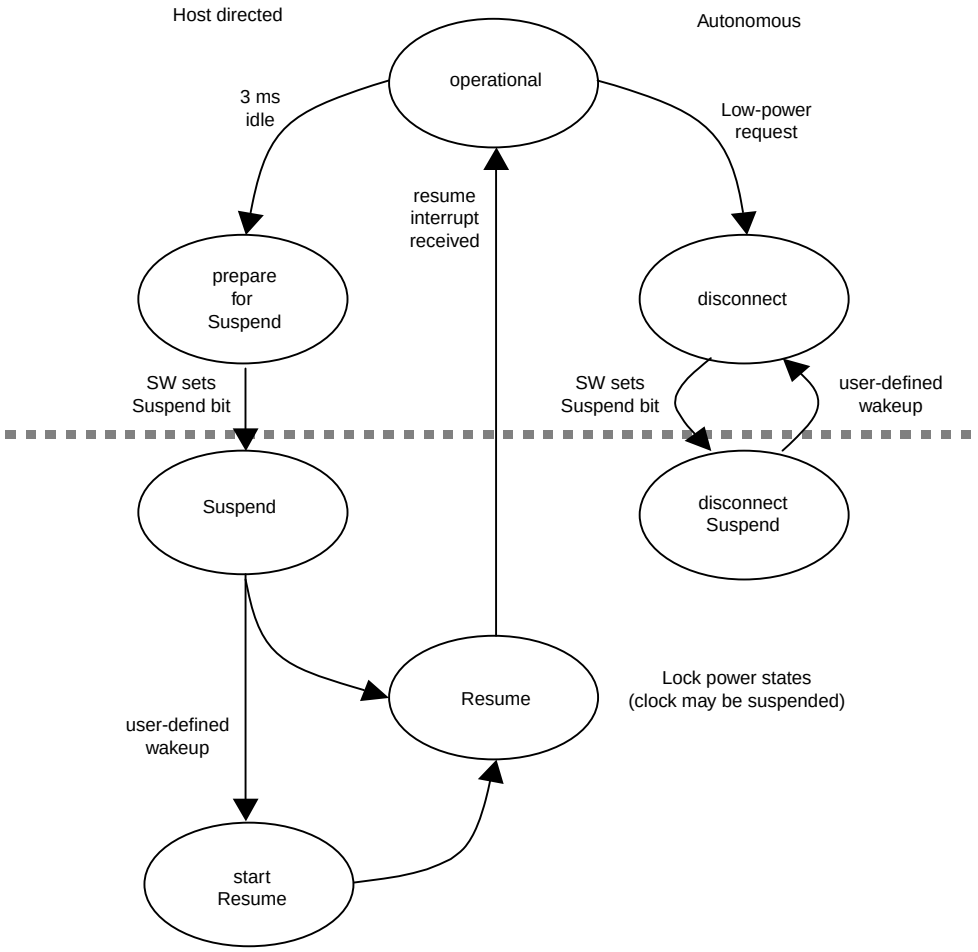


Fig 28. Device power state diagram

In the operational state both the transceiver clock and system clocks are running. Software can initiate a low power mode autonomously by disconnecting from the host to go into the disconnect state. Once in this state, the software can set the Suspend bit to turn off the transceiver clock putting the system in to the disconnect-suspend state. Since software cannot depend on the presents of a clock to clear the Suspend bit, a wakeup event must be defined which would clear the Suspend bit and allow the transceiver clock to resume.

The device can also go into suspend mode as a result of a suspend command from the host. Suspend is signaled on the bus by 3ms of idle time on the bus. This will generate a suspend interrupt to the software at which point the software must prepare to go into suspend then set the suspend bit. Once the Suspend bit is set the transceiver clock may turn off and the device will be in the suspended state. The device has two ways of getting out of suspend.

1. If remote wake-up is enabled, a wakeup event could be defined which would clear the Suspend bit. The software would then initiate the resume by setting the Resume bit in the port controller then waiting for a port change interrupt indicating that the port is in an operational state.
2. If the host puts resume signaling on the bus, it will clear the Suspend bit and generate a port change interrupt when the resume is finished.

In either case the system designer must insure an orderly restoration of the power and clocks to the suspended circuitry.

9.3 Host power states

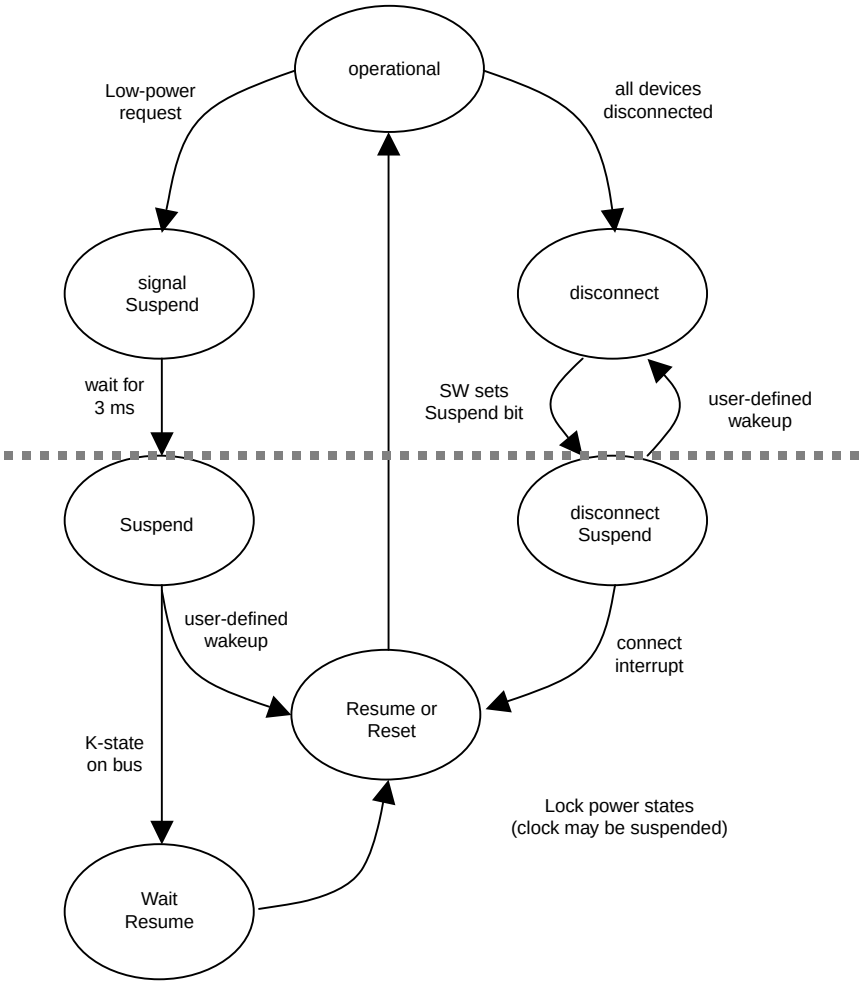


Fig 29. Host/OTG power state diagram

From an operational state when a host gets a low power request, it must set the suspend bit in the port controller. This will put an idle on the bus, block all traffic through the port, and turn off the transceiver clock. There are two ways for a host controller to get out of the suspend state. If it has enabled remote wake-up, a K-state on the bus will turn the transceiver clock and generate an interrupt. The software will then have to wait 20 ms for the resume to complete and the port to go back to an active state. Alternatively an external event could clear the suspend bit and start the transceiver clock running again. The software can then initiate a resume by setting the resume bit in the port controller, or force a reconnect by setting the reset bit in the port controller.

If all devices have disconnected from the host, the host can go into a low power mode by the software setting the suspend bit. From the disconnect-suspend state a connect event would start the transceiver clock and interrupt the software. The software would then need to set the reset bit to start the connect process.

9.4 Susp_CTRL module

The SUSP_CTRL module implements the power management logic of USB-OTG. It controls the suspend input of the transceiver. Asserting this suspend signal will put the transceiver in suspend mode and the generation of the 30 MHz clock and 60 MHz clock will be switched off.

A suspend control input of the transceiver (otg_on) that was previously tied high and prevented the transceiver to go into full suspend mode, has been connected to bit 0 of the USB_OTG_CFG register in the SYSCREG module (see Table 26–525). This bit is low by default and only needs to be set high in OTG Host mode operation.

In suspend mode, the transceiver will raise an output signal indicating that the PLL generating the 480 MHz clock can be switched off.

The SUSP_CTRL module also generates an output signal indicating whether the AHB clock is needed or not. If '0' the AHB clock is allowed to be switched off or reduced in frequency in order to save power.

The core will enter the low power state if:

- Software sets the PORTSC.PHCD bit.

When operating in host mode, the core will leave the low power state on one of the following conditions:

- software clears the PORTSC.PHCD bit
- a device is connected and the PORTSC.WKCN bit is set
- a device is disconnected an the PORTSC.WKDC bit is set
- an over-current condition occurs and the PORTSC.WKOC bit is set
- a remote wake-up from the attached device occurs (when USB bus was in suspend)
- the extra external user wake-up input is asserted (active low). This input is controlled by a register in the SYSCREG module (see Table 26–525).
- a change on vbusvalid occurs (= VBUS threshold at 4.4 V is crossed)
- a change on bvalid occurs (=VBUS threshold at 4.0 V is crossed).

When operating in device mode, the core will leave the low power state on one of the following conditions:

- software clears the PORTSC.PHCD bit.
- a change on the USB data lines (dp/dm) occurs.
- the extra external user wake-up input is asserted (active low). This input is controlled by a register in the sys_creg.
- a change on vbusvalid occurs (= VBUS threshold at 4.4 V is crossed).
- a change on bvalid occurs (= VBUS threshold at 4.0 V is crossed).

The vbusvalid and bvalid signals coming from the transceiver are not filtered in the SUSP_CTRL module. Any change on those signals will cause a wake-up event.

Input signals 'host_wakeup_n' and 'dev_wakeup_n' are extra external wake-up signals (for host mode and device mode respectively). However the detection of all USB related wake-up events is already handled in the SUSP_CTRL mode. Therefore in normal situations these signals can be tied high (= inactive).

Important note: it should be noted that the USB PLL cannot generate the required 480 MHz clock when the power supply is 0.9 V. In order to be able to generate 480 MHz, the supply level should be above 1.05 V.

1. Introduction

The DMA Controller (DMA) can perform DMA transfers on the AHB bus without using the CPU.

1.1 Features

The DMA module supports the following features:

- Supported transfer types:
  - Memory to Memory copy: Memory can be copied from the source address to the destination address with a specified length, while incrementing the address for both the source and destination.
  - Memory to peripheral: Data is transferred from incrementing memory to a fixed address of a peripheral. The flow is controlled by the peripheral.
  - Peripheral to memory: Data is transferred from a fixed address of a peripheral to incrementing memory. The flow is controlled by the peripheral.
- Supports single data transfers for all transfer types.
- Supports burst transfers for memory-to-memory transfers with following constraints:
  - A burst always consists of multiples of 4 (32-bit) words
  - Source and destination address have to be 4 words aligned for burst transfers
- The DMA controller has 12 channels
- Scatter gather possibility. This method is used to gather data which is located at different areas of memory. Two channels are needed per scatter gather action.
- Supports byte, half word and word transfers, and correctly aligns it over the AHB bus.
- Compatible with ARM flow control, for single requests (sreq), last single requests (lsreq), terminal count info (tc) and dma clearing (clr).
- Supports swapping in endianess of the transported data for file reading / MP3 decoding purposes.

Table 144. Peripherals that support DMA access

Peripheral name	Supported Transfer Types
NAND-flash controller	Memory to memory. NAND flow control is supported on channel 4 only.
SPI	Peripheral to memory and peripheral to memory
LCD	Memory to peripheral
UART	Memory to peripheral and peripheral to memory
I2C M/S	Memory to peripheral and peripheral to memory
ISRAM/IROM	Memory to memory

Table 144. Peripherals that support DMA access

Peripheral name	Supported Transfer Types
I2SRX_0/1	Peripheral to memory
I2STX_0/1	Memory to peripheral
PCM interface	Memory to peripheral and peripheral to memory

- Will do memory to-memory copies in 2 AHB cycles, and mem to peripheral or peripheral to mem in 3 AHB cycles (When zero waitstates of internal memory).
- Performance increase of burst transfers (with use of internal SRAM) compared to non-burst transfers will be about 30%.
- Contains masks for each IRQ input/source.
- Uses `HPROT' of the AHB bus for buffer disabling for peripheral transfers and the last DMA transfer.
- Most of the Flip-flops of the DMA are static like source, destination, control and length registers. These can be put on a gated clock-domain to conserve power.
- Supports external enabling of DMA channels, so other sources than the CPU can enable one or more DMA channels.
- External enabling possible for NAND flash Controller through channel 4.

2. General Description

2.1 Interface description

2.1.1 Clock Signals

Table 145. Clock Signals of the DMA Module

Clock Name	Acronym	I/O	Source/ Destination	Description
DMA_PCLK	PCLK	I	CGU	This is the clock the DMA operates on. This is the clock for both the APB and AHB bus.
DMA_CLK_GATED	CLK_GATED	I	CGU	Gated clock of PCLK. This clock is the same as PCLK, but it may be disabled during the inactive periods of the APB bus towards the DMA to reduce power. The clock disabling is done through the CGU, not in the DMA block. Clock gating is done on the active period of PSEL. CLK_GATED has to be synchronous with PCLK.

2.1.2 Bus Interface

The DMA module has an APB connection for register access.

The DMA module has a AHB bus connection connected to the multi layer AHB. Through this bus interface, the DMA data transfers are executed.

2.1.3 Interrupt Request Signals

The DMA module provides one interrupt request signal to the interrupt controller. The interrupt signal indicates when a channel is finished more than half-way, a soft interrupt, or a DMA abort.

2.1.4 Reset Signals

The CGU provides a synchronous reset signal to the DMA. It resets the whole DMA.

2.1.5 Internal signals of the DMA module

Table 146. DMA Signals of the DMA Module

Name	Type	Description
SDMA_SREQ	I	Signals going from DMA slaves, which is used for flow control. 0 = the DMA slave is not ready for single transfer request' 1 = the DMA slave has data available / can accept data.
SDMA_LREQ	I	Signals going from DMA slaves, which is used for flow control. 0 = the DMA slave is not ready for last transfer 1 = the DMA slave has just indicated a last transfer can be performed.
SDMA_TC	O	Signals going to DMA slaves, which is used for flow control. 0 = performed transfer was not the last transfer. 1 = performed transfer was the last transfer.This signal is not used in the LPC3130/31.
SDMA_CLR	O	If the slave supports DMA flow control, this signal is used to indicate to the slave that the single word / half word / byte transfer is complete, so the slave can restart a the request handshake. 0 = means the transfer is not complete 1 = means the transfer is complete.
SDMA_EXT_ENABLE	I	Signals arriving from external hardware that can also enable a DMA channel. Each bit corresponds to an equivalent SDMA channel. These signals may be generated on a different clock domain.External enabling only flash possible for NAND flash Controller. Channel 4 is used for this.
SDMA_EXT_ENABLE_ACK	O	The acknowledge of the external enabling, when a channels has finished its transfer. Each bit corresponds to a equivalent DMA channel.If this signal will be used on a different clock domain, the receiving side must take care of the clock-crossing. External enabling only possible for NAND flash Controller. Channel 4 is used for this.

3. Register overview

Table 147. Register overview: DMA controller (base address 0x1700 0000)

Name	R/W	Address Offset	Description
Channel 0 registers			
SOURCE ADDRESS	R/W	0x000	Source address register of channel 0
DESTINATION_ADDRESS	R/W	0X004	Destination address register of channel 0
TRANSFER_LENGTH	R/W	0X008	Transfer length register for channel 0
CONFIGURATION	R/W	0x00C	Configuration register for channel 0
ENABLE	R/W	0x010	Enable register for channel 0
<reserved>	-	0x014 – 0x018	Reserved
TRANSFER_COUNTER	R/W	0x01C	Transfer counter register for channel 0
Channel 1 registers			
SOURCE ADDRESS	R/W	0x020	Source address register of channel 1
DESTINATION_ADDRESS	R/W	0X024	Destination address register of channel 1
TRANSFER_LENGTH	R/W	0X028	Transfer length register for channel 1
CONFIGURATION	R/W	0x02C	Configuration register for channel 1
ENABLE	R/W	0x030	Enable register for channel 1
<reserved>	-	0x034 – 0x038	Reserved for channel 1
TRANSFER_COUNTER	R/W	0x03C	Transfer counter register for channel 1
Channel 2 registers			
SOURCE ADDRESS	R/W	0x040	Source address register of channel 2
DESTINATION_ADDRESS	R/W	0X044	Destination address register of channel 2
TRANSFER_LENGTH	R/W	0X048	Transfer length register for channel 2
CONFIGURATION	R/W	0x04C	Configuration register for channel 2
ENABLE	R/W	0x050	Enable register for channel 2
<reserved>	-	0x054 – 0x058	Reserved for channel 2
TRANSFER_COUNTER	R/W	0x05C	Transfer counter register for channel 2
Channel 3 registers			
SOURCE ADDRESS	R/W	0x060	Source address register of channel 3
DESTINATION_ADDRESS	R/W	0X064	Destination address register of channel 3
TRANSFER_LENGTH	R/W	0X068	Transfer length register for channel 3
CONFIGURATION	R/W	0x06C	Configuration register for channel 3
ENABLE	R/W	0x070	Enable register for channel 3
<reserved>	-	0x074 – 0x078	Reserved for channel 3
TRANSFER_COUNTER	R/W	0x07C	Transfer counter register for channel 3
Channel 4 registers			
SOURCE ADDRESS	R/W	0x080	Source address register of channel 4
DESTINATION_ADDRESS	R/W	0X084	Destination address register of channel 4

Table 147. Register overview: DMA controller (base address 0x1700 0000) *…continued*

Name	R/W	Address Offset	Description
TRANSFER_LENGTH	R/W	0X088	Transfer length register for channel 4
CONFIGURATION	R/W	0x08C	Configuration register for channel 4
ENABLE	R/W	0x090	Enable register for channel 4
<reserved>	-	0x094 – 0x098	Reserved for channel 4
TRANSFER_COUNTER	R/W	0x09C	Transfer counter register for channel 4
Channel 5 registers			
SOURCE ADDRESS	R/W	0x0A0	Source address register of channel 5
DESTINATION_ADDRESS	R/W	0X0A4	Destination address register of channel 5
TRANSFER_LENGTH	R/W	0X0A8	Transfer length register for channel 5
CONFIGURATION	R/W	0x0AC	Configuration register for channel 5
ENABLE	R/W	0x0B0	Enable register for channel 5
<reserved>	-	0x0B4 – 0x0B8	Reserved for channel 5
TRANSFER_COUNTER	R/W	0x0BC	Transfer counter register for channel 5
Channel 6 registers			
SOURCE ADDRESS	R/W	0x0C0	Source address register of channel 6
DESTINATION_ADDRESS	R/W	0X0C4	Destination address register of channel 6
TRANSFER_LENGTH	R/W	0X0C8	Transfer length register for channel 6
CONFIGURATION	R/W	0x0CC	Configuration register for channel 6
ENABLE	R/W	0x0D0	Enable register for channel 6
<reserved>	-	0x0D4 – 0x0D8	Reserved for channel 6
TRANSFER_COUNTER	R/W	0x0DC	Transfer counter register for channel 6
Channel 7 registers			
SOURCE ADDRESS	R/W	0x0E0	Source address register of channel 7
DESTINATION_ADDRESS	R/W	0X0E4	Destination address register of channel 7
TRANSFER_LENGTH	R/W	0X0E8	Transfer length register for channel 7
CONFIGURATION	R/W	0x0EC	Configuration register for channel 7
ENABLE	R/W	0x0F0	Enable register for channel 7
<reserved>	-	0x0F4 – 0x0F8	Reserved for channel 7
TRANSFER_COUNTER	R/W	0x0FC	Transfer counter register for channel 7
Channel 8 registers			
SOURCE ADDRESS	R/W	0x100	Source address register of channel 8
DESTINATION_ADDRESS	R/W	0X104	Destination address register of channel 8
TRANSFER_LENGTH	R/W	0X108	Transfer length register for channel 8
CONFIGURATION	R/W	0x10C	Configuration register for channel 8
ENABLE	R/W	0x110	Enable register for channel 8
<reserved>	-	0x114 – 0x118	Reserved for channel 8

Table 147. Register overview: DMA controller (base address 0x1700 0000) *…continued*

Name	R/W	Address Offset	Description
TRANSFER_COUNTER	R/W	0x11C	Transfer counter register for channel 8
Channel 9 registers			
SOURCE ADDRESS	R/W	0x120	Source address register of channel 9
DESTINATION_ADDRESS	R/W	0X124	Destination address register of channel 9
TRANSFER_LENGTH	R/W	0X128	Transfer length register for channel 9
CONFIGURATION	R/W	0x12C	Configuration register for channel 9
ENABLE	R/W	0x130	Enable register for channel 9
<reserved>	-	0x134 – 0x138	Reserved for channel 9
TRANSFER_COUNTER	R/W	0x13C	Transfer counter register for channel 9
Channel 10 registers			
SOURCE ADDRESS	R/W	0x140	Source address register of channel 10
DESTINATION_ADDRESS	R/W	0X144	Destination address register of channel 10
TRANSFER_LENGTH	R/W	0X148	Transfer length register for channel 10
CONFIGURATION	R/W	0x14C	Configuration register for channel 10
ENABLE	R/W	0x150	Enable register for channel 10
<reserved>	-	0x154 – 0x158	Reserved for channel 10
TRANSFER_COUNTER	R/W	0x15C	Transfer counter register for channel 10
Channel 11 registers			
SOURCE ADDRESS	R/W	0x160	Source address register of channel 11
DESTINATION_ADDRESS	R/W	0X164	Destination address register of channel 11
TRANSFER_LENGTH	R/W	0X168	Transfer length register for channel 11
CONFIGURATION	R/W	0x16C	Configuration register for channel 11
ENABLE	R/W	0x170	Reserved for channel 11
<reserved>	-	0x174 – 0x178	Reserved for channel 11
TRANSFER_COUNTER	R/W	0x17C	Transfer counter register for channel 11
-		0x180 – 0x1FC	Reserved
Alternate Channel 0 registers			
ALT_SOURCE_ADDRESS	W	0x200	Alt source address register for channel 0
ALT_DESTINATION_ADDRESS	W	0x204	Alt destination address register for channel 0
ALT_TRANSFER_LENGTH	W	0x208	Alt transfer length address register for channel 0
ALT_CONFIGURATION	W	0x20C	Alt configuration register for channel 0
Alternate Channel 1 registers			
ALT_SOURCE_ADDRESS	W	0x210	Alt source address register for channel 1
ALT_DESTINATION_ADDRESS	W	0x214	Alt destination address register for channel 1

Table 147. Register overview: DMA controller (base address 0x1700 0000) *…continued*

Name	R/W	Address Offset	Description
ALT_TRANSFER_LENGTH	W	0x218	Alt transfer length address register for channel 1
ALT_CONFIGURATION	W	0x21C	Alt configuration register for channel 1
Alternate Channel 2 registers			
ALT_SOURCE_ADDRESS	W	0x220	Alt source address register for channel 2
ALT_DESTINATION_ADDRESS	W	0x224	Alt destination address register for channel 2
ALT_TRANSFER_LENGTH	W	0x228	Alt transfer length address register for channel 2
ALT_CONFIGURATION	W	0x22C	Alt configuration register for channel 2
Alternate Channel 3 registers			
ALT_SOURCE_ADDRESS	W	0x230	Alt source address register for channel 3
ALT_DESTINATION_ADDRESS	W	0x234	Alt destination address register for channel 3
ALT_TRANSFER_LENGTH	W	0x238	Alt transfer length address register for channel 3
ALT_CONFIGURATION	W	0x23C	Alt configuration register for channel 3
Alternate Channel 4 registers			
ALT_SOURCE_ADDRESS	W	0x240	Alt source address register for channel 4
ALT_DESTINATION_ADDRESS	W	0x244	Alt destination address register for channel 4
ALT_TRANSFER_LENGTH	W	0x248	Alt transfer length address register for channel 4
ALT_CONFIGURATION	W	0x24C	Alt configuration register for channel 4
Alternate Channel 5 registers			
ALT_SOURCE_ADDRESS	W	0x250	Alt source address register for channel 5
ALT_DESTINATION_ADDRESS	W	0x254	Alt destination address register for channel 5
ALT_TRANSFER_LENGTH	W	0x258	Alt transfer length address register for channel 5
ALT_CONFIGURATION	W	0x25C	Alt configuration register for channel 5
Alternate Channel 6 registers			
ALT_SOURCE_ADDRESS	W	0x260	Alt source address register for channel 6
ALT_DESTINATION_ADDRESS	W	0x264	Alt destination address register for channel 6
ALT_TRANSFER_LENGTH	W	0x268	Alt transfer length address register for channel 6
ALT_CONFIGURATION	W	0x26C	Alt configuration register for channel 6
Alternate Channel 7 registers			
ALT_SOURCE_ADDRESS	W	0x270	Alt source address register for channel 7
ALT_DESTINATION_ADDRESS	W	0x274	Alt destination address register for channel 7
ALT_TRANSFER_LENGTH	W	0x278	Alt transfer length address register for channel 7

Table 147. Register overview: DMA controller (base address 0x1700 0000) *…continued*

Name	R/W	Address Offset	Description
ALT_CONFIGURATION	W	0x27C	Alt configuration register for channel 7
Alternate Channel 8 registers			
ALT_SOURCE_ADDRESS	W	0x280	Alt source address register for channel 8
ALT_DESTINATION_ADDRESS	W	0x284	Alt destination address register for channel 8
ALT_TRANSFER_LENGTH	W	0x288	Alt transfer length address register for channel 8
ALT_CONFIGURATION	W	0x28C	Alt configuration register for channel 8
Alternate Channel 9 registers			
ALT_SOURCE_ADDRESS	W	0x290	Alt source address register for channel 9
ALT_DESTINATION_ADDRESS	W	0x294	Alt destination address register for channel 9
ALT_TRANSFER_LENGTH	W	0x298	Alt transfer length address register for channel 9
ALT_CONFIGURATION	W	0x29C	Alt configuration register for channel 9
Alternate Channel 10 registers			
ALT_SOURCE_ADDRESS	W	0x2A0	Alt source address register for channel 10
ALT_DESTINATION_ADDRESS	W	0x2A4	Alt destination address register for channel 10
ALT_TRANSFER_LENGTH	W	0x2A8	Alt transfer length address register for channel 10
ALT_CONFIGURATION	W	0x2AC	Alt configuration register for channel 10
Alternate Channel 11 registers			
ALT_SOURCE_ADDRESS	W	0x2B0	Alt source address register for channel 11
ALT_DESTINATION_ADDRESS	W	0x2B4	Alt destination address register for channel 11
ALT_TRANSFER_LENGTH	W	0x2B8	Alt transfer length address register for channel 11
ALT_CONFIGURATION	W	0x2BC	Alt configuration register for channel 11
		0x2C0 – 0x3FC	reserved
Global control registers			
ALT_ENABLE	R/W	0x400	Alternative enable register
IRQ_STATUS_CLEAR	R/W	0x404	IRQ status clear register
IRQ_MASK	R/W	0x408	IRQ mask register
TEST_FIFO_RESP_STATUS	R	0x40C	Test FIFO response status register
SOFT_INT	W	0x410	Software interrupt register

4. Register description

4.1 Channel registers 0 to 11

Table 148. SOURCE_ADDRESS (addresses 0x1700 0000 (channel 0) to 0x1700 0160 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:0	Source address	R/W	0x0	This register contains the address from where the data is read from. This data will remain static during all memory transfers, and can only be changed by re-programming.

Table 149. DESTINATION_ADDRESS (addresses 0x1700 0004 (channel 0) to 0x1700 0164 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:0	Destination address	R/W	0x0	This register contains the address to where the data is written to. This data will remain static during all memory transfers, and can only be changed by re-programming.

Table 150. TRANSFER_LENGTH (addresses 0x1700 0008 (channel 0) to 0x1700 0168 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:21	-	-	0x0	Reserved
20:0	Transfer length	R/W	0x1FFFFF	This register contains the amount of cycles to transfer. This can be bytes, half-words or words. This data will remain static throughout the memory transfer, and can only be changed by re-programming. The number of transfers performed is: <The number programmed in this register> + 1 If for example 0x100 is programmed in this register, a total of 0x101 transfers will be done. Current maximum is 2048K transfers. Note: If burst transfer is configured in the CONFIGURATION register, than the TRANFER_LENGTH register will determine the amount of burst transfers. Note: one burst transfer will contain a transfer of 4 words.

If the transfer length programmed is 0 (which means 1 transfer of byte/half word/word/or burst of 4 word), the interrupt halfway is not generated.

Table 151. CONFIGURATION (addresses 0x1700 000C (channel 0) to 0x1700 016C (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:19	-	R	0x0	Reserved
18	CIRCULAR_BUFFER	R/W	0x0	When this bit is set, the enable bit inside the enable register will never be cleared, and the channel will keep looping. However the CPU can still be interrupted at the end of each loop and halfway each loop. This is a good alternative to channel-companions, since this technique requires only one channel to be enabled, saving channels
17	COMPANION_CHANNEL_ENABLE	R/W	0x0	If this bit is set, the channel number programmed inside the copy_companion_channel_n bits will be enabled at the end of the current transfer.
16	-	R	0x0	Reserved
15:13	COMPANION_CHANNEL_NR	R/W	0x0	If the companion_channel_enable bit is set, the channel number programmed in this register will be enabled when the current channel has finished transfer. This allows also the use of a linked-list / scatter-gather method. See Section 9–5.2.
12	INVERT_ENDIANESS	R/W	0x0	0x0: no endianess inversion while transferring 0x1: In this setting the endianess of the word, or half-word is switched. This is for instance required for MP3 decoding from PC-files. If the transfer is word aligned, then: Byte 3 and byte 0 are swapped Byte 2 and byte 1 are swapped If the transfer is half-word aligned then: Byte 1 of the half-word is swapped with byte 0 Note: It is allowed to use the same source and destination address to change the endianess of the data without copying it to a different location.

Table 151. CONFIGURATION (addresses 0x1700 000C (channel 0) to 0x1700 016C (channel 11)) *…continued*

Bit	Symbol	Access	Reset Value	Description
11:10	TRANSFER_SIZE	R/W	0x0	This register contains the size of each transfer: 0x0: Transfer of words 0x1: Transfer of half-words 0x2: Transfer of bytes. 0x3: Transfer of bursts If half-words or bytes are used, these will be correctly aligned over the AHB bus.
9:5	READ_SLAVE_NR	R/W	0x0	When 0x0 is written to this register, means that the transfer is unconditional for each read, and the read-address is incremented each read-cycle. If a number higher than 0x0 is programmed here, then flow control is used for the status of that specific slave-FIFO pin, decremented by 1. The address is NOT incremented for each write, so the same FIFO is read from each time. Use the pin number listed in Table 9–165 to use the flow control of the peripherals. So if a peripheral is connected to pin 3 of the DMA, and flow control for this slave will be used, then 0x4 must be programmed for these bits (0x3 + 1)
4:0	WRITE_SLAVE_NR	R/W	0x0	When 0x0 is written to this register, means that the transfer is unconditional for each write, and the write-address is incremented each write-cycle. If a number higher than 0x0 is programmed here, then flow control for each write is used, checking for the status of that specific slave-FIFO pin, decremented by 1. The address is NOT incremented for each write, so the same FIFO is written each time. Use the pin number listed in Table 9–165 to use the flow control of the peripherals. Example: if a peripheral is connected to pin 2 of the DMA, and flow control for this slave must be used, then 0x3 must be programmed (0x2 + 1 = 0x3)

Table 152. ENABLE (addresses 0x1700 0010 (channel 0) to 0x1700 0170 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:1	-	-	-	Reserved
0	enable	R/W	0	0x0: disable channel. 0x1: enable channel. This register will be automatically be disabled when the transfer is finished, OR when the slave has indicated the last transfer using `SDMA_LSREQ'. If the transfer is disabled by the CPU in the middle of a transfer, software might want to reset the counter, by writing to the `TRANSFER_COUNTER' register of that channel. Counter should be reset even if the DMA was stopped because `LSREQ' was active.

Table 153. TRANSFER_COUNTER (addresses 0x1700 001C (channel 0) to 0x1700 017C (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:21	-	-	0x0	Reserved
20:0	Transfer Counter	R/W	0x0	Reading this register returns the current counter status of the channel. The counter starts at `0' and ends at the length programmed in the `length' register.When a transfer is finished, the counter is reset to `0'. When a transfer is stopped by a slave, by using the `SDMA_LSREQ' signal, the counter indicates the amount of transfers performed.Writing this register resets the counter to `0'. Example: If this counter reads 0x100, then 0x101 transfers have been performed. Note: If a channel is disabled in the middle of a transfer by the CPU, OR the counter was stopped by using the `SDMA_LSREQ', then writing to this register is the only method of resetting the counter to `0'.

4.2 Alternate channel registers 0 to 11

Table 154. ALT_SOURCE_ADDRESS (addresses 0x1700 0200 (channel 0) to address 0x170002B0 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:0	Source address	W	0x0	This register is a mirror of the SOURCE_ADDRESS register, and can only be written to. See Section 9–5.2 for more details.

Table 155. ALT_DESTINATION_ADDRESS (addresses 0x1700 0204 (channel 0) to address 0x170002B4 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:0	Destination address	W	0x0	This register is a mirror of the DESTINATION_ADDRESS register, and can only be written to. See Section 9–5.2 for more details.

Table 156. ALT_TRANSFER_LENGTH (addresses 0x1700 0208 (channel 0) to 0x1700 02B8 (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:21	-	-	0x0	Reserved
20:0	Transfer Length	W	0x1FFFFF	This register is a mirror of the TRANSFER_LENGTH register, and can only be written to. See Section 9–5.2. for more details.

Table 157. ALT_CONFIGURATION (addresses 0x1700 020C (channel 0) to 0x1700 02BC (channel 11))

Bit	Symbol	Access	Reset Value	Description
31:0	-	W	0x0	This register is a mirror of the CONFIGURATION register, and can only be written to. See Section 9–5.2. for more details.

4.3 General control registers

Table 158. ALT_ENABLE (address 0x1700 0400)

Bit	Symbol	Access	Reset Value	Description
31:12	-	-	-	reserved
11:0	ALT_CH_EN	R/W	0x0	This register allows enabling and disabling of multiple channels at the same time. Each bit represents a channel number, so Bit 0 = Channel 0 Bit 1 = Channel 1.etc... Please use the individual `enable' register of each channel, instead of using this register! A read-modify-write to this register might write the wrong data, since there is a chance the enable register of one of the channels has updated itself in between the read-modify-write, because:- it is either finished,- was activated by a `copy-table' setting,- or when external enabling was involved

The IRQ_STATUS_CLR register contains information if a channel has finished its transfer OR if the channel is halfway.

A bit which has been set can only be cleared by writing a `1' to this bit in this register.

The `finish' bit will only be set when the channel is finished. After clearing, it will only be set again if the channel finishes again.

The `half-way' bit will only be set when the channel has passed the half of the transfer. After clearing, this bit will only be set when the channels passes halfway again.

The soft interrupt bit will be set when the SOFT_INT register is written to. This is for scatter-gather interrupt control: The last transfer of a scatter-gather operation can write towards this register, so the CPU knows the scatter-gather operation has finished.

The DMA_abort bit will be active if any transfer done by the DMA controller resulted in an abort on the AHB bus.

Clearing interrupts: Writing a `1' to a bit will clear the interrupt belonging to that bit.

Table 159. IRQ_STATUS_CLR (address 0x1700 0404)

Bit	Symbol	Access	Reset Value	Description
31	DMA_abort			DMA abort
30	Soft_interrupt	R/W	0x0	Soft interrupt, scatter gather
29:24	-	R/W	0x0	reserved
23	Half_way_11	R/W	0x0	Channel 11 is more than half-way
22	Finished_11	R/W	0x0	Channel 11 is finished
21	Half_way_10	R/W	0x0	Channel 10 is more than half-way
20	Finished_10	R/W	0x0	Channel 10 is finished
19	Half_way_9	R/W	0x0	Channel 9 is more than half-way
18	Finished_9	R/W	0x0	Channel 9 is finished
17	Half_way_8	R/W	0x0	Channel 8 is more than half-way
16	Finished_8	R/W	0x0	Channel 8 is finished
15	Half_way_7	R/W	0x0	Channel 7 is more than half-way
14	Finished_7	R/W	0x0	Channel 7 is finished
13	Half_way_6	R/W	0x0	Channel 6 is more than half-way
12	Finished_6	R/W	0x0	Channel 6 is finished
11	Half_way_5	R/W	0x0	Channel 5 is more than half-way
10	Finished_5	R/W	0x0	Channel 5 is finished
9	Half_way_4	R/W	0x0	Channel 4 is more than half-way
8	Finished_4	R/W	0x0	Channel 4 is finished
7	Half_way_3	R/W	0x0	Channel 3 is more than half-way
6	Finished_3	R/W	0x0	Channel 3 is finished
5	Half_way_2	R/W	0x0	Channel 2 is more than half-way
4	Finished_2	R/W	0x0	Channel 2 is finished
3	Half_way_1	R/W	0x0	Channel 1 is more than half-way
2	Finished_1	R/W	0x0	Channel 1 is finished
1	Half_way_0	R/W	0x0	Channel 0 is more than half-way
0	Finished_0	R/W	0x0	Channel 0 is finished

Table 160. IRQ_MASK (address 0x1700 0404)

Bit	Symbol	Access	Reset Value	Description
31	IRQ DMA abort		0x1	Mask IRQ of DMA abort
30	IRQ of Soft interrupt is masked		0x1	Mask IRQ of Soft interrupt
29:24	-	-	0x0	Reserved
23	Mask Half_way_11	R/W	0x1	Mask Channel 11 is more than half-way interrupt
22	Mask Finished_11	R/W	0x1	Mask Channel 11 is finished interrupt
21	Mask Half_way_10	R/W	0x1	Mask Channel 10 is more than half-way interrupt
20	Mask Finished_10	R/W	0x1	Mask Channel 10 is finished interrupt
19	Mask Half_way_9	R/W	0x1	Mask Channel 9 is more than half-way interrupt

Table 160. IRQ_MASK (address 0x1700 0404) *…continued*

Bit	Symbol	Access	Reset Value	Description
18	Mask Finished_9	R/W	0x1	Mask Channel 9 is finished interrupt
17	Mask Half_way_8	R/W	0x1	Mask Channel 8 is more than half-way interrupt
16	Mask Finished_8	R/W	0x1	Mask Channel 8 is finished interrupt
15	Mask Half_way_7	R/W	0x1	Mask Channel 7 is more than half-way interrupt
14	Mask Finished_7	R/W	0x1	Mask Channel 7 is finished interrupt
13	Mask Half_way_6	R/W	0x1	Mask Channel 6 is more than half-way interrupt
12	Mask Finished_6	R/W	0x1	Mask Channel 6 is finished interrupt
11	Mask Half_way_5	R/W	0x1	Mask Channel 5 is more than half-way interrupt
10	Mask Finished_5	R/W	0x1	Mask Channel 5 is finished interrupt
9	Mask Half_way_4	R/W	0x1	Mask Channel 4 is more than half-way interrupt
8	Mask Finished_4	R/W	0x1	Mask Channel 4 is finished interrupt
7	Mask Half_way_3	R/W	0x1	Mask Channel 3 is more than half-way interrupt
6	Mask Finished_3	R/W	0x1	Mask Channel 3 is finished interrupt
5	Mask Half_way_2	R/W	0x1	Mask Channel 2 is more than half-way interrupt
4	Mask Finished_2	R/W	0x1	Mask Channel 2 is finished interrupt
3	Mask Half_way_1	R/W	0x1	Mask Channel 1 is more than half-way interrupt
2	Mask Finished_1	R/W	0x1	Mask Channel 1 is finished interrupt
1	Mask Half_way_0	R/W	0x1	Mask Channel 0 is more than half-way interrupt
0	Mask Finished_0	R/W	0x1	Mask Channel 0 is finished interrupt

Table 161. TEST_FIFO_RESP_STAT (address 0x1700 0408)

Bit	Symbol	Access	Reset Value	Description
31:0	TEST_FIFO_RESP_STAT	R	0	Test register only. Not useful for functional operation. This register reads out the exact status of the FIFO-response pins going towards the DMA. Bit 30-0 represents the connected SREQ signals. By reading this register, it can be tested if the SREQ pins are correctly connected on a system level.

Table 162. SOFT_INT (address 0x1700 040C)

Bit	Symbol	Access	Reset Value	Description
31:0	-	-	0	Reserved
0	Enable_soft_interrupt	R/W	0	Writing to this bit will enable the soft_interrupt IRQ, in the IRQ status register. This register exists purely for linked-list, so the last transfer can be a write towards this address. This way the CPU knows exactly when a scatter-gather operation is finished.

5. Functional description

5.1 Channel arbitration

If the dma contains more than 1 channel (in our case 12 channels), arbitration is required so that each channel gets equal access over the AHB bus.

The channels are checked in a round-robin (circular) motion. First channel 0 receives a transfer, then 1, then 2... then 0 again and so forth.

When the current transfer is finished, the next channel will be checked first. The arbitration goes like this:

```
IF (next channel = enabled)
     IF (mem-to-mem) Current_channel = this channel
     else IF (peripheral to mem) OR (mem to peripheral) OR (peripheral to peripheral)
          If (selected peripheral(s) = ready) Current_channel = this channel
Else IF (next channel + 1 = enabled)
     IF (mem-to-mem) Current_channel = this channel
     else IF (peripheral to mem) OR (mem to peripheral) OR (peripheral to peripheral)
          If (selected peripheral(s) = ready) Current_channel = this channel
Else IF (next channel + 2 = enabled)
Else Current_channel = UNCHANGED
```

A new channel is known within a single clock-cycle. Every new AHB cycle the channels are re-arbitrated. As soon as a channel is ready to start a transfer, this channel gets access, and arbitration will continue from this point.

5.2 Scatter gathering / Building a linked-list

Scatter gathering is a method where data is located in lots of different areas of memory and needs to be `gathered' to one location as a whole. This might be mem to mem, but also mem/peripheral combinations.

In memory, the CPU can program a linked-list which consists of source, destination and length entries. Each entry will be executed as if a channel was programmed by these values and started, in a sequential order.

The dma supports a linked-list by using 2, sequential, channels.

The first channel will execute the contents of the linked list.

The second channel will perform the updating of the linked list entry.

The way it is implemented in the dma, is that the dma actually re-programs one of its own channels (the previous one), replacing the contents of a channel with the contents of the linked-list, then enables this channel.

A linked-list entry consists of 5 words. The first 4 words reprograms the first DMA channel (the one that executes the contents of the link-list). The fifth word overwrites the source address of the current DMA channel, thus updating the linked list entry to the next location. See the dma register map: the ALT_addresses.

The fifth word will overwrite the source address of the next channel.

See: Table 9–163

Table 163. Linked List Example

Address	Label	Remark#
n=0: LINKED_LIST_BASE_ADDESS + n*0x14 + 0x0	Source address n	0
n*0x14 + 0x4	Destination address n	
n*0x14 + 0x8	Transfer Length n	
n*0x14 + 0xc	Configuration n	1
n*0x14 + 0x10	Linked_list_Base_address + (n+1)*0x14	Addr next entry
(n+1)*0x14 + 0x0	Source address (n+1)	
(n+1)*0x14 + 0x4	Destination address (n+1)	
(n+1)*0x14 + 0x8	Transfer Length (n+1)	
(n+1)*0x14 + 0xc	Configuration (n+1)	
(n+1)*0x14 + 0x10	Linked_list_Base_address + (n+2)*0x14	
-	Room for more entries	2
Last_entry (Method 1). Finish list transfer:		
y=last entry number: (n+y)*0x14 + 0x0	Source address (n+y)	
(n+y)*0x14 + 0x4	Destination address (n+y)	
(n+y)*0x14 + 0x8	Transfer Length (n+y)	
(n+y)*0x14 + 0xc	Configuration (n+y)	Disable companion table option
(n+y)*0x14 + 0x10	0x0	Don't care
Last_entry (Method 2). Implement circular linked list: .		
y=last entry number: (n+y)*0x14 + 0x0	Source address (n+y)	
(n+y)*0x14 + 0x4	Destination address (n+y)	
(n+y)*0x14 + 0x8	Transfer Length (n+y)	
(n+y)*0x14 + 0xc	Configuration (n+y)	
(n+y)*0x14 + 0x10	Linked_list_Base_address	Jump to the start of the linked list, making it a circular linked list

Table 163. Linked List Example

Address		Label	Remark#
Last_entry (Method 3). Generate soft interrupt at the end of transfer:			3
y=last entry number:	(n+y)*0x14 + 0x0	<Any readable memory address>	
	(n+y)*0x14 + 0x4	<dma SOFT_INT address>	4
(n+y)*0x14 + 0x8		0x1	5
(n+y)*0x14 + 0xC		0x0	6
(n+y)*0x14 + 0x10		0x0	Don't care
Last_entry (Method 4). Generate soft interrupt and also disentangle the companion channels.			3
y=last entry number:	(n+y-1)*0x14 + 0	<Any readable memory address>	
(n+y-1)*0x14 + 0x4		<dma SOFT_INT address>	4
(n+y-1)*0x14 + 0x8			
0x1			5
(n+y-1)*0x14 + 0xC		<word transfer + Enable_next_table>	7
(n+y-1)*0x14 + 0x10		Linked_List_Base_address + (n+y)*0x14	Addr last entry
y=last entry number:	(n+y)*0x14 + 0x0	Source address y	
(n+y)*0x14 + 0x4		Destination address y	
	(n+y)n*0x14 + 0x8	Transfer Length y	
	(n+y)*0x14 + 0xc	Configuration y	Disable companion table option
	(n+y)*0x14 + 0x10	0x0	Don't care

Remark: 0: n = Channel entry number.

Remark: 1: The configuration register can be programmed any way the programmers likes it, but should always have the `companion channel' set to the next channel, or otherwise the linked-list execution stops. An exception in this case is when the programmer wants to stop the linked list execution (for instance in case of the last entry)

Remark: 2: As many entries can be set here as the programmer would like, each entry consuming 20 bytes of local memory. Remember that every entry has 5 addresses, and that address 5 contains the address of the following entry.

Remark: 3: This last entry will soft-interrupt the DMA controller, so the CPU knows the last transfer has finished. The source address is `don't care', but preferably a memory mapped location (like SRAM).

Remark: 4: The destination address is the soft-interrupt register of this DMA controller.

Remark: 5: Length `1' is sufficient

Remark: 6: This setting disables companion-channels, so basically this is the last transfer.

Remark: 7: The `enable companion channel' option must be set, since there is still1 transfer left.

5.2.1 Ending a linked-list transfer inside the linked list

The last entry of a linked list is a special one. In the example linked-list, there are 4 methods described how to end the linked list transfer:

Method 1 - Just stop the linked list, by disabling the copy-table-enabling in the config setting.

This can be done for instance if the CPU is not interested in an IRQ on this event.

Method 2 - Re-start the linked list by pointing the last pointer back to the beginning of the linked list, effectively creating a circular buffer made out of linked list entries.

Method 3 - Write towards the DMA soft interrupt register, which enables the CPU to be interrupted, so that it can act on the event that a linked-list has ended.

Method 4 - This is a combination of 1 and 3. Now the CPU gets a soft interrupt 1 entry before the last entry is executed.

Method x - Whatever suits the application.

5.2.2 To start a linked-list operation

- Program into memory the linked-list to <linked-list_base_address>.
- Reserve 2 sequential DMA channels/channels.
- Program the second channel as follows:

Table 164. Linked List Example

Source Address	<Linked_list_base_address>
Destination address	<The alternative source address of the previous DMA channel>
Transfer Length	0x4 (for 5 addresses per entry)
Configuration	DMA_word + enable_companion_channel (previous channel)
Enable address	0x1 (start the transfer)

Remark: Please make sure that the ALT_SOURCE_REG is used as the destination address, and not the normal SOURCE_REG.

5.2.3 Ending a linked list transfer in the middle of a linked-list execution

There may be cases where the CPU needs to stop a linked-list from execution. There are more ways to do this:

1. Brutally stop the execution. This can be done by:
   - Enable [first_channel]=0
   - Enable [second_channel]=0
   - Enable [first_channel]=0

Now both channels are disabled, and the execution is stopped.

To continue, do the following:

```
if counter[first_channel] > 0 then
     Enable[first_channel] = 1
else
     Enable[second_channel] = 1
```

2. Stop the linked-list, but let the current entry finish first.
3. This can be done by read-modify-write the config[second_channel] register, and disable the `companion channel' bit.
4. When this is done, the linked list entry will finish, and a new linked-list entry will be programmed in the previous channel.
5. However, the first channel will not be enabled, thus stopping the transfer.
6. To continue, do the following:
   - read-modify-write the config[second_channel] register, and enable the `companion channel' bit.
   - enable[first_channel] = 1

5.3 DMA flow control

The DMA flow control is compatible to the ARM flow control used in the ARM PL080 and PL081 DMA controllers.

Not supported by the DMA is the `Burst Request' and `Last Burst Request' signal, if this signal indicates a burst of more than one word.

For information about the timing of the DMA flow control, please refer to the ARM PL080 data sheet.

Is allowed for DMA slaves to assert an asynchronous request signal, either slower or faster than the DMA clock. This is synchronized inside the DMA. Please take care that the slave may receive an asynchronous `DMA_CLR' signal as well, if there is an intention to use asynchronous clocks.

If a DMA slave is behind a write-buffered bridge or has its own write buffer, this DMA slave may only de-assert `SREQ' when the transfer is truly completed. If the slave does not comply to this rule, then the write-buffered bridge must support the disabling of the write-buffer by reading the AHB-HPROT[2] signal.

Extra flip-flops are added for the Flow Control signals. This is needed for synchronization between blocks, which run on an other frequency than the DMA.

5.4 Connectivity

Table 165. SOFT_INT

Module	DMA Source	Peripheral Pin[1]	Interrupt
IPINT/PCM	Ipint_tx	0	IP_int tx single request indication
	Ipint_rx	1	IP_int0 rx single request indication
UART	uart_ rx	2	UART rx receive FIFO request information
	uart tx	3	UART tx transmit FIFO request information
I2C0	I2c0	4	i2c0 FIFO DMA request
I2C1	I2c1	5	i2c1 FIFO DMA request
I2STX0	I2STX0_dma_req_left	6	I2STX0 Left channel DMA request
	I2STX0_dma_req_right	7	I2STX0 Right channel DMA request
I2STX1	I2STX1_dma_req_left	8	I2STX1 Left channel DMA request
	I2STX1_dma_req_right	9	I2STX1 Right channel DMA request
I2SRX0	I2SRX0_dma_req_left	10	I2SRX0 left channel DMA request
	I2SRX0_dma_req_right	11	I2SRX0 right channel DMA request
I2SRX1	I2SRX1_dma_req_left	12	I2SRX1 left channel DMA request
	I2SRX1_dma_req_right	13	I2SRX1 right channel DMA request
-	-	14	reserved
-	-	15	reserved
LCD interface	lcd_interface_dma_req	16	LCD interface FIFO transmit DMA request
SPI	spi_tx_dmareq	17	SPI dma request for transmitting data.
	spi_rx_dmareq	18	SPI dma request for receiving data.
SD/MMC	sd_mmc_dmasreq	19	SD_MMC dma request for transmitting/receiving data.

[1] When programmed, add `0x1' to this number.

5.4.1 Using FIFO level slaves, which have no flow control

The DMA can support FIFO level slaves, with the following rules:

- For a FIFO level slave which is read, the FIFO level must be updated directly after the read has taken place.
- For a FIFO level slave which is written, the FIFO level must be updated directly after the write has taken place.If a write-slave is behind a write-buffered bridge, then the write-buffered bridge must support the disabling of the write-buffer by reading the AHB-HPROT[2] signal. If this is not done, it is then possible that the DMA sometimes performs two transfers to the slave in stead of one. This may be acceptable for a slave, but it is then mandatory the FIFO of this slave can accept two transfers. The DMA will never exceed its maximal transfer count.
- On top-level connectivity, the FIFO level request signal must be ANDed with the inverted DMA_CLR[x] signal. This new signal is then connected to the same numbered DMA_SREQ[x] pin.

5.5 External enable flow control

Other devices than the CPU can enable a DMA channel by activating one of the `Ext_Enable' flow control pins.

The DMA has an equal number of these pins available on the DMA top-level as there are channels available, and each external enable pin-number has a direct connection to its equivalent DMA channel number. The activation of one of these pins immediately enables its corresponding DMA channel.

So:

- Ext_en[0] can enable channel 0;
- Ext_en[1] can enable channel 1;
- Ext_en[2] can enable channel 2;
- Ext_en[3] can enable channel 3;
- Ext_en[4] can enable channel 4;
- Ext_en[5] can enable channel 5;
- Ext_en[6] can enable channel 6;
- Ext_en[7] can enable channel 7;
- Ext_en[8] can enable channel 8;
- Ext_en[9] can enable channel 9;
- Ext_en[10] can enable channel 10;
- Ext_en[11] can enable channel 11;

Table 166. Connections to DMA EXT_EN Pins

Ext_en	Connected Device
0	-
1	-
2	-
3	-
4	NAND flash_ctrl_DMA_ext_enable
5	-
6	-
7	-
8	-
9	-
10	-
11	-

It uses the following flow control, using Ext_en[4] as an example:

Ext_en[4] is asserted by the NAND flash Controller

The DMA starts channel 4 because of this assertion.

Ext_en[4] must remain asserted until Ext_en_ack[4] is asserted by the DMA.

The DMA asserts Ext_en_ack[4], when channel 4 is finished with all its transfers.

Ext_en[4] can be de-asserted.

The DMA will de-assert Ext_en_ack[4] because of this.

Another transfer is allowed, and the flow may be restarted.

A DMA channel can only be enabled by this method, not disabled.

If `Ext_en[4]' is asserted, the DMA will only enable channel 4 once. If the transfer is complete, and Ext_en[4] remains asserted, channel 4 will remain inactive. So a toggle is required on Ext_en[4] before multiple enables are possible, and this has to be done according the Ext_en flow control as described above.

It is allowed to have the `Ext_en' signals on different clock domains, so the Nandflash Controller can run faster or slower than the DMA

The `Ext_en_ack' of a channel is always asserted for at least a single clock-cycle if that channel is finished, disregarding if Ext_en is asserted or not. Any peripheral can recognize this way if a channel is finished.

6. Power optimization

Further more the module has clock gating. The gated clock CLK_GATED is requested when necessary. This will be requested as long as there is a transfer going on at the APB bus to/from the DMA. Setting the external enabling bit of PCR CGU registers of this clock, enables the clock gating of the this clock.

UM10314

Chapter 10: LPC3130/31 Interrupt controller

Rev. 1 — 4 March 2009 **User manual**

1. Introduction

The Interrupt Controller (INTC) collects interrupt requests from multiple devices, masks interrupt requests, and forwards the combined requests to the processor (see Figure 10–30). The interrupt controller also provides facilities to identify the interrupt requesting devices to be served.

This module has the following features:

- The interrupt controller decodes all the interrupt requests issued by the on-chip peripherals.
- Two interrupt lines (nFIQ, nIRQ) are provided to the to the ARM core. The ARM core supports two distinct levels of priority on all interrupt sources, nFIQ for high priority interrupts and nIRQ for normal priority interrupts.
- Software interrupt request capability associated with each request input.
- Visibility of the interrupt's request state before masking.
- Support for nesting of interrupt service routines.
- Interrupts routed to nIRQ and to nFIQ are vectored.

The following blocks can generate interrupts:

- Nand flash controller
- USB 2.0 HS OTG
- Event router
- 10-bit ADC
- UART
- LCD
- SPI
- I2C Master/Slave 0 and I2C Master/Slave 1
- Timer0, Timer1, Timer2, and Timer3
- I2STX_0/1
- I2SRX_0/1
- DMA

2. General description

The Vectored Interrupt Controller (INTC) collects interrupt requests from multiple devices, masks interrupt requests, and forwards the combined requests to the processor (see Figure 10–30). The interrupt controller also provides facilities to identify the interrupt requesting devices to be served.

The interrupt controller decodes all the interrupt requests issued by the on-chip peripherals. It supports total 29 interrupt sources and has two outputs (nFIQ, nIRQ). The ARM core can have two distinct levels of priority on all interrupt sources, nFIQ higher priority and nIRQ has lower. The interrupt controller shall operate as AHB slave.

Common features are:

- Software interrupt request capability associated to each request input
- Visibility of interrupt request state before masking
- Support for nesting of interrupt service routines.

Interrupts routed to nIRQ and to nFIQ are vectored. That is to say that the processor can execute the interrupt handler corresponding to the current interrupt without testing each interrupt individually. Thus the software is minimized.

The interrupt vector register contains the index of a specific ISR and an associated priority limiter value can be delivered if nesting of ISR is to be done.

In this interrupt controller, a set of software accessible variables is provided to control interrupt request generation. It is essentially used in debug mode.

The interrupt controller supports interrupts, which are level sensitive, asynchronous and can be active low or high.

2.1 Interface description

2.1.1 Clock signals

Table 167. Clock signals of the INTC module

Clock Name	Acronym	I/O	Source/ Destination	Description
INTC_CLK	CLK	I	CGU	Main clock, Identical to AHB CLK

2.1.2 Interrupt request signals

The LPC3130/31 interrupt controller supports 29 interrupt lines as asynchronous interrupt requests from interrupt devices (level active, any polarity). The INTC provides two outputs (level active low) to the ARM processor.

2.1.2.1 Processor Interrupt Request Inputs

The 29 interrupt request input signals are level active (of any polarity) and assumed glitch-free. They are treated as asynchronous to the interrupt controller clock INT_CLK. Interrupt requests once asserted, must be kept asserted until served by a software ISR.

2.1.2.2 Processor Interrupt Request Outputs

Processor interrupt requests are provided as level active output signals glitch-free at both polarities. Once asserted, processor interrupt request signals stay asserted until the interrupt device removes the request or the request becomes masked due to some state change of interrupt controller variables.

The interrupt controller introduces an interrupt latency (measured from assertion of an interrupt request signal to an assertion of interrupt signal to the CPU of less than 2 clk periods.

2.1.3 Reset signals

Table 168. Reset signals of the INTC module

Name	Type	Description
DTL_MMIO_RST_AN	I	Reset

The interrupt controller gets as an input a fully asynchronous reset signal (rst_an), which is internally synchronized to INT_CLK. The state of interrupt controller and bus adapter is initialized synchronous to clk. The minimum period of active reset (rst_an = 0) is 1 clk period.

2.1.4 DMA transfer signals

INTC block does not have any interface with the DMA block for DMA functionality. However, there is an interrupt request signal coming from DMA controller, which is one of the total 29 interrupts the INTC block receives.

2.2 Available interrupts

Table 169. Available interrupts

Module	Interrupt Source	Interrupt Number	Interrupt
Event Router	CASCADED_IRQ_0	1	Event Router IRQ0
	CASCADED_IRQ_1	2	Event Router IRQ1
	CASCADED_IRQ_2	3	Event Router IRQ2
	CASCADED_IRQ_3	4	Event Router IRQ3
TIMER0	TIMER0_INTCT	5	Count INT Timer0
TIMER1	TIMER1_INTCT	6	Count INT Timer1
TIMER2	TIMER2_INTCT	7	Count INT Timer2
TIMER3	TIMER3_INTCT	8	Count INT Timer3
ADC 10 Bit	ADC_INT	9	ADC INT
UART	UART_INTREQ	10	RECEIVER ERROR FLAG
			RECEIVE DATA AVAILABLE
			TIME-OUT
			TRANSMIT HOLDING EMPTY

Table 169. Available interrupts *…continued*

Module	Interrupt Source	Interrupt Number	Interrupt
I2C0 Master/Slave	I2C0_NINTR	11	TRANSMIT DONE
			TRANSMIT ARBITRATION FAILURE
			TRANSMIT NO ACK
			MASTER TRANSMITTER DATA REQUEST
			SLAVE TRANSMITTER DATA REQUEST
			RECEIVE FIFO FULL
			RECEIVE DATA AVAILABLE
			TRANSMIT FIFO NOT FULL
			SLAVE TRANSMIT FIFO NOT FULL
I2C1 Master/Slave	I2C1_NINTR	12	TRANSMIT DONE
			TRANSMIT ARBITRATION FAILURE
			TRANSMIT NO ACK
			MASTER TRANSMIT DATA REQUEST
			SLAVE TRANSMITTER DATA REQUEST
			RECEIVE FIFO FULL
			RECEIVE DATA AVAILABLE
			TRANSMIT FIFO NOT FULL
			SLAVE TRANSMIT FIFO NOT FULL
I2S Subsystem	I2STX0_IRQ	13	I2S0 TRANSMIT INTERRUPT
	I2STX1_IRQ	14	I2S1 TRANSMIT INTERRUPT
	I2SRX0_IRQ	15	I2S0 RECEIVE INTERRUPT
	I2SRX1_IRQ	16	I2S1 RECEIVE INTERRUPT
	reserved	17	-
LCD INTERFACE	LCD_INTERFACE_IRQ	18	LCD FIFO EMPTY
			LCD FIFO HALF EMPTY
			LCD FIFO OVERRUN
			LCD READ VALID

Table 169. Available interrupts *…continued*

Module	Interrupt Source	Interrupt Number	Interrupt
SPI	SPI_SMS_INT	19	SPI SMS
	SPI_TX_INT	20	SPI Transmit
	SPI_RX_INT	21	SPI Receive
	SPI_OV_INT	22	SPI OV
	SPI_INT	23	SPI Interrupt
DMA	DMA_IRQ	24	DMA DATA TRANSFER COMPLETE
NANDFLASH CTRL	NANDFLASH_CTRL_IRQ_NAND	25	NANDFLASH CTRL Interrupt
MCI	SD_MMC_INTR	26	MCI Interrupt
USB OTG	USB_OTG_IRQ	27	USB OTG Interrupt
ISRAM0	ISRAM0_MRC_FINISHED	28	ISRAM0 Interrupt
ISRAM1	ISRAM1_MRC_FINISHED	29	ISRAM1 Interrupt

In the above table, for UART, I2C0, I2C1 and LCD-Interface modules, any one or more of the multiple sources (4th column) can cause the interrupt (2nd column) shared by them, as they are assigned to that single interrupt bit. The exact source which caused the interrupt can be distinguished by reading the appropriate bits from registers inside the module. To find these details, please refer to the chapters for these specific modules.

2.3 Block Diagram

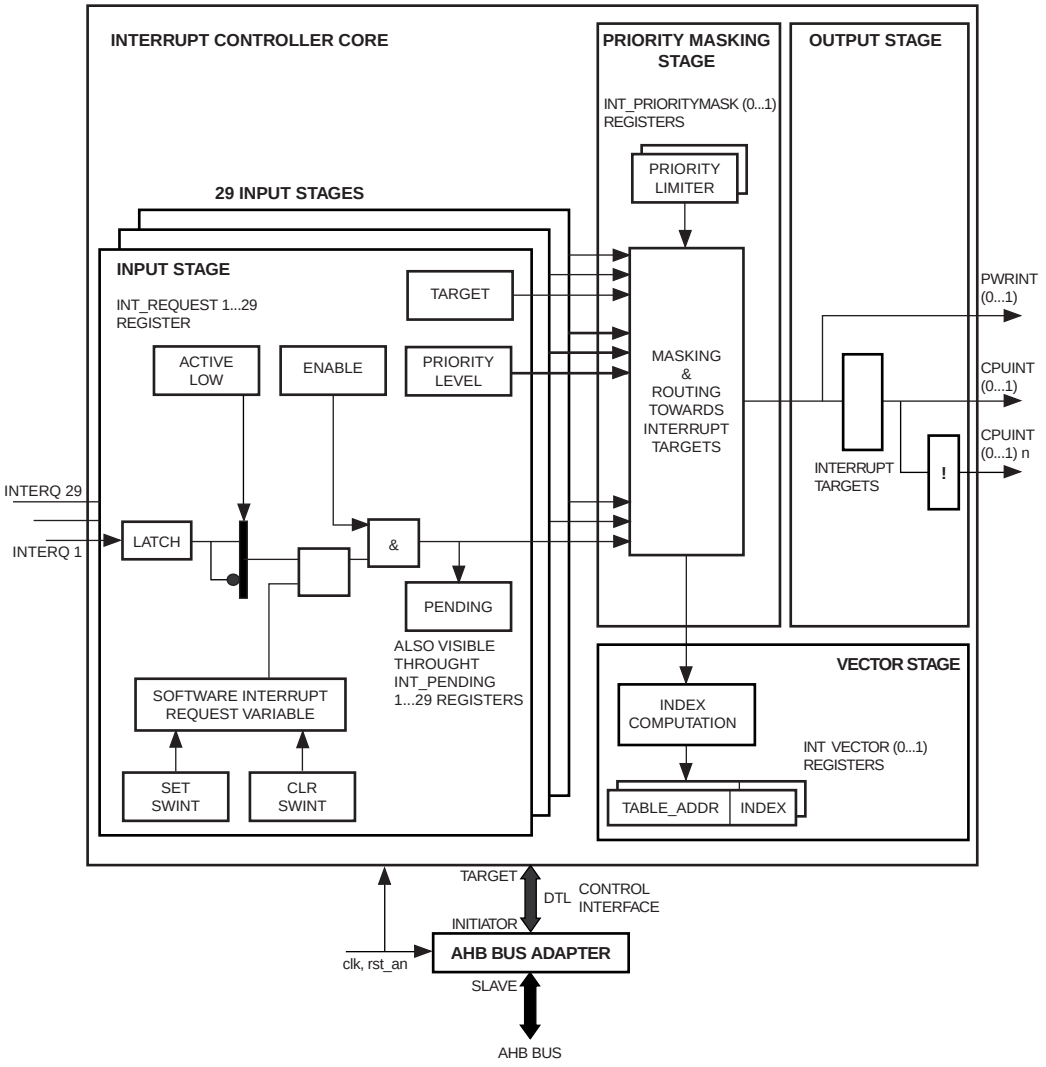


Fig 30. Interrupt controller block diagram

2.4 Short Description of sub blocks

2.4.1 Input Stage

An input stage performs the following tasks (see Figure 10–30):

- Input of one interrupt request (intreq) signal
- Latch the interrupt request state during computation of the interrupt vector, otherwise keep the latch transparent
- Invert the request polarity if the interrupt request signal is active low (controlled by the variable ACTIVE_LOW),
- Combine the interrupt request with the state of a local software interrupt request variable

- Enable or disable the resulting interrupt request (controlled by variable ENABLE), and finally
- Forward the request to the priority masking stage together with attributes characterizing the interrupt request. These attributes are: - the priority level assigned to the request (variable PRIORITY_LEVEL) - the interrupt target defined for the request (variable TARGET).

In addition, the input stage provides means to set and clear the software interrupt request variable (SET_SWINT and CLR_SWINT commands) and to observe the request status before priority masking (variable PENDING).

While no interrupt vector is being computed, the signal path of an interrupt request throughout the input stage (including the latch) is asynchronous and requires no active interrupt controller clock. Immediately before vector computation the latch synchronously captures the state of the intreq line and thereby blocks any signal changes to propagate into the vector stage potentially leading to incorrect index computation. Sufficient time to resolve potential metastability of the latched request will be allowed. After vector computation, the latch becomes transparent again.

There is one input stage per interrupt request input. The number of input stages is from 1 to 29 (see Table 10–169). The vector stage references an input stage through an index. There is no input stage defined for index 0 as this index is reserved for a special purpose by the vector stage.

2.4.2 Priority Masking Stage

The priority masking stage performs the following tasks:

- For each of the 2 interrupt targets, input all interrupt requests selected for the target and mask pending interrupts which are at lower or equal priority than a target specific priority threshold (PRIORITY_LIMITER)
- For each of the 2 interrupt targets, combine pending interrupt requests with priority above the priority threshold through a logical OR and route the result towards the interrupt target.

The signal path of interrupt requests throughout the priority masking stage towards the output stage is asynchronous and requires no active interrupt controller clock.

2.4.3 Output Stage

The output stage performs the following tasks:

- For each interrupt target, produce processor interrupt request output signals cpuint{0...1}(_n) at both active high and low level by registering the interrupt request information of the priority masking stage.

The interrupt controller introduces an interrupt latency (measured from assertion of an intreq. signal to an assertion of cpuint{0..1}(_n)) of less than 2 clock periods.

2.4.4 Vector Stage

The vector stage provides one vector register per interrupt target (INT_VECTOR_{0...1}). It performs the following tasks triggered by a read action to one of these registers:

- For a read of register INT_VECTOR_t, process the PRIORITY information of input states with pending interrupt requests selected for TARGET = t,
- Then identify the input stage with the highest PRIORITY value above the target specific PRIORITY_LIMITER threshold (if this condition is true for a multitude of input stages, then the input stage with the highest index is taken), and finally
- Present the index of that input stage through the INDEX variable in the INT_VECTOR_t register. If no interrupt request exceeds the PRIORITY_LIMITER threshold, then INDEX=0 is given.

The above process is performed upon any INT_VECTOR_* read action - there is no storage of a previously computed vector.

The information from the INT_VECTOR_* register facilitates a generic software ISR in identifying the interrupt requesting device to be serviced. To invoke the ISR of that device, the INDEX variable can be taken as offset into a table of address pointers towards device specific ISR. Alternatively, the total content of the INT_VECTOR_* register, consisting of a table base address (variable TABLE_ADDR) plus INDEX, can be taken as pointer into a table.

INDEX = 0 identifies the special case that no interrupt request requires service when the INT_VECTOR_* register is read.

For correct vector computation, it is required that the ISR always reads the INT_VECTOR_* register that corresponds to the interrupt target.

If 2 interrupts with same priority get activated at the same time, then the interrupt with lower identity number (see Table 10–169) takes priority.

3. Register overview

The purpose of the control interface is to give a processor read and write access to internal registers of the interrupt controller.

When reading one of the INT_VECTOR_{0...T} registers, then read access time is extended in respect to read access time of other registers by 2 additional wait cycles. This extra wait time covers the needs of vector computation and meta-stability resolution on latched intreq. signals even at maximum clk frequency.

Table 170. Register overview: Interrupt controller (base address 0x6000 0000)

Name	R/W	Address offset	Description
INT_PRIORITYMASK_0	R/W	0x000	interrupt target 0 priority threshold
INT_PRIORITYMASK_1	R/W	0x004	interrupt target 0 priority threshold
INT_VECTOR_0	W	0x100	Vector register for target 0 => nIRQ
INT_VECTOR_1	R/W	0x104	Vector register for target 1 => nFIQ
INT_PENDING_1_31	W	0x200	status of interrupt request 1..29 (3 bits don’t care)
INT_FEATURES	R/W	0x300	interrupt controller configuration features
INT_REQUEST_1	R/W	0x404	interrupt request 1 configuration features
INT_REQUEST_2	R/W	0x408	interrupt request 2 configuration features

Table 170. Register overview: Interrupt controller (base address 0x6000 0000)

Name	R/W	Address offset	Description
INT_REQUEST_3	R/W	0x40C	interrupt request 3 configuration features
INT_REQUEST_4	R/W	0x410	interrupt request 4 configuration features
INT_REQUEST_5	R/W	0x414	interrupt request 5 configuration features
INT_REQUEST_6	R/W	0x418	interrupt request 6 configuration features
INT_REQUEST_7	R/W	0x41C	interrupt request 7 configuration features
INT_REQUEST_8	R/W	0x420	interrupt request 8 configuration features
INT_REQUEST_9	R/W	0x424	interrupt request 9 configuration features
INT_REQUEST_10	R/W	0x428	interrupt request 10 configuration features
INT_REQUEST_11	R/W	0x42C	interrupt request 11 configuration features
INT_REQUEST_12	R/W	0x430	interrupt request 12 configuration features
INT_REQUEST_13	R/W	0x434	interrupt request 13 configuration features
INT_REQUEST_14	R/W	0x438	interrupt request 14 configuration features
INT_REQUEST_15	R/W	0x43C	interrupt request 15 configuration features
INT_REQUEST_16	R/W	0x440	interrupt request 16 configuration features
INT_REQUEST_17	R/W	0x444	interrupt request 17 configuration features
INT_REQUEST_18	R/W	0x448	interrupt request 18 configuration features
INT_REQUEST_19	R/W	0x44C	interrupt request 19 configuration features
INT_REQUEST_20	R/W	0x450	interrupt request 20 configuration features
INT_REQUEST_21	R/W	0x454	interrupt request 21 configuration features
INT_REQUEST_22	R/W	0x458	interrupt request 22 configuration features
INT_REQUEST_23	R/W	0x45C	interrupt request 23 configuration features
INT_REQUEST_24	R/W	0x460	interrupt request 24 configuration features
INT_REQUEST_25	R/W	0x464	interrupt request 25 configuration features
INT_REQUEST_26	R/W	0x468	interrupt request 26 configuration features
INT_REQUEST_27	R/W	0x46C	interrupt request 27 configuration features
INT_REQUEST_28	R/W	0x470	interrupt request 28 configuration features
INT_REQUEST_29	R/W	0x474	interrupt request 29 configuration features

4. Register description

4.1 Interrupt Priority Mask Register

Table 171. INT_PRIORITYMASK register (INT_PRIORITYMASK 0, address 0x6000 0000 and INT_PRIORITYMASK1 address 0x6000 0004)

Bit	Symbol	Access	Reset Value	Description
[31:8]	inter_slave_dly	Reserved	X	Reserved for future extensions; should be written as 0
[7:0]	PRIORITY_LIMITER	R/W	X	Priority Limiter: this variable determines a priority threshold that incoming interrupt requests must exceed to trigger interrupt requests towards processor. See text below.

Legal PRIORITY_LIMITER values are 0 …15; other values are reserved and lead to undefined behavior.

PRIORITY_LIMITER = 0: incoming interrupt requests with priority >0 can trigger interrupt requests towards processor.

PRIORITY_LIMITER = n: only interrupt requests at a priority level above n can trigger interrupt requests towards processor.

PRIORITY_LIMITER = 15: no incoming interrupt requests can trigger interrupt requests towards processor.

The PRIORITY_LIMITER variable can be used to define the minimum priority level for nesting interrupts: typically, the PRIORITY_LIMITER variable is set to the priority level of the ISR that is currently being executed. By doing this, only interrupt requests at a higher priority level will lead to a nested interrupt service. Nesting can be disabled by setting PRIORITY_LEVEL = 15 or by disabling interrupt exceptions within the processor.

4.2 Interrupt Vector Register

These registers identify, individually for each interrupt target, the highest priority enabled pending interrupt request that is present at the time when a register is being read.

Table 172. INT_VECTOR registers (INT_VECTOR0, address 0x6000 0100 and INT_VECTOR1, address 0x6000 0104)

Bit	Symbol	Access	Reset Value	Description
31:11	TABLE_ADDR	R/W	X	Table start address: Indicates the lower address boundary of a 2048 byte aligned interrupt vector table in memory
10:3	INDEX	R	X	Index: Indicates the intreq line number of the interrupt request to be served by the processor: INDEX = 0: no interrupt request to be served INDEX = 1: serve interrupt request at input intreq1 INDEX = 2: serve interrupt request at input intreq2 .... INDEX = 29: serve interrupt request at input intreq29
2:0	NULL	R	0	bit field always read as zero

The software ISR must always read the vector register that corresponds to the interrupt tar-get, e.g.:

- read INT_VECTOR_0 for interrupt target 0 service. => nIRQ
- read INT_VECTOR_1 for interrupt target 1 service. => nFIQ

The INT_VECTOR_n register content can be used as a vector into a memory based table. This table has N+1 entries. To be able to use the register content as a full 32 bit address pointer, the table must be aligned to a 2048 byte address boundary. If only the INDEX variable is used as offset into the table, then this address alignment is not required.

Each table entry has 64 bit data. It is recommended to pack per table entry:

the start address of a device specific ISR, plus the associated priority limiter value (if nesting of ISR shall be performed).

64 bit packing will optimize the speed of nested interrupt handling due to caches. In the (likely) case of a cache miss when reading data from the table, the priority limiter value to be programmed into the INT_PRIORITYMASK register will be loaded into the cache along with the ISR start address, saving several clock cycles interrupt processing time compared to a solution where the priority limiter value would have to be established from an INT_REQUEST_* register.

A vector with INDEX = 0 indicates that no interrupt with priority above the priority threshold is pending. The vector table should implement for this entry a "no interrupt" handler to treat this special case.

Remark: Due to the special purpose of INDEX = 0 no interrupt request input intreq0 and thus no INT_REQUEST_0 register exists.

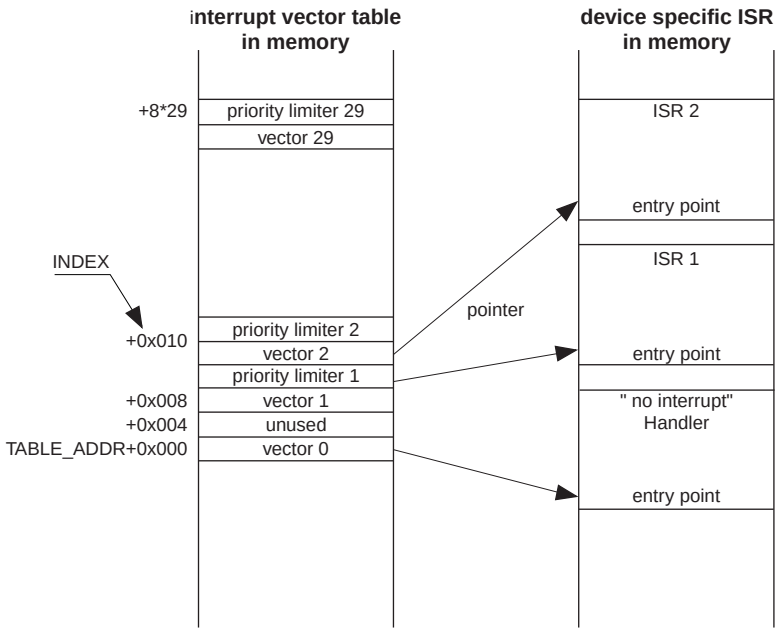


Fig 31. Memory based interrupt table

4.3 Interrupt Pending Register

This register gathers the PENDING variables of all interrupt requests.

Software can make use of INT_PENDING_1_31[29:1] register to gain a faster overview on pending interrupt requests than by reading individual INT_REQUEST_n registers. For certain software this may lead to a benefit in interrupt processing time.

INT_PENDING_1_31[29:1] reflects the state of signals intreq1...29.

Table 173. INT_PENDING register (INT_PENDING1_31, address 0x6000 0200)

Bit	Symbol	Access	Reset	Description
[31:30]	-	R	x	reserved
[29:1]	PENDING[i]	R	X	Pending interrupt request: This variable reflects the state of the intreq[i] line (if needed, converted to active high) OR'ed by the state of the local soft-ware interrupt request variable at the time the register is read. Note that the pending variables are also reflected by the INT_REQUEST_{1..29} registers are present individually for each interrupt request input. PENDING[i] = 0: no interrupt request PENDING[i] = 1: interrupt request is pending
0	-	R	x	reserved

4.4 Interrupt Controller Features Register

This register indicates the hardware configuration parameters chosen during the creation of the interrupt controller. Software can make use of the INT_FEATURES register to implement interrupt controller configuration specific behavior.

Table 174. INT_FEATURES register (address 0x6000 0300)

Bit	Symbol	Access	Reset Value	Description
31:22	Reserved	R	X	Reserved
21:16	T	R	0x01	IC Configuration parameter T: Number of interrupt targets supported (plus one). This is not configurable by Software, hence is a Read-Only parameter.
15:8	P	R	0x0F	Configuration parameter P: Number of priority levels supported. This is not configurable by Software, hence is a Read-Only parameter.
7:0	N	R	0x26[1]	Configuration parameter N: Number of interrupt request inputs. This is not configurable by Software, hence is a Read-Only parameter.

[1] Although this number indicates 38 interrupt sources, the hardware supports only 29 incoming interrupts as described in rest of the document.

4.5 Interrupt Request Registers

These sets of registers holds configuration information related to interrupt request inputs of the interrupt controller and allows issuing software interrupt requests.

There are 29 interrupt request registers, one for each intreq input signal.

Table 175. INT_REQUEST registers (INT_REQUEST1, address 0x6000 0404 to INT_REQUEST29, address 0x6000 0474)

Bit	Symbol	Access	Reset Value	Description
31	PENDING	R	X	Pending interrupt request: This variable reflects the state of the intreq line (if needed, converted to active high) OR'ed by the state of the local soft-ware interrupt request variable at the time the register is read. Note that the PENDING variable is also visible from the INT_PENDING_* registers. PENDING = 0: no interrupt request PENDING = 1: interrupt request pending
30	SET_SWINT	W	0	Set software interrupt request SET_SW_INT = 0 (write): no effect on the state of the local software interrupt request variable SET_SWINT = 1 (write): set the state of the local software interrupt request variable to '1' SET_SWINT is always reads as 0
29	CLR_SWINT	W	0	Clear software interrupt request: CLR_SWINT = 0 (write): clear the state of the local software interrupt request variable to '0' CLR_SWINT is always read as 0
28	WE_PRIORITY_LEVEL	W	X	Write Enable PRIORITY_LEVEL WE_PRIORITY_LEVEL = 0 (write): no change of PRIORITY_LEVEL variable state WE_PRIORITY_LEVEL = 1 (write): PRIORITY_LEVEL variable state may be changed WE_PRIORITY_LEVEL is always read as 0
27	WE_TARGET	W	X	Write Enable TARGET WE_TARGET = 0 (write): no change of TARGET variable state WE_TARGET = 1 (write): TARGET variable state may be changed WE_TARGET is always read as 0
26	WE_ENABLE	W	X	Write Enable ENABLE WE_ENABLE = 0 (write): no change of ENABLE variable state WE_ENABLE = 1 (write): ENABLE variable state may be changed WE_ENABLE is always read as 0

Table 175. INT_REQUEST registers (INT_REQUEST1, address 0x6000 0404 to INT_REQUEST29, address 0x6000 0474) *…continued*

Bit	Symbol	Access	Reset Value	Description
25	WE_ACTIVE_LOW	W	X	Write Enable ACTIVE_LOW WE_ACTIVE_LOW = 0 (write): no change of ACTIVE_LOW variable state WE_ACTIVE_LOW = 1 (write): ACTIVE_LOW variable state may be changed WE_ACTIVE_LOW is always read as 0
24:18	Reserved	R	X	Reserved; should be written as zeros
17	ACTIVE_LOW	R/W	0	Active Low This variable selects the polarity of the interrupt request input signal. See also WE_ACTIVE_LOW. ACTIVE_LOW = 1: the intreq signal is interpreted as active low ACTIVE_LOW = 0: the intreq signal is interpreted as active high
16	ENABLE	R/W	0	Enable interrupt request This variable controls whether an interrupt request is enabled for further processing by the interrupt controller. See also WE_ENABLE. ENABLE = 0: the interrupt request is discarded. It cannot cause a processor interrupt request. ENABLE = 1: the interrupt request may cause a processor interrupt request when further conditions for this become true.

Table 175. INT_REQUEST registers (INT_REQUEST1, address 0x6000 0404 to INT_REQUEST29, address 0x6000 0474) *…continued*

Bit	Symbol	Access	Reset Value	Description
15:14	Reserved	R	X	Reserved; should be written as zeros
13:8	TARGET	R/W	0	Interrupt target: This variable defines the interrupt target of an interrupt request. Legal values are 0 ... 1; other values are reserved and lead to undefined behaviour. See also WE_TARGET. TARGET = 0: the interrupt request shall lead to a processor interrupt request 0 (cpuint0) => nIRQ TARGET = 1: the interrupt request shall lead to a processor interrupt request 1 (cpuint1) => nFIQ High order bits not required for TARGET encoding are read-only 0.
7:0	PRIORITY_LEVEL	R/W	X	Priority level This variable determines the priority level of the interrupt request. Legal values are 0 ... P; other values are reserved and lead to undefined behaviour. See also WE_PRIORITY_LEVEL. PRIORITY_LEVEL = 0: the interrupt request has priority level 0 (masked); it is ignored PRIORITY_LEVEL = 1: the interrupt request has priority level 1 (lowest) ... PRIORITY_LEVEL = 15: the interrupt request has priority level 15 (highest) High order bits not required for PRIORITY_LEVEL encoding are read-only 0

Remark: There is no INT_REQUEST_0 register.

For changing the TARGET variable state dynamically, software must first disable the interrupt request (ENABLE = 0), then change TARGET and finally re-enable the request (ENABLE = 1).

Write enable commands are provided to allow the modification of individual INT_REQUEST_* variables by simple write operations instead of atomic read-modify-write operations. This feature allows to access INT_REQUEST_* registers simultaneously by multiple software threads.

5. Functional description

5.1 Why Vectored?

For each incoming interrupt, its source & priority are determined by INTC hardware. Being vectored helps IRQ handler to be simple & quick in response.

5.2 Interrupt Targets

The application of interrupt targets is not prescribed by the architecture. It may be specific to a system hardware/software design and may depend on the capabilities of the processor handling the interrupts. The interrupt architecture, as specified by ARM, recommends the following use of interrupt targets:

- ARM processor: target 0 = nIRQ (standard interrupt service with full context state save/restore) target 1 = nFIQ (fast interrupt service with minimal context save/restore)

These recommendations are adapted in the LPC3130/31 hardware also.

The interrupt target is configured for each interrupt request input of the interrupt controller through the TARGET variable in the INT_REQUEST_* registers.

5.3 Interrupt Priority

Interrupt request masking is performed individually per interrupt target by comparing the priority level assigned to a specific interrupt request input (variable PRIORITY_LEVEL in the INT_REQUEST_* registers) with a target specific priority threshold (variable PRIORITY_LIMITER in the INT_PRIORITYMASK_* registers).

Priority levels are defined as follows:

- Priority level 0 corresponds to 'masked'. Interrupt requests with priority 0 will never lead to an interrupt request towards processor.
- Priority level 1 corresponds to lowest priority.
- Priority level 15 corresponds to highest priority.

Programming the INT_REQUEST_* register variable ENABLE = 0 is an alternative to PRIORITY_LEVEL = 0 which is typically applied when an interrupt request input shall be temporarily disabled without the need to save and restore the current PRIORITY_LEVEL setting.

6. Power optimization

To reduce the power consumption of the interrupt controller, the 'clock gating' option is chosen. With clock gating, the clock for all software accessible registers is provided only during the course of a write operation and during synchronous reset. Outside these conditions, the clock is disabled and internal power dissipation of registers is close to zero.

7. Programming guide

7.1 Software interrupts

Software interrupt support is provided through variables in the INT_REQUEST_n registers. Software interrupts can be applied for:

- test the RTOS interrupt handling without using a device specific ISR.
- software emulation of an interrupt requesting device, including interrupts.

1. Introduction

The multi-layer AHB is an interconnection scheme based on the AHB protocol that enables parallel access paths between multiple masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth and a more flexible system architecture.

Multiple masters can have access to different slaves at the same time. When multiple masters want to have access to the same slave, a so called Round-Robin mechanism is used for bus arbitration.

1.1 Features

This module has the following features:

- Supports all combinations of 32-bit masters and slaves (fully connected interconnect matrix).
- Round-Robin priority mechanism for bus arbitration: All masters have the same priority and get bus access in their natural order.
- 4 devices on a master port (listed in their natural order for bus arbitration):
  - DMA
  - ARM926 Instruction port
  - ARM926 Data port
  - USB OTG
- 14 slave ports:
  - AHB to APB bridge 0
  - AHB to APB bridge 1
  - AHB to APB Bridge 2
  - AHB to APB Bridge 3
  - AHB to APB Bridge 4
  - Interrupt controller
  - NAND buffer
  - MCI SD/SDIO
  - USB OTG
  - ISRAM0 (96 kB)
  - ISRAM1 (96 kB) (LPC3131 only)
  - ISROM (128 kB)
  - MPMC configuration block
  - MPMC controller
- Zero wait state operation, up to 100% bandwidth usage possible.

- Bus implementation includes address decoding, arbitration and signal mixing.
- Designed to work according to the Multi layer AMBA Advanced System Bus (AHB Lite) concept.

1.2 About AHB and multilayer AHB

AHB (Advanced High-performance Bus) is a generation of AMBA bus, which is intended to address the requirements of high-performance designs. AMBA AHB is a level of bus, which sits above the APB and implements the features required for high-performance, high clock frequency systems, including:

- Burst transfers between bus masters and slaves on one layer
- Single cycle bus master hand over
- Single clock edge operation
- Non-tri state implementation.

In a multilayer AHB each layer has only one master. The benefits of a multilayer AHB are:

- This allows an increased bandwidth in comparison to one layer with more than one master.
- The master to slave mixing can be done without arbitration.
- The arbitration effectively becomes point arbitration at each peripheral and is only necessary when more than one master wants to access the same slave simultaneously.

2. General Description

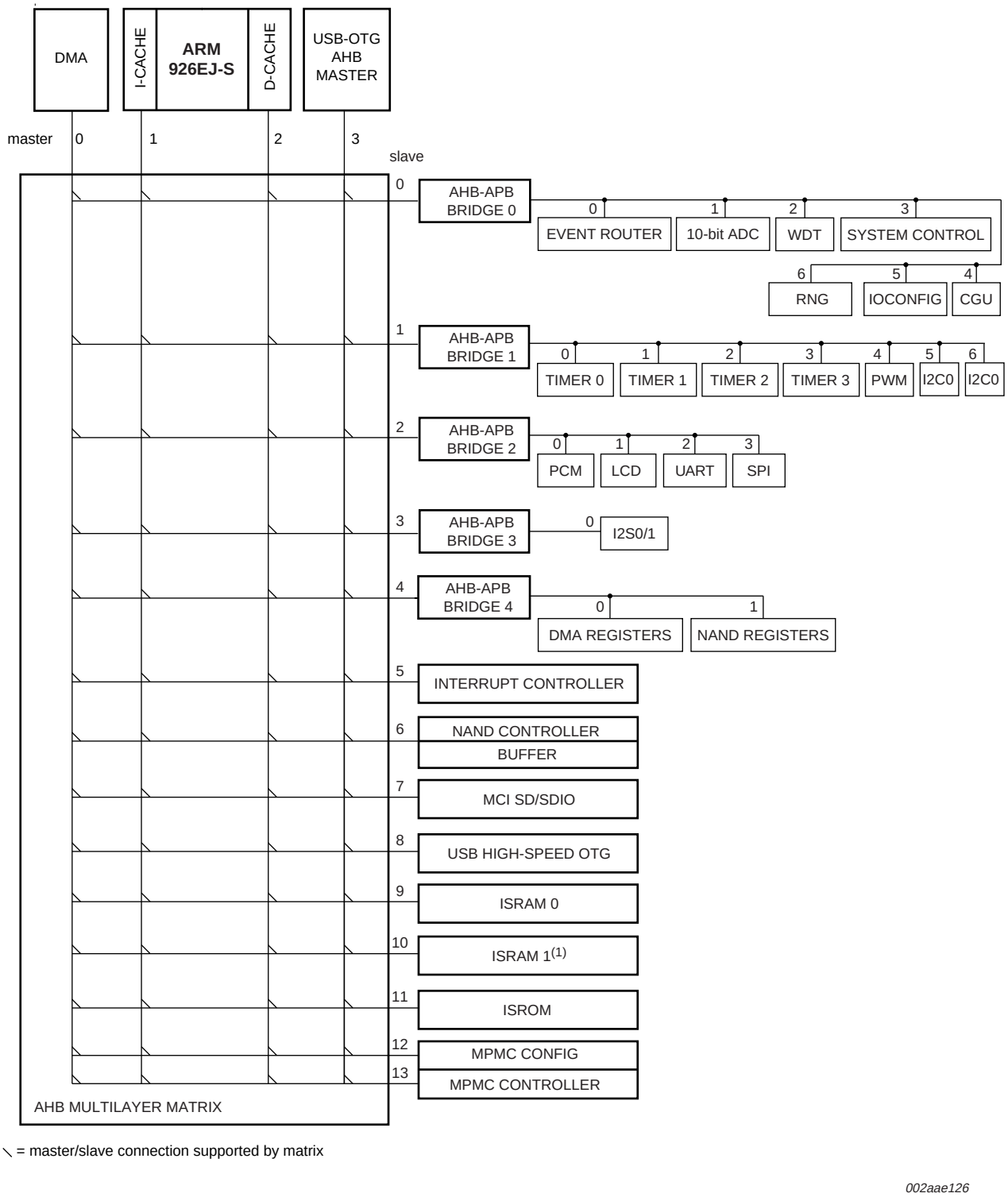


Fig 32. AHB multi-layer block diagram

2.1 Interface description

2.1.1 Clock Signals

Table 176. Clock signals of the AHB Module

Clock Name	I/O	Source/ Destination	Description
Clock			
AHB0_CLK	I	CGU	Main clock of the module. The logic in this module runs on this clock. Its frequency depends on the required speed for connected devices. The maximum frequency of AHB_CLK is 75 MHz.
Enable signals (see Section 13–5.3.2)			
AHB_M<x>_DISABLE_REQ	I	CGU	Master bus access deny request. When this signal becomes high the master may complete its current AHB transfer. After this transfer no new bus access is allowed.
AHB_M<x>_DISABLE_GRANT	I	CGU	Master bus access denied. This signal indicates that the deny request is received and the last allowed bus transfer has finished. The master IP clock can be safely disabled by the CGU.

2.1.2 Reset Signals

The AHB is reset by the (active low) AHB reset through the CGU. Reset is de-asserted synchronously to AHB_CLK; assertion may be done asynchronously to AHB_CLK. NOTE: This signal must be asserted upon a power_on reset.

2.1.3 System control register (Syscreg) signals

Table 177. External priority signals of the AHB Module (see Table 26–544)

Name	Type	Description
AHB_M0_EXTPRIO	I	External priority signal for master zero. If this bit is set, this master has higher priority on the bus for slave x than the masters without its external priority signal set.
AHB_M1_EXTPRIO	I	External priority signal for master one. If this bit is set, this master has higher priority on the bus for slave x than the masters without its external priority signal set.
AHB_M2_EXTPRIO	I	External priority signal for master two. If this bit is set, this master has higher priority on the bus for slave x than the masters without its external priority signal set.
AHB_M3_EXTPRIO	I	External priority signal for master three. If this bit is set, this master has higher priority on the bus for slave x than the masters without its external priority signal set.

Table 178. Shadow signals of the AHB Module (see Table 26–545)

Name	Type	Description
AHB_M1_SHADOW_POINTER	I	This signal is provided to be able to change the memory mapping for ARM instruction bus. The actual re-mapping pointer is a 32-bitvector of which the lower10 bits are '0'. It is freely programmable in increments of 1 kByte.
AHB_M2_SHADOW_POINTER	I	This signal is provided to be able to change the memory mapping for ARM data bus. The actual re-mapping pointer is a 32-bitvector of which the lower10 bits are '0'. It is freely programmable in increments of 1 kByte.

3. Register Overview

Not applicable. The AHB has no internal registers.

4. Functional description

This multi-layer AHB is based on the AHB-Lite protocol. This means that the layers do not support request and grant. Also retry and split transactions are not supported. The bus selects the master to the wanted slave. This selection is done without waitstates. The address decoding of a master to a slave port and routing of the address and command signals to the slave is done within one cycle.

Each master has a full address decoder for all slave peripherals in the system. A key issue in a multi-layer AHB system is the overall system complexity, arising from the number of concurrent actions possible in the system. To reduce the system complexity, the AHB multi-layer system only supports a unified memory map. This means all masters and slaves share a single, global 32-bit address space, and any master can select any slave in the system.

To avoid breaking the unified memory space, a specific section of the unified memory map is assigned as a shadow memory section. This memory section is virtual, i.e. no actual memory is present at the shadow address. It can be seen as a copy of a section of unified memory, specific to each master.

The bus arbiter is integrated in the AHB Multilayer and provides bus arbitration for a total of 4 bus masters. The scheduler determines the priority of the master by making use of the external priority.

The following list summarizes the rules which determine which master for the slave x is granted the bus:

1. A master requesting the bus is given the priority over a master not requesting the bus
2. If only one master with the highest externally assigned priority is requesting the bus, it will receive the bus regardless of the underlying scheduling algorithm
3. If no external priority is specified or all masters are of the same external priority, then the master selected by the scheduler is given the bus
4. If no master is requesting the bus layer then the layer will generate idle cycles
5. A master that locks the bus can keep it indefinitely.

The user set the external priority bits in the configuration register block. The re-arbitration is done in one cycle (at the same time as the next access to a slave port), i.e there are no waitstates.

This chip supports 4 masters and 14 slaves. An overview is given in following table:

Table 179. Shadow signals of the AHB Module

Masters	Description	Slaves	Description
Master 0	Simple DMA	Slave port 0	APB0
Master 1	ARM926 Instruction	Slave port 1	APB1
Master 2	ARM926 Data	Slave port 2	APB2
Master 3	USB OTG	Slave port 3	APB3
		Slave port 4	APB4
		Slave port 5	Interrupt controller
		Slave port 6	Internal RAM0
		Slave port 7	Nand flash controller
		Slave port 8	Mobile storage SD/MMC
		Slave port 9	USB OTG slave
		Slave port 10	Internal sram0
		Slave port 11	Internal sram1
		Slave port 12	MPMC configuration
		Slave port 13	MPMC

5. Power optimization

When a master wishes to enter a power down mode, its IP clock can be stopped using the CGU. For this, the CGU can apply a ‘disable request’. The corresponding master will finish its current bus transfer, else this will cause the corresponding slave device to be locked. As soon as the transfers is completed the ‘disable grant’ signal becomes high and the CGU can safely remove the IP clock of the master.

Slave devices connected to the AHB can use the clock-enable feature to selectively gate the clock as long as required.

1. Introduction

The AHB_TO_APB is a bus bridge between AMBA Advanced High-performance Bus (AHB) and the AMBA Peripheral Bus (APB).

1.1 Features

This module has the following features:

- Supports a uni-directional (slave only) APB data bus interface.
- One word deep write buffer.
- Single clock architecture with one clock domain (APB and AHB clock are synchronal). On LPC3130/31 AHB-APB Bridge 4 uses single clock architecture.
- Dual clock architecture with independent AHB and APB clock domains. On LPC3130/31 AHB-APB Bridge 0, AHB-APB Bridge 1, AHB-APB Bridge 2, and AHB-APB Bridge 3 use this architecture.

2. General description

2.1 Block diagram

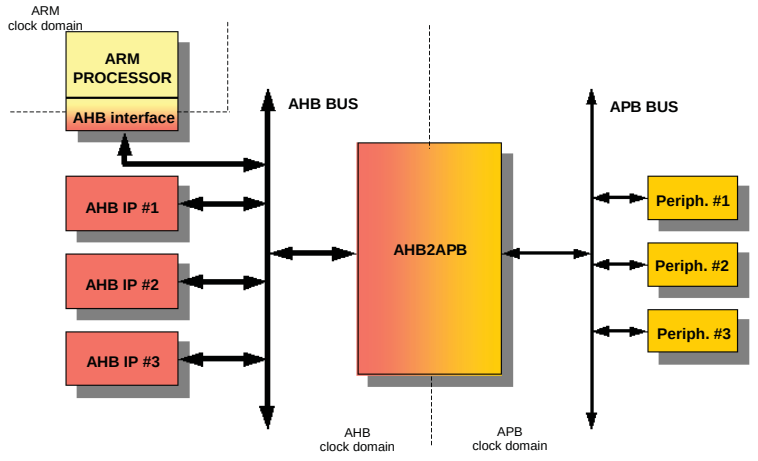


Fig 33. AHB Block Diagram

The AHB_TO_APB Bridge translates bus transactions generated by an AHB master to APB bus transactions. The bridge can de-couple a slow peripheral access by parking the AHB transaction on the bridge and free the high speed/performance AHB bus resource.

2.2 Interface description

The AHB_TO_APB bridge interface consists of a APB bus interface and a AHB bus interface. The AHB_TO_APB bridge has separate and independent clocks and reset signals for the AHB and the APB interface.

The APB interface is connected to the peripheral blocks of the APB subsystem.

The AHB interface is connected to the multi-layer AHB bus.

2.2.1 Clock Signals

The LPC3130/31 uses two architecture types of the AHB_TO_APB Bridge: a single clock architecture, which has only one clock input, and a dual clock architecture, which has two clocks.

The clocks can be asynchronous i.e. need not have a frequency or phase relation. Because the APB subsystem usually hosts slow peripherals, the APB_clk frequency is lower than the AHB_clk frequency.

Table 180. Clock Signals of the AHB_TO_ABP

Clock Name	I/O	Source/ Destination	Description
APB[0:3]_PCLK	I	CGU	Determines the operating frequency of the APB interface of the bridge. Operates the APB interface.
APB[0:4]_CLK	I	CGU	Determines the operating frequency of the AHB interface of the bridge. Operates the AHB interface.

2.2.2 Reset signals

The CGU provides two reset signals to the AHB_TO_APB bridges: AHB_RST_AN, AHB interface global asynchronous reset and APB_RST_AN, APB interface global asynchronous reset.

3. Register overview

The AHB_TO_APB does not have specific configuration registers.

4. Detailed architecture and functional modes description

4.1 Memory Endianess

The bus bridge operation is independent of the endianess memory format.

4.2 Data Steering

Data steering for peripherals that have a narrow data bus (8 or 16 bits) is not supported. These peripherals are assumed to be accessed with word (32-bit) aligned addresses.

Peripherals with sub-word aligned addressing can be connected to the AHB_TO_APB Bridge by shifting the address bits. Bits [x:0] of a byte-aligned peripheral must be connected to bits [x+2:2] of the bridge. Bits [x:0] of a half-word (16-bit) aligned peripheral must be connected to bits [x+1:1] of the bridge. In both cases, the master on the AHB side of the bridge (software) must use word aligned addressing.

4.3 Write Buffer

The AHB_TO_APB Bridge contains a one-word deep write buffer. Any APB device that DOES NOT have a APB_err signal (like all APB devices) will take advantage of the write buffer provided ahb_prot is 1, when ahb_prot is 0 the devices can not use the write buffer. On LPC3130/31 only SPI module generates APB_err signal. Hence write buffer is used by all other devices/modules connected to APB bus except SPI.

Devices with a APB_err signal can not use the write buffer. The write buffer alleviates putting wait states on AHB. However, consecutive write access or write-read accesses to the bridge will insert some wait states because the write buffer is only one word deep.

4.4 Address Alignment

The AHB_TO_APB Bus Bridge allows the user to enter system specifications and information about each of the peripherals connected to the bridge. The address space allocated to each peripheral is described in ‘Memory Map’ section. The bridge will assign the APB memory map based on these parameters.

5. Power optimization

The AHB_TO_APB module has an asynchronous clock domain crossing, allowing the APB clock frequency to be independent from the AHB clock frequency. This allows power saving by lowering the APB bus frequency while keeping a AHB interface with high clock frequency.

When using the AHB_TO_APB you must aim to meet the following guidelines:

- Operate at APB clock speed, when possible.
- Independently from the APB clock, reduce the AHB_CLK if possible.
- Switch off clocks when the device and its subsystem is not in use.
- Use the bridge write buffer for a more efficient data transfer.

6. Programming guide

AHB and APB bus clock frequencies can be set and/or disabled via the CGU registers.

1. Introduction

The Clock Generation Unit (CGU) is used for delivering all the clocks which are needed for the blocks of the digital die.

1.1 Features

This module has the following features:

- Several advanced features to optimize the system for low power:
  - All output clocks can be disabled individually for flexible power optimization. Some modules have automatic clock gating, which means that they are only active when (bus) access to the module is required.
  - Variable clock scaling for automatic power optimization of the AHB bus (high clock frequency when the bus is active, low clock frequency when the bus is idle)
  - Clock wake-up feature: when switched off, module clocks can be programmed to be activated automatically on the basis of an (external) event detected by the Event Router. An example of the use of this feature would be that all clocks (including the ARM / bus clocks) are off and activated automatically when a button is pressed
- Seven Clock sources:
  - Reference clock is generated by the oscillator with an external 12 MHz crystal.
  - Two external clock signals from the I2SRX_BCK0 and I2SRX_WS0 pins (used for generating audio frequencies in I2SRX0 / I2STX0 slave mode).
  - Two external clock signals from the I2SRX_BCK1 and I2SRX_WS1 pins (used for generating audio frequencies in I2SRX1 / I2STX1 slave mode).
  - Programmable system clock frequency is generated by the System PLL.
  - Programmable audio clock frequency (typically 512 x fs) is generated by the Audio PLL.

  Both the System PLL and the Audio PLL generate their own frequencies based on their (individual) reference clocks. The reference clocks can be programmed to the oscillator clock, or one of the external clock signals.
- Highly flexible switchbox to distribute the signals from the clock sources to the module clocks:
  - Each clock generated by the CGU is derived from one of the base clocks and optionally divided by a fractional divider
  - Each base clock can be programmed to have any one of the clock sources as an input clock
  - Fractional dividers can be used to divide a base clock by a fractional number
  - Fractional dividers support clock stretching to obtain a (near) 50% duty cycle output clock
- Register interface to reset all modules under software control.

- Based on the input of the Watchdog timer, the CGU can generate a system-wide reset in the case of a system hang-up.

2. General description

The CGU generates all the clock signals in the system and controls the reset signals for all modules. As shown in the block diagram of the CGU in Figure 13–34, the CGU has a regular structure. Each output clock generated by the CGU belongs to one of the system or audio clock domains. Each clock domain is fed by a single base clock that originates from one of the available clock sources.

Within a clock domain, fractional dividers are available to divide the base clock into a lower frequency. Within most clock domains, the output clocks are again grouped into one or more sub domains.

All output clocks within one sub domain are either all generated by the same fractional divider or they are connected directly to the base clock. Therefore all output clocks within one sub domain have the same frequency and all output clocks within one clock domain are synchronous because they originate from the same base clock The CGU has a reference clock (generated by the oscillator) and several external clock inputs.

The CGU also has several phase locked loop (PLL) circuits to generate clock signals that can be used for system clocks and/or audio clocks. All clock sources, except the output of the PLLs, can be used as reference input for the PLLs.

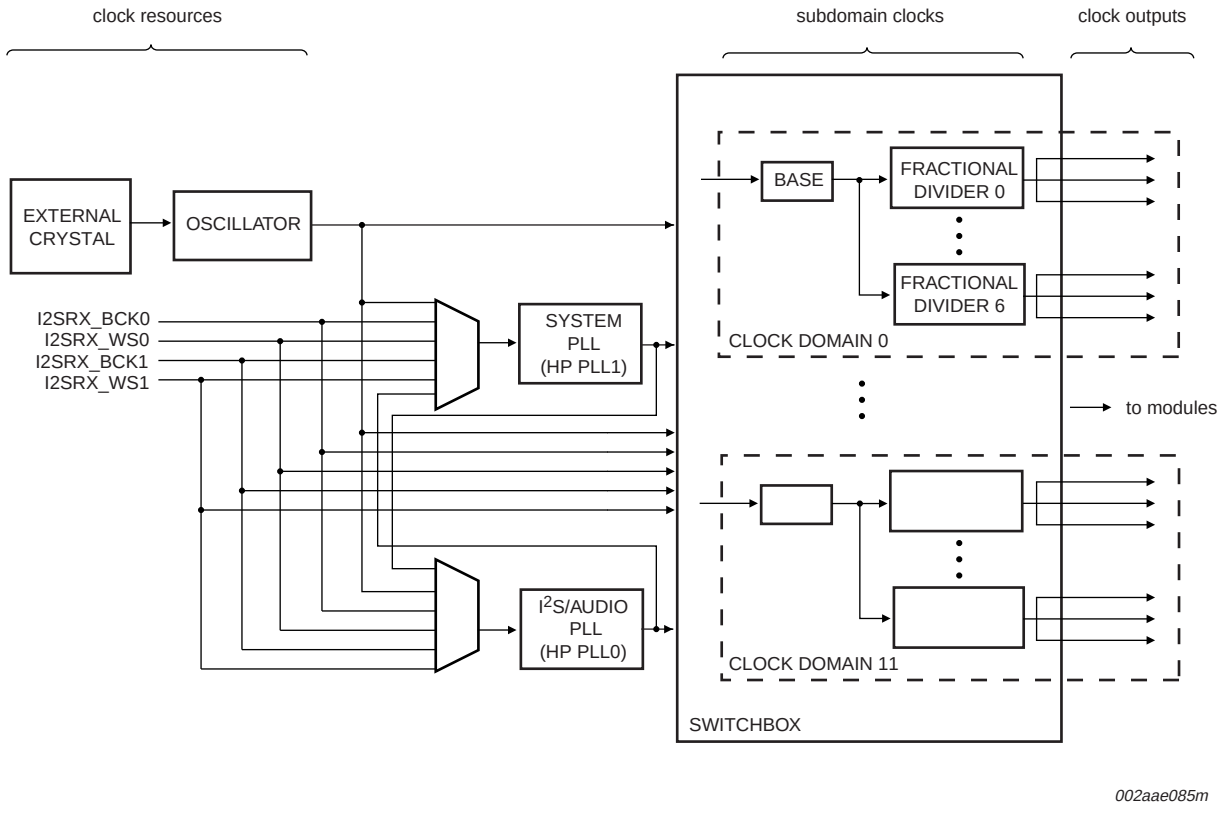


Fig 34. CGU block diagram

Table 181. CGU base clock domains and associated fractional dividers

Base clock domain	Domain #	Fractional dividers
SYS_BASE	0	0 to 7
AHB_APB0_BASE	1	7 and 8
AHB_APB1_BASE	2	9 and 10
AHB_APB2_BASE	3	11 to 13
AHB_APB3_BASE	4	14
PCM_BASE	5	15
UART_BASE	6	16
CLK1024FS_BASE	7	17 to 22
I2SRX_BCK0_BASE	8	none
I2SRX_BCK1_BASE	9	none
SPI_CLK_BASE	10	23
SYSCLK_O_BASE	11	none

2.1 Interface description

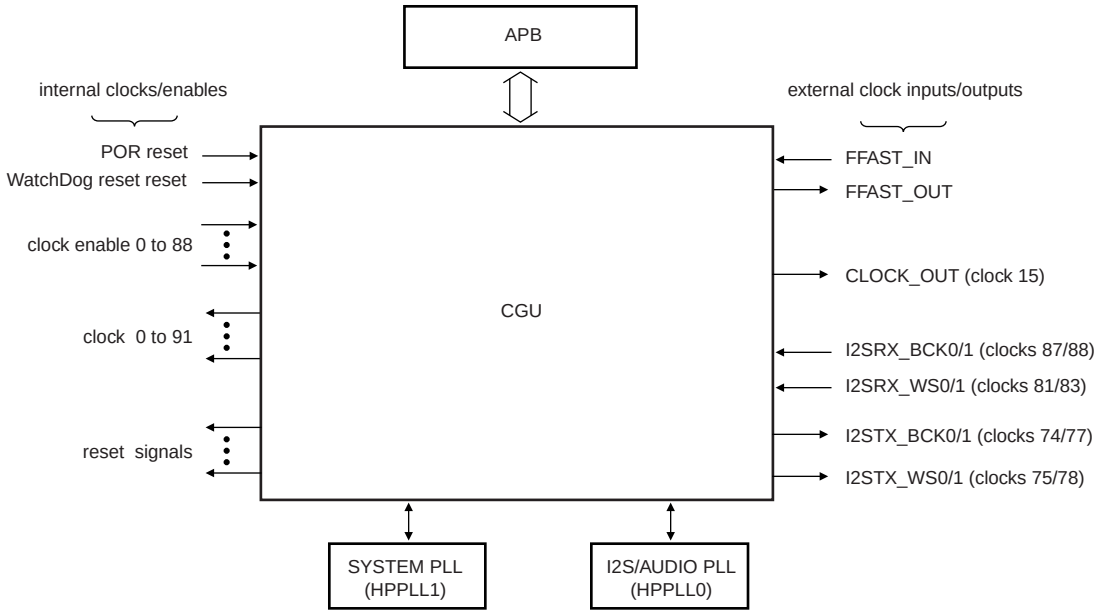


Fig 35. Interface block diagram

2.1.1 Clock signals

In the following table all base clocks and their derived clocks are listed. Each clock is assigned a number which is used in the corresponding Power Control Registers (PCR0 to PCR91) and the Power Status Registers (PSR0 to PSR91), see Table 13–183.

The LPC3130/31 has a total of 24 fractional dividers FDC0 to FDC23 which are distributed among the 12 base clock domains. Each base clock domain has pre-assigned fractional dividers which can be used to further divide the base clock (see Table 13–181). The output of the fractional dividers or base clock can be used as source for the clocks belonging to that domain.

In addition the SYS_BASE clock domain has seven dynamic fractional dividers (DYN_FDC0 to DYN_FDC6) to generate slow clocks corresponding to FDC0 to FDC6. When dynamic fractional dividers are enabled, LPC313x automatically switches to slow clocks (DYNC_FDC0 - DYNC_FDC6) from fast clocks (FDC0 - FDC6) when there is no AHB bus activity. For more details, see Section 13–5.1.5.

For a detailed description of the CGU switchbox see Section 13–5.1.

Table 182. Clock signals of the CGU

Base clock domain (selection stage)	Clock source/spreading stage	Clock Name	Clock #	Description
SYS_BASE	FDC0_CLK to FDC6_CLK or SYS_BASE_CLK	APB0_CLK	0	Clock for AHB side of AHB_TO_APB0 bridge.
		APB1_CLK	1	Clock for AHB side of AHB_TO_APB1 bridge.
		APB2_CLK	2	Clock for AHB side of AHB_TO_APB2 bridge.
		APB3_CLK	3	Clock for AHB side of AHB_TO_APB3 bridge
		APB4_CLK	4	Clock for AHB side of AHB_TO_APB4 bridge. Note that AHB_TO_APB4 is a synchronous bridge. So no separate clock is needed for the APB side of the bridge.
		AHB_TO_INTC_CLK	5	Clock for INTC bridge. This bridge is needed for DTL interface of the Interrupt Controller.
		AHB0_CLK	6	Clock for AHB Multi-layer.
		EBI_CLK	7	Clock for EBI
		DMA_PCLK	8	Clock for APB interface of DMA
		DMA_CLK_GATED	9	Clock for AHB interface of DMA
		NANDFLASH_S0_CLK	10	AHB port clock of the module
		NANDFLASH_ECC_CLK	11	Main clock for ECC part in the module
		reserved	12	-
		NANDFLASH_NAND_CLK	13	Main clock for the module
		NANDFLASH_PCLK	14	APB port clock of the module
		CLOCK_OUT	15	Free to use clock, with restriction that this clock is derived from SYS_base. This is the clock for the CLK_OUT pin.
		ARM926_CORE_CLK	16	Core clock of ARM926
		ARM926_BUSIF_CLK	17	AHB clock for ARM.

Table 182. Clock signals of the CGU *…continued*

Base clock domain (selection stage)	Clock source/spreading stage	Clock Name	Clock #	Description
SYS_BASE	FDC0_CLK to FDC6_CLK or SYS_BASE_CLK	ARM926_RETIME_CLK	18	The retime clock of the ARM is used for signifying the rising edge of the AHB clock for the instruction and data of the AHB by making use of IHCLKEN and DHCLKEN. The frequency of the retime clock must be equal to its base clock.
		SD_MMC_HCLK	19	AHB interface clock of the MCI
		SD_MMC_CCLK_IN	20	The card interface input clock of MCI
		USB_OTG_AHB_CLK	21	AHB clk of USB_OTG
		ISRAM0_CLK	22	AHB clock for internal SRAM0 controller
		RED_CTL_RSCLK	23	Clock used for Redundancy Controller of Internal memories
		ISRAM1_CLK	24	AHB clock for internal SRAM1 controller
		ISROM_CLK	25	AHB clock for internal SROM controller
		MPMC_CFG_CLK	26	AHB clock for MPMC
		MPMC_CFG_CLK2	27	Clock for timing all external memory transfers. Should be synchronous to HCLK, where this MPMC_CFG_CLK2 (MPMCCLK) can be twice the frequency of HCLK
		MPMC_CFG_CLK3	28	Clock used for External Refresh Generator. This clock has to run at the SYS_base frequency
		INTC_CLK	29	Clock for Interrupt Controller Clock at the DTL interface
AHB_APB0_BASE	FDC7_CLK to FDC8_CLK or AHB_APB0_BASE	AHB_TO_APB0_PCLK	30	Asynchronous Clock for APB interface of AHB_TO_APB0 bridge
		EVENT_ROUTER_PCLK	31	APB clock for Event Router
		ADC_PCLK	32	APB clock for 10-bit ADC
		ADC_CLK	33	10-bit ADC clock
		WDOG_PCLK	34	APB clock for WDOG
		IOCONF_PCLK	35	APB clock for IOCONFIG
		CGU_PCLK	36	APB clock for CGU
		SYSCREG_PCLK	37	APB clock for SYSREG
		reserved	38	-
		RNG_PCLK	39	Clock for Random number generator

Table 182. Clock signals of the CGU *…continued*

Base clock domain (selection stage)	Clock source/spreading stage	Clock Name	Clock #	Description
AHB_APB1_BASE	FDC9_CLK to FDC10_CLK or AHB_APB1_BASE	AHB_TO_APB1_PCLK	40	Asynchronous Clock for APB interface of AHB_TO_APB1 bridge
		TIMER0_PCLK	41	APB clock for Timer0
		TIMER1_PCLK	42	APB clock for Timer1
		TIMER2_PCLK	43	APB clock for Timer2
		TIMER3_PCLK	44	APB clock for Timer3
		PWM_PCLK	45	APB clock for PWM
		PWM_PCLK_REGS	46	Gated APB clock, used for register access of PWM
		PWM_CLK	47	Clock used for generating the output of the PWM
		I^2C0_PCLK	48	APB clock for I^2C0
		I^2C1_PCLK	49	APB clock for I^2C1
AHB_APB2_BASE	FDC11_CLK to FDC13_CLK or AHB_APB2_BASE	AHB_TO_APB2_PCLK	50	Clock for APB interface of AHB_TO_APB2 bridge.
		PCM_PCLK	51	APB clock for PCM. Used to synchronize the DMA handshake signals; needs to run continuously.
		PCM_APB_PCLK	52	APB Interface clock for PCM. Used to perform register accesses.
		UART_APB_CLK	53	APB clock for UART
		LCD_PCLK	54	APB clock for LCD
		LCD_CLK	55	Clock used by data and control flow towards the external LCD Controller.
		SPI_PCLK	56	APB bus clock of SPI.
		SPI_PCLK_GATED	57	Gated version of APB bus clock of SPI
AHB_APB3_BASE	FDC14_CLK or AHB_APB3_BASE	AHB_TO_APB3_PCLK	58	Asynchronous Clock for APB interface of AHB_TO_APB3 bridge
		I2S_CFG_PCLK	59	APB clock for I2S configuration block
		EDGE_DET_PCLK	60	APB clock for EDGE_DET
		I2STX_FIFO_0_PCLK	61	APB clock for I2STX_FIFO_0 (I2STX_0)
		I2STX_IF_0_PCLK	62	APB clock for I2STX_IF_0 (I2STX_0)
		I2STX_FIFO_1_PCLK	63	APB clock for I2STX_FIFO_1 (I2STX_1)
		I2STX_IF_1_PCLK	64	APB clock for I2STX_IF_1 (I2STX_1)
		I2SRX_FIFO_0_PCLK	65	APB clock for I2SRX_FIFO_0 (I2SRX_0)
		I2SRX_IF_0_PCLK	66	APB clock for I2SRX_IF_0 (I2SRX_0)
		I2SRX_FIFO_1_PCLK	67	APB clock for I2SRX_FIFO_1 (I2SRX1)
		I2SRX_IF_1_PCLK	68	APB clock for I2SRX_IF_1 (I2SRX1)
		-	69	reserved
		-	70	reserved

Table 182. Clock signals of the CGU *…continued*

Base clock domain (selection stage)	Clock source/spreading stage	Clock Name	Clock #	Description
PCM_BASE	FDC15_CLK or PCM_BASE	PCM_CLK_IP	71	Clock for Timing of PCM
UART_BASE	FDC16_CLK or UART_BASE	UART_U_CLK	72	Used for UART baud-rate generation
CLK1024FS_BASE	FDC17_CLK to FDC22_CLK or CLK1024FS_BASE	I2S_EDGE_DETECT_CLK	73	Sampling frequency clock. Used to generate NEWSAM flag from edge_detection.
		I2STX_BCK0_N	74	I2S Bit Clock of I2STX_0 (I2STX_0)
		I2STX_WS0	75	I2S Word Select of I2STX_0 (I2STX_0)
		I2STX_CLK0	76	System clock for external reference of I2STX_IF_0 (I2STX_0)
		I2STX_BCK1_N	77	I2S Bit Clock of I2STX_1 (I2STX_1)
		I2STX_WS1	78	I2S Word Select of I2STX_1 (I2STX_1)
		CLK_256FS	79	256 fs system clock for external reference. Also used as system clock for external reference of I2STX_1.
		I2SRX_BCK0_N	80	I2S Bit Clock of I2SRX_IF_0 in master mode (I2SRX_0)
		I2SRX_WS0	81	I2S Word Select of I2SRX_IF_0 (I2SRX_0)
		I2SRX_BCK1_N	82	I2S Bit Clock of I2SRX_IF_1 in master mode (I2SRX_1)
		I2SRX_WS1	83	I2S Word Select of I2SRX_IF_1 (I2SRX_1)
		-	84 to 86	reserved
I2SRX_BCK0_BASE	I2SRX_BCK0_BASE	I2SRX_BCK0	87	I2S Bit clock of I2SRX_0. This clock is used in both master and slave modes.
I2SRX_BCK1_BASE	I2SRX_BCK1_BASE	I2SRX_BCK1	88	I2S Bit clock of I2SRX_1. This clock is used in both master and slave modes.
SPI_CLK_BASE	FDC16_CLK or SPI_CLK_BASE	SPI_CLK	89	Main clock of the SPI module.
		SPI_CLK_GATED	90	Gated version of main clock of the SPI module.
SYSCLK_O_BASE	SYSCLK_O_BASE	SYSCLK_O	91	Clock for SYSCLK_O pin.

2.1.2 Interrupt request signals of CGU

The CGU does not generate interrupts.

2.1.3 DMA transfer signals of CGU

The CGU has no DMA transfer signals.

2.1.4 Reset signals of the CGU

CGU generates system wide reset based on POR and WatchDog reset events. Apart from generating a system wide reset, the CGU also provides a register interface to generate individual reset to the peripherals, memories, and bridges present on the chip (see Table 13–184).

Remark: The AHB_TO_APB0 resets are reserved. It is not allowed to use this reset, as it cannot be disabled again afterwards.

3. Register overview

The CGU consists of two register parts: the clock switchbox registers and the configuration registers. Both register parts use different base addresses.

3.1 Register overview of clock switchbox

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
Switch configuration registers for base clocks			
SCR0	R/W	0x000	Switch Configuration Register for SYS base
SCR1	R/W	0X004	Switch Configuration Register for AHB0_APB0 base
SCR2	R/W	0X008	Switch Configuration Register for AHB0_APB1 base
SCR3	R/W	0x00C	Switch Configuration Register for AHB0_APB2 base
SCR4	R/W	0x010	Switch Configuration Register for AHB0_APB3 base
SCR5	R/W	0x014	Switch Configuration Register for PCM base
SCR6	R/W	0x018	Switch Configuration Register for UART base
SCR7	R/W	0x01C	Switch Configuration Register for CLK1024FS base
SCR8	R/W	0x020	Switch Configuration Register for I2SRX_BCK0 base
SCR9	R/W	0X024	Switch Configuration Register for I2SRX_BCK1 base
SCR10	R/W	0X028	Switch Configuration Register for SPI_CLK base
SCR11	R/W	0x02C	Switch Configuration Register for SYSCLK_O base
Frequency select registers 1 for base clocks			
FS1_0	R/W	0x030	Frequency Select Register 1 for SYS base
FS1_1	R/W	0x034	Frequency Select Register 1 for AHB0_APB0 base
FS1_2	R/W	0x038	Frequency Select Register 1 for AHB0_APB1 base
FS1_3	R/W	0x03C	Frequency Select Register 1 for AHB0_APB2 base
FS1_4	R/W	0x040	Frequency Select Register 1 for AHB0_APB3 base
FS1_5	R/W	0X044	Frequency Select Register 1 for PCM base
FS1_6	R/W	0X048	Frequency Select Register 1 for UART base
FS1_7	R/W	0x04C	Frequency Select Register 1 for CLK1024FS base
FS1_8	R/W	0x050	Frequency Select Register 1 for I2SRX_BCK0 base
FS1_9	R/W	0x054	Frequency Select Register 1 for I2SRX_BCK1 base
FS1_10	R/W	0x058	Frequency Select Register 1 for SPI_CLK base
FS1_11	R/W	0x05C	Frequency Select Register 1 for SYSCLK_O base

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
Frequency select registers 2 for base clocks			
FS2_0	R/W	0x060	Frequency Select Register 2 for SYS base
FS2_1	R/W	0X064	Frequency Select Register 2 for AHB0_APB0 base
FS2_2	R/W	0X068	Frequency Select Register 2 for AHB0_APB1 base
FS2_3	R/W	0x06C	Frequency Select Register 2 for AHB0_APB2 base
FS2_4	R/W	0x070	Frequency Select Register 2 for AHB0_APB3 base
FS2_5	R/W	0x074	Frequency Select Register 2 for PCM base
FS2_6	R/W	0x078	Frequency Select Register 2 for UART base
FS2_7	R/W	0x07C	Frequency Select Register 2 for CLK1024FS base
FS2_8	R/W	0x080	Frequency Select Register 2 for I2SRX_BCK0 base
FS2_9	R/W	0X084	Frequency Select Register 2 for I2SRX_BCK1 base
FS2_10	R/W	0X088	Frequency Select Register 2 for SPI_CLK base
FS2_11	R/W	0x08C	Frequency Select Register 2 for SYSCLK_O base
Switch status registers for base clocks			
SSR0	R	0x090	Switch Status Register for SYS base
SSR1	R	0x094	Switch Status Register for AHB0_APB0 base
SSR2	R	0x098	Switch Status Register for AHB0_APB1 base
SSR3	R	0x09C	Switch Status Register for AHB0_APB2 base
SSR4	R	0x0A0	Switch Status Register for AHB0_APB3 base
SSR5	R	0X0A4	Switch Status Register for PCM base
SSR6	R	0X0A8	Switch Status Register for UART base
SSR7	R	0x0AC	Switch Status Register for CLK1024FS base
SSR8	R	0x0B0	Switch Status Register for I2SRX_BCK0 base
SSR9	R	0x0B4	Switch Status Register for I2SRX_BCK1 base
SSR10	R	0x0B8	Switch Status Register for SPI_CLK base
SSR11	R	0x0BC	Switch Status Register for SYSCLK_O base
Power control registers, spreading stage			
PCR0	R/W	0x0C0	Power Control Register for APB0_CLK
PCR1	R/W	0X0C4	Power Control Register for APB1_CLK
PCR2	R/W	0X0C8	Power Control Register for APB2_CLK
PCR3	R/W	0x0CC	Power Control Register for APB3_CLK
PCR4	R/W	0x0D0	Power Control Register for APB4_CLK
PCR5	R/W	0x0D4	Power Control Register for AHB_TO_INTC_CLK
PCR6	R/W	0x0D8	Power Control Register for AHB0_CLK
PCR7	R/W	0x0DC	Power Control Register for EBI_CLK
PCR8	R/W	0x0E0	Power Control Register for DMA_PCLK
PCR9	R/W	0X0E4	Power Control Register for DMA_CLK_GATED
PCR10	R/W	0X0E8	Power Control Register for NANDFLASH_S0_CLK
PCR11	R/W	0x0EC	Power Control Register for NANDFLASH_ECC_CLK
PCR12	R/W	0x0F0	reserved. Write 0 to this register.

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
PCR13	R/W	0x0F4	Power Control Register for NANDFLASH_NAND_CLK
PCR14	R/W	0x0F8	Power Control Register for NANDFLASH_PCLK
PCR15	R/W	0x0FC	Power Control Register for CLOCK_OUT
PCR16	R/W	0x100	Power Control Register for ARM926_CORE_CLK
PCR17	R/W	0X104	Power Control Register for ARM926_BUSIF_CLK
PCR18	R/W	0X108	Power Control Register for ARM926_RETIME_CLK
PCR19	R/W	0x10C	Power Control Register for SD_MMC_HCLK
PCR20	R/W	0x110	Power Control Register for SD_MMC_CCLK_IN
PCR21	R/W	0x114	Power Control Register for USB_OTG_AHB_CLK
PCR22	R/W	0x118	Power Control Register for ISRAM0_CLK
PCR23	R/W	0x11C	Power Control Register for RED_CTL_RSCLK
PCR24	R/W	0x120	Power Control Register for ISRAM1_CLK. For the LPC3130, write 0 to this register. The ISRAM1 block is not available on the LPC3130.
PCR25	R/W	0X124	Power Control Register for ISROM_CLK
PCR26	R/W	0X128	Power Control Register for MPMC_CFG_CLK
PCR27	R/W	0x12C	Power Control Register for MPMC_CFG_CLK2
PCR28	R/W	0x130	Power Control Register for MPMC_CFG_CLK3
PCR29	R/W	0x134	Power Control Register for INTC_CLK
PCR30	R/W	0x138	Power Control Register for AHB_TO_APB0_PCLK
PCR31	R/W	0x13C	Power Control Register for EVENT_ROUTER_PCLK
PCR32	R/W	0x140	Power Control Register for ADC_PCLK
PCR33	R/W	0X144	Power Control Register for ADC_CLK
PCR34	R/W	0X148	Power Control Register for WDOG_PCLK
PCR35	R/W	0x14C	Power Control Register for IOCONF_PCLK
PCR36	R/W	0x150	Power Control Register for CGU_PCLK
PCR37	R/W	0x154	Power Control Register for SYSCREG_PCLK
PCR38	R/W	0x158	reserved. Write 0 to this register.
PCR39	R/W	0x15C	Power Control Register for RNG_PCLK
PCR40	R/W	0x160	Power Control Register for AHB_TO_APB1_PCLK
PCR41	R/W	0X164	Power Control Register for TIMER0_PCLK
PCR42	R/W	0X168	Power Control Register for TIMER1_PCLK
PCR43	R/W	0x16C	Power Control Register for TIMER2_PCLK
PCR44	R/W	0x170	Power Control Register for TIMER3_PCLK
PCR45	R/W	0x174	Power Control Register for PWM_PCLK
PCR46	R/W	0x178	Power Control Register for PWM_PCLK_REGS
PCR47	R/W	0x17C	Power Control Register for PWM_CLK
PCR48	R/W	0x180	Power Control Register for I2C0_PCLK
PCR49	R/W	0x184	Power Control Register for I2C1_PCLK
PCR50	R/W	0x188	Power Control Register for AHB_TO_APB2_PCLK
PCR51	R/W	0x18C	Power Control Register for PCM_PCLK

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
PCR52	R/W	0x190	Power Control Register for PCM_APB_PCLK
PCR53	R/W	0x194	Power Control Register for UART_APB_CLK
PCR54	R/W	0x198	Power Control Register for LCD_PCLK
PCR55	R/W	0x19C	Power Control Register for LCD_CLK
PCR56	R/W	0x1A0	Power Control Register for SPI_PCLK
PCR57	R/W	0x1A4	Power Control Register for SPI_PCLK_GATED
PCR58	R/W	0x1A8	Power Control Register for AHB_TO_APB3_PCLK
PCR59	R/W	0x1AC	Power Control Register for I2S_CFG_PCLK
PCR60	R/W	0x1B0	Power Control Register for EDGE_DET_PCLK
PCR61	R/W	0x1B4	Power Control Register for I2STX_FIFO_0_PCLK
PCR62	R/W	0x1B8	Power Control Register for I2STX_IF_0_PCLK
PCR63	R/W	0x1BC	Power Control Register for I2STX_FIFO_1_PCLK
PCR64	R/W	0x1C0	Power Control Register for I2STX_IF_1_PCLK
PCR65	R/W	0x1C4	Power Control Register for I2SRX_FIFO_0_PCLK
PCR66	R/W	0x1C8	Power Control Register for I2SRX_IF_0_PCLK
PCR67	R/W	0x1CC	Power Control Register for I2SRX_FIFO_1_PCLK
PCR68	R/W	0x1D0	Power Control Register for I2SRX_IF_1_PCLK
PCR69	R/W	0x1D4	reserved. Write 0 to this register.
PCR70	R/W	0x1D8	reserved. Write 0 to this register.
PCR71	R/W	0x1DC	Power Control Register for PCM_CLK_IP
PCR72	R/W	0x1E0	Power Control Register for UART_U_CLK
PCR73	R/W	0x1E4	Power Control Register for I2S_EDGE_DETECT_CLK
PCR74	R/W	0x1E8	Power Control Register for I2STX_BCK0_N
PCR75	R/W	0x1EC	Power Control Register for I2STX_WS0
PCR76	R/W	0x1F0	Power Control Register for I2STX_CLK0
PCR77	R/W	0x1F4	Power Control Register for I2STX_BCK1_N
PCR78	R/W	0x1F8	Power Control Register for I2STX_WS1
PCR79	R/W	0x1FC	Power Control Register for CLK_256FS
PCR80	R/W	0x200	Power Control Register for I2SRX_BCK0_N
PCR81	R/W	0x204	Power Control Register for I2SRX_WS0
PCR82	R/W	0x208	Power Control Register for I2SRX_BCK1_N
PCR83	R/W	0x20C	Power Control Register for I2SRX_WS1
PCR84	R/W	0x210	reserved. Write 0 to this register.
PCR85	R/W	0x214	reserved. Write 0 to this register.
PCR86	R/W	0x218	reserved.Write 0 to this register.
PCR87	R/W	0x21C	Power Control Register for I2SRX_BCK0
PCR88	R/W	0x220	Power Control Register for I2SRX_BCK1
PCR89	R/W	0x224	Power Control Register for SPI_CLK
PCR90	R/W	0x228	Power Control Register for SPI_CLK_GATED
PCR91	R/W	0x22C	Power Control Register for SYSCLK_O

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
Power status registers, spreading stage			
PSR0	R	0x230	Power Status Register for APB0_CLK
PSR1	R	0x234	Power Status Register for APB1_CLK
PSR2	R	0x238	Power Status Register for APB2_CLK
PSR3	R	0x23C	Power Status Register for APB3_CLK
PSR4	R	0x240	Power Status Register for APB4_CLK
PSR5	R	0x244	Power Status Register for AHB_TO_INTC_CLK
PSR6	R	0x248	Power Status Register for AHB0_CLK
PSR7	R	0x24C	Power Status Register for EBI_CLK
PSR8	R	0x250	Power Status Register for DMA_PCLK
PSR9	R	0x254	Power Status Register for DMA_CLK_GATED
PSR10	R	0x258	Power Status Register for NANDFLASH_S0_CLK
PSR11	R	0x25C	Power Status Register for NANDFLASH_ECC_CLK
PSR12	R	0x260	reserved
PSR13	R	0x264	Power Status Register for NANDFLASH_NAND_CLK
PSR14	R	0x268	Power Status Register for NANDFLASH_PCLK
PSR15	R	0x26C	Power Status Register for CLOCK_OUT
PSR16	R	0x270	Power Status Register for ARM926_CORE_CLK
PSR17	R	0x274	Power Status Register for ARM926_BUSIF_CLK
PSR18	R	0x278	Power Status Register for ARM926_RETIME_CLK
PSR19	R	0x27C	Power Status Register for SD_MMC_HCLK
PSR20	R	0x280	Power Status Register for SD_MMC_CCLK_IN
PSR21	R	0x284	Power Status Register for USB_OTG_AHB_CLK
PSR22	R	0x288	Power Status Register for ISRAM0_CLK
PSR23	R	0x28C	Power Status Register for RED_CTL_RSCLK
PSR24	R	0x290	Power Status Register for ISRAM1_CLK
PSR25	R	0x294	Power Status Register for ISROM_CLK
PSR26	R	0x298	Power Status Register for MPMC_CFG_CLK
PSR27	R	0x29C	Power Status Register for MPMC_CFG_CLK2
PSR28	R	0x2A0	Power Status Register for MPMC_CFG_CLK3
PSR29	R	0x2A4	Power Status Register for INTC_CLK
PSR30	R	0x2A8	Power Status Register for AHB_TO_APB0_PCLK
PSR31	R	0x2AC	Power Status Register for EVENT_ROUTER_PCLK
PSR32	R	0x2B0	Power Status Register for ADC_PCLK
PSR33	R	0x2B4	Power Status Register for ADC_CLK
PSR34	R	0x2B8	Power Status Register for WDOG_PCLK
PSR35	R	0x2BC	Power Status Register for IOCONF_PCLK
PSR36	R	0x2C0	Power Status Register for CGU_PCLK
PSR37	R	0x2C4	Power Status Register for SYSCREG_PCLK
PSR38	R	0x2C8	reserved

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
PSR39	R	0x2CC	Power Status Register for RNG_PCLK
PSR40	R	0x2D0	Power Status Register for AHB_TO_APB1_PCLK
PSR41	R	0x2D4	Power Status Register for TIMER0_PCLK
PSR42	R	0x2D8	Power Status Register for TIMER1_PCLK
PSR43	R	0x2DC	Power Status Register for TIMER2_PCLK
PSR44	R	0x2E0	Power Status Register for TIMER3_PCLK
PSR45	R	0x2E4	Power Status Register for PWM_PCLK
PSR46	R	0x2E8	Power Status Register for PWM_PCLK_REGS
PSR47	R	0x2EC	Power Status Register for PWM_CLK
PSR48	R	0x2F0	Power Status Register for I2C0_PCLK
PSR49	R	0x2F4	Power Status Register for I2C1_PCLK
PSR50	R	0x2F8	Power Status Register for AHB_TO_APB2_PCLK
PSR51	R	0x2FC	Power Status Register for PCM_PCLK
PSR52	R	0x300	Power Status Register for PCM_APB_PCLK
PSR53	R	0x304	Power Status Register for UART_APB_CLK
PSR54	R	0x308	Power Status Register for LCD_PCLK
PSR55	R	0x30C	Power Status Register for LCD_CLK
PSR56	R	0x310	Power Status Register for SPI_PCLK
PSR57	R	0x314	Power Status Register for SPI_PCLK_GATED
PSR58	R	0x318	Power Status Register for AHB_TO_APB3_PCLK
PSR59	R	0x31C	Power Status Register for I2S_CFG_PCLK
PSR60	R	0x320	Power Status Register for EDGE_DET_PCLK
PSR61	R	0x324	Power Status Register for I2STX_FIFO_0_PCLK
PSR62	R	0x328	Power Status Register for I2STX_IF_0_PCLK
PSR63	R	0x32C	Power Status Register for I2STX_FIFO_1_PCLK
PSR64	R	0x330	Power Status Register for I2STX_IF_1_PCLK
PSR65	R	0x334	Power Status Register for I2SRX_FIFO_0_PCLK
PSR66	R	0x338	Power Status Register for I2SRX_IF_0_PCLK
PSR67	R	0x33C	Power Status Register for I2SRX_FIFO_1_PCLK
PSR68	R	0x340	Power Status Register for I2SRX_IF_1_PCLK
PSR69	R	0x344	reserved
PSR70	R	0x348	reserved
PSR71	R	0x34C	Power Status Register for PCM_CLK_IP
PSR72	R	0x350	Power Status Register for UART_U_CLK
PSR73	R	0x354	Power Status Register for I2S_EDGE_DETECT_CLK
PSR74	R	0x358	Power Status Register for I2STX_BCK0_N
PSR75	R	0x35C	Power Status Register for I2STX_WS0
PSR76	R	0x360	Power Status Register for I2STX_CLK0
PSR77	R	0x364	Power Status Register for I2STX_BCK1_N
PSR78	R	0x368	Power Status Register for I2STX_WS1

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
PSR79	R	0x36C	Power Status Register for CLK_256FS
PSR80	R	0x370	Power Status Register for I2SRX_BCK0_N
PSR81	R	0x374	Power Status Register for I2SRX_WS0
PSR82	R	0x378	Power Status Register for I2SRX_BCK1_N
PSR83	R	0x37C	Power Status Register for I2SRX_WS1
PSR84	R	0x380	reserved
PSR85	R	0x384	reserved
PSR86	R	0x388	reserved
PSR87	R	0x38C	Power Status Register for I2SRX_BCK0
PSR88	R	0x390	Power Status Register for I2SRX_BCK1
PSR89	R	0x394	Power Status Register for SPI_CLK
PSR90	R	0x398	Power Status Register for SPI_CLK_GATED
PSR91	R	0x39C	Power Status Register for SYSCLK_O
Enable select registers, spreading stage			
ESR0	R/W	0x3A0	Enable Select and enable Register for APB0_CLK
ESR1	R/W	0x3A4	Enable Select and enable Register for APB1_CLK
ESR2	R/W	0x3A8	Enable Select and enable Register for APB2_CLK
ESR3	R/W	0x3AC	Enable Select and enable Register for APB3_CLK
ESR4	R/W	0x3B0	Enable Select and enable Register for APB4_CLK
ESR5	R/W	0x3B4	Enable Select and enable Register for AHB_TO_INTC_CLK
ESR6	R/W	0x3B8	Enable Select and enable Register for AHB0_CLK
ESR7	R/W	0x3BC	Enable Select and enable Register for EBI_CLK
ESR8	R/W	0x3C0	Enable Select and enable Register for DMA_PCLK
ESR9	R/W	0x3C4	Enable Select and enable Register for DMA_CLK_GATED
ESR10	R/W	0x3C8	Enable Select and enable Register for NANDFLASH_S0_CLK
ESR11	R/W	0x3CC	Enable Select and enable Register for NANDFLASH_ECC_CLK
ESR12	R/W	0x3D0	reserved
ESR13	R/W	0x3D4	Enable Select and enable Register for NANDFLASH_NAND_CLK
ESR14	R/W	0x3D8	Enable Select and enable Register for NANDFLASH_PCLK
ESR15	R/W	0x3DC	Enable Select and enable Register for NANDFLASH_CLOCK_OUT
ESR16	R/W	0x3E0	Enable Select and enable Register for ARM926_CORE_CLK
ESR17	R/W	0x3E4	Enable Select and enable Register for ARM926_BUSIF_CLK
ESR18	R/W	0x3E8	Enable Select and enable Register for ARM926_RETIME_CLK
ESR19	R/W	0x3EC	Enable Select and enable Register for SD_MMC_HCLK
ESR20	R/W	0x3F0	Enable Select and enable Register for SD_MMC_CCLK_IN
ESR21	R/W	0x3F4	Enable Select and enable Register for USB_OTG_AHB_CLK
ESR22	R/W	0x3F8	Enable Select and enable Register for ISRAM0_CLK
ESR23	R/W	0x3FC	Enable Select and enable Register for RED_CTL_RSCLK

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
ESR24	R/W	0x400	Enable Select and enable Register for ISRAM1_CLK
ESR25	R/W	0x404	Enable Select and enable Register for ISROM_CLK
ESR26	R/W	0x408	Enable Select and enable Register for MPMC_CFG_CLK
ESR27	R/W	0x40C	Enable Select and enable Register for MPMC_CFG_CLK2
ESR28	R/W	0x410	Enable Select and enable Register for MPMC_CFG_CLK3
ESR29	R/W	0x414	Enable Select and enable Register for INTC_CLK
ESR30	R/W	0x418	Enable Select and enable Register for AHB_TO_APB0_PCLK
ESR31	R/W	0x41C	Enable Select and enable Register for EVENT_ROUTER_PCLK
ESR32	R/W	0x420	Enable Select and enable Register for ADC_PCLK
ESR33	R/W	0x424	Enable Select and enable Register for ADC_CLK
ESR34	R/W	0x428	Enable Select and enable Register for WDOG_PCLK
ESR35	R/W	0x42C	Enable Select and enable Register for IOCONF_PCLK
ESR36	R/W	0x430	Enable Select and enable Register for CGU_PCLK
ESR37	R/W	0x434	Enable Select and enable Register for SYSCREG_PCLK
ESR38	R/W	0x438	reserved
ESR39	R/W	0x43C	Enable Select and enable Register for RNG_PCLK
ESR40	R/W	0x440	Enable Select and enable Register for AHB_TO_APB1_PCLK
ESR41	R/W	0x444	Enable Select and enable Register for TIMER0_PCLK
ESR42	R/W	0x448	Enable Select and enable Register for TIMER1_PCLK
ESR43	R/W	0x44C	Enable Select and enable Register for TIMER2_PCLK
ESR44	R/W	0x450	Enable Select and enable Register for TIMER3_PCLK
ESR45	R/W	0x454	Enable Select and enable Register for PWM_PCLK
ESR46	R/W	0x458	Enable Select and enable Register for PWM_PCLK_REGS
ESR47	R/W	0x45C	Enable Select and enable Register for PWM_CLK
ESR48	R/W	0x460	Enable Select and enable Register for I2C0_PCLK
ESR49	R/W	0x464	Enable Select and enable Register for I2C1_PCLK
ESR50	R/W	0x468	Enable Select and enable Register for AHB_TO_APB2_PCLK
ESR51	R/W	0x46C	Enable Select and enable Register for PCM_PCLK
ESR52	R/W	0x470	Enable Select and enable Register for PCM_APB_PCLK
ESR53	R/W	0x474	Enable Select and enable Register for UART_APB_CLK
ESR54	R/W	0x478	Enable Select and enable Register for LCD_PCLK
ESR55	R/W	0x47C	Enable Select and enable Register for LCD_CLK
ESR56	R/W	0x480	Enable Select and enable Register for SPI_PCLK
ESR57	R/W	0x484	Enable Select and enable Register for SPI_PCLK_GATED
ESR58	R/W	0x488	Enable Select and enable Register for AHB_TO_APB3_PCLK
ESR59	R/W	0x48C	Enable Select and enable Register for I2S_CFG_PCLK
ESR60	R/W	0x490	Enable Select and enable Register for EDGE_DET_PCLK
ESR61	R/W	0x494	Enable Select and enable Register for I2STX_FIFO_0_PCLK
ESR62	R/W	0x498	Enable Select and enable Register for I2STX_IF_0_PCLK

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
ESR63	R/W	0x49C	Enable Select and enable Register for I2STX_FIFO_1_PCLK
ESR64	R/W	0x4A0	Enable Select and enable Register for I2STX_IF_1_PCLK
ESR65	R/W	0x4A4	Enable Select and enable Register for I2SRX_FIFO_0_PCLK
ESR66	R/W	0x4A8	Enable Select and enable Register for I2SRX_IF_0_PCLK
ESR67	R/W	0x4AC	Enable Select and enable Register for I2SRX_FIFO_1_PCLK
ESR68	R/W	0x4B0	Enable Select and enable Register for I2SRX_IF_1_PCLK
ESR69	R/W	0x4B4	reserved
ESR70	R/W	0x4B8	reserved
ESR71	R/W	0x4BC	Enable Select and enable Register for PCM_CLK_IP
ESR72	R/W	0x4C0	Enable Select and enable Register for UART_U_CLK
ESR73	R/W	0x4C4	Enable Select and enable Register for I2S_EDGE_DETECT_CLK
ESR74	R/W	0x4C8	Enable Select and enable Register for R_I2STX_BCK0_N
ESR75	R/W	0x4CC	Enable Select and enable Register for I2STX_WS0
ESR76	R/W	0x4D0	Enable Select and enable Register for I2STX_CLK0
ESR77	R/W	0x4D4	Enable Select and enable Register for I2STX_IF_BCK1_N
ESR78	R/W	0x4D8	Enable Select and enable Register for I2STX_WS1
ESR79	R/W	0x4DC	Enable Select and enable Register for CLK_256FS
ESR80	R/W	0x4E0	Enable Select and enable Register for I2SRX_BCK0_N
ESR81	R/W	0x4E4	Enable Select and enable Register for I2SRX_WS0
ESR82	R/W	0x4E8	Enable Select and enable Register for I2SRX_BCK1_N
ESR83	R/W	0x4EC	Enable Select and enable Register for I2SRX_WS1
ESR84	R/W	0x4F0	reserved
ESR85	R/W	0x4F4	reserved
ESR86	R/W	0x4F8	reserved
ESR87	R/W	0x4FC	Enable Select and enable Register for SPI_CLK
ESR88	R/W	0x500	Enable Select and enable Register for SPI_CLK_GATED
Base control registers for SYS base			
BCR0	R/W	0x504	Base Control Register for SYS base
BCR1	R/W	0x508	Base Control Register for AHB0_APB0 base
BCR2	R/W	0x50C	Base Control Register for AHB0_APB1 base
BCR3	R/W	0x510	Base Control Register for AHB0_APB2 base
BCR7	R/W	0x514	Base Control Register for CLK1024FS base
Fractional divider configuration registers			
FDC0	R/W	0x518	Fractional Divider Configuration Register for Fractional Divider 0 (SYS base)
FDC1	R/W	0x51C	Fractional Divider Configuration Register for Fractional Divider 1 (SYS base)
FDC2	R/W	0x520	Fractional Divider Configuration Register for Fractional Divider 2 (SYS base)

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
FDC3	R/W	0x524	Fractional Divider Configuration Register for Fractional Divider 3 (SYS base)
FDC4	R/W	0x528	Fractional Divider Configuration Register for Fractional Divider 4 (SYS base)
FDC5	R/W	0x52C	Fractional Divider Configuration Register for Fractional Divider 5 (SYS base)
FDC6	R/W	0x530	Fractional Divider Configuration Register for Fractional Divider 6 (SYS base)
FDC7	R/W	0x534	Fractional Divider Configuration Register for Fractional Divider 7 (AHB0_APB0 base)
FDC8	R/W	0x538	Fractional Divider Configuration Register for Fractional Divider 8 (AHB0_APB0 base)
FDC9	R/W	0x53C	Fractional Divider Configuration Register for Fractional Divider 9 (AHB0_APB1 base)
FDC10	R/W	0x540	Fractional Divider Configuration Register for Fractional Divider 10 (AHB0_APB1 base)
FDC11	R/W	0x544	Fractional Divider Configuration Register for Fractional Divider 11 (AHB0_APB2 base)
FDC12	R/W	0x548	Fractional Divider Configuration Register for Fractional Divider 12 (AHB0_APB2 base)
FDC13	R/W	0x54C	Fractional Divider Configuration Register for Fractional Divider 13 (AHB0_APB2 base)
FDC14	R/W	0x550	Fractional Divider Configuration Register for Fractional Divider 14 (AHB0_APB3 base)
FDC15	R/W	0x554	Fractional Divider Configuration Register for Fractional Divider 15 (PCM base)
FDC16	R/W	0x558	Fractional Divider Configuration Register for Fractional Divider 16 (UART base)
FDC17	R/W	0x55C	Fractional Divider Configuration Register for Fractional Divider 17 (CLK1024FS base)
FDC18	R/W	0x560	Fractional Divider Configuration Register for Fractional Divider 18 (CLK1024FS base)
FDC19	R/W	0x564	Fractional Divider Configuration Register for Fractional Divider 19 (CLK1024FS base)
FDC20	R/W	0x568	Fractional Divider Configuration Register for Fractional Divider 20 (CLK1024FS base)
FDC21	R/W	0x56C	Fractional Divider Configuration Register for Fractional Divider 21 (CLK1024FS base)
FDC22	R/W	0x570	Fractional Divider Configuration Register for Fractional Divider 22 (CLK1024FS base)
FDC23	R/W	0x574	Fractional Divider Configuration Register for Fractional Divider 23 (SPI_CLK base)
Dynamic fractional divider configuration registers (SYS base only)			
DYN_FDC0	R/W	0x578	Dynamic Fractional Divider Configuration Register for Fractional Divider 0 (SYS base)

Table 183. Register overview: CGU clock switchbox (register base address 0x1300 4000)

Name	R/W	Address Offset	Description
DYN_FDC1	R/W	0x57C	Dynamic Fractional Divider Configuration Register for Fractional Divider 1 (SYS base)
DYN_FDC2	R/W	0x580	Dynamic Fractional Divider Configuration Register for Fractional Divider 2 (SYS base)
DYN_FDC3	R/W	0x584	Dynamic Fractional Divider Configuration Register for Fractional Divider 3 (SYS base)
DYN_FDC4	R/W	0x588	Dynamic Fractional Divider Configuration Register for Fractional Divider 4 (SYS base)
DYN_FDC5	R/W	0x58C	Dynamic Fractional Divider Configuration Register for Fractional Divider 5 (SYS base)
DYN_FDC6	R/W	0x590	Dynamic Fractional Divider Configuration Register for Fractional Divider 6 (SYS base)
Dynamic fractional divider selection registers (SYS base only)			
DYN_SEL0	R/W	0x594	Dynamic Selection Register for Fractional Divider 0 (SYS base)
DYN_SEL1	R/W	0x598	Dynamic Selection Register for Fractional Divider 1 (SYS base)
DYN_SEL2	R/W	0x59C	Dynamic Selection Register for Fractional Divider 2 (SYS base)
DYN_SEL3	R/W	0x5A0	Dynamic Selection Register for Fractional Divider 3 (SYS base)
DYN_SEL4	R/W	0x5A4	Dynamic Selection Register for Fractional Divider 4 (SYS base)
DYN_SEL5	R/W	0x5A8	Dynamic Selection Register for Fractional Divider 5 (SYS base)
DYN_SEL6	R/W	0x5AC	Dynamic Selection Register for Fractional Divider 6 (SYS base)

3.2 Register overview of the CGU configuration registers

Table 184. Register overview: CGU configuration block (register base address 0x1300 4C00)

Name	R/W	Address Offset	Description
Power and oscillator control			
POWERMODE	R/W	0x000	Power mode register; Power up reset initiated by an external signal 'POR'
WD_BARK	R	0X004	Watch dog bark register; Power up reset is initiated by an internal signal from the watchdog.
FFAST_ON	R/W	0X008	Activate fast oscillator register
FFAST_BYPASS	R/W	0x00C	Bypass comparator register fast oscillator
Reset control			
APB0_RESETN_SOFT	R/W	0x010	Reset register for AHB part of AHB_TO_APB0 bridge; reserved.[1]
AHB_TO_APB0_PNRES_SOFT	R/W	0x014	Reset register for APB part of AHB_TO_APB0 bridge; reserved.[1]
APB1_RESETN_SOFT	R/W	0x018	Reset register for AHB part of AHB_TO_APB1 bridge
AHB_TO_APB1_PNRES_SOFT	R/W	0x01C	Reset register for APB part of AHB_TO_APB1 bridge

Table 184. Register overview: CGU configuration block (register base address 0x1300 4C00)
…continued

Name	R/W	Address Offset	Description
APB2_RESETN_SOFT	R/W	0x020	Reset register for AHB part of AHB_TO_APB2 bridge
AHB_TO_APB2_PNRES_SOFT	R/W	0X024	Reset register for APB part of AHB_TO_APB2 bridge
APB3_RESETN_SOFT	R/W	0X028	Reset register for AHB part of AHB_TO_APB3 bridge
AHB_TO_APB3_PNRES_SOFT	R/W	0x02C	Reset register for APB part of AHB_TO_APB3 bridge
APB4_RESETN_SOFT	R/W	0x030	Reset register for AHB_TO_APB4 bridge; Only one reset is needed, because this bridge is synchronous.
AHB_TO_INTC_RESETN_SOFT	R/W	0x034	Reset register for AHB_TO_INTC
AHB0_RESETN_SOFT	R/W	0x038	Reset register for AHB0; reserved.[1]
EBI_RESET_N_SOFT	R/W	0x03C	Reset register for EBI
PCM_PNRES_SOFT	R/W	0x040	Reset register for APB domain of PCM; Asynchronous APB domain reset for PCM.
PCM_RESET_N_SOFT	R/W	0X044	Reset register for synchronous clk_ip domain of PCM
PCM_RESET_ASYNC_N_SOFT	R/W	0X048	Reset register for asynchronous clk_ip domain of PCM
TIMER0_PNRES_SOFT	R/W	0x04C	Reset register for Timer0
TIMER1_PNRES_SOFT	R/W	0x050	Reset register for Timer1
TIMER2_PNRES_SOFT	R/W	0x054	Reset register for Timer2
TIMER3_PNRES_SOFT	R/W	0x058	Reset register for Timer3
ADC_PRESETN_SOFT	R/W	0x05C	Reset register for controller of 10 bit ADC Interface
ADC_RESETN_ADC10BITS_SOFT	R/W	0x060	Reset register for A/D converter of ADC Interface; global reset
PWM_RESET_AN_SOFT	R/W	0X064	Reset register for PWM; asynchronous.
UART_SYS_RST_AN_SOFT	R/W	0X068	Reset register UART/IrDA; asynchronous.
I2C0_PNRES_SOFT	R/W	0x06C	Reset register for I2C0
I2C1_PNRES_SOFT	R/W	0x070	Reset register for I2C1
I2S_CFG_RST_N_SOFT	R/W	0x074	Reset register for I2S_Config; I2S_config APB domain reset
I2S_NSOF_RST_N_SOFT	R/W	0x078	Reset register for NSOF counter of I2S_CONFIG
EDGE_DET_RST_N_SOFT	R/W	0x07C	Reset register for Edge_det
I2STX_FIFO_0_RST_N_SOFT	R/W	0x080	Reset register for I2STX_FIFO_0
I2STX_IF_0_RST_N_SOFT	R/W	0X084	Reset register for I2STX_IF_0
I2STX_FIFO_1_RST_N_SOFT	R/W	0X088	Reset register for I2STX_FIFO_1
I2STX_IF_1_RST_N_SOFT	R/W	0x08C	Reset register for I2STX_IF_1

Table 184. Register overview: CGU configuration block (register base address 0x1300 4C00)
…continued

Name	R/W	Address Offset	Description
I2SRX_FIFO_0_RST_N_SOFT	R/W	0x090	Reset register for I2SRX_FIFO_0
I2SRX_IF_0_RST_N_SOFT	R/W	0x094	Reset register for I2SRX_IF_0
I2SRX_FIFO_1_RST_N_SOFT	R/W	0x098	Reset register for I2SRX_FIFO_1
I2SRX_IF_1_RST_N_SOFT	R/W	0x09C	Reset register for I2SRX_IF_1
reserved	R/W	0X0A0 to 0x0B0	-
LCD_PNRES_SOFT	R/W	0x0B4	Reset register for LCD Interface
SPI_PNRES_APB_SOFT	R/W	0x0B8	Reset register for apb_clk domain of SPI
SPI_PNRES_IP_SOFT	R/W	0x0BC	Reset register for ip_clk domain of SPI
DMA_PNRES_SOFT	R/W	0x0C0	Reset register for DMA
NANDFLASH_CTRL_ECC_RESET_N_SOFT	R/W	0X0C4	Reset register for ECC clock domain of Nandflash Controller
reserved	R/W	0X0C8	-
NANDFLASH_CTRL_NAND_RESET_N_SOFT	R/W	0x0CC	Reset register for of Nandflash Controller; Reset signal for the nand_clk domain of NAND flash controller
SD_MMC_PNRES_SOFT	R/W	0x0D4	Reset register for MCI synchronous with AHB clock; The AND function of this reset signal and sd_mmc_nres_cclk_in will result in the reset signal for MCI.
SD_MMC_NRES_CCLK_IN_SOFT	R/W	0x0D8	Reset register for MCI synchronous with IP clock; The AND function of this reset signal and sd_mmc_nres will result in the reset signal for MCI.
USB_OTG_AHB_RST_N_SOFT	R/W	0x0DC	Reset register for USB_OTG
RED_CTL_RESET_N_SOFT	R/W	0x0E0	Reset register for Redundancy Controller
AHB_MPMC_HRESTN_SOFT	R/W	0X0E4	Reset register for MPMC
AHB_MPMC_REFRESH_RESETN_SOFT	R/W	0X0E8	Reset register for refresh generator used for MPMC
INTC_RESERTN_SOFT	R/W	0x0EC	Reset register for Interrupt Controller.
PLL control (audio PLL)			
HP0_FIN_SELECT	R/W	0x0F0	Register for selecting input to high HPPLL0
HP0_MDEC	R/W	0x0F4	M-divider register of HP0 PLL
HP0_NDEC	R/W	0x0F8	N-divider register of HP0 PLL
HP0_PDEC	R/W	0x0FC	P-divider register of HP0 PLL
HP0_MODE	R/W	0x100	Mode register of HP0 PLL
HP0_STATUS	R	0X104	Status register of HP0 PLL
HP0_ACK	R	0X108	Ratio change acknowledge register of HP0 PLL
HP0_REQ	R/W	0x10C	Ratio change request register of HP0 PLL
HP0_INSELR	R/W	0x110	Bandwidth selection register of HP0 PLL
HP0_INSELI	R/W	0x114	Bandwidth selection register of HP0 PLL

Table 184. Register overview: CGU configuration block (register base address 0x1300 4C00)
…continued

Name	R/W	Address Offset	Description
HP0_INSELP	R/W	0x118	Bandwidth selection register of HP0 PLL
HP0_SELR	R/W	0x11C	Bandwidth selection register of HP0 PLL
HP0_SELI	R/W	0x120	Bandwidth selection register of HP0 PLL
HP0_SELP	R/W	0X124	Bandwidth selection register of HP0 PLL
PLL control (system PLL)			
HP1_FIN_SELECT	R/W	0X128	Register for selecting input to high HP1 PLL
HP1_MDEC	R/W	0x12C	M-divider register of HP1 PLL
HP1_NDEC	R/W	0x130	N-divider register of HP1 PLL
HP1_PDEC	R/W	0x134	P-divider register of HP1 PLL
HP1_MODE	R/W	0x138	Mode register of HP1 PLL
HP1_STATUS	R	0x13C	Status register of HP1 PLL
HP1_ACK	R	0x140	Ratio change acknowledge register of HP1 PLL
HP1_REQ	R/W	0X144	Ratio change request register of HP1 PLL
HP1_INSELR	R/W	0X148	Bandwidth selection register of HP1 PLL
HP1_INSELI	R/W	0x14C	Bandwidth selection register of HP1 PLL
HP1_INSELP	R/W	0x150	Bandwidth selection register of HP1 PLL
HP1_SELR	R/W	0x154	Bandwidth selection register of HP1 PLL
HP1_SELI	R/W	0x158	Bandwidth selection register of HP1 PLL
HP1_SELP	R/W	0x15C	Bandwidth selection register of HP1 PLL

[1] The AHB_TO_APB0 resets are reserved. It is not allowed to use this reset, as it cannot be disabled again afterwards.

4. Register description

4.1 Register description of clock switchbox

Table 185. Switch configuration register SCR<base number> (SCR0 to SCR11, addresses 0x1300 0000 to 0x1300 002C)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3	STOP	R/W	0	Forces switch in disable mode (No frequency selected)
2	RESET	R/W	0	Asynchronous reset of both switches
1	ENF2	R/W	0	Enable side #2 of switch
0	ENF1	R/W	1	Enable side #1 of switch

Table 186. Frequency select register 1 FS1_<base number> (FS1_0 to FS1_11, addresses 0x1300 0030 to 0x1300 005C)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2:0	FS1	R/W	0x0	The value of FS1 selects the input frequency for side #1 of the frequency switch. At reset, this side of the switch is enabled. The following input frequencies can be selected for FS1: 0x0: ffast 12 MHz 0x1: I2SRX_BCK0 0x2: I2SRX_WS0 0x3: I2SRX_BCK1 0x4: I2SRX_WS1 0x5: HPPLL0 (Audio/I2S PLL) 0x6:HPPLL1 (System PLL)

Table 187. Frequency Select register 2 FS2_<base number> (FS2_0 to FS2_11, addresses 0x1300 0060 to 0x1300 008C)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2:0	FS2	R/W	0x0	The value of FS2 selects the input frequency for side #2 of the frequency switch. At reset, this side of the switch is disabled. The following input frequencies can be selected: 0x0: ffast 12 MHz 0x1: I2SRX_BCK0 0x2: I2SRX_WS0 0x3: I2SRX_BCK1 0x4: I2SRX_WS1 0x5: HPPLL0 (Audio/I2S PLL) 0x6:HPPLL1 (System PLL)

Table 188. Switch Status register SSR<base number> (SSR0 to SSR11, addresses 0x1300 0090 to 0x1300 00BC)

Bit	Symbol	R/W	Reset Value	Description
31:5	Reserved	-	-	Reserved
4:2	FS	R	0x0	Feedback of currently used frequency selection
1	FS2STAT	R	0x0	If true, side #2 of the frequency switch is currently enabled
0	F1STAT	R	0x1	If true, side #1 of the frequency switch is currently enabled

Table 189. Power Control register PCR<clock number> (PCR0 to PCR91, addresses 0x1300 0030 to 0x1300 022C)

Bit	Symbol	R/W	Reset Value	Description
31:5		-	-	Reserved
4	ENOUT_EN	R/W	0	When true, the clk enabling preview signal <clk>_enableout reflects the enable state for the second active edge of <clk>. Only the enableout's of the following clocks can be used in the CGU of the LPC3130/31: ARM926_BUSIF_CLK (7) MPMC_CFG_CLK (26)
3	EXTEN_EN	R/W	0	Enable external enabling (= enable generated from outside module). It allows a clock to be controlled by an input signal that the template names '<clockname>_enable'. An example are 'pclk' of APB busses used for configuration registers, these need only to be active when the register is accessed and can be controlled by the APB psel (only for 3 clock cycle accesses) signal through the external enable input. **This bit can be set for these clocks**: PCM_APB_PCLK (52) EVENT_ROUTER_PCLK (31), ADC_PCLK 32), IOCONFIG_PCLK (35), CGU_PCLK (36), SYSCREG_PCLK (37), DMA_CLK_GATED (9), SPI_PCLK_GATED (57), SPI_CLK_GATED (90), PCM_CLK_IP (71), PWM_CLK_REGS (46) **This bit should be kept zero (not used) for**: I2C0_PCLK (48) I2C1_PCLK (49) WDOG_PCLK (34) UART_APB_CLK (53) LCD_PCLK (54)

Table 189. Power Control register PCR<clock number> (PCR0 to PCR91, addresses 0x1300 0030 to 0x1300 022C) *…continued*

Bit	Symbol	R/W	Reset Value	Description
2	WAKE_EN	R/W	1	When '0' wake up is overruled and the module clock remains active when wakeup is low. This control exists to support for power down modes. With the input signal 'wakeup' all the clocks that have WAKE_EN set can be centrally controlled. When wakeup becomes low, the clocks will be disabled and when wakeup is set high they will be enabled.
1	AUTO	R/W	1	When false Wakeup and External enable are overruled, only internal enabling via configuration and fractional divider remain active. This control is primary meant for debugging purposes, as when it is set to zero the clock is no longer affected by power saving modes.
0	RUN	R/W	1	When '0' clock is disabled. The following clocks are not powered down: ARM core clock, AHB bus clock, CGU clock, APB0 clock INTC clock, and the clocks from the IP that the code is running from(e.g. internal SRAM)

Table 190. Power Status register PSR<clock number> (PSR0 to PSR91, addresses 0x1300 0230 to 0x1300 039C)

Bit	Symbol	R/W	Reset Value	Description
31:2		-	-	Reserved
1	WAKEUP	R	1	Indicates the wakeup condition for this clock.
0	ACTIVE	R	1	Indicates clock is functional.

Table 191. Enable Select register ESR0 to ESR29 (ESR0 to ESR29, addresses 0x1300 03A0 to 0x1300 0414)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:1	ESR_SEL	R/W	0	Selection of fractional dividers 0 to 6 can be made for clocks of SYS base. 0 selects FDC0 1 selects FDC1 2 selects FDC2 3 selects FDC3 4 selects FDC4 5 selects FDC5 6 selects FDC6
0	ESR_EN	R/W	0	When the ESR_EN is true an enable is generated from the fractional divider indexed by ESR_SEL if 0 SYS_BASE_CLK is used.

Table 192. Enable Select register ESR30 to ESR39 (ESR30 to ESR39, addresses 0x1300 0418 to 0x1300 043C)

Bit	Symbol	R/W	Reset Value	Description
31:2		-	-	Reserved
1	ESR_SEL	R/W	0	Selection of fractional dividers 7 and 8 can be made for clocks of AHB0_APB0 base. 0 selects FDC7 1 selects FDC8
0	ESR_EN	R/W	0	When the ESR_EN is true an enable is generated from the fractional divider indexed by ESR_SEL if 0 AHB_APB0_BASE_CLK is used.

Table 193. Enable Select register ESR40 to ESR49 (ESR40 to ESR49, addresses 0x1300 0440 to 0x1300 0464)

Bit	Symbol	R/W	Reset Value	Description
31:2		-	-	Reserved
1	ESR_SEL	R/W	0	Selection of fractional dividers 9 and 10 can be made for clocks of AHB0_APB1 base. 0 selects FDC9 1 selects FDC10
0	ESR_EN	R/W	0	When the ESR_EN is true an enable is generated from the fractional divider indexed by ESR_SEL if 0 AHB_APB1_BASE_CLK is used.

Table 194. Enable Select register ESR50 to ESR57 (ESR50 to ESR57, addresses 0x1300 0468 to 0x1300 0484)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2:1	ESR_SEL	R/W	0	Selection of fractional dividers 11 to 13 can be made for clocks of AHB0_APB2 base. 0 selects FDC11 1 selects FDC12 2 selects FDC13
0	ESR_EN	R/W	0	When the ESR_EN is true an enable is generated from the fractional divider indexed by ESR_SEL if 0 AHB_APB2_BASE_CLK is used.

Table 195. Enable Select register ESR58 to ESR72 (ESR58 to ESR72, addresses 0x1300 0488 to 0x1300 04C0)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	ESR_EN	R/W	0	When the ESR_EN is true an enable is generated from the fractional divider. If 1, -FDC14 for ESR58 to ESR 70, -FDC15 for ESR71, -FDC16 for ESR72. If 0, -AHB_APB3_BASE_CLK is used for ESR58 to ESR70 -PCM_BASE_CLK is used for ESR71 -UART_BASE_CLK is used for ESR72

Table 196. Enable Select register ESR73 to ESR86 (ESR73 to ESR86, addresses 0x1300 04C4 to 0x1300 04F8)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:1	ESR_SEL	R/W	0	Selection of fractional dividers 17 to 22 can be made for clocks of CLK1024FS base. 0 - selects FDC17 1 - selects FDC18 2 - selects FDC19 3 - selects FDC20 4 - selects FDC21 5 - selects FDC22
0	ESR_EN	R/W	0	When the ESR_EN is true an enable is generated from the fractional divider indexed by ESR_SEL if 0 CLK1024FS_BASE_CLK is used.

Table 197. Enable Select register ESR87 to ESR88 (ESR87 to ESR88, addresses 0x1300 04FC to 0x1300 0500)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	ESR_EN	R/W	0	When the ESR_EN is 1, an enable is generated from the fractional 23. If 0 SPI_CLK_BASE_CLK is used.

Table 198. Base control register 0 (BCR0, address 0x1300 0504)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FDRUN	R/W	0	When true, fractional dividers belonging to SYS base are allowed to function. This bit overrules the run bit in the control register of the fractional dividers. So when FDRUN is set low all fractional dividers will be disabled. The purpose is to be able to activate all fractional dividers of a certain base simultaneously, so that they run in sync.

Table 199. Base Control register 1 (BCR1, address 0x1300 0508)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FDRUN	R/W	0	When true, fractional dividers belonging to AHB0_APB0 base are allowed to function. This bit overrules the run bit in the control register of the fractional dividers. So when FDRUN is set low all fractional dividers will be disabled. The purpose is to be able to activate all fractional dividers of a certain base simultaneously, so that they run in sync.

Table 200. Base Control register 2 (BCR2, address 0x1300 050C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FDRUN	R/W	0	When true, fractional dividers belonging to AHB0_APB1 base are allowed to function. This bit overrules the run bit in the control register of the fractional dividers. So when FDRUN is set low all fractional dividers will be disabled. The purpose is to be able to activate all fractional dividers of a certain base simultaneously, so that they run in sync.

Table 201. Base Control register 3 (BCR3, address 0x1300 0510)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FDRUN	R/W	0	When true, fractional dividers belonging to AHB0_APB2 base are allowed to function. This bit overrules the run bit in the control register of the fractional dividers. So when FDRUN is set low all fractional dividers will be disabled. The purpose is to be able to activate all fractional dividers of a certain base simultaneously, so that they run in sync.

Table 202. Base Control register 7 (BCR7, address 0x1300 0514)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FDRUN	R/W	0	When true, fractional dividers belonging to CLK1024FS base are allowed to function. This bit overrules the run bit in the control register of the fractional dividers. So when FDRUN is set low all fractional dividers will be disabled. The purpose is to be able to activate all fractional dividers of a certain base simultaneously, so that they run in sync.

Table 203. Fractional divider register 0 to 23 (except FDC17) (FDC0 to FDC23 (except FDC17), addresses 0x1300 0518 to 0x1300 0574)

Bit	Symbol	R/W	Reset Value	Description
31:19		-	-	Reserved
18:11	MSUB	R/W	0xff	Modulo subtraction value. The MSUB value can be calculated according: Fdiv = n/m * f MSUB = -n
10:3	MADD	R/W	0x1	Modulo addition value. The MADD value can be calculated according: Fdiv = n/m * f MADD = m-n
2	FDCTRL_STRETCH	R/W	0x0	Enables the stretching option. When stretching the generated clocks will have approximate 50% duty cycle
1	FDCTRL_RESET	R/W	0x0	Asynchronous reset of the fractional divider
0	FDCTRL_RUN	R/W	0x0	Enables the fractional divider

Table 204. Fractional Divider register 17 (FDC17, address 0x1300 055C)

Bit	Symbol	R/W	Reset Value	Description
31:29		-	-	Reserved
28:16	MSUB	R/W	0xff	Modulo subtraction value. The MSUB value can be calculated according: Fdiv = n/m * f MSUB = -n
15:3	MADD	R/W	0x1	Modulo addition value. The MADD value can be calculated according: Fdiv = n/m * f MADD = m-n
2	FDCTRL_STRETCH	R/W	0x0	Enables the stretching option. When stretching the generated clocks will have approximate 50% duty cycle
1	FDCTRL_RESET	R/W	0x0	Asynchronous reset of the fractional divider
0	FDCTRL_RUN	R/W	0x0	Enables the fractional divider

Table 205. Dynamic Fractional Divider register (DYN_FDC0 to DYN_FDC6, addresses 0x1300 0578 to 0x1300 0590)

Bit	Symbol	R/W	Reset Value	Description
31:20		-	-	Reserved
19	STOP_AUTO_RESET	R/W	0	Disable auto reset of fractional divider when changing from high-to-low or from low-to-high divider values.
18:11	MSUB	R/W	0xff	The MSUB value can be calculated according: Fdiv = n/m * f MSUB = -n
10:3	MADD	R/W	0x1	The MADD value can be calculated according: Fdiv = n/m * f MADD = m-n
2	DYN_FDCTRL_STRETCH	R/W	0x0	Enables the stretching option, during low speed operations. Advised to use the same value as in the corresponding FDC stretch bit.
1	DYN_FDC_ALLOW	R/W	0x0	Setting this bit enables the dynamic fractional divider. Then: - FDC settings are the settings for high speed operations - DYN_FDC settings are the settings for slow speed operations
0	DYN_FDCTRL_RUN	R/W	0x0	Enables the fractional divider during low speeds.

Table 206. Dynamic Fractional Divider Selection register (DYN_SEL0 to DYN_SEL6, addresses 0x1300 0594 to 0x1300 05AC)

Bit	Symbol	R/W	Reset Value	Description
31:19		-	-	Reserved
8	mpmc_refresh_req	R/W	0	External SDRAM refresh generator transfers can enable high speed. There is a special register setting in the 'external_refresh_generator' in which the duration (in SYS_BASE_CLK cycles) can be programmed of how long this 'mpmc_refresh_req' bit should be active at every refresh request. This allows every refresh cycle to trigger the high-speed operation. The purpose is to reduce SDRAM power consumption.
7	ecc_ram_busy	R/W	0	Hispeed mode during ECC activity of Nandflash Controller. Note: Has always to be enabled during variable clock scaling.
6	usb_otg_mst_trans	R/W	0	USB OTG transfers can enable high speed
5	arm926_lp_d_ready	R/W	0	ARM926 data transfers can enable high-speed
4	arm926_lp_d_trans	R/W	0	ARM926 data transfers can enable high-speed

Table 206. Dynamic Fractional Divider Selection register (DYN_SEL0 to DYN_SEL6, addresses 0x1300 0594 to 0x1300 05AC) *…continued*

Bit	Symbol	R/W	Reset Value	Description
3	arm926_lp_i_ready	R/W	0	ARM926 instruction last transfers can enable high-speed
2	arm926_lp_i_trans	R/W	0	ARM926 instruction transfers can enable high-speed
1	dma_ready	R/W	0	dma last transfers can enable high-speed
0	dma_trans	R/W	0	dma transfers can enable high-speed

4.2 Register Description of Configuration Registers

4.2.1 Power and oscillator control registers

See Section 13–5.3 for a detailed description of the power mode.

Table 207. Powermode register (POWERMODE, address 0x1300 4C00)

Bit	Symbol	R/W	Reset Value	Description
31:2		-	-	Reserved
1:0	Powermode	R/W	0x1	00: Unsupported, results in unpredictable behaviour.
				01: Normal operational mode.
				10: Unsupported, results in unpredictable behaviour.
				11: Wakeup enabled clocks are disabled until a wakeup event occurs.

Table 208. Watchdog Bark register (WD_BARK, address 0x1300 4C04)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	WD_BARK	R	0x1	Is set when a watchdog reset has occurred (read only). This bit is cleared only by a power on reset.

Table 209. Fast Oscillator activate register (FFAST_ON, 0x1300 4C08)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FFAST_ON	R/W	0x1	Activate fast oscillator

Table 210. Fast Oscillator Bypass comparator register (FFAST_BYPASS, 0x1300 4C0C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	FFAST_BYPASS	R/W	0x1	Oscillator test mode

4.2.2 Reset control registers

See Section 13–5.3 for a detailed description of the reset configuration.

Table 211. APB0_RESETN_SOFT register (address 0x1300 4C10)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	APB0_RESETN_SOFT	R/W	0x1	Reserved (It is not allowed to use this reset, as it cannot be disabled again afterwards).

Table 212. AHB_TO_APB0_PNRES_SOFT register (address 0x1300 4C14)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	AHB_TO_APB0_PNRES_SOFT	R/W	0x1	Reserved (It is not allowed to use this reset, as it cannot be disabled again afterwards)

Table 213. APB1_RESETN_SOFT register (address 0x1300 4C18)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	APB1_RESETN_SOFT	R/W	0x1	Reset for AHB part of AHB_TO_APB1 bridge

Table 214. AHB_TO_APB1_PNRES_SOFT register (address 0x1300 4C1C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	AHB_TO_APB1_PNRES_SOFT	R/W	0x1	Reset for APB part of AHB_TO_APB1 bridge

Table 215. APB2_RESETN_SOFT register (address 0x1300 4C20)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	APB2_RESETN_SOFT	R/W	0x1	Reset for AHB part of AHB_TO_APB2 bridge

Table 216. AHB_TO_APB2_PNRES_SOFT register (address 0x1300 4C24)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	AHB_TO_APB2_PNRES_SOFT	R/W	0x1	Reset for APB part of AHB_TO_APB2 bridge

Table 217. APB3_RESETN_SOFT register (address 0x1300 4C28)

Bit	Symbol	Access	Reset Value	Description
31:1		-	-	Reserved
0	APB3_RESETN_SOFT	R/W	0x1	Reset for AHB part of AHB_TO_APB3 bridge

Table 218. AHB_TO_APB3_PNRES_SOFT register (address 0x1300 4C2C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	AHB_TO_APB3_PNRES_SOFT	R/W	0x1	Reset for APB part of AHB_TO_APB3 bridge

Table 219. APB4_RESETN_SOFT register (address 0x1300 4C30)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	APB4_RESETN_SOFT	R/W	0x1	Reset for AHB part of AHB_TO_APB4 bridge

Table 220. AHB_TO_INTC_RESETN_SOFT register (address 0x1300 4C34)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	AHB_TO_INTC_resetn_soft	R/W	0x1	Reset for AHB_TO_INTC

Table 221. AHB0_RESETN_SOFT register (address 0x1300 4C38)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	ahb0_resetn_soft	R/W	0x1	Reserved (It is not allowed to use this reset, as it cannot be disabled again afterwards)

Table 222. EBI_RESETN_SOFT register (address 0x1300 4C3C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	ebi_resetn_soft	R/W	0x1	Reset for EBI

Table 223. PCM_PNRES_SOFT UNIT register (address 0x1300 4C40)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	PCM_pnres_soft unit	R/W	0x1	Reset for APB domain of PCM

Table 224. PCM_RESET_N_SOFT register (address 0x1300 4C44)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	PCM_reset_n_soft	R/W	0x1	Reset for synchronous clk_ip domain of PCM

Table 225. PCM_RESET_ASYNC_N_SOFT register (address 0x1300 4C48)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	PCM_reset_async_n_soft	R/W	0x1	Reset for asynchronous clk_ip domain of PCM

Table 226. TIMER0_PNRES_SOFT register (address 0x1300 4C4C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	timer0_pnres_soft	R/W	0x1	Reset for Timer0

Table 227. TIMER1_PNRES_SOFT register (address 0x1300 4C50)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	timer1_pnres_soft	R/W	0x1	Reset for Timer1

Table 228. TIMER2_PNRES_SOFT register (address 0x1300 4C54)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	timer2_pnres_soft	R/W	0x1	Reset for Timer2

Table 229. TIMER3_PNRES_SOFT register (address 0x1300 4C58)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	timer3_pnres_soft	R/W	0x1	Reset for Timer3

Table 230. ADC_PRESETN_SOFT register (address 0x1300 4C5C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	adc_presetn_soft	R/W	0x1	Reset for controller of 10 bit ADC Interface

Table 231. ADC_RESETN_ADC10BITS_SOFT register (address 0x1300 4C60)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	adc_resetn_adc10bits_soft	R/W	0x1	Reset for A/D converter of ADC Interface

Table 232. PWM_RESET_AN_SOFT register (address 0x1300 4C64)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	pwm_reset_an_soft	R/W	0x1	Reset for PWM

Table 233. UART_SYS_RST_AN_SOFT register (address 0x1300 4C68)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	uart_sys_rst_an_soft	R/W	0x1	Reset for UART/IrDA

Table 234. I2C0_PNRES_SOFT register (address 0x1300 4C6C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	i2c0_pnres_soft	R/W	0x1	Reset for I2C0

Table 235. I2C1_PNRES_SOFT register (address 0x1300 4C70)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	i2c1_pnres_soft	R/W	0x1	Reset for I2C1

Table 236. I2S_CFG_RST_N_SOFT register (address 0x1300 4C74)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2S_cfg_rst_n_soft	R/W	0x1	Reset for I2S_Config

Table 237. I2S_NSOF_RST_N_SOFT register (address 0x1300 4C78)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2S_nsof_rst_n_soft	R/W	0x1	Reset for NSOF counter of I2S_CONFIG

Table 238. EDGE_DET_RST_N_SOFT register (address 0x1300 4C7C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2S_nsof_rst_n_soft	R/W	0x1	Reset for Edge_det

Table 239. I2STX_FIFO_0_RST_N_SOFT register (address 0x1300 4C80)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2STX_FIFO_0_rst_n_soft	R/W	0x1	Reset for I2STX_FIFO_0

Table 240. I2STX_IF_0_RST_N_SOFT register (address 0x1300 4C84)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2STX_IF_0_rst_n_soft	R/W	0x1	Reset for I2STX_IF_0

Table 241. I2STX_FIFO_1_RST_N_SOFT register (address 0x1300 4C88)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2STX_FIFO_1_rst_n_soft	R/W	0x1	Reset for I2STX_FIFO_1

Table 242. I2STX_IF_1_RST_N_SOFT register (address 0x1300 4C8C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2STX_IF_1_rst_n_soft	R/W	0x1	Reset for I2STX_IF_1

Table 243. I2SRX_FIFO_0_RST_N_SOFT register (address 0x1300 4C90)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2SRX_FIFO_0_rst_n_soft	R/W	0x1	Reset for I2SRX_FIFO_0

Table 244. I2SRX_IF_0_RST_N_SOFT register (address 0x1300 4C94)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2SRX_IF_0_rst_n_soft	R/W	0x1	Reset for I2SRX_IF_0

Table 245. I2SRX_FIFO_1_RST_N_SOFT register (address 0x1300 4C98)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2SRX_FIFO_1_rst_n_soft	R/W	0x1	Reset for I2SRX_FIFO_1

Table 246. I2SRX_IF_1_RST_N_SOFT register (address 0x1300 4C9C)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	I2SRX_IF_1_rst_n_soft	R/W	0x1	Reset for I2SRX_IF_1

Table 247. LCD_PNRES_SOFT register (address 0x1300 4CB4)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	lcd_pnres_soft	R/W	0x1	Reset for LCD Interface

Table 248. SPI_PNRES_APB_SOFT register (address 0x1300 4CB8)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	spi_pnres_apb_soft	R/W	0x1	Reset register for apb_clk domain of SPI

Table 249. SPI_PNRES_IP_SOFT register (address 0x1300 4CBC)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	spi_pnres_ip_soft	R/W	0x1	Reset for ip_clk domain of SPI

Table 250. DMA_PNRES_SOFT register (address 0x1300 4CC0)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	dma_pnres_soft	R/W	0x1	Reset for DMA

Table 251. NANDFLASH_CTRL_ECC_RESET_N_SOFT register (address 0x1300 4CC4)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	nandflash_ctrl_ecc_reset_n_soft	R/W	0x1	Reset for ECC clock domain of Nandflash Controller

Table 252. NANDFLASH_CTRL_NAND_RESET_N_SOFT register (address 0x1300 4CCC)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	nandflash_ctrl_NAND_reset_n_soft	R/W	0x1	Reset for Nandflash Controller

Table 253. SD_MMC_PNRES_SOFT register (address 0x1300 4CD4)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	sd_mmc_pnres_soft	R/W	0x1	Reset for MCI synchronous with AHB clock

Table 254. SD_MMC_NRES_CCLK_IN_SOFT register (address 0x1300 4CD8)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	sd_mmc_nres_cclk_in_soft	R/W	0x1	Reset register for MCI synchronous with IP clock

Table 255. USB_OTG_AHB_RST_N_SOFT (address 0x1300 4CDC)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	usb_otg_ahb_rst_n_soft	R/W	0x1	Reset for USB_OTG

Table 256. RED_CTL_RESET_N_SOFT (address 0x1300 4CE0)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	red_ctl_reset_n_soft	R/W	0x1	Reset for Redundancy Controller

Table 257. AHB_MPMC_HRESETN_SOFT (address 0x1300 4CE4)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	ahb_mpmc_hresetn_soft	R/W	0x1	Reset for MPMC

Table 258. AHB_MPMC_REFRESH_RESETN_SOFT (address 0x1300 4CE8)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	ahb_mpmc_refresh_resetn_soft	R/W	0x1	Reset for refresh generator used for MPMC

Table 259. INTC_RESETN_SOFT (address 0x1300 4CEC)

Bit	Symbol	R/W	Reset Value	Description
31:1		-	-	Reserved
0	intc_resetn_soft	R/W	0x1	Reset for Interrupt Controller.

4.2.3 PLL control registers

See Section 13–5.5 for a detailed description of the PLLs.

Table 260. HP0 Frequency Input Select register (HP0_FIN_SELECT, address 0x1300 4CF0)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:0	hp0_fin_select	R/W	0x0	Select input to high HPPLL0: 0x0: ffast (12 Mhz) 0x1: I2SRX_BCK0 0x2: I2SRX_WS0 0x3: I2SRX_BCK1 0x4: I2SRX_WS1 0x5: Reserved 0x6: HP1_FOUT 0x7 to 0x15: Reserved

Table 261. HP0 M-divider register (HP0_MDEC, address 0x1300 4CF4)

Bit	Symbol	R/W	Reset Value	Description
31:17		-	-	Reserved
16:0	hp0_mdec	R/W	0x0	Decoded divider ratio code for feedback divider (M-divider)

Table 262. HP0 N-divider register (HP0_NDEC, address 0x1300 4CF8)

Bit	Symbol	R/W	Reset Value	Description
31:10		-	-	Reserved
9:0	hp0_ndec	R/W	0x0	Decoded divider ratio code for pre-divider (N-divider)

Table 263. HP0 P-divider register (HP0_PDEC, address 0x1300 4CFC)

Bit	Symbol	R/W	Reset Value	Description
31:7		-	-	Reserved
6:0	hp0_pdec	R/W	0x0	Decoded divider ratio code for post-divider (P-divider)

Table 264. HP0 Mode register (HP0_MODE, address 0x1300 4D00)

Bit	Symbol	R/W	Reset Value	Description
31:9		-	-	Reserved
8	hp0_mode_bypass	R/W	0	Bypass mode
7	hp0_mode_limup_off	R/W	0	Up limiter: 0x0 : In spread spectrum and fractional PLL applications. 0x1: In other applications.

Table 264. HP0 Mode register (HP0_MODE, address 0x1300 4D00) *…continued*

Bit	Symbol	R/W	Reset Value	Description
6	hp0_mode_bandsel	R/W	0	Bandwidth adjustment pin (to modify externally the bandwidth of the PLL) (Warning: In normal application this pin must be made low('0'). When this pin is high('1') the bandwidth depends on the value of the pins inselr[3:0], inseli[3:0] and inselp[4:0].
5	hp0_mode_frm	R/W	0	Free Running Mode
4	hp0_mode_directi	R/W	0	Normal operation with DIRECTO
3	hp0_mode_directo	R/W	0	Normal operation with DIRECTI
2	hp0_mode_pd	R/W	1	Power down mode
1	hp0_mode_skew_en	R/W	1	Skew mode
0	hp0_mode_clken	R/W	0	Enable mode

Table 265. HP0 Status register (HP0_STATUS, address 0x1300 4D04)

Bit	Symbol	R/W	Reset Value	Description
31:2		-	-	Reserved
1	hp0_status_fr	R	0	Free running detector
0	hp0_status_lock	R	0	Lock detector

Table 266. HP0 Acknowledge register (HP0_ACK, address 0x1300 4D08)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2	hp0_ack_p	R	0	Post-divider ratio change acknowledge
1	hp0_ack_n	R	0	Pre-divider ratio change acknowledge
0	hp0_ack_m	R	0	Feedback divider ratio change acknowledge

Table 267. HP0 request register (HP0_REQ, address 0x1300 4D0C)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2	hp0_req_p	R/W	0	Post-divider ratio change request
1	hp0_req_n	R/W	0	Pre-divider ratio change request
0	hp0_req_m	R/W	0	Feedback divider ratio change request

Table 268. HP0 Bandwith Selection register (HP0_INSELR, address 0x1300 4D10)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:0	hp0_inselr	R/W	0x0	pins to select the bandwidth (does not matter when bandsel = '0')

Table 269. HP0 Bandwith Selection register (HP0_INSELI, address 0x1300 4D14)

Bit	Symbol	R/W	Reset Value	Description
31:6		-	-	Reserved
5:0	hp0_inseli	R/W	0x0	Bandwidth selection register of HP0 PLL (does not matter when bandsel = ’0’)

Table 270. HP0 Bandwith Selection register (HP0_INSELP, address 0x1300 4D18)

Bit	Symbol	R/W	Reset Value	Description
31:5		-	-	Reserved
4:0	hp0_inselp	R/W	0x0	Bandwidth selection register of HP0 PLL (does not matter when bandsel = ’0’)

Table 271. HP0 Bandwith Selection register (HP0_SELR, address 0x1300 4D1C)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:0	hp0_selr	R/W	0x0	Bandwidth selection register of HP0 PLL

Table 272. HP0 Bandwith Selection register (HP0_SELI, address 0x1300 4D20)

Bit	Symbol	R/W	Reset Value	Description
31:6		-	-	Reserved
5:0	hp0_seli	R/W	0x0	Bandwidth selection register of HP0 PLL

Table 273. HP0 Bandwith Selection register (HP0_SELP, address 0x1300 4D24)

Bit	Symbol	R/W	Reset Value	Description
31:5		-	-	Reserved
4:0	hp0_selp	R/W	0x0	Bandwidth selection register of HP0 PLL

Table 274. HP1 Frequency Input Select register (HP1_FIN_SELECT, address 0x1300 4D28)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:0	hp1_fin_select	R/W	0x0	Select input to high HPPLL1: 0x0: ffast (12 Mhz) 0x1: I2SRX_BCK0 0x2: I2SRX_WS0 0x3: I2SRX_BCK1 0x4: I2SRX_WS1 0x5: HP0_FOUT 0x6 to 0x15: Reserved

Table 275. HP1 M-divider register (HP1_MDEC, address 0x1300 4D2C)

Bit	Symbol	R/W	Reset Value	Description
31:17		-	-	Reserved
16:0	hp1_mdec	R/W	0x0	Decoded divider ratio code for feedback divider (M-divider)

Table 276. HP1 N-divider register (HP1_NDEC, address 0x1300 4D30)

Bit	Symbol	R/W	Reset Value	Description
31:10		-	-	Reserved
9:0	hp1_ndec	R/W	0x0	Decoded divider ratio code for pre-divider (N-divider)

Table 277. HP1 P-diver register (HP1_PDEC, address 0x1300 4D34)

Bit	Symbol	R/W	Reset Value	Description
31:7		-	-	Reserved
6:0	hp1_pdec	R/W	0x0	Decoded divider ratio code for pre-divider (P-divider)

Table 278. HP1 Mode register (HP1_MODE, address 0x1300 4D38)

Bit	Symbol	R/W	Reset Value	Description
31:9		-	-	Reserved
8	hp1_mode_bypass	R/W	0	Bypass mode
7	hp1_mode_limup_off	R/W	0	Up limiter: 0x0 : In spread spectrum and fractional PLL applications. 0x1: In other applications.
6	hp1_mode_bandsel	R/W	0	Bandwidth adjustment pin (to modify externally the bandwidth of the PLL) (Warning: In normal application this pin must be made low('0'). When this pin is high('1') the bandwidth depends on the value of the pins inselr[3:0], inseli[3:0] and inselp[4:0].
5	hp1_mode_frm	R/W	0	Free Running Mode
4	hp1_mode_directi	R/W	0	Normal operation with DIRECTO
3	hp1_mode_directo	R/W	0	Normal operation with DIRECTI
2	hp1_mode_pd	R/W	1	Power down mode
1	hp1_mode_skew_en	R/W	1	Skew mode
0	hp1_mode_clken	R/W	0	Enable mode

Table 279. HP1 Status register (HP1_STATUS, address 0x1300 4D3C)

Bit	Symbol	R/W	Reset Value	Description
31:2		-	-	Reserved
1	hp1_status_fr	R	0	Free running detector
0	hp1_status_lock	R	0	Lock detector

Table 280. HP1 Acknowledge register (HP1_ACK, address 0x1300 4D40)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2	hp10_ack_p	R	0	Post-divider ratio change acknowledge
1	hp1_ack_n	R	0	Pre-divider ratio change acknowledge
0	hp1_ack_m	R	0	Feedback divider ratio change acknowledge

Table 281. HP1 Request register (HP1_REQ, address 0x1300 4D44)

Bit	Symbol	R/W	Reset Value	Description
31:3		-	-	Reserved
2	hp1_req_p	R/W	0	Post-divider ratio change request
1	hp1_req_n	R/W	0	Pre-divider ratio change request
0	hp1_req_m	R/W	0	Feedback divider ratio change request

Table 282. HP1 bandwith Selection register (HP1_INSELR, address 0x1300 4D48)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:0	hp1_inselr	R/W	0x0	pins to select the bandwidth (does not matter when bandsel = '0')

Table 283. HP1 bandwith Selection register (HP1_INSELI, address 0x1300 4D4C)

Bit	Symbol	R/W	Reset Value	Description
31:6		-	-	Reserved
5:0	hp1_inseli	R/W	0x0	Bandwidth selection register of HP1 PLL (does not matter when bandsel = '0')

Table 284. HP1 bandwith Selection register (HP1_INSELP, address 0x1300 4D50)

Bit	Symbol	R/W	Reset Value	Description
31:5		-	-	Reserved
4:0	hp1_inselp	R/W	0x0	Bandwidth selection register of HP1 PLL (does not matter when bandsel = '0')

Table 285. HP1 bandwith Selection register (HP1_SELR, address 0x1300 4D54)

Bit	Symbol	R/W	Reset Value	Description
31:4		-	-	Reserved
3:0	hp1_selr	R/W	0x0	Bandwidth selection register of HP1 PLL

Table 286. HP1 bandwith Selection register (HP1_SELI, address 0x1300 4D58)

Bit	Symbol	R/W	Reset Value	Description
31:6		-	-	Reserved
5:0	hp1_seli	R/W	0x0	Bandwidth selection register of HP1 PLL

Table 287. HP1 bandwith Selection register (HP1_SELP, address 0x1300 4D5C)

Bit	Symbol	R/W	Reset Value	Description
31:5		-	-	Reserved
4:0	hp1_selp	R/W	0x0	Bandwidth selection register of HP1 PLL

5. Functional description

The Clock Generation Unit contains:

- Clock switch block
- Configuration register block
- Reset and power block
- 12 MHz oscillator
- 2 PLLs to generate audio sample frequencies and to generate system clock frequencies.

5.1 Clock switch box

5.1.1 Overview Switchbox Module

The switchbox consists of the following stages:

- A selection stage that allows a selection between a number of references into a number of base frequencies
- A spreading stage that for each base frequency provides individual enabling towards a set of module clocks.

These two stages are controlled via APB configuration registers. Table 13–182 shows the switchbox configuration.

5.1.2 Selection stage

Selection Multiplexer switches allow each reference frequency to be passed to each base frequency.

Frequency switches allow safe run-time changes of base frequency selection.

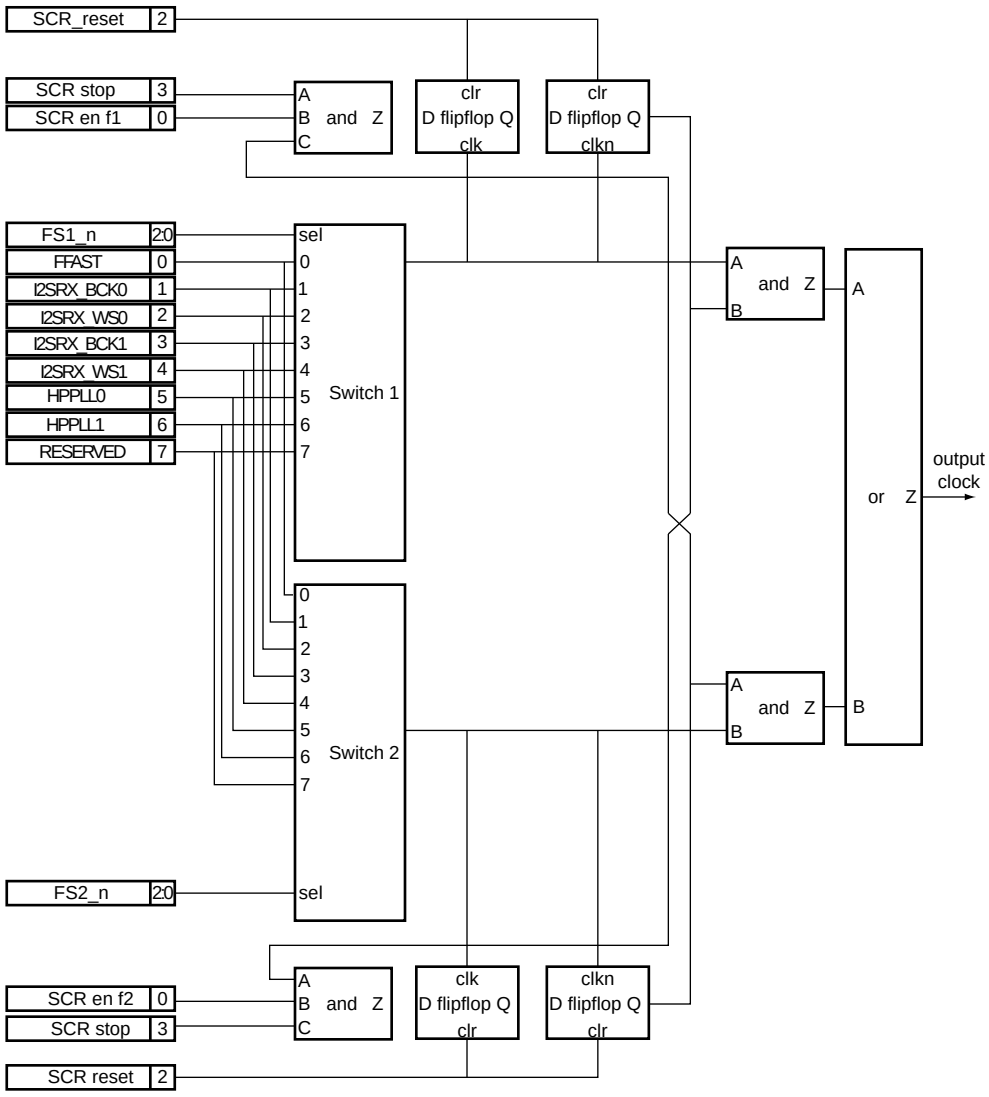


Fig 36. CGU Switchbox selection stage block diagram

A two path switch is used because there are no guaranteed clocks, making the use of state machines difficult. The frequency selector uses two multiplexers on all of the reference frequency inputs, resulting in the frequency F1 and F2. Two multiplexers are used to avoid glitches. Glitches can occur either, when multiplexing, or because the two frequencies are asynchronous (which they usually are) and the switching results in pulse clipping. The switch, as implemented above, solves these problems. This is done by selecting the new frequency on the multiplexer towards the switch side that is not activated, then by both removing the enable on one side, and activating the enable on the other side, switching take place. The construction of the switch is such that first base_clk (the base frequency) is stopped when its level is low and after (at least) a complete period, at the falling edge of the new frequency, base_clk will start to run again with this new frequency. The delay of at least a complete period in both the disabling and enabling ensures that no pulse width violations can occur.

Because it takes a certain amount of time to switch between F1 and F2, care must be taken when switching to or from very low-speed clocks. There may be a significant delay, in ARM clock cycles, between the clock switch programming, and the actual clock activation.

The SSR register can be used by software to wait for a clock switch to complete. When an active F1 or F2 is stopped externally (by stopping a PLL or as result of external activity) then the switch will enter a deadlock. (e.g.: When f2enabled is high, and F2 is stopped, then you cannot switch to F1.) This deadlock can be detected (after a software time out) by looking at the SSR register, both F1STAT and F2STAT will be '0'. To recover from this, the frequency switch must be reset in the software using the RESET bit of the SCR register.

All frequency select registers FS1 and FS2 in the clock switchbox are reset to 0, making FFAST as default clock source.

5.1.3 Spreading stage

- Each base frequency can be used to drive a set of module clocks.
- Each clock has its own enabling that can be controlled by a selectable fractional divider, via an (optional) external enable input, by its configuration register or with the wake_up signal.
- Fractional dividers on a base can be synchronized using the BCR registers.
- In test mode base clocks are overruled by a test clock.
- Positive and inverted clocks sharing the same enabling controls.

5.1.4 Fractional dividers

Fractional dividers give the possibility to generate derivatives of a base frequency. A derivative is generated by enabling/masking clock pulses and optionally stretching these pulses to obtain 50% duty cycle clock approximations can be done by software control.

The fraction n/m must always be smaller than one and greater than zero:

- When using clock stretching, the fraction must be smaller or equal to 1/2.
- To obtain the best possible 50% duty cycle when clock stretching is used, n/m should equal a division by a 2 power value (i.e.: 1/2, 1/4, 1/8, ..). Using other fractions will result in a best approximation.
- To minimize power consumption madd and msub should be chosen to have as many trailing zero's as possible (i.e. Shift values left until the bitwidth boundary reached).
- In case of multiple fractional dividers exist on a base and they need to run in sync use the base control register (BCR0 - BCR3, BCR7) fd_run bit to disable the fractional dividers. Then program the dividers and set the base control register fd_run bit to high. This ensures that all fractional dividers on this base will start running at the same instant.

Example calculation of modula add (madd) and modula subtract (msub) values:

Say an input frequency of 13 MHz is given while a frequency of 12 MHz is required. In this case we want a frequency
f' = 12/13 * f

So n = 12 and m = 13. This then gives

madd = m - n = 13 - 12 = 1

msub = -n = -12

Note that clock stretching is not allowed since n/m > 1/2.

In order to minimize power consumption madd and msub must be as large as possible. The limit of their values is determined by the madd/msub bit width. In this case msub is the largest value, in order to express -12, five bits are required. However since msub is always negative the fractional divider does not need the sign bit, leaving 4bits. If madd/msub bit width has been set to say 8 bits, it is allowed to shift 4 bits, giving:

msub' = -(12<<4)= -12 * 2^4 = -12 * 16 = -192

madd' = 1<<4 = 2^4 = 16

5.1.5 Dynamic fractional dividers

The dynamic fractional dividers allow hardware (mostly AHB busmasters) to directly control the speed of the AHB bus. This will give several advantages:

- Hardware will decide the most optimum AHB frequency. Software engineers do not have to decide the optimum AHB frequency for their application. The Hardware will do this as efficiently as possible.
- The fast clock will only be needed when data needs to be transferred. This means that when data does not need to be transferred, the AHB bus and its connected IP will consume very little power during IDLE modes.

The operation is as follows:

- All the fractional dividers of SYS-base have 'partner' registers.
- The register 'DYN_FDC' for the slow-speed clock-setting and the 'DYN_SEL' register for the selection of AHB busmasters that can trigger the dynamic operation.

When these partner registers are not programmed, the fractional dividers run in normal mode. The values programmed in the normal 'FDC' register determine the speed of the fractional divider, and this speed remains the same until re-programmed.

However, when bit ' DYN_FDC_ALLOW' of a 'DYN_FDC' register is set, then the AHB busmasters can control the speed of the fractional divider to either a 'slow' or a 'high' speed setting.

When this 'DYN_FDC_ALLOW' bit is set, the 'FDC' register is then the register for the 'high' speed setting, register 'DYN_FDC' is the register for the 'slow' speed setting.

In register 'DYN_SEL' each 'set' bit enables an AHB busmaster to set the fractional divider to the 'high' speed setting. As shown in Figure 13–37, the logical AND operation between this enable signal (DYN_SEL bit) and the signal which comes from the AHB master (DYN Connection) will select the FDC MADD bits. The result of this is that the fractional divider will be configured for a 'high' speed setting. In this case has DYN_FDC_ALLOW to be activated.

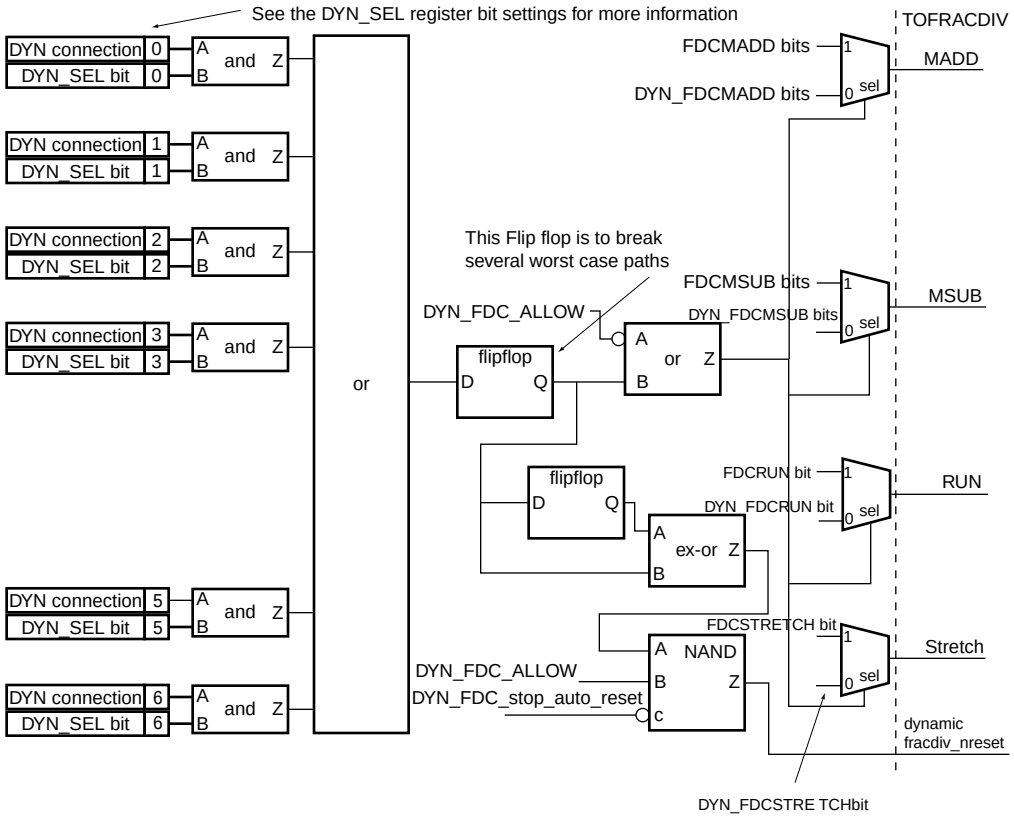


Fig 37. Dynamic fractional dividers block diagram

When any of the selected inputs change from high-to-low or from low-to-high a reset may be generated automatically to the fractional divider. This forces the fractional divider to re-program itself to the new value, to improve the speed of the transition from low to high speed operation.

This also synchronizes dynamic fractional dividers 0 and 1 with each other when they are both programmed with the same selection bits. This is required when the CPU and the AHB bus are both programmed behind a different fractional divider.

When required, this dynamic reset behavior can be disabled by setting bit 'STOP_AUTO_RESET' of the DYN_FDC register.

5.1.6 External enabling

External enabling provides the ability to use signals from outside the switchbox to enable the module clocks. These signals are latched with the base reference to ensure correct timing.

This functionality is typically used to reduce power consumption by disabling a clock whenever it is not required. It allows clock control from a module level as opposed to the power control register that represents clock control from the system level.

5.1.7 Wake_up feature

To support power modes a clock can be made controllable with the wake_up signal via its PCR wakeup_en bit. When active this clock will then, together with all other clocks that have the wakeup_en bit set, be disabled when wake_up becomes low and enabled when wake_up becomes high.

Through the wake_up feature, a group of clocks can be made controllable with a single signal, wake_up. The intention of this signal is, when used in combination with a wake_up event detection mechanism and a power mode selection, it will then disable clocks that are wake_up enabled until a wake_up event has occurred.

Clock enabling/disabling takes two to three base periods to take effect.

The wake_up feature will be disabled for a clock when its auto bit in its pcr register is set to '0'.

5.2 Configuration register block

5.2.1 Watchdog identification register

To find out whether a reset was caused by an external 'RSTIN_N', or a watchdog reset, a special 'wd_bark' register is used. When a watchdog reset has occurred, the 'wd_bark' register will be 1 after reset.

5.2.2 Controlling the frequency sources

There are two types of analog devices in the CGU. These are used as frequency sources:

- 12 MHz Oscillator
- Two PLLs: HPPLL (Phase Locked Loop).

5.2.3 Programming PLLs

To program a Phase Locked Loop (PLL) device, do the following sequence:

1. Disable the device by activating the power down mode or place it in reset mode when the module is digital
2. Set the correct operating mode and multiplication/division factor
3. Enable the device by placing it in functional mode
4. Wait until the generated frequency is stable.

When re-programming a frequency source, make sure that no clock is being generated from it.

5.3 Reset and power block

Power up reset is initiated by the external signal RSTIN_N, or by the internal signal 'wdr', from the watchdog. The RSTIN_N signal is intended as a ‘battery insertion’ type of reset, active low. Both the 'RSTIN_N' and the 'wdr' initialize the clockgen module into its reset state. A 'vddalways_resetn' reset signal is created that is stretched until the first rising 'pclk' edge after the power up signal has become inactive. 'vddalways_resetn' activates the other reset signal(s), defined as reset domains.

The CGU is able to generate as many resets, synchronized to specific clock domains, as required. All resets domains specified have a re-synchronization clock input. This is used to keep the reset active until the second rising edge of the resync clock, after 'vddalways_resetn' has become inactive. If the reset is asynchronous, it is kept active for another half clock cycle to prevent hold violations. The reset can be selected either as active high or active low polarity. It can also be selected for synchronous or asynchronous use. The implementation difference is with the handling of the signal during scan test mode and its deactivation timing. During production test (scantestmode=1), all asynchronous reset outputs are directly, asynchronously connected to RSTIN_N. All synchronous signals remain controlled by the synchronization flip flops.Optionally a software reset, active low, can be generated. This will add a configuration register, the signal is given the 'AND' command with the normal reset and serves as data input to the synchronization flip flops.

5.3.1 Clock disabling

By means of the powermode register it is possible to disable module clocks using the wake_up mechanism. Clocks that are disabled in this way will be reactivated when a wake_up event occurs.

Clock disabling is done by first, enabling the wake_up mechanism for a set of clocks by setting the 'wake_en' bit in the clk's power control register. Then by writing 2'b11 in the powermode register, clock disabling is initiated, resulting in following sequence of actions:

1. Masters for which disabling controls are provided will be denied bus access.
2. Wakeup enabled clocks will disappear after their second active edge.
3. When wake_up become high (a wake_up event occurred) the power status register is asynchronous reset into normal mode.
4. Disabled clocks become active again after two clock periods.

5.3.2 AHB Master disabling feature

The CGU template offers support for individual AHB master disabling. AHB Masters must not be performing bus access when put into clock disable mode. To ensure the master is removed from the bus a disable request is generated. This signal is used by the bus (ahb_multilayer) to deny any new access requests. After the master has finished its current request a disable grant is generated by the bus. The internal wakeup_i signal going to the clock switchbox is allowed to become low when all grants have been received.

The following is a detailed overview of the clk disable mechanism sequence:

1. Clock disabling is preceded by a software setup in which for a set of clocks the wakeup_en control bit is set in their PCR (see clock switchbox). It also configures the event_router to respond to certain events that can generate a wakeup towards the CGU.
2. The software then activates the clock disabling by writing b11 to the powermode register.
3. The CGU will before entering powermode ensure that all masters for which the wakeup_en control bit has been set and that support master disabling are no longer performing bus access. To do this it sets the corresponding master_disable_req signal high.

4. The multilayer uses the disable request to mask bus access requests from the master. The master is allowed to finish its current activities, but will not be able to initiate a new access.
5. When the master is no longer performing access (this might be immediately) the bus will return a master_disable_grant signal to the CGU.
6. As soon as all masters, for which the wakeup_en control bit has been set, are disabled the clocks will be disabled.
7. Clocks will become quiescent after two of their active edges.
8. When a wakeup event occurs the wake_up signal output of the event router becomes high and clears (asynchronously) the power mode register in the CGU, thereby re enabling the clocks after two of their active edges.
9. One cycle after a master clock starts running its master_disable_req signal becomes low and the bus will again process its access requests.

5.4 12 MHz oscillator

The oscillator is a 50 MHz Pierce crystal oscillator with amplitude control. It can be used in many applications e.g. as a digital reference for digital circuits, A/D and D/A clocking, etc. It is a robust design and can be used across a large frequency range.

The features of the 12MHz oscillator are as follows:

- On chip biasing resistance
- Amplitude controlled
- Large frequency range: 1 MHz to 50 MHz
- Slave mode
- Power down mode
- Bypass test mode.

5.4.1 Oscillation mode

In oscillation mode, the oscillator gain stage can have a normal or large transconductance, determined by the hf pin. A large transconductance is required for higher oscillation frequencies, higher series resistance of the crystal and higher external load capacitors. In table below the values of the external components for frequency ranges between 1 MHz and 20 MHz are given.

Table 288. Crystal oscillator interface register

Oscillation frequency fc	Max. series resistance Rs	External load capacitors Cx1, Cx2
12 MHz	< 160	18 pF, 18 pF
	< 160	39 pF, 39 pF

5.5 PLLs for generating audio clocks (HPPLL0) and system clocks (HPPLL1)

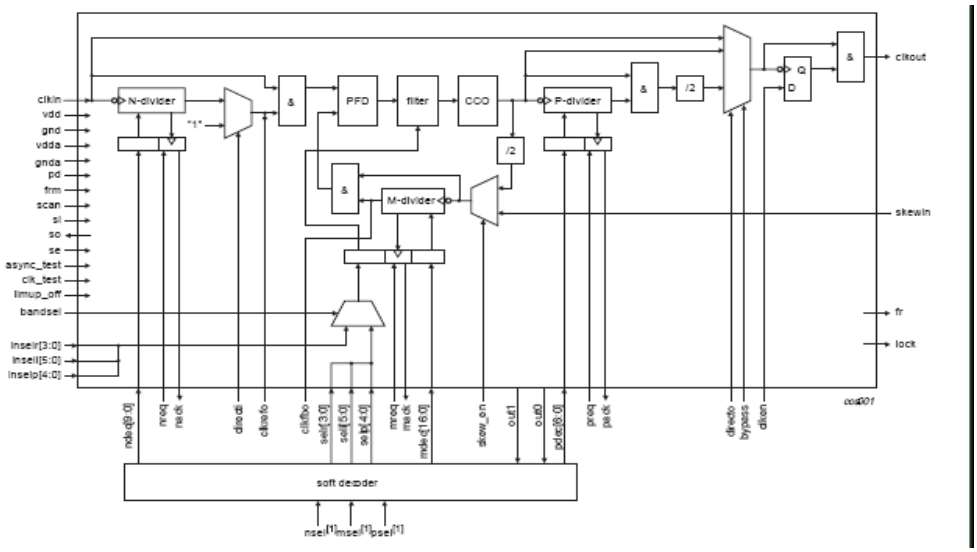


Fig 38. PLL for generating audio and system clocks

5.5.1 Functional description of the PLL

The clock input has to be fed to pinclkin. Pin clkout is the PLL clock output. The analog part of the PLL consists of a Phase Frequency Detector (PFD), filter and a Current Controlled Oscillator (CCO). The PFD has two inputs, a reference input from the (divided) external clock and one input from the divided CCO output clock. The PFD compares the phase/frequency of these input signals and generates a control signal when they do not match. This control signal is fed to a filter that drives the CCO. The PLL contains three programmable dividers: pre-divider (N), feedback-divider (M) and post-divider (P). Every divider contains a bus (xsel[n:0] in which x is n, m or p) to load a divider ratio. The dividers also possess the handshake signals xreq and xack to select a new divider ratio. The PLL contains a lock detector (use the lock pin to monitor whether or not the PLL is in lock, so don't use this pin to reset a system) that measures the phase difference between the rising edges of the input and feedback clocks. Only when this difference is smaller than the so called ‘lock criterion’ for more than seven consecutive input clock periods, the lock output switches from low to high. A single too large phase difference immediately resets the counter and causes the lock signal to drop (when it was high). Requiring seven phase measurements in a row to be below a certain figure ensures that the lock detector will not indicate lock until both the phase and frequency of the input and feedback clocks are very well aligned. This effectively prevents false lock indications, and thus ensures a glitch free lock signal. To avoid frequency hang-up, the PLL contains a frequency limiter. This feature is built in to prevent the CCO from running too fast, this can occur when, for example, a wrong feedback-divider (M) ratio is applied to the PLL. For analog test purposes there are also the pins clkfbo and clkrefo to monitor the output of the feedback and pre-divider.

5.5.2 Use of PLL operating modes

Table 289. PLL operating modes

	HP0/1_Mode bit settings:							
Mode	**Pd**	**Clken**	**Bypass**	**DirectI**	**DirectO**	**Skew_en**	**frm**	**scan**
1: Normal	0	1	0	1/0	1/0	0	0	0
2: Reserved	0	1	0	1/0	1/0	1	0	0
3: Power Down	1	x	x	x	x	x	x	x
4: Bypass	0	0/1	1	x	x	x	0	x
5: Reserved	0	x	0	x	x	x		
6: Scan	1	x	x	x	x	x	x	1
7: Enable	x	0/1	x	x	x	x	x	x

5.5.2.1 Normal Mode

Mode 1 is the normal operating mode.

The pre- and post-divider can be selected to give:

- mode 1a: Normal operating mode without post-divider and without pre-divider
- mode 1b: Normal operating mode with post-divider and without pre-divider
- mode 1c: Normal operating mode without post-divider and with pre-divider
- mode 1d: Normal operating mode with post-divider and with pre-divider

To get at the output of the PLL (clkout) the best phase-noise and jitter performance, the highest possible reference clock (clkref) at the PFD has to be used. Therefore mode 1a and 1b are recommended, when it is possible to make the right output frequency without pre-divider.

By using the post-divider the clock at the output of the PLL (clkout) the divider ratio is always even because the divide-by-2 divider after the post-divider.

Table 290. DirectI and Directo bit settings in HP0/1_Mode register

Mode	DirectI	DirectO
1a	1	1
1b	1	0
1c	0	1
1d	0	0

5.5.2.2 Mode 1a: Normal operating mode without post-divider and without pre-divider

In normal operating mode 1a the post-divider and pre-divider are bypassed. The operating frequencies are:

Fout = Fcco = 2 x M x Fin Λ (275 MHz $\leq$Fcco $\leq$550 MHz, 4 kHz $\leq$ Fin $\leq$150 MHz)

The feedback divider ratio is programmable:

- Feedback-divider M (M, 1 to 2^{15})

5.5.2.3 Mode 1b: Normal operating mode with post-divider and without pre-divider

In normal operating mode 1b the pre-divider is bypassed. The operating frequencies are:

Fout = Fcco /(2 x P) = (M / P) x Fin Λ (275 MHz ≤Fcco ≤550 MHz, 4 kHz ≤Fin ≤ 150 MHz)

The divider ratios are programmable:

- Feedback-divider M (M, 1 to 2^{15})
- Post-divider P (P, 1 to 32)

5.5.2.4 Mode 1c: Normal operating mode without post-divider and with pre-divider

In normal operating mode 1c the post-divider with divide-by-2 divider is bypassed. The operating frequencies are:

Fout = Fcco = 2 x M x Fin / N Λ (275 MHz ≤Fcco ≤550 MHz, 4 kHz ≤Fin/N ≤150 MHz)

The divider ratios are programmable:

- Pre-divider N (N, 1 to 256)
- Feedback-divider M (M, 1 to 2^{15})

5.5.2.5 Mode 1d: Normal operating mode with post-divider and with pre-divider

In normal operating mode 1d none of the dividers are bypassed. The operating frequencies are:

Fout = Fcco /(2 x P) = M x Fin /(N x P) Λ (275 MHz ≤Fcco ≤550 MHz, 4 kHz ≤Fin/N ≤150 MHz)

The divider ratios are programmable:

- Pre-divider N (N, 1 to 256)
- Feedback-divider M (M, 1 to 2^{15})
- Post-divider P (P, 1 to 32)

5.5.2.6 Mode 2: Reserved

Reserved for future use.

5.5.2.7 Mode 3: Power down mode (pd)

In this mode (pd = '1'), the oscillator will be stopped, the lock output will be made low, and the internal current reference will be turned off. During pd it is also possible to load new divider ratios at the input buses (msel, psel, nsel). Power-down mode is ended by making pd low, causing the PLL to start up. The lock signal will be made high once the PLL has regained lock on the input clock.

5.5.2.8 Mode 4: Bypass mode

In the bypass mode the input clock (clkin) will be bypassed to the output (clkout) of the PLL. Precaution has to be taken that no spikes will occur at the output of the PLL (clkout) by switching into and out of the bypass mode. To avoid spikes the output has to be disabled during switching into and out of the bypass mode. This can be done with pin clken.

5.5.2.9 Mode 5: Reserved

Reserved for future use.

5.5.2.10 Mode 6: Test mode for digital part

In this mode the digital logic in the PLL can be scanned on faults. All the digital circuitry in the PLL is connected to one scan-chain with the input and output pins si and so and the enable pin se and test clock clk_test. By setting the PLL into test mode (scan = '1'), the test clock is connected to the scan-chain. During scan mode the PLL has to be set into power down mode. To test the synchronous circuitry the Shift Mode and Normal Mode is needed. The asynchronous circuitry can be tested with the Shift Mode and the Async Mode.

5.5.2.11 Mode 7: Enable mode

In the enable mode the output clkout of the PLL is enabled. When clken = '0' the output of the PLL is low (clkout = '0'). Precaution is already taken that no spikes will occur at the output of the PLL (clkout) by switching into and out of the enable mode.

5.5.3 Settings for Audio PLL

Table 13–291 shows the divider settings used for configuring a certain output frequency F_{out} by a certain sample frequency Fs for the Audio PLL.

Table 291. Audio PLL divider ratio settings for 12 MHz

Fs (kHz)	Fout (MHz)	FCCo (MHz)	Ndec	Mdec	Pdec	SELR	SELI	SELP
256 Fs (Settings for Class-AB)								
96	24.576	491.52	63	13523	14	0	8	31
88.2	22.5792	406.4256	131	29784	23	0	8	31
64	16.384	327.68	102	7482	14	0	8	31
48	12.288	368.64	63	2665	24	0	8	31
44.1	11.2896	406.4256	131	29784	7	0	8	31
32	8.192	327.68	102	7482	31	0	8	31
512 Fs (Settings for Class-AB)								
24	12.288	368.64	63	2665	24	0	8	31
22.05	11.2896	406.4256	131	29784	7	0	8	31
16	8.192	327.68	102	7482	31	0	8	31
12	6.144	307.20	5	30580	6	0	56	31
11.025	5.6448	282.24	187	17508	6	0	8	31
1024 Fs (Settings for Class-AB)								
8	8.192	327.68	102	7482	31	0	8	31
Settings for Class-D								
-	45158.4	451.584	251	27221	5	0	8	31
-	49152	491.52	63	13523	5	0	8	31
-	65536	524.288	55	10662	2	0	4	31

5.5.4 Settings for System PLL

Table 13–292 shows the divider settings used for configuring a certain output frequency F_{out} for the System PLL.

Table 292. System PLL divider ratio settings for 12 MHz

Fout (MHz)	FCCo (MHz)	Ndec	Mdec	Pdec	SELR	SELI	SELP
24	288	770	1023	10	0	16	7
30	300	514	32597	5	0	28	13
36	288	770	1023	2	0	16	7
42	336	770	4095	2	0	16	8
48	288	770	1023	1	0	16	7
54	324	514	32085	1	0	28	14
60	360	770	8191	1	0	16	8
66	396	514	21844	1	0	36	17
72	288	770	1023	66	0	16	7
78	312	770	2047	66	0	16	7
84	336	770	4095	66	0	16	8
90	360	770	8191	66	0	16	8
96	384	770	16383	66	0	20	9
102	408	770	32767	66	0	20	9
108	432	770	32766	66	0	20	10
114	456	770	32765	66	0	20	10
120	480	770	32762	66	0	24	11
126	504	770	32757	66	0	24	11
132	528	770	32746	66	0	24	12
138	276	514	32725	98	0	24	12
144	288	770	1023	98	0	16	7
150	300	514	32597	98	0	28	13
156	312	770	2047	98	0	16	7
160	320	1	10854	98	0	44	21
164	328	1	21708	98	0	44	21
168	336	770	4095	98	0	16	8
170	340	11	5686	98	0	40	31
180	360	770	8191	98	0	16	8

5.6 Typical performance settings

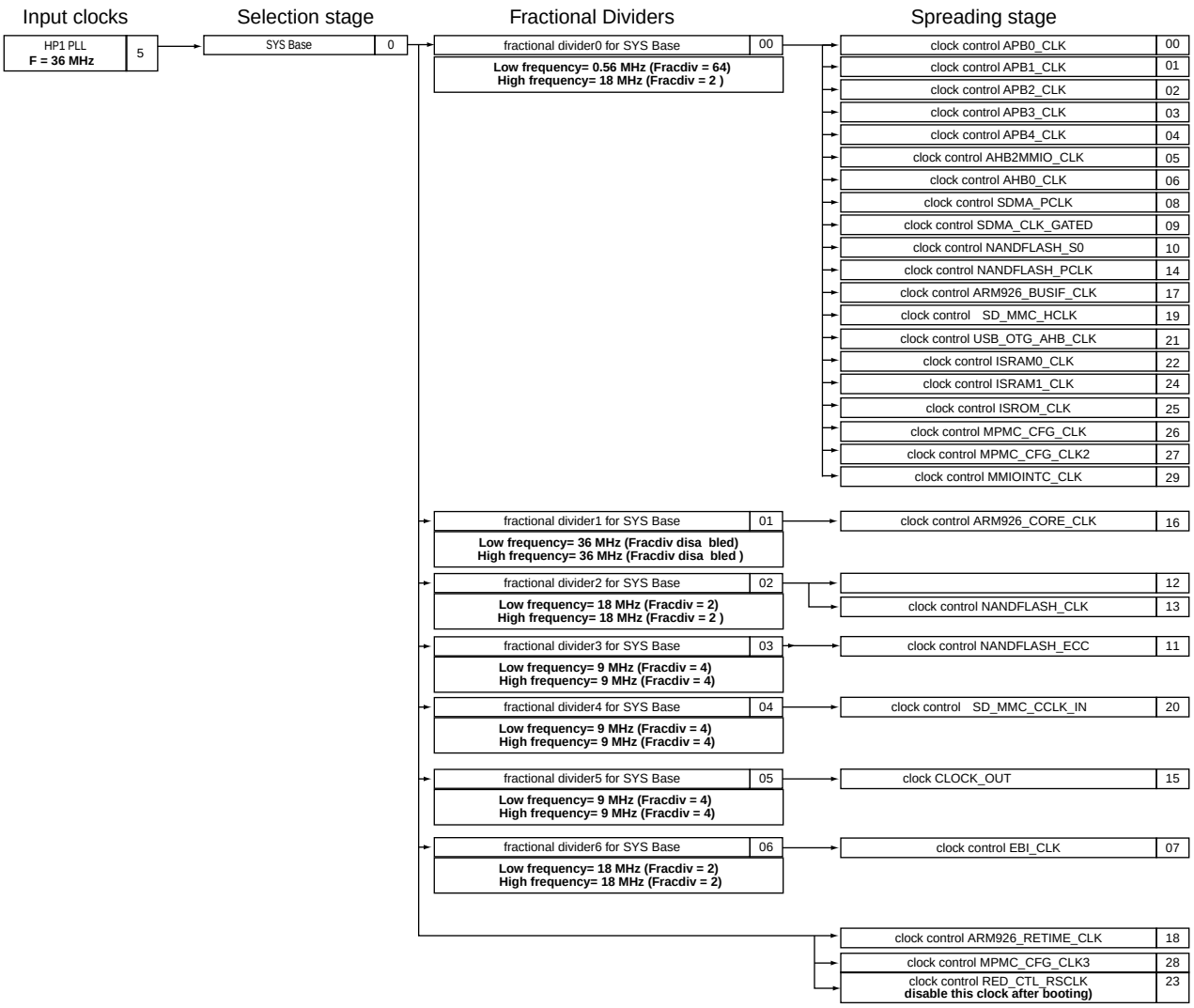


Fig 39. Performance setting - Example 1

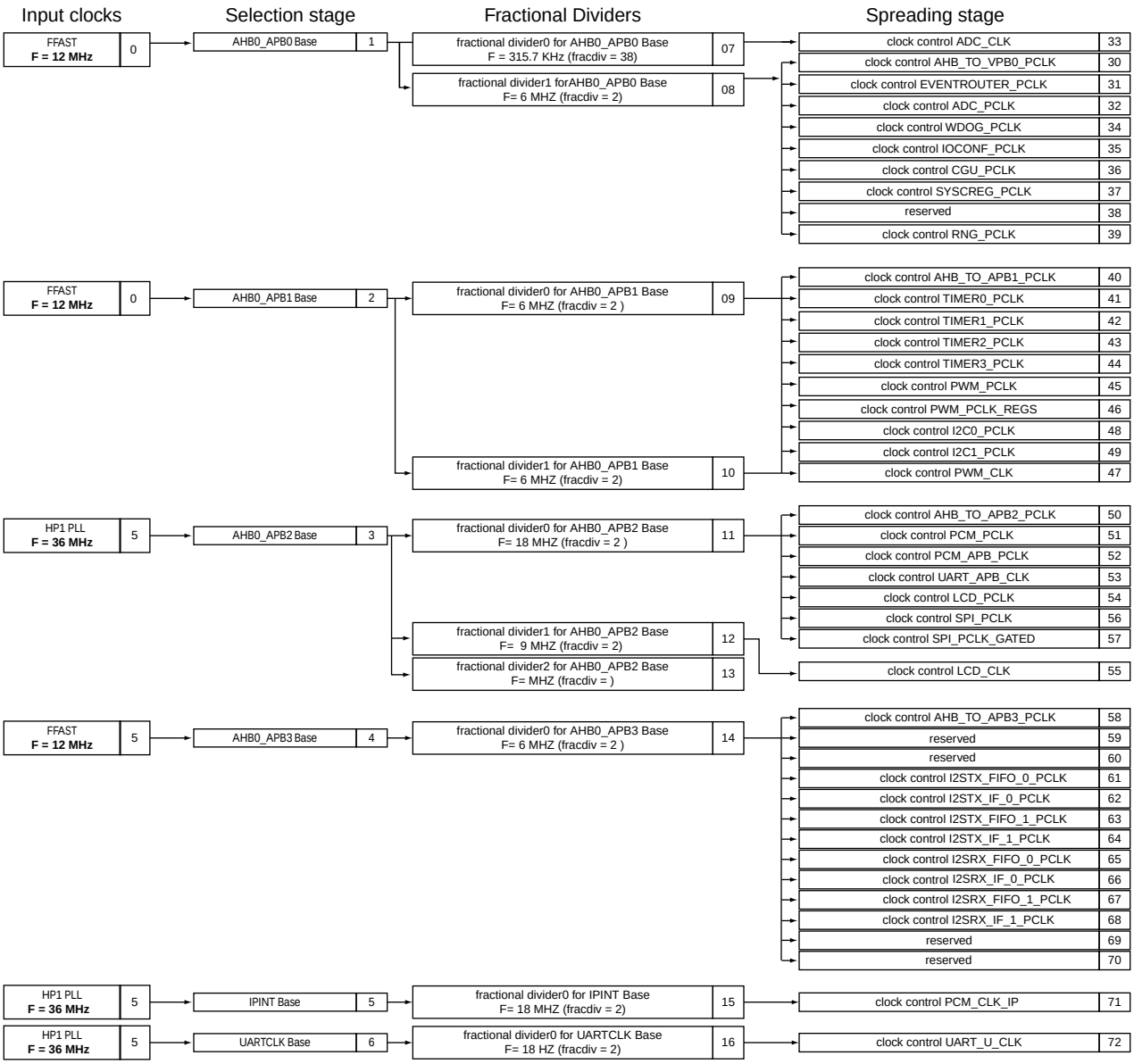


Fig 40. Performance setting - Example 2

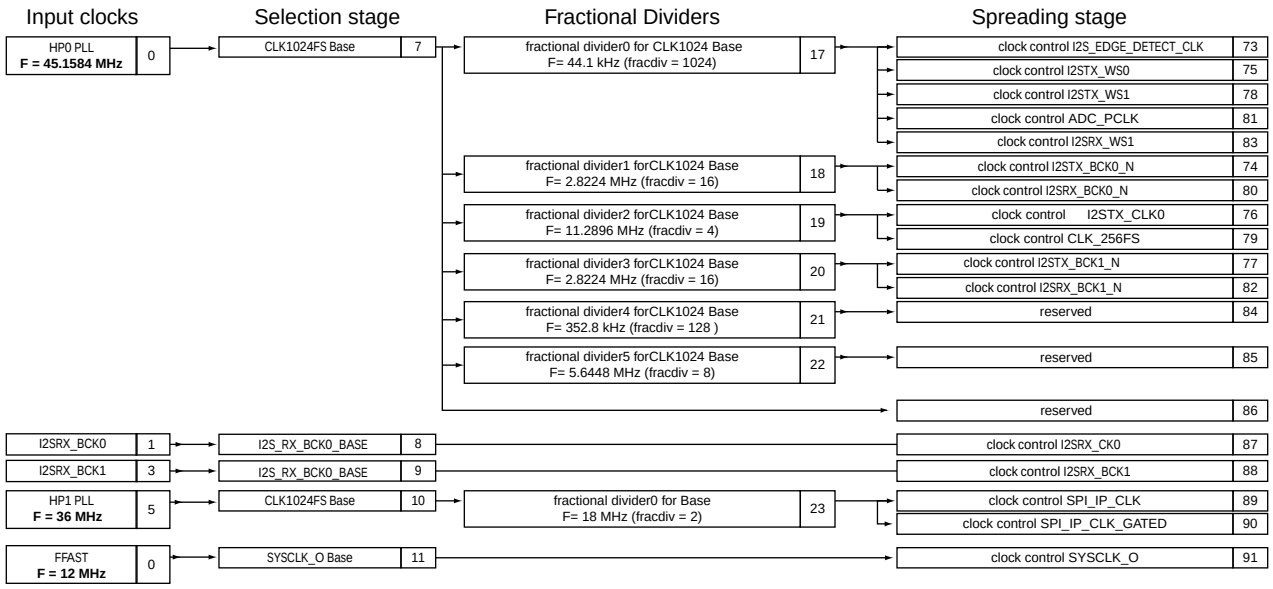


Fig 41. Performance setting - Example 3

6. Power optimization

The CGU supports variable clock scaling and external clock enabling. These items reduce power dissipation.

Clocks of blocks that are not used in a certain application can be disabled by the CGU. The same is true for PLLs and base.

7. Programming guide

7.1 Maximum Frequencies

Setting the ARM speed at a specific voltage also has implications to the AHB bus speed settings. AHB bus speed has to be a fractional (integer) division of the ARM speed but is always limited by the maximum AHB speed. The maximum frequencies of the AHB Multi-layer and the ARM926 are shown in Table 13–293 for different voltages.

Table 293. Maximum clock speed AHB Multi-layer and ARM at WC/85

Maximum speed at WC/85 in MHz				
Voltage (V)	0.9	1.0	1.1	1.2
AHB Multi-layer speed	41	57	73	90
ARM926 speed	90	130	170	180

7.2 Changing fractional dividers values

Steps to follow for changing the fractional dividers values when base frequency is lower or equal than the maximum possible frequency of one of the clocks:

1. Clear the BCR bit of the base, that consists the fractional divider(s), that you want to change. (Reset all fractional dividers of the base is not needed. This will be done by BCR bit).
2. Change the divider values of the fractional dividers.
3. Set the BCR bit of the base, that consists the fractional divider(s), that are changed.

Steps to follow for changing the fractional dividers values when base frequency is higher than the maximum possible frequency of one of the clocks:

1. Switch base to 12 MHz clock
2. Clear the BCR bit of the base, that consists the fractional divider(s), that you want to change. (Reset all fractional dividers of the base is not needed. This will be done by BCR bit)
3. Change the divider values of the fractional dividers
4. Set the BCR bit of the base, that consists the fractional divider(s), that are changed
5. Switch base to the required reference clock.

7.3 Programming variable clock-scaling

7.3.1 Programming order variable clock-scaling

1. Set the BCR bit of Base 0 to ‘0x0’.
2. Program all AHB IP to fractional divider 0.
3. Deselect these clocks from the fractional divider
   - AHB_MPMC_CFG_CLK3
   - ARM926_CORE_CLK
   - ARM926_RETIME_CLK
4. Set the PCR EN_OUT bit of the following clocks:
   - ARM926_BUSIF_CLK
   - AHB_MPMC_CFG_CLK (This is for the refresh logic for the SDRAM)
5. Set the RUN-bit of fractional divider 0 when the clock needs to be divided. When fractional divider must be the same as the SYS_BASE_CLK, then keep the RUN bit disabled.
6. Program the DYN_SEL register of dynamic fractional divider 0 with all the masters that may enable the high-speed clock.
7. Program the DYN_FDC register of fractional divider 0 (AHB) to the slow clock value and enable it by setting the DYN_FDC_ALLOW bit.
8. Program dynamic fractional divider register DYN_FDC of fractional divider 1 with the same settings as FDC, but don't set the DYN_FDC_ALLOW bit.
9. When the ARM926 needs to run slower than the SYS_BASE_CLK then:
   - Program the ARM926 clock (ARM926_CORE_CLK) to fractional divider 1
   - Program fractional divider 1. Set the RUN bit when the ARM926 clock needs to be divided
   - Program the DYN_SEL bits of dynamic fractional divider 1 with the same value as DYN_SEL of dynamic fractional divider 0

- When you want to run the ARM always on the same frequency, program register DYN_FDC of fractional divider 1 with the same settings as FDC and set the DYN_FDC_ALLOW bit. Otherwise program a different value in DYN_FDC and set the DYN_FDC_ALLOW bit.

10. Set the BCR register of SYS_BASE (BCR0) to 0x1. The clock-shop is now running with dynamic frequencies
11. Because of the now unpredictable average clock-speed of the AHB bus, the refresh of the SDRAM needs to be programmed using the alternative refresh generator. This generator can be programmed inside the ‘syscreg’ block, register ‘mpmc_testmode0’. See the programming example below.

Remark: The alternative refresh register needs to be re-calculated when the base frequency is changed.

It is very important that when both fractional dividers are enabled, it is mandatory to clear the BCR register before reprogramming fractional dividers.

When just fractional divider 0 is enabled, it can be reprogrammed on-the-fly.

One has to take the following into account for the CGU driver:

- Fractional divider should be selected but not enabled when fractional divider setting = '0' (otherwise dynamic clock scaling does not work).

7.3.2 Programming example for variable clock-scaling

```
#define FDID_FOR_AHB_IP 0
#define FDID_FOR_ARM926 1

clkid = vhGetClockId(VHISRAM_ID, 0, 0);
baseid = clkid2baseid(clkid);
     // we work on te base_id of the SRAM (and ARM, and AHB...)

// Disable all fracdivs of base 0
SWITCHBOX_REGS -> base_bcr[baseid]=0;

//Setup fracdivs. Clear first
vhClkFracDivClearAll();

vhClkFracDivConfig_fixed_fdid(FDID_FOR_AHB_IP, 1, 4, 1, 1);  // devide clock by 4

for (i=baseid2firstclk(baseid);i<= baseid2lastclk(baseid); i++) {vhClkFracDivSelect(i,
     FDID_FOR_AHB_IP);
     }

vhClkFracDivDeselect(CGU_SWITCHBOX_AHB_MPMC_PL172_CFG_CLK3_ID); // to allow the
     alternative refresh generator to run faster
vhClkFracDivDeselect(CGU_SWITCHBOX_ARM926EJS_CORE_CLK_ID); // to allow the ARM to run
     faster
vhClkFracDivDeselect(CGU_SWITCHBOX_ARM926EJS_RETIME_CLK_ID); // to allow the retimer
     of the ARM9 to work.
```

```
vhPrintfMessage("Setup CGU pcr register;\n");
     SWITCHBOX_REGS -> clk_pcr[CGU_SWITCHBOX_ARM926EJS_BUSIF_CLK_ID] |= PCR_ENOUT_EN;
     SWITCHBOX_REGS -> clk_pcr[CGU_SWITCHBOX_AHB_MPMC_PL172_CFG_CLK_ID] |=
     PCR_ENOUT_EN;

//Setting dynamic ARM9 half speed to half the speed
     vhClkFracDivConfig_fixed_fdid(FDID_FOR_ARM926, 1, 2, 1, 1);
     vhClkFracDivSelect(CGU_SWITCHBOX_ARM926EJS_CORE_CLK_ID, FDID_FOR_ARM926);

// AHB dynamic clock

vhDynFracDivSelect(FDID_FOR_AHB_IP,0xffffffff); // all masters can trigger fast speed
vhClkFracDivConfig_Dyn(FDID_FOR_AHB_IP, 1, 40,0,1,1); // divided by 40!

// ARM926 dynamic clock
vhDynFracDivSelect(0x1,0xffffffff); // all masters can trigger fast speed
vhClkFracDivConfig_Dyn(FDID_FOR_ARM926, 1, 2,0,1,1); // devided by 2

// Enable the dynamic clocks
SWITCHBOX_REGS -> base_bcr[baseid]=1;

vhPrintfMessage("System to 96 Mhz\n");
vhClkReferenceSelect(VH_PWRCLK_SYS_BASE, CGU_FIN_SELECT_FFAST);
vhClkLpPllConfig (0,CGU_FIN_SELECT_FFAST, 7, 0, 1); // 96 Mhz
vhClkReferenceSelect(VH_PWRCLK_SYS_BASE, CGU_FIN_SELECT_LPPLL0);

// Use the alternative refresh generator to generate a AHB clock-indendent refresh
     towards the SDRAM
// The base clock has just been set to 96 Mhz. The hyphotatical SDRAM has a
     auto-refresh timing of 15 us.
// Calculation: 96 Mhz = 10.42ns
// 15 us auto refresh /10.42ns = 1440 clocks.

gpSYSCREG_REGS->mpmc_testmode0=0x1000 + (1440/16);
// Enable bit + 1260 base clocks @ 96 Mhz = 15 us

// This is to set the dynamic clock temporarily high during refreshes. This will
     improve power on certain SDRAMs
gpSYSCREG_REGS->mpmc_testmode1=0x20; // Dynamic activity of 32 clock cycles
```

1. Introduction

The LPC3130/31contains a special timer module that can be used to generate a software reset in case of CPU/software crash.

The watchdog timer can also be used as an ordinary timer.

1.1 Features

This module has the following features:

- Generates a chip-wide reset request when its programmed time-out period has expired, in the event of a software or hardware failure.
- Watchdog counter can be reset by a periodical software trigger.
- After a reset, a register will indicate whether a reset has occurred because of a watchdog generated reset.
- Apart from watchdog functionality, it can also be used as a normal interval timer.

2. General description

2.1 Clock signals

Table 294. Clock Signals of the WatchDog Timer

Clock Name	Acronym	I/O	Source/Destination	Description
WDOG_PCLK	PCLK	I	CGU	Main clock for WatchDog timer block

Remark: The clock is asynchronous to the AHB Clock

2.2 Interrupt requests

The Watchdog Timer module has 2 interrupt request signals. The first is connected to the Event Router while the second is connected to the CGU.

Table 295. Interrupt Requests of the WatchDog Timer

Name	Type	Description
M0	O	Soft WatchDog interrupt to Event Router.
M1	O	Hard WatchDog Reset to CGU.

2.3 Reset signals

The CGU provides an APB Asynchronous Reset signal (PNRES). It is active low.

3. Register overview

Table 296. Register overview: WDT (register base address 0x1300 2400)

Name	Access	Address offset	Description
IR	R/W	0x00	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of two possible interrupt sources are generating an interrupt.
TCR	R/W	0x04	Timer Control Register. The TCR is used to control the Timer Counter and Prescale Counter functions. The Timer Counter and Prescale Counter can be disabled or reset through the TCR.
TC	R	0x08	Timer Counter. The TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.
PR	R/W	0x0C	This Prescale Register (PR) is an 32 bits register. It specifies the maximum value (MAXVAL) for the prescale Counter (PC). It allows the user to specify that the TC be incremented every PR+1 cycles of pclk.
PC	R	0x10	Prescale Counter. The PC is increment every WDOG_PCLK cycle when timer counter is enabled in TCR register.
MCR	R/W	0x14	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when one of the Match Registers matches the value in the TC.
MR0	R/W	0x18	Match Register 0. The MR0 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches TC.
MR1	R/W	0x1C	Match Register 1. The MR1 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR1 matches that TC.
-	-	0x38 - 0x20	Reserved
EMR	R/W	0x3C	External Match Register. The EMR contains control and status bits related to the external match pins m(0-1). EMR[7:4] is used to determine how EMR[1:0] and m(0-1) change when a match occurs.

4. Register Description

Table 297. Interrupt Register (IR) of the Watchdog Timer (address 0x1300 2400)

Bit	Symbol	R/W	Reset Value	Description
0	intr_m0	R/W	0	Interrupt bit for a MR0 and TC macth If an interrupt is being generated then the this bit will be one. Otherwise, the bit will be zero. Writing logic one will reset the interrupt. Writing a zero has no effect. Writing a one instead of a zero allows the user to write the contents of the Interrupt Register to itself thus providing a quick method of clearing.
1	intr_m1	R/W	0	Interrupt bit for a MR1 and TC macth.Operation is similar to intr_m0
31-2	-	-	-	Reserved

Table 298. Timer Control Register (TCR) of the Watchdog Timer (address 0x1300 2404)

Bit	Symbol	R/W	Reset Value	Description
0	Counter Enable	R/W	0	1- the Timer Counter is enabled for counting. 0- the counters are disabled.
1	Counter Reset	R/W	0	When one, the Timer Counter is synchronously reset on the next positive edge of WDOG_PCLK. The counters remain reset until TCR[1] is brought back to zero.
31-2	-	-	-	Reserved

Table 299. TimerCounter Register (TC) of the Watchdog Timer (address 0x1300 2408)

Bit	Symbol	R/W	Reset Value	Description
31-0	VAL	R	0	A read reflects the current value of the Watchdog Timer counter. A write loads a new value into the Timer counter. The TC is incremented every PR+1 cycles of WDOG_PCLK.

Table 300. Prescale register (PR) of the Watchdog Timer (address 0x1300 240C)

Bit	Symbol	R/W	Reset Value	Description
31-0	MAXVAL	R/W	0	It specifies the maximum value (MAXVAL) for the prescale Counter (PC). It allows the user to specify that the TC be incremented every PR+1 cycles of WDOG_PCLK.

Table 301. Prescale counter Register (PC) of the Watchdog Timer (address 0x1300 2410)

Bit	Symbol	R/W	Reset Value	Description
31-0	VAL	R	0	A read reflects the current value of the Watchdog Prescale counter. A write loads a new value into the Prescale counter. The PC is increment every WDOG_PCLK cycle when timer counter is enabled in TCR register.

Table 302. Match Control Register (MCR, address 0x1300 2414)

Bit	Symbol	R/W	Reset Value	Description
0	Interrupt on MR0	R/W	0	When one, an interrupt is generated through Event router when MR0 matches the value in the TC. When zero this interrupt feature is disabled.
1	Reset on MR0	R/W	0	When one, the TC will be synchronously reset if MR0 matches TC. When zero this feature is disabled.
2	Stop on MR0	R/W	0	When one, the TC and PC will stop counting and TCR[0] will be set to 0 if MR0 matches TC. When zero this feature is disabled.
3	Interrupt on MR1	R/W	0	When one, a system wide reset is generated through CGU when MR1 matches the value in the TC. When zero this interrupt feature is disabled.
4	Reset on MR1	R/W	0	When one, the TC will be synchronously reset if MR1 matches the TC. When zero this feature is disabled.
5	Stop on MR1	R/W	0	When one, the TC and PC will stop counting and TCR[0] will be set to 0 if MR1 matches TC. When zero this feature is disabled.
31-6	-	-	-	Reserved

Table 303. Match Register (MR) of the Watchdog Timer (MR0, address 0x1300 2418; MR1, address 0x1300 241C)

Bit	Symbol	R/W	Reset Value	Description
31-0	VAL	R/W	0	Holds the match value for the Timer Counter.

Table 304. External Match Registers (EMR) of the Watchdog Timer (address 0x1300 243C)

Bit	Symbol	R/W	Reset Value	Description
0	External Match 0	R/W	0	When match register 0 (MR0) equals the timer counter (TC) this output can either toggle, go to zero, go to one, or do nothing. EMR[4:5] controls the functionality of this output. Clearing M(0-1) can be done by writing directly to EMR.
1	External Match 1	R/W	0	When match register 0 (MR1) equals the timer counter (TC) this output can either toggle, go to zero, go to one, or do nothing. EMR[6:7] controls the functionality of this output.Clearing M(0-1) can be done by writing directly to EMR.
3-2	-	-	-	Reserved
5-4	External Match Control 0	R/W	0	Determines the functionality of External Match 0, table below shows the decode of these pins.
7-6	External Match Control 1	R/W	0	Determines the functionality of External Match 1, table below shows the decode of these pins.
31-8	-	-	-	Reserved

Table 305. External Match Control

CTRL_X[1]	CTRL_X[0]	Description
0	0	Do Nothing
0	1	Set LOW
1	0	Set HIGH
1	1	Toggle

5. Functional description

The watchdog timer block is clocked by WDOG_PCLK, which clocks a 32 bit counter. The timer counter is enabled through Timer control register (TCR). The module also has a prescale counter which can be used to further divide WDOG_PCLK clock feeding the 32 bit Timer counter.

The counter block is configured for two Match Registers. Each Match Register is 32 bit wide, same as Timer Counter. Each Match Register can be configured through the Match Control Register to stop the Timer Counter, thus maintaining their value at the time of the match, restart the Timer Counter at zero, allow the counter to continue counting, and/or generate an interrupt when its contents match those of the TC. When MRx=TC, Reset on MRx is enabled through the MCR, and the TC is enabled through the TCR, then the Timer Counter is reset on rising edge of pclk. It should be noted that stop on match has higher priority than reset on match.

External Match register provides both control and status of the external match pins M(0-1). EMR[0:1] and M(0-1) can either toggle, go to zero, go to one, or maintain state when the contents of MRx is equal to the contents of TC. EMR[4:7] are used to specify the action taken by EMR[0:1] and M(0-1). Clearing M(0-1) can be done by writing directly to EMR. Figure 14–42 shows how the Watchdog Timer is located in the LPC3130/31. Watch dog timer can be used in following different ways.

- As a watchdog The m1 output is used for generating an event to the CGU, which requests a reset.
- As a timer The m0 output is used for generating an event to the Event Router, which generates an interrupt to the Interrupt Controller. Note that the latency between the occurrence of an event at pin m0 and when the IRQ or FIQ will be asserted due this event, will be longer when the Timer Modules are used. Because the interrupts generated by the Timer Modules are directly connected to the Interrupt Controller, while the event of the Watchdog Timer goes via the Event Controller
- As watchdog and a timer The value of the MCR0 (Match Register 0) has to be a lower than the value of MCR1. Otherwise not desired resets will be generated by the CGU.

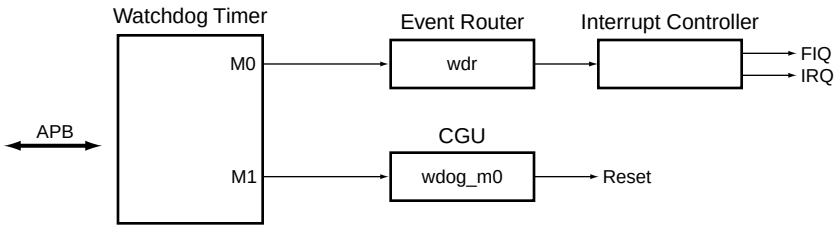


Fig 42. Watchdog Timer

Once the watchdog is enabled, it will monitor the programmed time-out period. The counter counts in modulo 2n fashion. An interrupt is generated if one of the Match Registers matches the contents of the Timer Counter indicating time-out. In normal operation the watchdog is triggered periodically, resetting the watchdog counter and ensuring that no reset is generated. In the event of a software or hardware failure preventing the CPU from triggering the watchdog, the time-out period will be exceeded and a reset requested from the CGU.

UM10314

Chapter 15: LPC3130/31 Input/Output configuration modules GPIO/IOCONFIG

Rev. 1 — 4 March 2009 User manual

1. Introduction

The General Purpose Input/Output (GPIO) pins can be controlled through the register interface provided in the Input/Output Configuration module (IOCONFIG). Next to several dedicated GPIO pins, most digital IO pins can also be used as GPIO if not required for their normal, dedicated function.

1.1 Features

The IOCONFIG module supports the following features:

- Provides control for the digital pins that can double as GPIO (besides their normal function).
- Each controlled pin can be configured for 4 operational modes:
  - Normal operation (i.e. controlled by a function block - not GPIO)
  - Driven low (GPIO)
  - Driven high (GPIO)
  - High impedance/input
- The register interface provides 'set' and 'reset' access methods for choosing the operational mode.
- Conforms to APB interface specification

2. General description

The IOCONFIG is comprised of a set of registers for individual control and visibility over a (relatively large) set of pads. By means of a set of pad multiplexers, individual pads can be switched to operate either in their normal mode, or in 'GPIO' mode. Such pads are referred to as functional pads, and are allocated to also service function blocks. In the normal mode of operation, the pad services the function block to which it is connected. In GPIO mode, a pad is fully controllable by way of dedicated bits in the mode registers, namely Mode1 and Mode0.

This block conforms to the ARM Peripheral Bus (APB) specification for ease of use with other APB peripherals.

Remark: Note that the pin multiplexing between different non-GPIO functions is controlled through the SYSCREG block, see Section 26–4.8.

3. Interface description

3.1 Clock Signals

Table 306. Clock Signals of the IOCONFIG module

Clock Name	Clock Name Acronym	I/O	Source/ Destination	Description
IOCONF_PCLK	pclk	I	CGU	APB bus clock; All registers are clocked on this clock.

3.2 Reset Signals

The IOCONFIG module is reset by a synchronous APB bus reset.

For all functional pins controlled by the IOCONFIG registers, the reset signal sets all the MODE1 register bits to '0' and the MODE0 register bits to '1'. Hence, the subsystem modules themselves control their output at reset.

If the pins are GPIO only (GPIO0 to GPIO20), they are set as inputs at reset: Their mode register bits are set to MODE1 = 0 and MODE0 = 0 by the reset signal.

4. Register overview

Table 307. Register overview: IOCONFIG (Function block level) (register base address 0x1300 3000)

Name	Address offset	Description
EBI_MCI	0x000	Base address of the register set pertaining to the first set of 32 multiplexed pads.
EBI_I2STX_0	0X040	Base address of the register set pertaining to the second set of 32 of multiplexed pads.
CGU	0X080	Base address of the register set pertaining to the Clock Generation Unit function block.
I2SRX_0	0x0C0	Base address of the register set pertaining to I2SRX function block 0.
I2SRX_1	0x100	Base address of the register set pertaining to I2SRX function block 1.
I2STX_1	0x140	Base address of the register set pertaining to I2STX function block 1.
EBI	0x180	Base address of the register set pertaining to the External Bus Interface function block.
GPIO	0x1C0	Base address of the register set pertaining to the general purpose IO
I2C1	0x200	Base address of the register set pertaining to the I2C function block
SPI	0x240	Base address of the register set pertaining to the Serial Peripheral Interface function block.

Table 307. Register overview: IOCONFIG (Function block level) (register base address 0x1300 3000)

Name	Address offset	Description
NANDFLASH_CTRL	0x280	Base address of the register set pertaining to the NANDFLASH function block.
PWM	0x2C0	Base address of the register set pertaining to the Pulse Width Modulator function block.
UART	0x300	Base address of the register set pertaining to the Universal Asynchronous Receiver/Transmitter function block.L

Each function block contains the registers in Table 15–308. Address offsets are with respect to the address of each function block in Table 15–307.

Table 308. Register level for each function block

Name	R/W	Address offset	Write Operation Description	Read Operation Description
PINS	R	0x000	Disabled	Input pin state register. Reads the state of input pins.
-			Reserved	
-			Reserved	
-			Reserved	
MODE0	R/W	0x010	Load	
MODE0_SET	R/W	0x014	Set Bits	Read Mode 0.
MODE0_RESET	R/W	0x018	Reset Bits	
-		0x01C	Reserved	
MODE1	R/W	0x020	Load	
MODE1_SET	R/W	0x024	Set Bits	Read Mode 1.
MODE1_RESET	R/W	0x028	Reset Bits	
-		0x02C	Reserved	
-		0x030	Reserved	
-		0x034	Reserved	
-		0x038	Reserved	
-		0x03C	Reserved	

Each bit in the PINS and MODEn registers correspond to the functionality of one pin. See Section 15–6 for a description of how to set the mode bits for each pin.

The PINS register reflects the current state (external) of the pins which are configured as GPIO input pins.

5. Register description

Table 309. EBI_MCI registers (EBI_MCI_PINS, address 0x1300 3000; EBI_MCI_MODE0, address 0x1300 3010; EBI_MCI_MODE0_SET, address 0x1300 3014; EBI_MCI_MODE0_RESET, address 0x1300 3018; EBI_MCI_MODE1, address 0x1300 3020; EBI_MCI_MODE1_SET, address 0x1300 3024; EBI_MCI_MODE1_RESET, address 0x1300 3028)

Bit Number in MODE0 and MODE1 Registers	Reset State	I/O Name
0	Input	mGPIO9
1	Input	mGPIO6
2	Driven by IP	mLCD_DB_7
3	Driven by IP	mLCD_DB_4
4	Driven by IP	mLCD_DB_2
5	Driven by IP	mNAND_RYBN0
6	Driven by IP	mI2STX_CLK0
7	Driven by IP	mI2STX_BCK0
8	Driven by IP	EBI_A_1_CLE
9	Driven by IP	EBI_NCAS_BLOUT_0
10	Driven by IP	mLCD_DB_0
11	Driven by IP	EBI_DQM_0_NOE
12	Driven by IP	mLCD_CSB
13	Driven by IP	mLCD_DB_1
14	Driven by IP	mLCD_E_RD
15	Driven by IP	mLCD_RS
16	Driven by IP	mLCD_RW_WR
17	Driven by IP	mLCD_DB_3
18	Driven by IP	mLCD_DB_5
19	Driven by IP	mLCD_DB_6
20	Driven by IP	mLCD_DB_8
21	Driven by IP	mLCD_DB_9
22	Driven by IP	mLCD_DB_10
23	Driven by IP	mLCD_DB_11
24	Driven by IP	mLCD_DB_12
25	Driven by IP	mLCD_DB_13
26	Driven by IP	mLCD_DB_14
27	Driven by IP	mLCD_DB_15
28	Input	mGPIO5
29	Input	mGPIO7
30	Input	mGPIO8
31	Input	mGPIO10

Table 310. EBI_I2STX_0 register (EBI_I2STX_0_PINS, address 0x1300 3040; EBI_I2STX_0_MODE0, address 0x1300 3050; EBI_I2STX_0_MODE0_SET, address 0x1300 3054; EBI_I2STX_0_MODE0_RESET, address 0x1300 3058; EBI_I2STX_0_MODE1, address 0x1300 3060; EBI_I2STX_0_MODE1_SET, address 0x1300 3064; EBI_I2STX_0_MODE1_RESET, address 0x1300 3068)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	mNAND_RYBN1
1	Driven by IP	mNAND_RYBN2
2	Driven by IP	mNAND_RYBN3
3	Driven by IP	mUART_CTS_N
4	Driven by IP	mUART_RTS_N
5	Driven by IP	mI2STX_DATA0
6	Driven by IP	mI2STX_WS0
7	Driven by IP	EBI_NRAS_BLOUT_1
8	Driven by IP	EBI_A_0_ALE
9	Driven by IP	EBI_NWE

Table 311. CGU register (CGU_PINS, address 0x1300 3080; CGU_MODE0, address 0x1300 3090; CGU_MODE0_SET, address 0x1300 3094; CGU_MODE0_RESET, address 0x1300 3098; CGU_MODE1, address 0x1300 30A0; CGU_MODE1_SET, address 0x1300 30A4; CGU_MODE1_RESET, address 0x1300 30A8)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	CGU_SYSCLK_O

Table 312. I2SRX_0 register (I2SRX_0_PINS, address 0x1300 30C0; I2SRX_0_MODE0, address 0x1300 30D0; I2SRX_0_MODE0_SET, address 0x1300 30D4; I2SRX_0_MODE0_RESET, address 0x1300 30D8; I2SRX_0_MODE1, address 0x1300 30E0; I2SRX_0_MODE1_SET, address 0x1300 30E4; I2SRX_0_MODE1_RESET, address 0x1300 30E8)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	I2SRX_BCK0
1	Driven by IP	I2SRX_DATA0
2	Driven by IP	I2SRX_WS0

Table 313. I2SRX_1 register (I2SRX_1_PINS, address 0x1300 3100; I2SRX_1_MODE0, address 0x1300 3110; I2SRX_1_MODE0_SET, address 0x1300 3114; I2SRX_1_MODE0_RESET, address 0x1300 3118; I2SRX_1_MODE1, address 0x1300 3120; I2SRX_1_MODE1_SET, address 0x1300 3124; I2SRX_1_MODE1_RESET, address 0x1300 3128)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	I2SRX_DATA1
1	Driven by IP	I2SRX_BCK1
2	Driven by IP	I2SRX_WS1

Table 314. I2STX_1 registers (I2STX_1_PINS, address 0x1300 3140; I2STX_1_MODE0, address 0x1300 3150; I2STX_1_MODE0_SET, address 0x1300 3154; I2STX_1_MODE0_RESET, address 0x1300 3158; I2STX_1_MODE1, address 0x1300 3160; I2STX_1_MODE1_SET, address 0x1300 3164; I2STX_1_MODE1_RESET, address 0x1300 3168)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	I2STX_DATA1
1	Driven by IP	I2STX_BCK1
2	Driven by IP	I2STX_WS1
3	Driven by IP	I2STX_256FS_O

Table 315. EBI registers (EBI_PINS, address 0x1300 3180; EBI_MODE0, address 0x1300 3190; EBI_MODE0_SET, address 0x1300 3194; EBI_MODE0_RESET, address 0x1300 3198; EBI_MODE1, address 0x1300 31A0; EBI_MODE1_SET, address 0x1300 31A4; EBI_MODE1_RESET, address 0x1300 31A8)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	EBI_D_9
1	Driven by IP	EBI_D_10
2	Driven by IP	EBI_D_11
3	Driven by IP	EBI_D_12
4	Driven by IP	EBI_D_13
5	Driven by IP	EBI_D_14
6	Driven by IP	EBI_D_4
7	Driven by IP	EBI_D_0
8	Driven by IP	EBI_D_1
9	Driven by IP	EBI_D_2
10	Driven by IP	EBI_D_3
11	Driven by IP	EBI_D_5
12	Driven by IP	EBI_D_6
13	Driven by IP	EBI_D_7
14	Driven by IP	EBI_D_8
15	Driven by IP	EBI_D_15

Table 316. GPIO registers (GPIO_PINS, address 0x1300 31C0; GPIO_MODE0, address 0x1300 31D0; GPIO_MODE0_SET, address 0x1300 31D4; GPIO_MODE0_RESET, address 0x1300 31D8; GPIO_MODE1, address 0x1300 31E0; GPIO_MODE1_SET, address 0x1300 31E4; GPIO_MODE1_RESET, address 0x1300 31E8)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Input	GPIO_GPIO1
1	Input	GPIO_GPIO0
2	Input	GPIO_GPIO2
3	Input	GPIO_GPIO3
4	Input	GPIO_GPIO4
5	Input	GPIO_GPIO11

Table 316. GPIO registers (GPIO_PINS, address 0x1300 31C0; GPIO_MODE0, address 0x1300 31D0; GPIO_MODE0_SET, address 0x1300 31D4; GPIO_MODE0_RESET, address 0x1300 31D8; GPIO_MODE1, address 0x1300 31E0; GPIO_MODE1_SET, address 0x1300 31E4; GPIO_MODE1_RESET, address 0x1300 31E8) *…continued*

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
6	Input	GPIO_GPIO12
7	Input	GPIO_GPIO13
8	Input	GPIO_GPIO14
9	Input	GPIO_GPIO15
10	Input	GPIO_GPIO16
11	Input	GPIO_GPIO17
12	Input	GPIO_GPIO18
13	Input	GPIO_GPIO19
14	Input	GPIO_GPIO20

Table 317. I2C1 registers (I2C1_PINS, address 0x1300 3200; I2C1_MODE0, address 0x1300 3210; I2C1_MODE0_SET, address 0x1300 3214; I2C1_MODE0_RESET, address 0x1300 3218; I2C1_MODE1, address 0x1300 3220; I2C1_MODE1_SET, address 0x1300 3224; I2C1_MODE1_RESET, address 0x1300 3288)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	I2C_SDA1
1	Driven by IP	I2C_SCL1

Table 318. SPI registers (SPI_PINS, address 0x1300 3240; SPI_MODE0, address 0x1300 3250; SPI_MODE0_SET, address 0x1300 3254; SPI_MODE0_RESET, address 0x1300 3258; SPI_MODE1, address 0x1300 3260; SPI_MODE1_SET, address 0x1300 3264; SPI_MODE1_RESET, address 0x1300 3268)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	SPI_MISO
1	Driven by IP	SPI_MOSI
2	Driven by IP	SPI_CS_IN
3	Driven by IP	SPI_SCK
4	Driven by IP	SPI_CS_OUT0

Table 319. NAND_FLASH registers (NAND_PINS, address 0x1300 3280; NAND_MODE0, address 0x1300 3290; NAND_MODE0_SET, address 0x1300 3294; NAND_MODE0_RESET, address 0x1300 3298; NAND_MODE1, address 0x1300 32A0; NAND_MODE1_SET, address 0x1300 32A4; NAND_MODE1_RESET, address 0x1300 32A8)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	NAND_NCS_3
1	Driven by IP	NAND_NCS_0
2	Driven by IP	NAND_NCS_1
3	Driven by IP	NAND_NCS_2

Table 320. PWM registers (PWM_PINS, address 0x1300 32C0; PWM_MODE0, address 0x1300 32D0; PWM_MODE0_SET, address 0x1300 32D4; PWM_MODE0_RESET, address 0x1300 32D8; PWM_MODE1, address 0x1300 32E0; PWM_MODE1_SET, address 0x1300 32E4; PWM_MODE1_RESET, address 0x1300 32E8)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	PWM_DATA

Table 321. UART registers (UART_PINS, address 0x1300 3300; UART_MODE0, address 0x1300 3310; UART_MODE0_SET, address 0x1300 3314; UART_MODE0_RESET, address 0x1300 3318; UART_MODE1, address 0x1300 3320; UART_MODE1_SET, address 0x1300 3324; UART_MODE1_RESET, address 0x1300 3328)

Bit Number in Mode0 and Mode 1 Registers	Reset State	I/O Name
0	Driven by IP	UART_RXD
1	Driven by IP	UART_TXD

6. Functional description

The IOCONFIG functionality is granulated into function blocks. Each function block addresses functional pads associated with one particular IP module. Figure 15–43 shows the pad multiplexer logic (MUX) and depicts the behavior of the IOCONFIG module. The IOCONFIG controls the selection between 'GPIO' and 'normal' mode. This selection controls the propagation of output (out) and active-low output enable (oen) signals from reaching the functional pad.

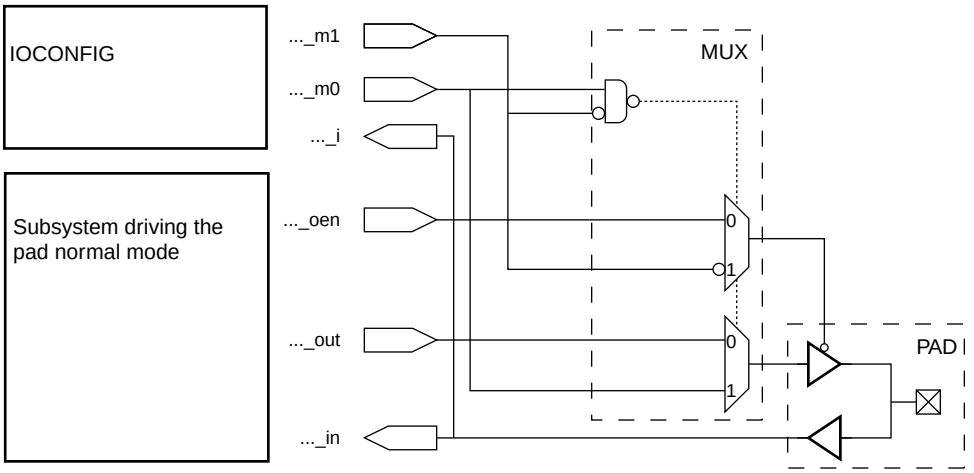


Fig 43. Pad multiplex logic connections

When 'm1' is enabled, 'm0' controls the data being output. When 'm1' is disabled, 'm0' controls selection between GPIO and normal mode. The normal (non-GPIO) mode occurs when 'm0'=1 and 'm1'=0, hence 'out' and 'oen' from the subsystem are the driving signals for the pad.

A specific bit position within the Mode0 and Mode1 registers correspond to the m0, m1 signals of a specific pad respectively. Programming the corresponding bits as depicted in Table 15–322 sets the operational state of a functional pad. Hence, changing modes of all functional pads requires at most two register access - i.e. to registers Mode0 and Mode1. As also depicted in Table 15–322, the Mode0 and Mode1 registers are also addressable for set-bits and reset-bits access.

Table 322. Functional Pad Mode Bits

Mode 1 bit (m1)	Mode 0 bit (m0)	Operating mode of functional pad
0	0	Input - Output driver is disabled
0	1	Output is controlled by the device
1	0	Output is driven low
1	1	Output is driven high

Input values are not registered and always read directly from the pad's input driver through a pad multiplex logic regardless of the mode of the pad.

7. Programming guide

To program a particular subsystem module's I/O:

- Get the IOCONFIG module's base address.
- Select the operating mode of a particular GPIO pin by programming specific bits of the Mode0, and Mode1 registers in accordance with Table 15–322.
- Writing to a register is done by writing to an address location = (IOCONFIG Base Address + Function blocks' offset + Register's offset), as shown in Table 15–307 and Table 15–308 respectively.

Note: Through the APB, only 32-bit registers are addressed. Hence bitwise-shifting operators are used to fix the bit value in the exact bit position required in the mode registers.

1. Introduction

The four-channel, 10-bit successive approximation Analog-to-Digital converter (ADC) is able to convert one of its 4 analog input signals with a conversion rate of 400kS/s for a 10-bit resolution. The resolution can be reduced down to 2-bit. The conversion rate increases in that case to 1500 kS/s.

1.1 Features

This module has the following features:

- Programmable ADC resolution from 2 to 10 bits
- Single A/D and continuous A/D conversion scan mode
- Four analog input channels, selected by an analog multiplexer
- Maximum conversion rate 400k samples/s for 10 bits resolution
- Individual result registers for each channel
- Power down mode performing minimal power dissipation
- Internal power management to switch off unused circuitry between conversion cycles
- No start-up cycles, no power down / power up recovery time

2. General description

2.1 Interface description

2.1.1 Clock signals

Table 323. Clock Signals of the ADC

Clock name	Acronym	I/O	Source/ Destination	Description
ADC_PCLK	pclk	I	CGU	APB Bus Clock. Clock from APB.
ADC_CLK	clk	I	CGU	Clock signal from CGU, frequency fclk = 31.25 kHz.

2.1.2 Pin connections

Table 324. ADC inputs

Name	Type	Description
ADC10B_GPA[3:0]	I	Analog inputs to be converted. The input voltage is between 0 V and 3.3 V.
ADC10B_VDDA33	I	ADC power
ADC10B_GNDA	I	ADC ground

2.1.3 Reset signals

The CGU provides two reset signals to the ADC block. PRESETN resets all registers and RESETN_ADC10BITS provides a global reset to the ADC.

3. Register overview

Table 325. Register overview: ADC (register base address 0x1300 2000)

Name	Access	Address Offset	Description	Reset Value
ADC_R0_REG	R	0x00	Digital conversion data for analog input channel 0	NA
ADC_R1_REG	R	0x04	Digital conversion data for analog input channel 1	NA
ADC_R2_REG	R	0x08	Digital conversion data for analog input channel 2	NA
ADC_R3_REG	W	0x0C	Digital conversion data for analog input channel 3	NA
Reserved		0x10-0x1C		
ADC_CON_REG	R/W	0X20	Controls the ADC operation modes and gives status information	0x0000 0000
ADC_CSEL _REG	R/W	0X24	Defines which analog input channels are included and defines resolution in an A/D conversion	NA
ADC_INT_ENABLE_REG	R/W	0X28	Contains a variable to enable/disable the interrupt request generation	NA
ADC_INT_STATUS_REG	R	0x2C	Contains interrupt status variable that indicates the presence of interrupt condition	NA
ADC_INT_CLEAR_REG	W	0x30	Clears interrupt status variable in ADC_INT_STATUS register	NA
Reserved		0x34	-	NA

4. Register description

ADC contains 4 registers for the converted input results, a control register, a channel selection register, and three registers for interrupt control.

4.1 ADC data registers

The ADC data registers ADC_Rx contain the output data, the output data is send to APB when CPU gives a read request. These registers store the result of an A/D conversion scan through 4 channels. Register ADC_R0 is associated to channel 0 and ADC_R1 to channel 1 and so on. The registers are read-only.

Table 326. ADC_Rx_REG (ADC_R0_REG, address 0x1300 2000; ADC_R1_REG, address 0x1300 2004, ADC_R2_REG, address 0x1300 2008; ADC_R3_REG, address 0x1300 200C)

Bit	Symbol	R/W	Reset Value	Description
9:0	ADC_Rx_DATA	R	NA	Digital conversion data with respect to analog input channel

4.2 ADC control register

This register controls the ADC operation modes and gives status information.

Table 327. ADC_CON_REG (address 0x1300 2020)

Bit	Symbol	R/W	Reset Value	Description
0	-	-	-	reserved
1	ADC_ENABLE	R/W	0	ADC enable: 0 = Non-operational mode 1: Operational mode
2	ADC_CSCAN	R/W	0	Continuous Scan: 0 = Single conversion 1 = Continuous conversion scan
3	ADC_START	R/W	0	Start command: 0 = No effect 1 = Start an A/D conversion scan
4	ADC_STATUS	R	0	ADC Status: 0 = no A/D conversion in progress 1 = A/D conversion scan is in progress Power down mode is not allowed when A/D conversion scan is in progress.
31:5	-	-	-	Reserved. Do not write ones to reserved bits.

4.3 ADC channel selection register

This register defines which analog input channels are included and defines resolution in an A/D conversion.

Table 328. ADC_CSEL_RES_REG (address 0x1300 2024)

Bit	Symbol	R/W	Reset Value	Description
3:0	CSEL0	R/W	0	By setting the bit-resolution channel 0 is selected (2-10)
7:3	CSEL1	R/W	0	By setting the bit-resolution channel 1 is selected (2-10)
11:8	CSEL2	R/W	0	By setting the bit-resolution channel 2 is selected (2-10)
15:12	CSEL3	R/W	0	By setting the bit-resolution channel 3 is selected (2-10)
31:16	-	-	-	Reserved. Do not write ones to reserved bits.

4.4 ADC interrupt enable register

This register contains a variable to enable/disable the interrupt request generation.

Table 329. ADC_INT_ENABLE_REG (address 0x1300 2028)

Bit	Symbol	R/W	Reset Value	Description
0	ADC_INT_ENABLE	R/W	0	Interrupt enable: 0 = Disable 1 = Enable An interrupt request is generated when theADC_SCAN_INT_STATUS flag is set.
31:1	-	-	-	Reserved. Do not write ones to reserved bits.

4.5 ADC interrupt status register

This register contains interrupt status variable that indicates the presence of interrupt condition. It is read-only.

Table 330. ADC_INT_STATUS_REG (address 0x1300 202C)

Bit	Symbol	R/W	Reset Value	Description
0	ADC_INT_STATUS	R	0	Interrupt status: 0 = No interrupt pending 1 = Interrupt pending.
31:1	-	-	-	Reserved. Do not write ones to reserved bits.

4.6 ADC interrupt clear register

A write action to this address location allows to clear interrupt status variable in the ADC_INT_STATUS register.

Table 331. ADC_INT_CLEAR_REG (address 0x1300 2030)

Bit	Symbol	R/W	Reset Value	Description
0	ADC_INT_CLEAR	W	0	Interrupt clear: 0 = No effect 1 = clear ADC_SCAN_INT_STATUS variable.
31:1	-	-	-	Reserved. Do not write ones to reserved bits.

5. Functional description

5.1 A/D conversion control

The ADC performs analog input channel multiplexing, sampling and successive digital approximation of analog signals. The protocol sequence starts with the sampling of the selected analog input channels. This is followed by the 'approximation loop' in which the DAC voltage is stepwise approximated to the sampled input voltage. The number of loop cycles depends on whether 2,3,4,..,10 bit resolution is selected. The A/D conversion result becomes valid when the ADC_INT_STATUS bit is set (see Table 16–330), and the result is moved to the ADC_Rx_REG register of the corresponding channel(s).

5.2 ADC Resolution

The resolution within the AD conversion process is software programmable through the ADC_CSEL_RES_REG (Table 16–328). The resolution can be adjust between 2 and 10 bits. The conversion rate is as follows:

$$ConversionRate = \frac{ClockFrequency}{(Resolution + 1)}$$

5.3 Multi-channel A/D conversion scan

Each analog input channel has a 10 bit result registers to store its A/D conversion result. By selecting a resolution from 2 to 10 bit through the ADC_CSEL_RES_REG, a channel is included in the ADC scan. A channel is excluded from the scan if its resolution is set to 0. For example, it is possible to scan channel 0, channel1, and channel 3without scanning channel 2. The A/D conversion scan process can be started by software.

There are two scan modes: Continuous Scan mode and Single Scan mode. In Continuous Scan mode, A/D conversion scans are carried out continuously: once one scan completed, the next one is started automatically. In Single Scan mode, only a single conversion scan is carried out, the next scan must be started explicitly by software.

5.4 Clocking

The clock for the ADC interface (ADC_PCLK) is provided by the Clock Generation Unit (CGU).

The frequency of the ADC clock (ADC_CLK) doesn't have to be very high because the number of samples can be low for measuring for example a battery voltage. Therefore the clock frequency ADC_CLK that is offered by the Clock Generation Unit can be used (max is 4.5 MHz).

5.5 Interrupts

The ADC interface implements one interrupt, a scan interrupt which indicates the completion of an A/D conversion scan process and the validity of the data in the result registers.

6. Power optimization

To minimize the power consumption in the ADC, all the unused circuitry is switched off between conversions. During these inactive periods, the analog part is in Power-down mode, and the current from ADC10B_VDDA33 is below 1 µA. The current from VDDI is reduced to the clock buffers operation. The analog part of the ADC can be powered down via the ADC_ENABLE bit in the ADC_CON_REG.

In addition, the analog part of the ADC can be set explicitly into power-down mode using a syscregister bit: SYSCREG_ADC_PD_ADC10BITS (see Table 26–519).

7. Programming guide

Reset ADC Interface:

1.Write PRESETN: reset ADC_CONTROLLER:

`ADC_ENABLE = ADC_START = ADC_SELFREF = ADC_CSCAN = ADC_CSEL0 = ADC_CSEL1 = ADC_INT_ENABLE = ADC_INT_STATUS =0`

Setup ADC Interface:

1. Read ADC Status => `ADC_INT_STATUS = 0`
2. Write ADC Interrupt Enable Register => `ADC_INT_ENABLE = 1`
3. Read ADC Interrupt Status Register => `ADC_INT_STATUS = 0`
4. Select reference voltage input (vrefp0) => `ADC_SELVREF = 1`
5. Write Select Channel and Resolution Register => `ADC_CSEL0` until `ADC_CSEL3`
6. Write ADC enable bit => `ADC_ENABLE = 1`

Run Single Conversion Mode (`ADC_CSCAN = 0`):

1. Write ADC Start Command => `ADC_START = 1`
2. Write ADC Start Command => `ADC_START = 0`
3. Wait for interrupt => `ADC_INT_STATUS`
4. Read ADC Interrupt Status Register => `ADC_INT_STATUS = 1`
5. Write ADC interrupt clear register => `ADC_INT_CLEAR = 1`
6. Read ADC Result Register for Channel 0 until Channel 4=> `ADC_R0.. ADC_R4`
7. Wait for ADC Interrupt Status Register => `ADC_INT_STATUS = 0`

Run Single Conversion Stops => Go back to Setup ADC Interface

Run Continuous Conversion Mode (`ADC_CSCAN = 1`):

1. Write ADC Start Command => `ADC_START = 1`
2. Write ADC Start Command => `ADC_START = 0`
3. Wait for interrupt => `ADC_INT_STATUS`
4. Read ADC Interrupt Status Register => `ADC_INT_STATUS = 1`
5. Write ADC interrupt clear register => `ADC_INT_CLEAR = 1`
6. Read ADC Result Register for Channel 0 until Channel 4=> `ADC_R0.. ADC_R4`
7. Wait for ADC Interrupt Status Register => `ADC_INT_STATUS = 0`
8. Run Continuous Conversion Start at point 3.
9. Stop Continuous Conversion => `ADC_CSCAN = 0`.

1. Introduction

The Event Router extends the interrupt capability of the system by offering a flexible and versatile way of generating interrupts. Combined with the wake-up functionality of the CGU, it also offers a way to wake-up the system from suspend mode (with all clocks deactivated).

1.1 Features

This module has the following features:

- Provides programmable routing of input events to multiple outputs for use as interrupts or wake-up signals.
- Input events can come from internal signals or from the pins that can be used as GPIO. Note that the GPIO pins can be used to trigger events when in normal, functional mode or in GPIO mode.
- Inputs can be used either directly or latched (edge detected) as an event source.
- The active level (polarity) of the input signal for triggering events is programmable.
- Direct events will disappear when the input becomes inactive.
- Latched events will remain active until they are explicitly cleared.
- Each input can be masked globally for all outputs at once.
- Each input can be masked for each output individually.
- Event detect status can be read for each output separately.
- Event detection is fully asynchronous (no active clock required).
- Module can be used to generate a system wake-up from suspend mode.

2. General description

The Event Router has four interrupt outputs that are connected to the interrupt controller and one wake-up output connected to the CGU as shown in Figure 17–44. The output signals are activated when an event (for instance a rising edge) is detected on one of the input signals. The input signals of the Event Router are numerous and are connected to relevant internal (control) signals in the system or to external signals through pins of the LPC3130/31.

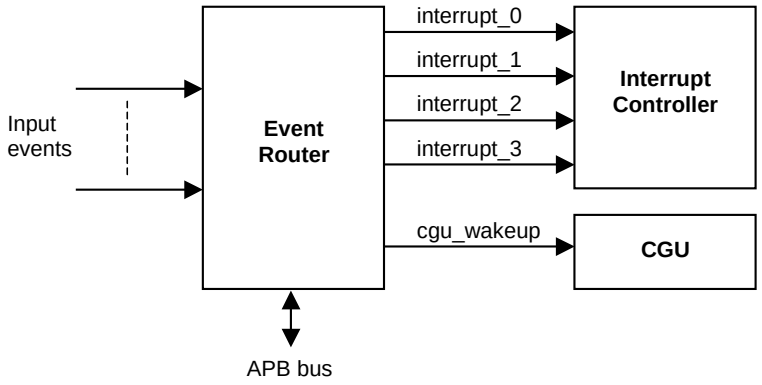


Fig 44. Event router block diagram

2.1 Interface description

2.1.1 Clock Signals

Table 332. Clock Signals of the Event Router

Clock Name	I/O	Source/Destination	Description
EVENT_ROUTER_PCLK	I	CGU	APB Clock. All registers are clocked on this clock.

2.1.2 Pin Connections

Table 17–333 shows the input signals of the Event Router which are connected to pins of the LPC3130/31.

Table 333. Event router signals connected to pins of the LPC3130/31

Name	Type	Reset Value	Description
EBI_D_6	I	0x0	EBI data 6
EBI_D_5	I	0x0	EBI data 5
EBI_D_4	I	0x0	EBI data 4
EBI_D_3	I	0x0	EBI data 3
EBI_D_2	I	0x0	EBI data 2
EBI_D_1	I	0x0	EBI data 1
EBI_D_0	I	0x0	EBI data 0
mNAND_RYBN3	I	0x0	EBI NAND ready/busy 3
mNAND_RYBN2	I	0x0	EBI NAND ready/busy 2
mNAND_RYBN1	I	0x0	EBI NAND ready/busy 1
mNAND_RYBN0	I	0x0	EBI NAND ready/busy 0
mLCD_RW_WR	I	0x0	LCD 6800 enable. 8080 write enable
mLCD_E_RD	I	0x0	LCD 6800 enable. 8080 write enable
mLCD_CSB	I	0x0	LCD chip select
mLCD_RS	I	0x0	LCD instruction register/data register

Table 333. Event router signals connected to pins of the LPC3130/31

Name	Type	Reset Value	Description
mLCD_DB_15	I	0x0	LCD data 15
mLCD_DB_14	I	0x0	LCD data 14
mLCD_DB_13	I	0x0	LCD data 13
mLCD_DB_12	I	0x0	LCD data 12
mLCD_DB_11	I	0x0	LCD data 11
mLCD_DB_10	I	0x0	LCD data 10
mLCD_DB_9	I	0x0	LCD data 9
mLCD_DB_8	I	0x0	LCD data 8
mLCD_DB_7	I	0x0	LCD data 7
mLCD_DB_6	I	0x0	LCD data 6
mLCD_DB_5	I	0x0	LCD data 5
mLCD_DB_4	I	0x0	LCD data 4
mLCD_DB_3	I	0x0	LCD data 3
mLCD_DB_2	I	0x0	LCD data 2
mLCD_DB_1	I	0x0	LCD data 1
mLCD_DB_0	I	0x0	LCD data 0
mGPIO10	I	0x0	general purpose IO pin 10
mGPIO9	I	0x0	general purpose IO pin 9
mGPIO8	I	0x0	general purpose IO pin 8
mGPIO7	I	0x0	general purpose IO pin 7
mGPIO6	I	0x0	general purpose IO pin 6
mGPIO5	I	0x0	general purpose IO pin 5
GPIO4	I	0x0	general purpose IO pin 4
GPIO3	I	0x0	general purpose IO pin 3
GPIO2	I	0x0	general purpose IO pin 2
GPIO1	I	0x0	general purpose IO pin 1
GPIO0	I	0x0	general purpose IO pin 0
EBI_NRAS_BLOUT_1	I	0x0	EBI upper lane byte select
EBI_NCAS_BLOUT_0	I	0x0	EBI lower lane byte select
EBI_DQM_0_NOE	I	0x0	EBI read enable
EBI_A_1_CLE	I	0x0	EBI clock latch enable
EBI_A_0_ALE	I	0x0	EBI address latch enable
EBI_NWE	I	0x0	EBI write enable
EBI_D_15	I	0x0	EBI data 15
EBI_D_14	I	0x0	EBI data 14
EBI_D_13	I	0x0	EBI data 13
EBI_D_12	I	0x0	EBI data 12
EBI_D_11	I	0x0	EBI data 11
EBI_D_10	I	0x0	EBI data 10
EBI_D_9	I	0x0	EBI data 9
EBI_D_8	I	0x0	EBI data 8

Table 333. Event router signals connected to pins of the LPC3130/31

Name	Type	Reset Value	Description
EBI_D_7	I	0x0	EBI data 7
PWM_DATA	I	0x0	PWM output
I2SRX_WS0	I	0x0	I2SRX word select
I2SRX_DATA0	I	0x0	I2SRX serial data output
I2SRX_BCK0	I	0x0	I2SRX bit clock
mI2STX_WS0	I	0x0	I2STX word select
mI2STX_DATA0	I	0x0	I2STX serial data output
mI2STX_BCK0	I	0x0	I2STX bit clock
mI2STX_CLK0	I	0x0	I2STX serial clock
mUART_RTS_N	I	0x0	UART ready to send
mUART_CTS_N	I	0x0	UART clear to send
UART_TXD	I	0x0	UART serial output
UART_RXD	I	0x0	UART serial input
SPI_CS_OUT0	I	0x0	SPI chip select output (master)
SPI_SCK	I	0x0	SPI clock input (slave) / clock output (master)
SPI_CS_IN	I	0x0	SPI chip select input (slave)
SPI_MOSI	I	0x0	SPI data output (master) / data input (slave)
SPI_MISO	I	0x0	SPI data input (master) / data output (slave)
NAND_NCS_3	I	0x0	EBI chip enable 3
NAND_NCS_2	I	0x0	EBI chip enable 2
NAND_NCS_1	I	0x0	EBI chip enable 1
NAND_NCS_0	I	0x0	EBI chip enable 0
USB_ID	I	0x0	USB indicates to the USB transceiver whether in device or host mode
USB_VBUS	I	0x0	USB supply detection line

2.1.3 Interrupt Request Signals

Table 334. Interrupt Request Signals of Event Router

Name	Type	Description
INTERRUPT_0	O	interrupt output 0 connected to Interrupt controller as CASCADED_IRQ0
INTERRUPT_1	O	interrupt output 1 connected to Interrupt controller as CASCADED_IRQ1
INTERRUPT_2	O	interrupt output 2 connected to Interrupt controller as CASCADED_IRQ2
INTERRUPT_3	O	interrupt output 3 connected to Interrupt controller as CASCADED_IRQ3

2.1.4 Reset Signals

The event router is reset by an APB bus reset (PNRES).

3. Register overview

All input event signals connected to the Event Router are grouped together in banks (k). If the number of input signals is larger than 32 a next bank is used (signal 33 becomes bit 0 of bank 1). The pending register shows the arrangement of the input event signal per bank. The bits of all other registers are arranged in the same way as the pending register but for reasons of simplicity the detailed bit information is left out

Table 335. Register overview: Event router (register base address 0x1300 0000)

Name	R/W	Address Offset	Description
-	-	0x000 - 0xBFF	reserved
pend[0]	R	0x0C00	input event pending register - bank 0
pend[1]	R	0x0C04	input event pending register - bank 1
pend[2]	R	0x0C08	input event pending register - bank 2
pend[3]	R	0x0C0C	input event pending register - bank 3
int_clr[0]	W	0x0C20	input event clear register - bank 0
int_clr[1]	W	0x0C24	input event clear register - bank 1
int_clr[2]	W	0x0C28	input event clear register - bank 2
int_clr[3]	W	0x0C2C	input event clear register - bank 3
int_set[0]	W	0x0C40	input event set register - bank 0
int_set[1]	W	0x0C44	input event set register - bank 1
int_set[2]	W	0x0C48	input event set register - bank 2
int_set[3]	W	0x0C4C	input event set register - bank 3
mask[0]	R/W	0x0C60	input event mask register - bank 0
mask[1]	R/W	0x0C64	input event mask register - bank 1
mask[2]	R/W	0x0C68	input event mask register - bank 2
mask[3]	R/W	0x0C6C	input event mask register - bank 3
mask_clr[0]	W	0x0C80	input event mask clear register - bank 0
mask_clr[1]	W	0x0C84	input event mask clear register - bank 1
mask_clr[2]	W	0x0C88	input event mask clear register - bank 2
mask_clr[3]	W	0x0C8C	input event mask clear register - bank 3
mask_set[0]	W	0x0CA0	input event mask set register - bank 0
mask_set[1]	W	0x0CA4	input event mask set register - bank 1
mask_set[2]	W	0x0CA8	input event mask set register - bank 2
mask_set[3]	W	0x0CAC	input event mask set register - bank 3
apr[0]	R/W	0x0CC0	input event activation polarity register - bank 0
apr[1]	R/W	0x0CC4	input event activation polarity register - bank 1
apr[2]	R/W	0x0CC8	input event activation polarity register - bank 2
apr[3]	R/W	0x0CCC	input event activation polarity register - bank 3
atr[0]	R/W	0x0CE0	input event activation type register - bank 0
atr[1]	R/W	0x0CE4	input event activation type register - bank 1
atr[2]	R/W	0x0CE8	input event activation type register - bank 2
atr[3]	R/W	0x0CEC	input event activation type register - bank 3

Table 335. Register overview: Event router (register base address 0x1300 0000)

Name	R/W	Address Offset	Description
rsr[0]	R	0x0D20	input event raw status register - bank 0
rsr[1]	R	0x0D24	input event raw status register - bank 1
rsr[2]	R	0x0D28	input event raw status register - bank 2
rsr[3]	R	0x0D2C	input event raw status register - bank 3
intout	R	0x0D40	status of interrupt output pins
-	R	0x0E00	reserved
-	R	0x0FFC	reserved
intoutPend[0][0]	R	0x1000	interrupt output 0 pending register - bank 0
intoutPend[0][1]	R	0x1004	interrupt output 0 pending register - bank 1
intoutPend[0][2]	R	0x1008	interrupt output 0 pending register - bank 2
intoutPend[0][3]	R	0x100C	interrupt output 0 pending register - bank 3
intoutPend[1][0]	R	0x1020	interrupt output 1 pending register - bank 0
intoutPend[1][1]	R	0x1024	interrupt output 1 pending register - bank 1
intoutPend[1][2]	R	0x1028	interrupt output 1 pending register - bank 2
intoutPend[1][3]	R	0x102C	interrupt output 1 pending register - bank 3
intoutPend[2][0]	R	0x1040	interrupt output 2 pending register - bank 0
intoutPend[2][1]	R	0x1044	interrupt output 2 pending register - bank 1
intoutPend[2][2]	R	0x1048	interrupt output 2 pending register - bank 2
intoutPend[2][3]	R	0x104C	interrupt output 2 pending register - bank 3
intoutPend[3][0]	R	0x1060	interrupt output 3 pending register - bank 0
intoutPend[3][1]	R	0x1064	interrupt output 3 pending register - bank 1
intoutPend[3][2]	R	0x1068	interrupt output 3 pending register - bank 2
intoutPend[3][3]	R	0x106C	interrupt output 3 pending register - bank 3
intoutPend[4][0]	R	0x1080	cgu_wakeup pending register - bank 0
intoutPend[4][1]	R	0x1084	cgu_wakeup pending register - bank 1
intoutPend[4][2]	R	0x1088	cgu_wakeup pending register - bank 2
intoutPend[4][3]	R	0x108C	cgu_wakeup pending register - bank 3
intoutMask[0][0]	R/W	0x1400	interrupt output 0 mask register -bank 0
intoutMask[0][1]	R/W	0x1404	interrupt output 0 mask register - bank 1
intoutMask[0][2]	R/W	0x1408	interrupt output 0 mask register - bank 2
intoutMask[0][3]	R/W	0x140C	interrupt output 0 mask register - bank 3
intoutMask[1][0]	R/W	0x1420	interrupt output 1 mask register - bank 0
intoutMask[1][1]	R/W	0x1424	interrupt output 1 mask register - bank 1
intoutMask[1][2]	R/W	0x1428	interrupt output 1 mask register - bank 2
intoutMask[1][3]	R/W	0x142C	interrupt output 1 mask register - bank 3
intoutMask[2][0]	R/W	0x1440	interrupt output 2 mask register - bank 0
intoutMask[2][1]	R/W	0x1444	interrupt output 2 mask register - bank 1
intoutMask[2][2]	R/W	0x1448	interrupt output 2 mask register - bank 2
intoutMask[2][3]	R/W	0x144C	interrupt output 2 mask register - bank 3
intoutMask[3][0]	R/W	0x1460	interrupt output 3 mask register - bank 0

Table 335. Register overview: Event router (register base address 0x1300 0000)

Name	R/W	Address Offset	Description
intoutMask[3][1]	R/W	0x1464	interrupt output 3 mask register - bank 1
intoutMask[3][2]	R/W	0x1468	interrupt output 3 mask register - bank 2
intoutMask[3][3]	R/W	0x146C	interrupt output 3 mask register - bank 3
intoutMask[4][0]	R/W	0x1480	cgu_wakeup mask register - bank 0
intoutMask[4][1]	R/W	0x1484	cgu_wakeup mask register - bank 1
intoutMask[4][2]	R/W	0x1488	cgu_wakeup mask register - bank 2
intoutMask[4][3]	R/W	0x148C	cgu_wakeup mask register - bank 3
intoutMaskClr[0][0]	W	0x1800	interrupt output 0 mask clear register - bank 0
intoutMaskClr[0][1]	W	0x1804	interrupt output 0 mask clear register - bank 1
intoutMaskClr[0][2]	W	0x1808	interrupt output 0 mask clear register - bank 2
intoutMaskClr[0][3]	W	0x180C	interrupt output 0 mask clear register - bank 3
intoutMaskClr[1][0]	W	0x1820	interrupt output 1 mask clear register - bank 0
intoutMaskClr[1][1]	W	0x1824	interrupt output 1 mask clear register - bank 1
intoutMaskClr[1][2]	W	0x1828	interrupt output 1 mask clear register - bank 2
intoutMaskClr[1][3]	W	0x182C	interrupt output 1 mask clear register - bank 3
intoutMaskClr[2][0]	W	0x1840	interrupt output 2 mask clear register - bank 0
intoutMaskClr[2][1]	W	0x1844	interrupt output 2 mask clear register - bank 1
intoutMaskClr[2][2]	W	0x1848	interrupt output 2 mask clear register - bank 2
intoutMaskClr[2][3]	W	0x184C	interrupt output 2 mask clear register - bank 3
intoutMaskClr[3][0]	W	0x1860	interrupt output 3 mask clear register - bank 0
intoutMaskClr[3][1]	W	0x1864	interrupt output 3 mask clear register - bank 1
intoutMaskClr[3][2]	W	0x1868	interrupt output 3 mask clear register - bank 2
intoutMaskClr[3][3]	W	0x186C	interrupt output 3 mask clear register - bank 3
intoutMaskClr[4][0]	W	0x1880	cgu_wakeup mask clear register - bank 0
intoutMaskClr[4][1]	W	0x1884	cgu_wakeup mask clear register - bank 1
intoutMaskClr[4][2]	W	0x1888	cgu_wakeup mask clear register - bank 2
intoutMaskClr[4][3]	W	0x188C	cgu_wakeup mask clear register - bank 3
intoutMaskSet[0][0]	W	0x1C00	interrupt output 0 mask set register - bank 0
intoutMaskSet[0][1]	W	0x1C04	interrupt output 0 mask set register - bank 1
intoutMaskSet[0][2]	W	0x1C08	interrupt output 0 mask set register - bank 2
intoutMaskSet[0][3]	W	0x1C0C	interrupt output 0 mask set register - bank 3
intoutMaskSet[1][0]	W	0x1C20	interrupt output 1 mask set register - bank 0
intoutMaskSet[1][1]	W	0x1C24	interrupt output 1 mask set register - bank 1
intoutMaskSet[1][2]	W	0x1C28	interrupt output 1 mask set register - bank 2
intoutMaskSet[1][3]	W	0x1C2C	interrupt output 1 mask set register - bank 3
intoutMaskSet[2][0]	W	0x1C40	interrupt output 2 mask set register - bank 0
intoutMaskSet[2][1]	W	0x1C44	interrupt output 2 mask set register - bank 1
intoutMaskSet[2][2]	W	0x1C48	interrupt output 2 mask set register - bank 2
intoutMaskSet[2][3]	W	0x1C4C	interrupt output 2 mask set register - bank 3
intoutMaskSet[3][0]	W	0x1C60	interrupt output 3 mask set register - bank 0

Table 335. Register overview: Event router (register base address 0x1300 0000)

Name	R/W	Address Offset	Description
intoutMaskSet[3][1]	W	0x1C64	interrupt output 3 mask set register - bank 1
intoutMaskSet[3][2]	W	0x1C68	interrupt output 3 mask set register - bank 2
intoutMaskSet[3][3]	W	0x1C6C	interrupt output 3 mask set register - bank 3
intoutMaskSet[4][0]	W	0x1C80	cgu_wakeup mask set register - bank 0
intoutMaskSet[4][1]	W	0x1C84	cgu_wakeup mask set register - bank 1
intoutMaskSet[4][2]	W	0x1C88	cgu_wakeup mask set register - bank 2
intoutMaskSet[4][3]	W	0x1C8C	cgu_wakeup mask set register - bank 3

4. Register description

4.1 Pending Register pend[0] to pend[3]

The pending registers indicate when a masked input event is active. Reading a '1' indicates the input event is active, reading a '0' means no input event.

Table 336. Pend [0] register (address 0x1300 0C00)

Bit	Symbol	R/W	Reset Value	Description
31	EBI_D_6	R	0x0	input event from GPIO pin
30	EBI_D_5	R	0x0	input event from GPIO pin
29	EBI_D_4	R	0x0	input event from GPIO pin
28	EBI_D_3	R	0x0	input event from GPIO pin
27	EBI_D_2	R	0x0	input event from GPIO pin
26	EBI_D_1	R	0x0	input event from GPIO pin
25	EBI_D_0	R	0x0	input event from GPIO pin
24	mNAND_RYBN3	R	0x0	input event from GPIO pin
23	mNAND_RYBN2	R	0x0	input event from GPIO pin
22	mNAND_RYBN1	R	0x0	input event from GPIO pin
21	mNAND_RYBN0	R	0x0	input event from GPIO pin
20	mLCD_RW_WR	R	0x0	input event from GPIO pin
19	mLCD_E_RD	R	0x0	input event from GPIO pin
18	mLCD_CSB	R	0x0	input event from GPIO pin
17	mLCD_RS	R	0x0	input event from GPIO pin
16	mLCD_DB_15	R	0x0	input event from GPIO pin
15	mLCD_DB_14	R	0x0	input event from GPIO pin
14	mLCD_DB_13	R	0x0	input event from GPIO pin
13	mLCD_DB_12	R	0x0	input event from GPIO pin
12	mLCD_DB_11	R	0x0	input event from GPIO pin
11	mLCD_DB_10	R	0x0	input event from GPIO pin
10	mLCD_DB_9	R	0x0	input event from GPIO pin
9	mLCD_DB_8	R	0x0	input event from GPIO pin
8	mLCD_DB_7	R	0x0	input event from GPIO pin

Table 336. Pend [0] register (address 0x1300 0C00) *…continued*

Bit	Symbol	R/W	Reset Value	Description
7	mLCD_DB_6	R	0x0	input event from GPIO pin
6	mLCD_DB_5	R	0x0	input event from GPIO pin
5	mLCD_DB_4	R	0x0	input event from GPIO pin
4	mLCD_DB_3	R	0x0	input event from GPIO pin
3	mLCD_DB_2	R	0x0	input event from GPIO pin
2	mLCD_DB_1	R	0x0	input event from GPIO pin
1	mLCD_DB_0	R	0x0	input event from GPIO pin
0	pcm_int	R	0x0	input event from PCM

Table 337. Pend [1] register (address 0x1300 0C04)

Bit	Symbol	R/W	Reset Value	Description
31	GPIO16	R	0x0	input event from GPIO pin
30	GPIO15	R	0x0	input event from GPIO pin
29	GPIO14	R	0x0	input event from GPIO pin
28	GPIO13	R	0x0	input event from GPIO pin
27	GPIO12	R	0x0	input event from GPIO pin
26	GPIO11	R	0x0	input event from GPIO pin
25	mGPIO10	R	0x0	input event from GPIO pin
24	mGPIO9	R	0x0	input event from GPIO pin
23	mGPIO8	R	0x0	input event from GPIO pin
22	mGPIO7	R	0x0	input event from GPIO pin
21	mGPIO6	R	0x0	input event from GPIO pin
20	mGPIO5	R	0x0	input event from GPIO pin
19	GPIO4	R	0x0	input event from GPIO pin
18	GPIO3	R	0x0	input event from GPIO pin
17	GPIO2	R	0x0	input event from GPIO pin
16	GPIO1	R	0x0	input event from GPIO pin
15	GPIO0	R	0x0	input event from GPIO pin
14	EBI_NRAS_BLOUT_1	R	0x0	input event from GPIO pin
13	EBI_NCAS_BLOUT_0	R	0x0	input event from GPIO pin
12	EBI_DQM_0_NOE	R	0x0	input event from GPIO pin
11	EBI_A_1_CLE	R	0x0	input event from GPIO pin
10	EBI_A_0_ALE	R	0x0	input event from GPIO pin
9	EBI_NWE	R	0x0	input event from GPIO pin
8	EBI_D_15	R	0x0	input event from GPIO pin
7	EBI_D_14	R	0x0	input event from GPIO pin
6	EBI_D_13	R	0x0	input event from GPIO pin
5	EBI_D_12	R	0x0	input event from GPIO pin
4	EBI_D_11	R	0x0	input event from GPIO pin
3	EBI_D_10	R	0x0	input event from GPIO pin

Table 337. Pend [1] register (address 0x1300 0C04) *…continued*

Bit	Symbol	R/W	Reset Value	Description
2	EBI_D_9	R	0x0	input event from GPIO pin
1	EBI_D_8	R	0x0	input event from GPIO pin
0	EBI_D_7	R	0x0	input event from GPIO pin

Table 338. Pend [2] register (address 0x1300 0C08)

Bit	Symbol	R/W	Reset Value	Description
31	PWM_DATA	R	0x0	input event from GPIO pin
30	I2C_SCL1	R	0x0	input event from GPIO pin
29	I2C_SDA1	R	0x0	input event from GPIO pin
28	CLK_256FS_O	R	0x0	input event from GPIO pin
27	I2STX_WS1	R	0x0	input event from GPIO pin
26	I2STX_BCK1	R	0x0	input event from GPIO pin
25	I2STX_DATA1	R	0x0	input event from GPIO pin
24	I2SRX_WS1	R	0x0	input event from GPIO pin
23	I2SRX_BCK1	R	0x0	input event from GPIO pin
22	I2SRX_DATA1	R	0x0	input event from GPIO pin
21	I2SRX_WS0	R	0x0	input event from GPIO pin
20	I2SRX_DATA0	R	0x0	input event from GPIO pin
19	I2SRX_BCK0	R	0x0	input event from GPIO pin
18	mI2STX_WS0	R	0x0	input event from GPIO pin
17	mI2STX_DATA0	R	0x0	input event from GPIO pin
16	mI2STX_BCK0	R	0x0	input event from GPIO pin
15	mI2STX_CLK0	R	0x0	input event from GPIO pin
14	mUART_RTS_N	R	0x0	input event from GPIO pin
13	mUART_CTS_N	R	0x0	input event from GPIO pin
12	UART_TXD	R	0x0	input event from GPIO pin
11	UART_RXD	R	0x0	input event from GPIO pin
10	SPI_CS_OUT0	R	0x0	input event from GPIO pin
9	SPI_SCK	R	0x0	input event from GPIO pin
8	SPI_CS_IN	R	0x0	input event from GPIO pin
7	SPI_MOSI	R	0x0	input event from GPIO pin
6	SPI_MISO	R	0x0	input event from GPIO pin
5	NAND_NCS_3	R	0x0	input event from GPIO pin
4	NAND_NCS_2	R	0x0	input event from GPIO pin
3	NAND_NCS_1	R	0x0	input event from GPIO pin
2	NAND_NCS_0	R	0x0	input event from GPIO pin
1	GPIO18	R	0x0	input event from GPIO pin
0	GPIO17	R	0x0	input event from GPIO pin

Table 339. Pend [3] register (address 0x1300 0C0C)

Bit	Symbol	R/W	Reset Value	Description
31:30	-			Reserved
29	isram1_mrc_finished	R	0x0	ISRAM1 redundancy controller event
28	isram0_mrc_finished	R	0x0	ISRAM0 redundancy controller event
27	USB_ID	R	0x0	input event from GPIO pin
26	usb_otg_vbus_pwr_en	R	0x0	input event from USB
25	usb_atx_pll_lock	R	0x0	USB PLL lock event
24	usb_otg_ahb_needclk	R	0x0	input event from USB
23	USB_VBUS	R	0x0	input event from USB_VBUS pin
22	MCI_CLK	R	0x0	input event from GPIO pin
21	MCI_CMD	R	0x0	input event from GPIO pin
20	MCI_DAT_7	R	0x0	input event from GPIO pin
19	MCI_DAT_6	R	0x0	input event from GPIO pin
18	MCI_DAT_5	R	0x0	input event from GPIO pin
17	MCI_DAT_4	R	0x0	input event from GPIO pin
16	MCI_DAT_3	R	0x0	input event from GPIO pin
15	MCI_DAT_2	R	0x0	input event from GPIO pin
14	MCI_DAT_1	R	0x0	input event from GPIO pin
13	MCI_DAT_0	R	0x0	input event from GPIO pin
12	arm926_lp_nirq	R	0x0	Reflects nIRQ signal going to ARM core
11	arm926_lp_nfiq	R	0x0	Reflects nFIQ signal going to ARM core
10	I2c1_scl_n	R	0x0	input event from I2C1
9	I2c0_scl_n	R	0x0	input event from I2C0
8	uart_rxd	R	0x0	input event from UART
7	wdog_m0	R	0x0	input event from Watch Dog Timer
6	adc_int	R	0x0	input event from ADC
5	timer3_intct1	R	0x0	input event from Timer 3
4	timer2_intct1	R	0x0	input event from Timer 2
3	timer1_intct1	R	0x0	input event from Timer 1
2	timer0_intct1	R	0x0	input event from Timer 0
1	GPIO20	R	0x0	input event from GPIO20
0	GPIO19	R	0x0	input event from GPIO19

4.2 Interrupt Clear Register int_clr[0] to int_clr[3]

These registers allow latched events to be cleared by writing a '1' to any bits corresponding to the interrupts to be cleared. The bits are arranged in the same way as the pending registers.

Table 340. int_clr register(int_clr0, address 0x1300 0C20; int_clr1, address 0x1300 0C24 int_clr_2, address 0x1300 0C28; int_clr3, address 0x1300 0C2C)

Bit	Symbol	R/W	Reset Value	Description
31:0	int_clr	W	0x0	interrupt clear register, write any bit '1' to clear that interrupt latch (one bit per input)

4.3 Interrupt set register int_set[0] to int_set[3]

This register allows the user to generate an event through software. This register could be used for debugging the event router driver or for generating artificial events.

Table 341. int_set register (int_set0, address 0x1300 0C40; int_set1, address 0x1300 0C44 int_set2, address 0x1300 0C48; int_set3, address 0x1300 0C4C)

Bit	Symbol	R/W	Reset Value	Description
31:0	int_set	W	0x0	interrupt set register, write any bit '1' to set that interrupt latch (one bit per input)

4.4 Mask Register mask[0] to mask[3]

The mask register allows the user to enable or disable input events globally across all outputs. An event, which is enabled in this register, will cause activation of any outputs, which have also been programmed in the intoutMask register. An event, which is disabled, will not cause activation of any outputs. The bits are arranged in the same way as the pending register. For multi-thread applications separate addresses are also provided for clearing and setting of latch bits, removing the need for read-modify-write operations.

Table 342. mask register (mask0, address 0x1300 0C60; mask1, address 0x1300 0C64 mask2, address 0x1300 0C68; mask3, address 0x1300 0C6C)

Bit	Symbol	R/W	Reset Value	Description
31:0	int_set	W	0xFFFFFFFF	global input event enable, one bit per input1 = enable, 0 = disable an input

4.5 Mask clear register mask_clr[0] to mask_clr[3]

These registers allow bits in the mask register to be clear by writing a '1' to any bits corresponding to the mask bits to be set. The bits are arranged in the same way as the pending register.

Table 343. mask register (mask_clr0, address 0x1300 0C80; mask_clr1, address 0x1300 0C84; mask_clr2, address 0x1300 0C88; mask_clr3, address 0x1300 0C8C)

Bit	Symbol	R/W	Reset Value	Description
31:0	mask_clr	W	0xFFFFFFFF	event enable clear register, write any bit '1' to clear an input event enable (one bit per input)

4.6 Mask set register mask_set[0] to mask_set[3]

These registers allow bits in the mask register to be set by writing a `1' to any bits corresponding to the mask bits to be set. The bits are arranged in the same way as the pending register.

Table 344. mask_set register (mask_set0, address 0x1300 0CA0; mask_set1, address 0x1300 0C4A4; mask_set2, address 0x1300 0CA8; mask_set3, address 0x1300 0C8C)

Bit	Symbol	R/W	Reset Value	Description
31:0	mask_set	W	0x0	event enable set register, write any bit '1' to set an input event enable bit (one bit per input)

4.7 Activation polarity register apr[0] to apr[3]

The activation polarity register (APR) is used to configure which level is the active state for the event sources. A high bit indicates that the event is high sensitive, a low bit that it is low sensitive. The bits are arranged in the same way as the pending register.

Table 345. apr[0] register (address 0x1300 0CC0)

Bit	Symbol	R/W	Reset Value	Description
31:0	apr	R/W	0x1	activation polarity register (one bit per input)1 = high sensitive, 0 = low sensitive

Table 346. apr[1] register (address 0x1300 0CC4)

Bit	Symbol	R/W	Reset Value	Description
31:0	apr	R/W	0x0	activation polarity register (one bit per input)1 = high sensitive, 0 = low sensitive

Table 347. apr[2] register (address 0x1300 0CC8)

Bit	Symbol	R/W	Reset Value	Description
31:0	apr	R/W	0x1	activation polarity register (one bit per input)1 = high sensitive, 0 = low sensitive

Table 348. apr[3] register (address 0x1300 0CCC)

Bit	Symbol	R/W	Reset Value	Description
31:30	-			Reserved
29:0	apr	R/W	0xFFFFFFC	activation polarity register (one bit per input)1 = high sensitive, 0 = low sensitive

4.8 Activation type register atr[0] to atr[3]

The activation type register (ATR) is used to configure whether an event signal is used directly or if it is latched. If it is latched, the interrupt will persist after its event source has become inactive until it is cleared by an int_clr write action. The event router includes an edge detection circuit, which prevents reassertion of an interrupt if the input remains at the active level after the latch is cleared. A high bit written to the ATR selects the latched event as the event source; a low bit uses the event directly. The bits are arranged in the same way as the pending register.

Table 349. atr[0] register (address 0x1300 0CE0)

Bit	Symbol	R/W	Reset Value	Description
31:0	atr	R/W	0x1	activation type register (one bit per input)1 = latched(edge), 0 = direct

Table 350. atr[1] register (address 0x1300 0CE4)

Bit	Symbol	R/W	Reset Value	Description
31:0	atr	R/W	0x1	activation type register (one bit per input)1 = latched(edge), 0 = direct

Table 351. atr[2] register (address 0x1300 0CE8)

Bit	Symbol	R/W	Reset Value	Description
31:0	atr	R/W	0xFFFFFFC	activation type register (one bit per input)1 = latched(edge), 0 = direct

Table 352. atr[3] register (address 0x1300 0CEC)

Bit	Symbol	R/W	Reset Value	Description
31:30	-	R/W		Reserved
29:0	atr	R/W	0x77FFFFC	activation type register (one bit per input)1 = latched(edge), 0 = direct

4.9 Raw status registers rsr[0] to rsr[3]

The Raw Status Register (RSR) shows unmasked events including latched events. A high bit read from the RSR indicates an event is (or has been) generated by the particular event source. A low bit read indicates the device is not generating an event. Level sensitive events are expected to be held and removed by the interrupt source. The bits are arranged in the same way as the pending register.

Table 353. rsr registers (rsr0, address 0x1300 0D20; rsr1, address 0x1300 0D24; rsr2, address 0x1300 0D28; rsr3, address 0x1300 0D2C)

Bit	Symbol	R/W	Reset Value	Description
31:0	rsr	R	0x0	raw status of input events or event latches in latched mode (one bit per input)

4.10 Intout register

These registers show the current state of the event router interrupt outputs.

Table 354. intout register (address 0x1300 0D40)

Bit	Symbol	R/W	Reset Value	Description
31:5	-			reserved
4	cgu_wakeup	R	0x0	Current state of cgu_wakeup output
3	intout3	R	0x0	current state of interrupt output 3
2	intout2	R	0x0	current state of interrupt output 2
1	intout1	R	0x0	current state of interrupt output 1
0	intout0	R	0x0	current state of interrupt output 0

4.11 Interrupt output pending register intoutPend[0:4][0:3]

With these registers the user can, for each individual interrupt output, enable/disable an input event to be routed to that output. The register/bit arrangement matches that of the pending register.

Table 355. intoutPend[m][n] register (m = 0 to 4, n = 0 to 3, intoutPend00, address 0x1300 1000; intoutPend01, address 0x1300 1004 to intoutPend43, address 0x1300 108C)

Bit	Symbol	R/W	Reset Value	Description
31:0	intoutPend	R	0x0	an array of status bits, one bit per input showing which events are pending for each interrupt output

4.12 Interrupt output mask register intoutMask[0:4][0:3]

With these registers the user can, for each individual interrupt output, enable/disable an input event to be routed to that output. The bits are arranged in the same way as the pending register. For multi-thread applications separate addresses are also provided for clearing and setting of latch bits, removing the need for read-modify-write operations.

Table 356. intoutMask[m][n] (m = 0 to 4, n = 0 to 3, intoutMask00, address 0x1300 1400; intoutMask01, address 0x1300 1404 to intoutMask43, address 0x1300 148C)

Bit	Symbol	R/W	Reset Value	Description
31:0	intoutMask	R/W	0x0	enable bits for each interrupt output, connecting input events to that output. 1: input event is enabled 0: input event is disabled

4.13 Interrupt output mask clear register intoutMaskClr[0:4][0:3]

Writing a bit to 1 in any of these registers clears the corresponding bit in the corresponding intoutMask register. The bits are arranged in the same way as the pending register.

Table 357. intoutMaskClr[m][n] register (m = 0 to 4, n = 0 to 3, intoutMask00, address 0x1300 1400; intoutMask01, address 0x1300 1404 to intoutMask43, address 0x1300 148C)

Bit	Symbol	R/W	Reset Value	Description
31:0	intoutMaskClr	W	0x0	event enable clear register for each interrupt output, write any bit '1' to clear the corresponding bit in the corresponding output mask

4.14 Interrupt output mask set register intoutMaskSet[0:4][0:3]

Writing a bit to 1 in any of these registers sets the corresponding bit in the corresponding intoutMask register. The bits are arranged in the same way as the pending register.

Table 358. intoutMaskSet[m][n] register (m = 0 to 4, n = 0 to 3, intoutMaskSet00, address 0x1300 1C00; intoutMaskSet01, address 0x1300 1404 to intoutMaskSet43, address 0x1300 1C8C)

Bit	Symbol	R/W	Reset Value	Description
31:0	intoutMaskSet	W	0x0	event enable set register for each interrupt output, write any bit '1' to clear the corresponding bit in the corresponding output mask

5. Functional Description

5.1 Wake-up Behavior

All event sources, which are connected to the event inputs, can cause an wake-up trigger to the CGU module.

5.2 Architecture

The event router block is accessible through a APB interface. The number of interrupt outputs that can be generated is limited due to the maximum APB data size of 32 bit.

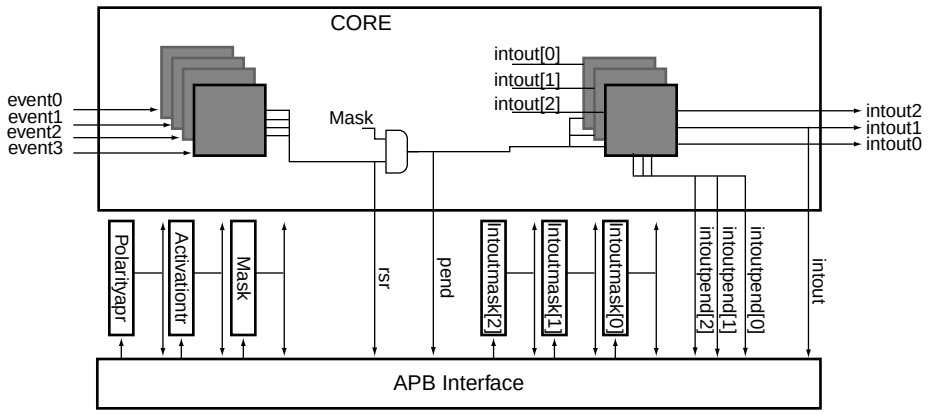


Fig 45. Architecture

Input events are processed in event slices, one for each event signal. Each of these slices generates one event signal and is visible in the rsr register. These events are then anded with enables from the mask register to give pending event status. All events are connected to an output slice for each output. In an output slice the signals from all inputs can be enabled or disabled to generate that output. There is a separate pending, mask, maskClr and maskSet register for each output slice.

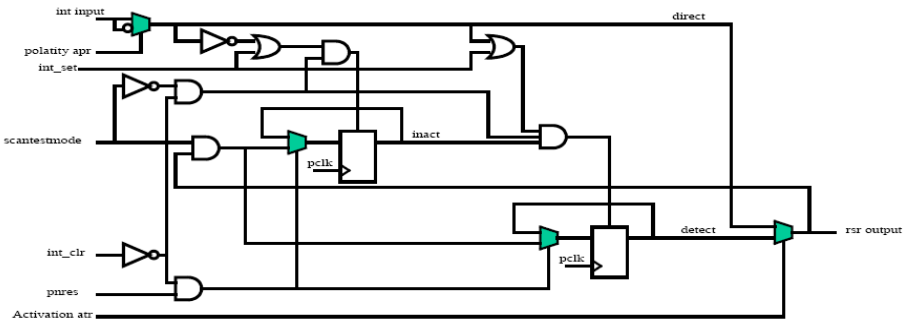


Fig 46. Input slice

An event slice is controlled through bits in the polarity, activation, intSet and intClr registers:

- The polarity setting conditionally inverts the interrupt input event
- The activation setting selects between latched or direct event
- The resulting interrupt event is visible through a read action on the raw status register
- Edge detection is performed by two registers with set functions. One detects the signal being inactive, the other detects the signal being active Latched
- These interrupt values are visible through read actions on the status registers
- A write '1' on the corresponding slice index in the intClr will clear the latched interrupt event synchronously.

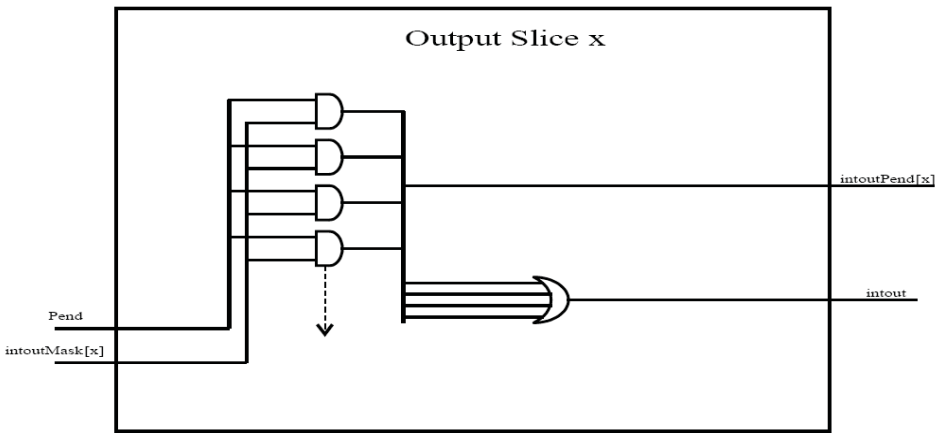


Fig 47. Output slice

For individual outputs, the results can be read on their specific intoutPend status register.

6. Power optimization

This module can be used in low power systems to request power-up or start of a clock on an external or internal event.

1. Introduction

The random number generator (RNG) generates true random numbers. Two independent ring oscillator clocks feed the RNG clock inputs. They provide a source clock, which varies from device to device, depending on the VLSI technology process and the device's own temperature. Because of this unstable clock source, the random numbers generated are highly unpredictable. Therefore, it is very unlikely that two random number generators, in two different systems, will generate the same random number sequence.

2. General Description

2.1 Features

- True random number generator.
- Two internal generators, each fed by a dedicated ring oscillator clock.
- The ring oscillator clock sources are unstable, depending on both VLSI technology process spread and chip's temperature. The instability of the RNG clock sources is essential to guarantee true and highly unpredictable random numbers.
- Each internal generator combines the pseudo-random output of an LFSR (Linear Feedback Shift Register) and a MWCG (Multiply-with-Carry Generator).
- The random number register does not rely on any kind of reset.
- The generators are free running in order to increase the level of randomness and security.

2.2 Interface Description

The RNG interface consists of a APB slave bus interface and two separate clock inputs. The independent clock inputs are directly fed to the RNG's internal random number generators

2.2.1 Clock Signals

Three clock signals are fed to the RNG.

Two clock sources, namely ANALOG_CLK_RNG[0], and ANALOG_CLK_RNG[1], are unstable. These two clocks are used to generate highly unpredictable random numbers. The input frequency of this clock sources should be in the range of 10 to 50 MHz.

The third clock, namely RNG_PCLK, is used to clock the RNG APB bus interface logic and the RNG internal registers. The three clock domains are asynchronous to each other. The random numbers generated by the internal pseudo-random generators are sampled using the `RNG_PCLK` to the `RNG_PCLK` random number register. The frequency of the RNG_PCLK is governed by the CGU and must be synchronous with the APB0 subsystem bus clock.

Table 359. Clock Signals of the RNG Module

Clock Name	I/O	Source/ Destination	Description
ANALOG_CLK_RNG[0]	I	CGU/RINGOSC0	The clock signal fed to the generator #0 of the RNG. The oscillator RINGOSC0 can be enabled via the SYSCREG block (Table 26–518) register.
ANALOG_CLK_RNG[1]	I	CGU/RINGOSC1	The clock signal fed to the generator #1 of the RNG. The oscillator RINGOSC0 can be enabled via the SYSCREG block (Table 26–518) register.
RNG_PCLK	I	CGU	RNG clock signal for the APB bus interface

2.2.2 Interrupt Request Signals

The RNG has no interrupt signals.

2.2.3 Reset Signals

The CGU provides two reset signals to the RNG, the PNRES in the APB clock domain and a global asynchronous reset (HRESET).

3. Register overview

Table 360. Register overview: RNG (base register address: 0x1300 6000)

Name	R/W	Address Offset	Description
RANDOM_NUMBER	R	0x000	Random number
-	R	0x004	Reserved
-	R	0x008	Reserved
POWERDOWN	R/W	0xFF4	Power-down mode

4. Register description

Table 361. RANDOM_NUMBER (address 0x1300 6000)

Bit	Symbol	R/W	Reset Value	Description
31:0	RANDOM_NUMBER	R	Random!	This register contains a random 32 bit number which is computed each time it is read

Table 362. POWERDOWN (address 0x1300 6FF4)

Bit	Symbol	R/W	Reset Value	Description
31:3	-	-	-	Reserved

Table 362. POWERDOWN (address 0x1300 6FF4)

Bit	Symbol	R/W	Reset Value	Description
2	Power down	R/W	0x0	When set all accesses to standard registers are blocked
1	Force soft-reset	R/W	0x0	When used in combination with soft-reset it forces an RNG reset immediately
0	Soft-reset	R/W	0x0	Request a software RNG reset, to be executed when the APB slave interface is deselected, and or pending APB register read/write operations are completed

5. Functional description

Each of the two internal generators combines the pseudo-random output of an LFSR (Linear Feedback Shift Register) and a MWCG (Multiply-with-Carry Generator). These generators are free running as long as the ring oscillator clock is available on the ANALOG_CLK_RNG input port.

The final random number is created by xor'ing the output of both internal generators. Each generator runs using a different input frequency. This guarantees that the random number generated will have good statistical random properties.

Each time the RANDOM_NUMBER is queried, the final xor'ed value is sampled and stored on the internal APB register. For a correct sampling process both the RNG_CLK and the ANALOG_CLK_RNG clocks must be running.

Once the Ring Oscillator clocks are enabled via the SYSCREG (Table 26–518), and the RNG_PCLK clock is enabled, the block provides a random number every time the RNG random register is read via the APB bus.

The block never allows the same number to be read more than once. To prevent two consecutive readings being performed via the APB bus, the RNG will complete the second APB bus read operation only when a new random number is available. The latency of producing random numbers depends on the frequency of the ring oscillator clocks.

A random number is generated after 5 clock cycles of the slower of the two clocks provided in the ANALOG_CLK_RNG input ports. During this time the RNG retains the control on the APB bus.

6. Power optimization

To reduce power, the ring oscillators can be switched off, via the system register, when no random number is needed. The RNG can be switched off either by setting the POWERDOWN register in power down mode, or by disabling the RNG_PCLK clock in the CGU.

7. Programming guide

Always make sure that when reading the RANDOM_NUMBER register, the RNG must be active, and the oscillators must be enabled.

7.1 Enabling the RNG

- Set the POWERDOWN bit to '0'.
- Enable the RNG_PCLK clock via the CGU.
- Enable both Ring Oscillator clocks via the system configuration register Table 26–518.

7.2 Reading a random number from the RNG

- Read register RANDOM_NUMBER.

7.3 Disable the RNG

- Switch off the RNG clock via the CGU.
- Switch off the Ring Oscillators' clocks via the system register Table 26–518.

1. Introduction

The Serial Peripheral Interface (SPI) module is used for synchronous serial data communication with other devices, that support the SPI/SSI protocol.

Examples of the devices that this SPI module can communicate with are memories, cameras, and WiFi-g. The SPI/SSI-bus is a 5-wire interface and is suitable for low, medium, and high data rate transfers.

1.1 Features

- Motorola SPI frame format with a word size of 8/16 bits
- Texas Instruments SSI frame format with a word size of 4…16 bits
- Serial clock rate master mode maximum 45 MHz
- Serial clock rate slave mode maximum 25 MHz
- Support for single data access DMA
- Full-Duplex operation
- Maskable interrupts
- Multiple slaves support (at the maximum of 3 slaves).

2. General description

The SPI is a serial bus standard that was established by Motorola and is supported in silicon products from various manufacturers. In operation, there is a clock, a ‘data in’, a ’data out,’ and a ’chip select’ for each integrated circuit that is to be controlled. Most serial digital devices can be controlled with this combination of signals.

Devices communicate using a master/slave relationship, in which the master generates the clock and selects a slave device. The data may be transferred in either, or both directions simultaneously. In fact, as far as SPI is concerned, data is always transferred in both directions. It is up to the master and slave devices to know whether a received byte is meaningful or not.

The SSI (Synchronous Serial Interface) is similar to the SPI protocol. It makes use of the same pins. However in this protocol the data is only clocked out on the falling edge and clocked in on the rising edge of the master. In the SPI protocol this may be swapped. The SSI protocol was established by Texas Instruments.

2.1 Interface description

2.1.1 Clock signals

Table 363. SPI Module Clock Signals

Clock Name	I/O	Source/ Destination	Description
SPI_CLK	I	CGU	Main clock of the module. Most of the logic in this module runs on this clock. Its frequency depends on the required speed for the external SPI interface. The maximum frequency is 90 MHz.
SPI_CLK_GATED	I	CGU	Gated clock of main clock, SPI_CLK. The logic, which generates the serial clock, runs on this clock. To enable the clock gating for this clock its CGU register has to be set.The maximum frequency is 90 MHz.
SPI_PCLK	I	CGU	APB bus clock. The registers, the DMA request circuitry and the interrupt request circuitry run on this clock. Its frequency can be chosen independent of the other clocks, except from the SPI_PCLK_GATED, in this module.
SPI_PCLK_GATED	I	CGU	Gated clock of SPI_PCLK. The logic, like registers, runs on this clock. To enable the clock gating for this clock its CGU register has to be set.
SPI_SCK_OUT	O	Pin	Serial SPI clock out for master mode. This clock is in master mode derived from SPI_CLK(_GATED) with a programmable divider ratio (oversampling ratio). In master mode the frequencies of SPI_SCK and SPI_CLK should satisfy $f_{SPI\_CLK} >= 2 \times f_{SPI\_SCK}$.
SPI_SCK_IN	I	Pin	Serial SPI clock in for slave mode. In slave mode the frequencies of SPI_SCK_IN and SPI_CLK should satisfy $f_{SPI\_CLK} >= 4 \times f_{SPI\_SCK}$.

The clock domain between the APB clock and SPI_CLK is asynchronous, allowing the APB clock frequency to be independent from the SPI clock frequency.

2.1.2 Bus interface

The SPI module has a APB bus connection that is connected to APB Bus 2. Through this bus interface the registers of the module are accessed.

2.1.3 Pin connections

Table 364. SPI pin connections

Pin name	Type (func)	Reset Value	Description
SPI_MOSI	I/O	-	Data output for master and data input for slave. This pin is fed by SPI_RXD port and the output of this pin goes to the SPI_TXD port.
SPI_MISO	I/O	-	Data input for master and data output for slave. This pin is fed by SPI_RXD port and the output of this pin goes to the SPI_TXD port.
SPI_SCK	I/O	0	Serial clock out (master mode, SPI_SCK_OUT port) and in (slave mode, SPI_SCK_IN).
SPI_CS_IN	I	-	Chip select in, which is used in slave mode

Table 364. SPI pin connections

Pin name	Type (func)	Reset Value	Description
SPI_CS_OUT0	O	0	Chip select out for slave 0, which is used in master mode.
SPI_CS_OUT1	O	0	Chip select out for slave 1, which is used in master mode. This pin is multiplexed to mUART_CTS_N.
SPI_CS_OUT2	O	0	Chip select out for slave 2, which is used in master mode. This pin is multiplexed to mUART_RTS_N.

2.1.4 Interrupt request signals

The SPI module has one interrupt request signal to the interrupt controller.

Table 365. Interrupt Request Signals

Name	Type	Description
SPI_INTREQ	O	Combined interrupt request.

2.1.5 Reset signals

The CGU provides two synchronous reset signals to the SPI block: SPI_RST_N, which resets the logic in the SPI_CLK domain and it is low active, and APB_RST_N, which resets the logic in the SPI_PCLK domain of the module and is low active.

These resets should be used at the same time to reset the module.

2.1.6 DMA transfer signals

Table 366. SDMA Signals

Name	Type	Description
DMA_RX_SREQ	O	Receive DMA single transfer request.
DMA_TX_SREQ	O	Transmit DMA single transfer request.

3. Register overview

Table 367. Register overview: SPI (register base adress 0x1500 2000)

Name	R/W	Address Offset	Description
SPI configuration registers			
SPI_CONFIG	R/W	0x000	Configuration register
SLAVE_ENABLE	R/W	0X004	Slave enable register
TX_FIFO_FLUSH	W	0X008	Transmit FIFO flush register
FIFO_DATA	R/W	0x00C	FIFO data register
NHP_POP	W	0x010	NHP pop register
NHP_MODE	R/W	0x014	NHP mode selection register
DMA_SETTINGS	R/W	0x018	DMA settings register
STATUS	R	0x01C	Status register
HW_INFO	R	0x020	Hardware information register
SPI slave registers			
SLV0_SETTINGS1	R/W	0x024	Slave settings register 1 (for slave 0)

Table 367. Register overview: SPI (register base adress 0x1500 2000)

Name	R/W	Address Offset	Description
SLV0_SETTINGS2	R/W	0x028	Slave settings register 2 (for slave 0)
SLV1_SETTINGS1	R/W	0x02C	Slave settings register 1 (for slave 1)
SLV1_SETTINGS2	R/W	0x030	Slave settings register 2 (for slave 1)
SLV2_SETTINGS1	R/W	0x034	Slave settings register 1 (for slave 2)
SLV2_SETTINGS2	R/W	0x038	Slave settings register 2 (for slave 2)
-		0x03C-0xFD0	Reserved
SPI interrupt registers			
INT_THRESHOLD	R/W	0xFD4	Tx/Rx threshold interrupt levels
INT_CLR_ENABLE	W	0xFD8	INT_ENABLE bits clear register
INT_SET_ENABLE	W	0xFDC	INT_ENABLE bits set register
INT_STATUS	R	0xFE0	Interrupt status register
INT_ENABLE	R	0xFE4	Interrupt enable register
INT_CLR_STATUS	W	0xFE8	INT_STATUS bits clear register
INT_SET_STATUS	W	0xFEC	INT_STATUS bits set register
-	-	0xFF0-0xFF8	Reserved

4. Register description

4.1 SPI configuration registers

Table 368. SPI Configuration register (SPI_CONFIG, address 0x1500 2000)

Bit	Symbol	Access	Reset Value	Description
31:16	inter_slave_dly	R/W	0x1	The minimum delay between two transfers to different slaves on the serial interface (measured in clock cycles of the SPI_CLK). The minimum value is 1.
15:8	-	-	0	Reserved
7	update_enable	W		Update enable bit. It must be set by software when the SLAVE_ENABLE register had been programmed. It will be automatically cleared when the new value is in use. 0: the current value in the SLAVE_ENABLEregister is being used for transmission. A new value may be programmed. As soon as update enable is cleared again the new value is used for transmission. 1: the newly programmed value in the SLAVE_ENABLE register is not used for transmission yet. As soon as the value will be used for transmission this bit will clear automatically.In SMS mode the newly programmed value will be used when the pending SMS transfer finishes. In normal transmission mode newly programmed value will be used right away (after some clock domain synchronization delay)
6	Software_reset	R/W	0	Software reset bit. Writing '1' to this bit will reset the block completely. This bit is self clearing.
5	-	-	-	Reserved
4	Slave_disable	R/W	0	Slave output disable is only relevant in slave mode. When multiple slaves are connected to a single chip select signal for broadcasting of a message by a master, only one slave may drive data on its transmit data line (since all transmit data lines of the slaves are tied together to the single master). 0: slave can drive its transmit data output 1: slave must not drive its transmit data output.
3	Transmit_mode	R/W	0	Transmit mode 0: normal mode 1: sequential multi-slave mode. For slave mode this bit must be 0.

Table 368. SPI Configuration register (SPI_CONFIG, address 0x1500 2000) *…continued*

Bit	Symbol	Access	Reset Value	Description
2	Loopback_mode	R/W	0	Loopback mode bit 0: normal serial interface operation 1: transmit data is internally looped-back and is received. Note: when the RX FIFO width is smaller than the TX FIFO width, then the most significant bit of the transmitted data is lost in loopback mode.
1	Ms_mode	R/W	0	master/slave mode bit0: master mode1: slave mode.
0	Spi_enable	R/W	0	SPI enable bit. When this bit is set the module is enabled. Slave mode: If the module is not enabled, it will not accept data from a master or send data to a master. Master mode: If there is data present in the transmit FIFO the module starts transmitting. Before setting this bit, at least one slave should be selected in the SLAVE_ENABLE register. In sequential multi-slave mode this bit is self-clearing. 0: disables SPI 1: enables SPI.

Table 369. Slave Enable register (SLAVE_ENABLE, address 0x1500 2004)

Bit	Symbol	Access	Reset Value	Description
31:6	-	-	0	Reserved
5:0	slave_enable	R/W	0x0	Slave enable bits (bits [1:0] -> slave 1, bits [3:2] -> slave 2, etc.) Per slave 2 bits are used. There are three possible values: 00: the corresponding slave is disabled 01: the corresponding slave is enabled 11: the corresponding slave is suspended (10: not supported). Note: in normal transmission mode only, one slave may be enabled and the others should be disabled. In sequential multi-slave mode more than one slave may be enabled. Slaves can also be suspended, which means they will be skipped during the transfer. This is used to avoid sending data to a slave while there is data in the transmit FIFO for the slave, so skipping data in the transmit FIFO. This register is only relevant in Master Mode.

Table 370. Transmit FIFO flush register (TX_FIFO_FLUSH, address 0x1500 2008)

Bit	Symbol	Access	Reset Value	Description
31:1	-	-	0	Reserved
0	tx_fifo_flush	W	0	Transmit FIFO flush bit. In sequential multi-slave mode the transmit FIFO keeps its data by default. This means the FIFO needs to be flushed before changing the FIFO contents. 1: flush transmit FIFO 0: no action.

Table 371. FIFO data register (FIFO_DATA, address 0x1500 200C)

Bit	Symbol	Access	Reset Value	Description
31:0	fifo_enable	R/W	0	This register is used to access the FIFOs: Writing data puts new data in the transmit FIFO. When the Tx FIFO width is smaller than 32 bits only LSBs written to this register will be put in the FIFO. Reading data reads a word from the receive FIFO. When the Rx FIFO width is smaller than 32 the MSBs read from this register are zero. Note: the NHP registers can change the effect of reading this register.

Table 372. NHP POP register (NHP_POP, address 0x1500 2010)

Bit	Symbol	Access	Reset Value	Description
31:1	-	-	0	Reserved.
0	nhp_pop	W		NHP pop bit Setting this bit will pop the first element from the receive FIFO. This is necessary in NHP mode because reading the FIFO_DATA register will not cause the receive FIFO pointer to be updated in this mode. (to protect the receive FIFO from losing data because of speculative reads). This bit will clear automatically.

Table 373. NHP mode register (NHP_MODE, address 0x1500 2014)

Bit	Symbol	Access	Reset Value	Description
31:1	-	-	-	Reserved
0	nhp_mode	R/W	0	NHP mode bit. Setting this bit puts the SPI in 'NHP mode' and protects the receive FIFO contents from speculative reads. Now a read of the FIFO_DATA register only returns the data from the FIFO, but will not result in updating of the FIFO's read pointer as a side effect. Speculative reads of the FIFO_DATA register will not cause data loss of the received FIFO. After every read of data, the NHP pop register needs to be written, in order to remove the read element from the FIFO, and to point to the next element. Clearing the bit disables the NHP mode. An explicit pop of the receive FIFO is no longer needed. Reading the FIFO_DATA register also updates the receive FIFO's read pointer as a side-effect.

Table 374. DMA setting register (DMA_SETTINGS, address 0x1500 2018)

Bit	Symbol	Access	Reset Value	Description
31:8	-	-	0x0	Reserved.
7:5	-	-	0x0	Reserved.
4:2	-	-	0x0	Reserved.
1	tx_dma_enable	R/W	0	Tx DMA enable bit 1: DMA enabled 0: DMA disabled
0	rx_dma_enable	R/W	0	Rx DMA enable bit 1: DMA enabled 0: DMA disabled.

Table 375. Status register (STATUS, address 0x1500 2018)

Bit	Symbol	Access	Reset Value	Description
31:6	-	-	0	Reserved.
5	Sms_mode_busy	R	0	Sequential multi-slave mode busy flag 1: SPI is currently transmitting in sequential multi-slave mode, once all data to all slaves has been sent, this bit is cleared 0: SPI is not in sequential multi-slave mode or not busy transmitting in this mode.
4	spi_busy	R	0	SPI busy flag 1: SPI is currently transmitting and/or receiving or the transmit FIFO is not empty 0: SPI is idle.
3	rx_fifo_full	R	0	Receive FIFO full bit 1: receive FIFO full 0: receive FIFO not full.

Table 375. Status register (STATUS, address 0x1500 2018) *…continued*

Bit	Symbol	Access	Reset Value	Description
2	rx_fifo_empty	R	1	Receive FIFO empty bit 1: receive FIFO empty 0: receive FIFO not empty.
1	tx_fifo_full	R	0	Transmit FIFO full bit 1: transmit FIFO full 0:transmit FIFO not full.
0	tx_fifo_empty	R	1	Transmit FIFO empty bit 1: transmit FIFO empty 0: transmit FIFO not empty.

Table 376. Hardware information register (HW_INFO, address 0x1500 2020)

Bit	Symbol	Access	Reset Value [decimal]	Description
31	-	-	0	Reserved.
30	fifoimpl	R	0	For software usage: The FIFO memory implementation, 0=flipflops 1=SRAM.
29:26	num_slaves	R	3	For software usage: The maximum number of slaves supported by this hardware configuration (minus 1 encoded).
25:21	tx_fifo_width	R	16	For software usage: The width of the transmit FIFO of this hardware configuration (minus 1 encoded).
20:16	rx_fifo_width	R	16	For software usage: The width of the receive FIFO of this hardware configuration (minus 1 encoded).
15:8	tx_fifo_depth	R	64	
7:0	rx_fifo_depth	R	64	

4.2 SPI slave registers

Table 377. Slave settings 1 (SLV0_SETTINGS1, address 0x1500 2024; SLV1_SETTINGS1, address 0x1500 202C; SLV2_SETTINGS1, address 0x1500 2034)

Bit	Symbol	Access	Reset Value	Description
31:24	inter_transfer_dly	R/W	0x0	The delay between transfers to this slave measured in serial clock cycles. This delay is minimal 0 serial clock cycles (SPI_SCK_OUT). This field is only relevant in master mode.
23:16	number_words	R/W	0x0	Number of words to send in sequential multi-slave mode. After this number of words have been transmitted to this slave the master starts transmitting to the next slave. If the sequential multi-slave mode is disabled this field is not used. (minus 1 encoded). This field is only relevant in master mode.
15:8	clk_divisor2	R/W	0x2	Serial clock rate divisor 2: A value from 2 to 254 (lsb bit is hard-coded 0).
7:0	clk_divisor1	R/W	0x0	Serial clock rate divisor 1: A value from 0 to 255.

The serial clock frequency is derived from the IP clock frequency using the values, which are programmed in the clk_divisor1 and clk_divisor2 fields:

Table 378. Slave settings 2 (SLV0_SETTINGS2, address 0x1500 2028; SLV1_SETTINGS2, address 0x1500 2030; SLV2_SETTINGS2, address 0x1500 2038)

Bit	Symbol	Access	Reset Value	Description
31:17	-	-	0	Reserved.
16:9	pre_post_cs_dly	R/W	0	Programmable delay that occurs twice in a transfer and is present between assertion of the chip select and transfer (sampling) of the first data bit AND between the transfer of the last data bit and de-assertion of chip select.The minimum delay is one serial clock cycle (SPI_SCK_OUT). This register is minus one encoded (0 gives a one cycle delay). This field is only relevant in SPI master mode.
8	cs_value	R/W	0	Chip select value between back-to-back transfers selection bit. 1: chip select has a steady state high value between transfers 0: chip select has a steady state low value between transfers The period in which the chip select has this value is programmed in the inter_transfer_dly field of the SLVx_SETTINGS1 registered. This field is only relevant in SPI master mode.

Table 378. Slave settings 2 (SLV0_SETTINGS2, address 0x1500 2028; SLV1_SETTINGS2, address 0x1500 2030; SLV2_SETTINGS2, address 0x1500 2038)

Bit	Symbol	Access	Reset Value	Description
7	transfer_format	R/W	0	Format of transfer 0: SPI format 1: SSI format.
6	spo	R/W	0	Serial clock polarity (only used if SPI mode is selected) 1: the serial clock has a steady state high value between transfers 0: the serial clock has a steady state low value between transfers.
5	sph	R/W	0	Serial clock phase (only used if SPI mode is selected). Determines on which edges of the serial clock data is captured during transfers. 1: first data bit is captured on the second clock edge transition of a new transfer 0: first data bit is captured on the first clock edge transition of a new transfer
4:0	word size	R/W	0x0	Word size of transfers to this slave (minus 1). SPI mode: 8/16 bits supported SSI mode: 4 ... 16 bits supported

- $F_{SPI\_SCK} = F_{SPI\_CLK}/(clkdivisor2 + (1 + clkdivisor1))$

However, in slave mode this formula does not count. The SPI_CLK may never be smaller than 4 times serial clock. In case of an oversampling ratio of 4 this means that the SPI_TXD transitions occur at the correct instant or at most one clock period of SPI_CLK earlier. However, for higher oversampling ratios the transitions occur too early (in a range from oversampling ratio/2-2 to oversampling ratio/2-1 clock cycles of the slave's SPI_CLK).

4.3 SPI interrupt registers

Table 379. Interrupt threshold register (INT_THRESHOLD, address 0x1500 2FD4)

Bit	Symbol	Access	Reset Value	Description
31:16	-	-	0	Reserved.
15:8	tx_threshold	R/W	0	A transmit threshold level interrupt is requested when the transmit FIFO contains less than this number of elements. When the value is higher than the FIFO size the behaviour of the threshold interrupt is undefined.
7:0	rx_threshold	R/W	0	A receive threshold level interrupt is requested when the receive FIFO contains more than this number of elements. When the value is higher than the FIFO size the behaviour of the threshold interrupt is undefined.

Table 380. Interrupt clear enable register (INT_CLR_ENABLE, address 0x1500 2FD8)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	0	Reserved.
4	clr_sms_int_enable	W	0	Writing '1' clears sequential multi-slave mode ready interrupt bit in the INT_ENABLE register.
3	clr_tx_int_enable	W	0	Writing '1' clears transmit threshold level interrupt bit in the INT_ENABLE register.
2	clr_rx_int_enable	W		Writing '1' clears receive threshold level interrupt bit in the INT_ENABLE register.
1	-	-	0	Reserved.
0	clr_ov_int_enable	W	0	Writing '1' clears receive overrun interrupt bit in the INT_ENABLE register.

Table 381. Interrupt set enable register (INT_SET_ENABLE, address 0x1500 2FDC)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	0	Reserved.
4	set_sms_int_enable	W	0	Writing '1' sets sequential multi-slave mode ready interrupt bit in the INT_ENABLE register.
3	set_tx_int_enable	W	0	Writing '1' sets transmit threshold level interrupt bit in the INT_ENABLE register.
2	set_rx_int_enable	W		Writing '1' sets receive threshold level interrupt bit in the INT_ENABLE register.
1	-	-	0	Reserved.
0	set_ov_int_enable	W	0	Writing '1' sets receive FIFO overrun interrupt bit in the INT_ENABLE register.

Table 382. Interrupt status register (INT_STATUS, address 0x1500 2FE0)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	0	Reserved.
4	sms_int_status	R	0	Sequential multi-slave mode ready interrupt status.
3	tx_int_status	R	0	Transmit threshold level interrupt status.
2	rx_int_status	R		Receive threshold level interrupt status.
1	-	-	0	Reserved.
0	ov_int_status	R	0	Receive FIFO overrun interrupt status.

Table 383. Interrupt enable register (INT_ENABLE, address 0x1500 2FE4)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	0	Reserved.
4	sms_int_enable	R	0	Sequential multi-slave mode ready interrupt enable.
3	tx_int_enable	R	0	Transmit threshold level interrupt enable.

Table 383. Interrupt enable register (INT_ENABLE, address 0x1500 2FE4)

Bit	Symbol	Access	Reset Value	Description
2	rx_int_enable	R		Receive threshold level interrupt enable.
1	-	-	0	Reserved.
0	tx_int_enable		0	Receive FIFO overrun interrupt enable.

Table 384. Interrupt clear status register (INT_CLR_STATUS, address 0x1500 2FE8)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	0	Reserved.
4	clr_sms_int_status	R	0	Writing '1' clears sequential multi-slave mode ready interrupt bit in the INT_STATUS register.
3	clr_tx_int_status	R	0	Writing '1' clears transmit threshold level interrupt bit in the INT_STATUS register.
2	clr_rx_int_status	R		Writing '1' clears receive threshold level interrupt bit in the INT_STATUS register.
1	-	-	0	Reserved.
0	clr_ov_int_status		0	Writing '1' clears receive FIFO overrun interrupt bit in the INT_STATUS register.

Table 385. Interrupt set status register (INT_SET_STATUS, address 0x1500 2FEC)

Bit	Symbol	Access	Reset Value	Description
31:5	-	-	0	Reserved.
4	set_sms_int_status	W	0	Writing '1' sets sequential multi-slave mode ready interrupt bit in the INT_STATUS register.
3	set_tx_int_status	W	0	Writing '1' sets transmit threshold level interrupt bit in the INT_STATUS register.
2	set_rx_int_status	W		Writing '1' sets receive threshold level interrupt bit in the INT_STATUS register.
1	-	-	0	Reserved.
0	set_ov_int_status	W	0	Writing '1' sets receive FIFO overrun interrupt bit in the INT_STATUS register.

5. Functional Description

The SPI module is a master or slave interface for synchronous serial communication with peripheral devices that have either Motorola SPI or Texas Instruments synchronous serial interfaces (SSI).

The SPI module performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information through the APB interface. The transmit and receive paths are buffered with FIFO memories. Serial data is transmitted on SPI_TXD and received on SPI_RXD.

The SPI module includes a programmable bit rate clock divider and prescaler to generate the serial output clock SPI_SCK_OUT from the input clock SPI_CLK. The SPI operating mode, frame format, and word size are programmed through the SLVx_SETTINGS registers.

A single combined interrupt request SPI_INTREQ output is asserted if any of the interrupts are asserted and unmasked. All five interrupts also have a separate interrupt request line.

A set of DMA signals is provided for interfacing with a DMA controller.

Depending on the operating mode selected, the SPI_CS_OUT outputs operate as an active HIGH frame synchronization output for Texas Instruments synchronous serial frame format or an active LOW chip select for SPI.

5.1 Formats

Each data frame is between 4 and 16 bits long depending on the size of words

programmed, and is transmitted starting with the MSB. There are two basic frame types that can be selected:

- Texas Instruments synchronous serial (SSI)
- Motorola Serial Peripheral Interface (SPI).

For these formats, the serial clock (SPI_SCK_OUT) is held inactive while the SPI module is idle, and transitions at the programmed frequency only during active transmission or reception of data.

For Motorola SPI, the chip select pin (SPI_CS_OUT) is active low, and is asserted during the entire transmission of the frame.

For Texas Instruments SSI, the chip select pin (SPI_CS_OUT) is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SPI module and the off-chip slave device drive their output data on the rising edge of the serial clock pin, and latch data from the other device on the falling edge.

The next sections describe the frame formats in more detail. Note that in this sections ’delay1’ is used to show the delay programmed in the pre_post_cs_dly field of the SLVx_SETTINGS2 register and “delay2” is used to show the delay programmed in the inter_slave_dly field of the SLVx_SETTINGS1 register.

5.1.1 SSI Format

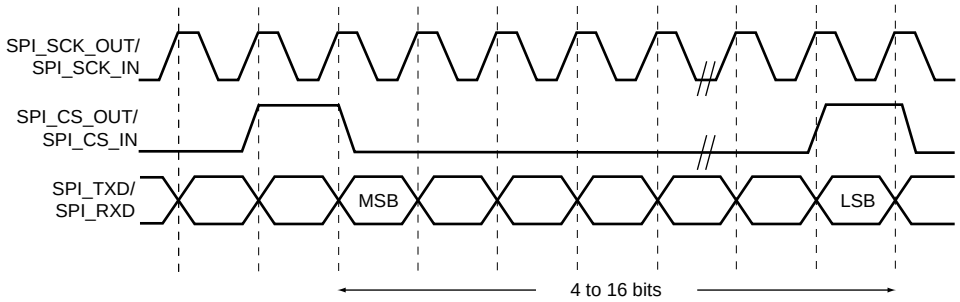


Fig 48. SSI frame format (single Transfer)

In this mode, the serial clock (SPI_SCK_OUT) and the chip select (SPI_CS_OUT) are forced LOW, and the transmission data line SPI_TXD (in slave mode: SPI_MISO, in master mode: SPI_MOSI) tri-stated whenever the SPI module is idle. Once the bottom entry of the transmit FIFO contains data, SPI_CS_OUT is pulsed HIGH for one SPI_SCK_OUT period. The data to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SPI_SCK_OUT, the MSB of the 4 to 16-bit data frame is shifted out on the SPI_TXD pin. Likewise, the MSB of the received data is shifted onto the SPI_RXD pin by the off-chip serial slave device.

Both the SPI module and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SPI_SCK_OUT. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SPI_SCK_OUT after the LSB has been latched.

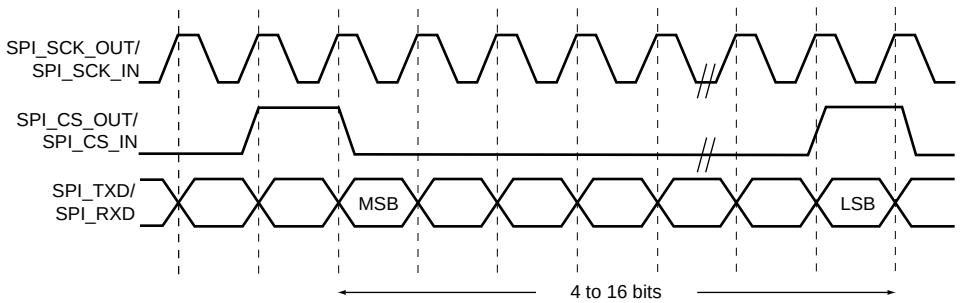


Fig 49. SSI frame format (back to back Transfer)

The inter_transfer_dly field in the SLVx_SETTINGS1 specifies the delay between back-to-back SSI transfers to the same slave. In Figure 19–49 this delay is zero cycles, the SPI_CS_OUT signal is asserted for signalling the beginning of the next transfer in the same cycle as the last bit of the previous transfer is transmitted. When the delay is programmed to be higher than 0, extra delay cycles are added before the next transfer will be started. During these delay cycles the SPI_SCK_OUT signal is low.

When a new slave is selected, there is a delay between the last transfer to the previous slave and the first transfer to the next slave. This delay is programmed in the inter_slave_dly field of the SPI_CONFIG register. In slave mode only zeros are transmitted in case of a FIFO under run.

5.1.2 SPI Format

The Motorola SPI interface is a four-wire interface where the SPI_CS_OUT signal behaves as a chip select. The main feature of the Motorola SPI format is that the inactive state and phase of the SPI_SCK_OUT signal are programmable through the SPO and SPH bits within the slave settings registers.

SPO clock polarity

When the SPO clock polarity control bit is LOW, it produces a steady state low value on the SPI_SCK_OUT pin. If the SPO clock polarity control bit is HIGH, a steady state high value is placed on the SPI_SCK_OUT pin when data is not being transferred.

SPH clock phase

The SPH control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.

When the SPH phase control bit is LOW, data is captured on the first clock edge transition. If the SPH clock phase control bit is HIGH, data is captured on the second clock edge transition.

The values of these bits determine the 4 modes of the SPI:

Table 386. External Signals

Mode	SPO	SPH
0 (00)	0	0
1 (01)	0	1
2 (10)	1	0
3 (11)	1	1

Single and continuous transmission signal sequences for Motorola SPI format in mode 0,1,2 and 3 are shown in Figure 19–49 to Figure 19–57 inclusive.

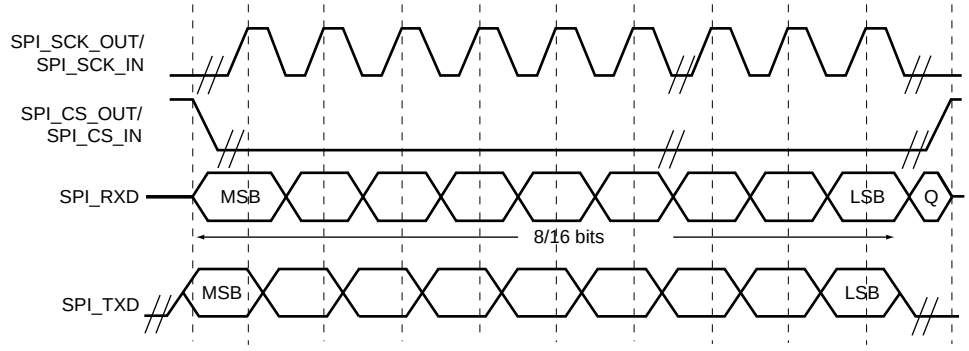


Fig 50. SPI frame format in Mode 0 (Single Transfer)

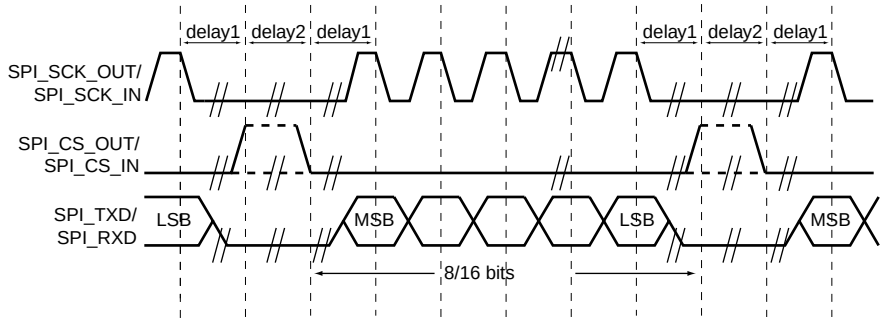


Fig 51. SPI format in Mode 0 (Back to Back Transfer)

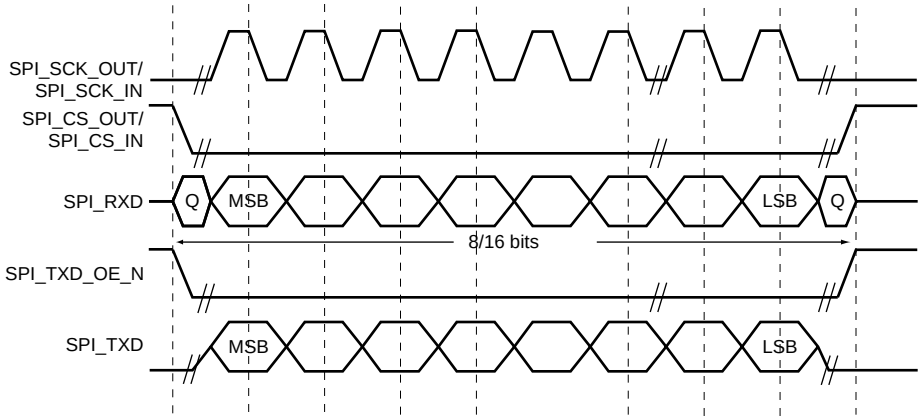


Fig 52. SPI Format in Mode 1 (Single Transfer)

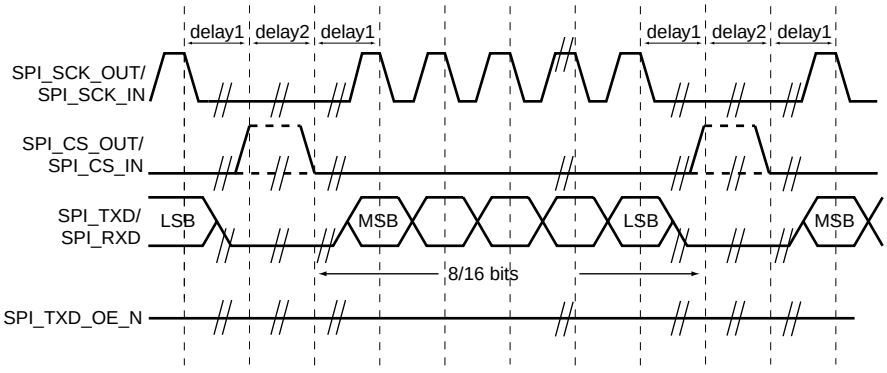


Fig 53. SPI Format in Mode 1 (Back to Back)

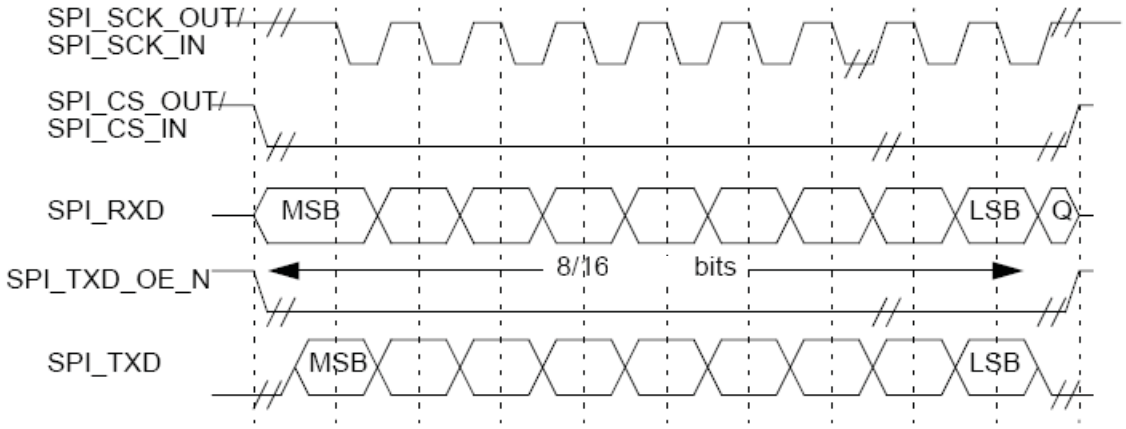


Fig 54. SPI Format in Mode 2 (Single Transfer)

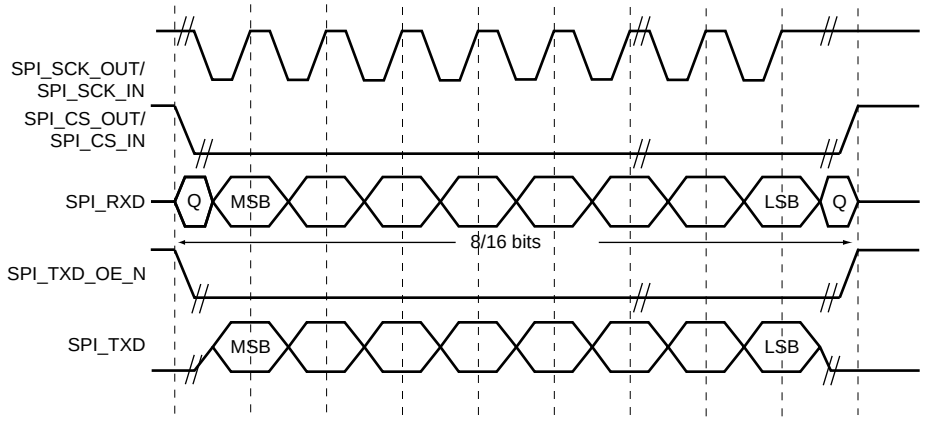


Fig 55. SPI Format in Mode 2 (Back to Back Transfer)

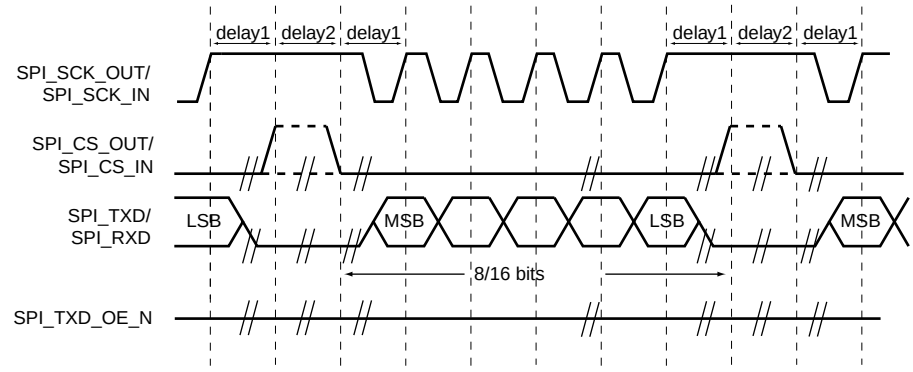


Fig 56. SPI Format in Mode 3 (Back to Back Transfer)

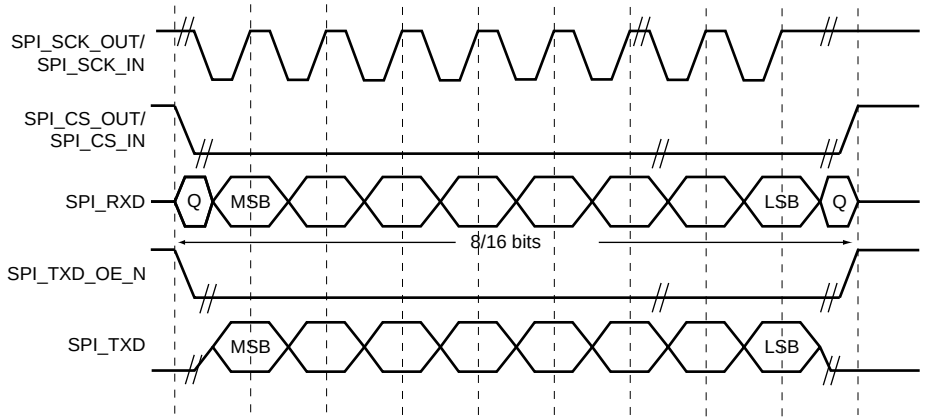


Fig 57. SPI Format in Mode 3 (Single Transfer)

During idle periods:

- The SPI_SCK_OUT signal is forced low for mode 0 and 1. This signal is forced high for mode 2 and 3.
- SPI_CS_OUT is forced HIGH if no more data is present in the transmit FIFO. In case more data is to be transmitted (back-to-back transfer), the signal can be either forced high or low depending on a register setting which can be programmed differently for every slave
- The transmit data line SPI_TXD is arbitrarily forced LOW
- The SPI_TXD_OE_N pad enable signal is forced HIGH, making the transmit pad high impedance
- When the SPI module is configured as a master, the SPI_SCK_OE_N is driven LOW, enabling the SPI_SCK_OUT pad (active LOW enable)
- When the module is configured as a slave, the SPI_SCK_OE_N line is driven HIGH, disabling the SPI_SCK_OUT pad (active LOW enable).

If the SPI module is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPI_CS_OUT master signal being driven LOW. This causes slave (which is enabled by the SLAVE_ENABLE register) data to be enabled onto the SPI_RXD line of the master. The SPI_TXD_OE_N line is driven LOW, enabling the master SPI_TXD output pad. After a minimum of half a SPI_SCK_OUT period, valid master data is transferred to the SPI_TXD pin. Now that both the master and the slave data have been set, the SPI_SCK_OUT master clock pin becomes HIGH (mode 0,1)/LOW (mode 2, 3) after half a SPI_SCK_OUT period (= delay1). This delay is programmable and can be programmed for every slave differently. In this delay period no positive edges of the SPI_SCK_OUT clock signal occur. The same delay is applied after the last SPI_SCK_OUT positive edge of this transfer and possibly de-asserting of the SPI_CS_OUT signal.

For mode 0 and 2 the data is captured on the rising and propagated on the falling edges of the SPI_SCK_OUT signal. This is repeated 8 or 16 times depending on the programmed word size.

For mode 1 and 3 the data is captured on the falling and propagated on the rising edges of the SPI_SCK_OUT signal. This is repeated 8 or 16 times depending on the programmed word size.

In case of a single word transmission, after all bits of the data word have been transferred, the SPI_CS_OUT signal line is returned to its idle HIGH state after a minimum delay of 1 master clock period. This delay is programmable per slave and is equal to the delay between assertion of the SPI_CS_OUT signal and the first positive edge of the SPI_SCK_OUT clock signal.

In case of continuous back-to-back transmissions, the SPI_CS_OUT signal may (after the before mentioned delay of minimal 1 period) be asserted between transfers or not. Delay 2 shows the period in which there might be a SPI_CS_OUT pulse or not. This delay is minimal zero SPI_SCK_OUT master clock cycles and is programmable for every slave separately.

On completion of a back-to-back transfer, the SPI_CS_OUT pin is returned to its idle state after the minimal delay of one SPI_SCK_OUT period.

If after a transfer, another slave is selected, there is a programmable delay in SPI_SCK_OUT cycles between de-assertion of the chip select of the previous slave, and assertion of the chip select of the new slave.

In slave mode, only zeros are transmitted in case of a FIFO under run.

5.2 Operation Modes

The module (in master mode) supports two modes of operation. One is the normal transmission mode in which software intervention is needed every time a new slave needs to be addressed and some interrupt handling.

The second operation mode is the sequential multi-slave mode. This mode reduces software intervention and interrupt load. Another advantage is that the data in the transmit FIFO can be transmitted again without the need of re-filling the FIFO, which reduces bus load.

5.2.1 Sequential Multi-Slave Mode

In this mode it is possible to sequentially transmit data to different slaves without having to re-program the module between transfers to different slaves. The purpose of this mode is to minimize interrupts/software intervention and bus traffic. This mode is only applicable when the module is in master mode.

In the example in Figure 19–58 the module supports addressing of four slaves. All four are sent data in sequential multi-slave mode. Three elements are transferred to slave 1, two to slave 2, three to slave 3 and finally one to slave 4. Now the module disables itself. When it is enabled again the same data is transmitted to the four slaves.

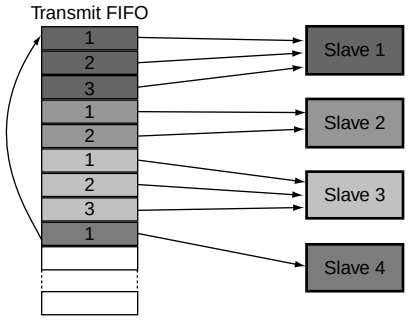


Fig 58. Sequential multi-slave mode (General Explanation)

Before entering this mode the transmit data needs to be present in the transmit FIFO. No data may be added after entering sequential multi-slave mode. When the data to be transferred needs to be changed the transmit FIFO must be flushed and sequential multi-slave mode is left and entered again to 'save' the new data present in the transmit FIFO. This is necessary because the FIFO contents are saved as a side-effect of entering sequential multi-slave mode from normal transmission mode. The data in the transmit FIFO is saved to allow transmitting it multiple times without the need to re-fill it with the same data.

All programming of the settings necessary to adapt to all slaves has to be done before enabling (starting the transfer) the module in sequential multi-slave mode. Once a transfer has started these settings may not be changed until the module has finished the transfer and is automatically disabled again. You can select only one slave in this mode.

Once a sequential multi-slave mode transfer has started it finishes completely, even if the module gets disabled before the transfer is over. When a transfer is finished, the module disables itself and requests a sequential multi-slave mode ready interrupt.

You can temporarily suspend/skip one or more of the slaves in a transfer. The data in the transmit FIFO does not need to be flushed. During the transfer this data will be skipped and on the serial interface nothing happens for the exact time that would have been used by transferring to the skipped slave. In the receive FIFO dummy (zero filled) words are written. The number of dummy words is equal to the number of words that would have been received by the suspended slave. When suspending slaves it is important to keep the corresponding SLVx_SETTINGS. The numbers_words field is necessary to skip the data for this slave. The other settings are needed to create the delay of the suspended transfer on the serial interface. Suspending a slave does not change anything for the duration of a sequential multi-slave transfer.

A slave can also be completely disabled. The transmit FIFO may not hold any data for this slave, which means the transmit FIFO may need to be flushed and re-programmed. The SLVx_SETTINGS for a disabled slave are ignored.

Setting up the sequentially multi-slave mode transfer:

- Programming the settings for transmitting to different slaves. Every slave can have different settings like SPI/TI mode, different word size or baud rates. The registers SLVx_SETTINGS1 and SLVx_SETTINGS2 are duplicated for the number of slaves

that are supported by the module. In SLVx_SETTINGS1 also the number of words to transmit to this slave must be programmed. These settings may not be changed during a sequential multi-slave mode transfer.

- Writing the transmit data to the transmit FIFO. All data for all slaves must be present before entering sequential multi-slave mode. The contents of the transmit FIFO is 'saved' when entering this mode to allow sending the same data multiple times without having to re-fill the FIFO with the same data. If the data in the transmit FIFO needs to be changed the FIFO first needs to be flushed using the TX_FIFO_FLUSH register. Note that when changing the slaves that are enabled/disabled or the number of elements to send to each slave, the transmit FIFO contents must be changed. After writing new data to the transmit FIFO the multi-sequence mode is left and entered again to 'save' the new data.
- Enabling/disabling/suspending slaves. In sequential multi-slave mode order of transmission is fixed. First slave 1 is addressed, then slave 2, until the last slave is reached. By programming the SLAVE_ENABLE register it can be specified which slave to address and which slaves to skip. Writing a '01' enables the corresponding slave and a '00' disables the corresponding slave. Note that when a slave is disabled, no data for this slave must be written to the transmit FIFO. The SLVx_SETTINGS register settings for a disabled slave are ignored. Writing '11' suspends/skips the corresponding slave. The data for this slave is still present in the transmit FIFO, is skipped. The SLVx_SETTINGS register settings for a suspended slave are still used to skip the data in the FIFO for this slave and to create a delay on the serial interface equal to the delay that would be used for transmitting to the suspended/skipped slave. Because a new value is written to the SLAVE_ENABLE register the update_enable bit in the SPI_CONFIG register must be set.
- Enabling/disabling DMA. The transmit DMA requests must be disabled. All transmit data should be present before enabling the SPI module so transmit DMA requests are not necessary. The receive DMA request can be enabled. But the receive FIFO can also be read only after the sequential multi-slave mode has finished, which requires the receive DMA request to be disabled.
- Enabling the sequential multi-slave mode ready interrupt request. This interrupt signals the end of a sequential multi-slave transfer. After a transfer the receive FIFO needs to be read, before a new transfer starts. If the new transfer can start before the receive FIFO is empty, the receive FIFO needs to be sized as such.
- Setting the transmit mode bit (SPI_CONFIG register). Setting this bit 'saves' the transmit FIFO contents (so it can be transmitted multiple times without re-filling) and enter the sequential multi-slave mode. This prevents FIFO overruns.
- Enabling the MODULE. Once the SPI module is enabled in sequential multi-slave mode it sends its transmit data to the slaves that are enabled. It will automatically re-program its settings if needed to adapt to different slaves. When all data has been transmitted, the sequential multi-slave ready interrupt is requested. The update_enable bit in the SPI_CONFIG register is cleared to allow re-programming of the SLAVE_ENABLE register. Now the receive FIFO needs to be emptied to allow a new transfer to be started. The SPI module disables itself and wait until it is enabled again to start a new transfer. After it is enabled again, it transmits the same data, unless the transmit FIFO is filled with new data after flushing the FIFO.
- Setting the enable bit in the SPI_CONFIG register enables the module in sequential multi-slave mode. If the sequential multi-slave mode needs to be left, the transmit mode bit in the SPI_CONFIG register has to be cleared. Also the transmit FIFO needs

to be flushed to allow its contents to be changed. If not in sequential multi-slave mode, only one slave may be enabled in the SLAVE_ENABLE register. Once the sequential multi-slave mode has been left, the module is in normal transmission mode.

5.2.2 Normal Transmission Mode

When the module is used as a master and is not programmed to be in sequential multi-slave mode, it is in normal transmission mode. In this mode software programs the settings of the SPI module, writes data to the transmit FIFO and then enables the module. The module transmits until all data has been sent or until it gets disabled before all data has been sent. When data needs to be transmitted to another slave, software needs to re-program the settings of the module, write new data and enable the module again. This mode requires software intervention every time a new slave needs to be addressed and more interrupt handling.

Remark: When re-programming any of the settings the module must be disabled first.

After changing the settings it can be enabled again. Transmit data can also be added when the module is still enabled, disabling is not necessary in this case.

Setting up a normal transmission mode transfer:

- Programming of the settings necessary to adapt to the slave. In the SLVx_SETTINGS1 and SLVx_SETTINGS2 register the settings for every slave can be programmed. The registers corresponding to the slave to be addressed will need to be programmed. The ’number_words’ field of SLV_REGISTER1 is not needed in this mode of transmission and is ignored.
- Enabling the slave to be addressed in the SLAVE_ENABLE register. The bits in this register corresponding to the slave to be addressed needs to be set to ’01’. Note that in normal transmission mode only one slave may be enabled. Setting more than one bit will result in undefined behavior. After programming the SLAVE_ENABLE register the update_enable bit in the SPI_CONFIG register needs to be set.
- Enabling the module by setting the enable bit in the SPI_CONFIG register.
- Writing data to the transmit FIFO. Once data is in the transmit FIFO and the module is enabled the module starts transmitting. Transmit data may also be written to the transmit FIFO before enabling the module.
- When a new slave needs to be addressed the module needs to be disabled. The SLVx_SETTINGS registers corresponding to the new slave must be programmed and the new slave must be enabled (and previous slave disabled) in the SLAVE_ENABLE register. If the module is disabled during a pending transfer the data being transferred is lost.

5.2.3 Slave Mode

The module can be used in slave mode by setting the ’ms_mode’ bit in the SPI_CONFIG register. The settings of the slave can be programmed in the SLV1_SETTINGS registers that would correspond to slave 1 (offsets 0x024 and 0x028). A slave must be programmed to be in normal transmission mode. SLAVE_ENABLE register is ignored in slave mode.

6. Power optimization

The SPI module has an asynchronous clock domain crossing allowing the APB clock frequency to be independent from the IP clock frequency. This allows power saving by lowering the APB bus frequency while receiving and transmitting on the serial interface with the same unchanged frequency.

For more power saving the oversampling ratio has to be set as low as possible to lower the SPI_CLK frequency.

The module has clock gating. The gated clocks are requested when necessary. They are requested as long as there is data in the transmit FIFO, or the serial interface control blocks remain busy. Setting the external enabling bit of PCR CGU registers of these clocks enables their clock gating.

7. Programming guide

To set-up a normal transmission the following registers have to be programmed.

- Program the settings in the SLVx_SETTINGS1 and SLVx_SETTINGS2
- Enable the slave to be addressed in the SLAVE_ENABLE register. When the module is slave the value 0x01 has to be written to this register. The number of the enabled slave determines which SLVx_SETTINGS1 and SLVx_SETTINGS2 are invoked
- Program the module as slave or master by (re-)setting the ms_mode bit in the SPI_CONFIG register.
- To start the data transmission directly when data is available in the FIFO, enable the module by setting the enable bit in the SPI_CONFIG register.
- Write data to the FIFO_DATA register. Once data in the FIFO, the data is transmitted if the module is the master.
- When a new slave has to be addressed the module must be disabled and the previous programming has to be re-done.

UM10314

Chapter 20: LPC3130/31 Memory Card Interface (MCI)

Rev. 1 — 4 March 2009 User manual

1. Introduction

MCI is an interface between the AHB and the memory card. It supports Secure Digital memory (SD Mem), Secure Digital I/O (SDIO), Multimedia Cards (MMC), and Consumer Electronics Advanced Transport Architecture (CE-ATA).

1.1 Features

This module has the following features:

- Supports Secure Digital memory protocol commands.
- Supports Secure Digital I/O protocol commands.
- Supports Multimedia Card protocol commands.
- Supports CE-ATA digital protocol commands.
- Supports Command Completion signal and interrupt to processor.
- Command Completion Signal disable feature.
- Supports 1 SD or MMC (3.3 or 4.0) or CE-ATA device.
- Supports CRC generation and error detection.
- Provides individual clock control to selectively turn ON or OFF clock to a card.
- Supports SDIO interrupts in 1-bit and 4-bit modes.
- Supports SDIO suspend and resume operation.
- Supports SDIO read wait.
- Supports block size of 1 to 65,535 bytes.
- Supports FIFO over-run and under-run prevention by stopping card clock.
- Supports little-endian mode of AHB operation.
- Support DMA access.
- Supports 2 FIFOs, TX and RX FIFO (FIFO depth = 32 and FIFO data width = 32 bits).

2. General description

2.1 Block diagram

The module consists of the following main functional blocks, which are illustrated in the figure below.

- Bus Interface Unit (BIU) - Provides AHB and DMA interfaces for register and data read/writes.
- Card Interface Unit (CIU) - Takes care of the card protocols and provides clock management.

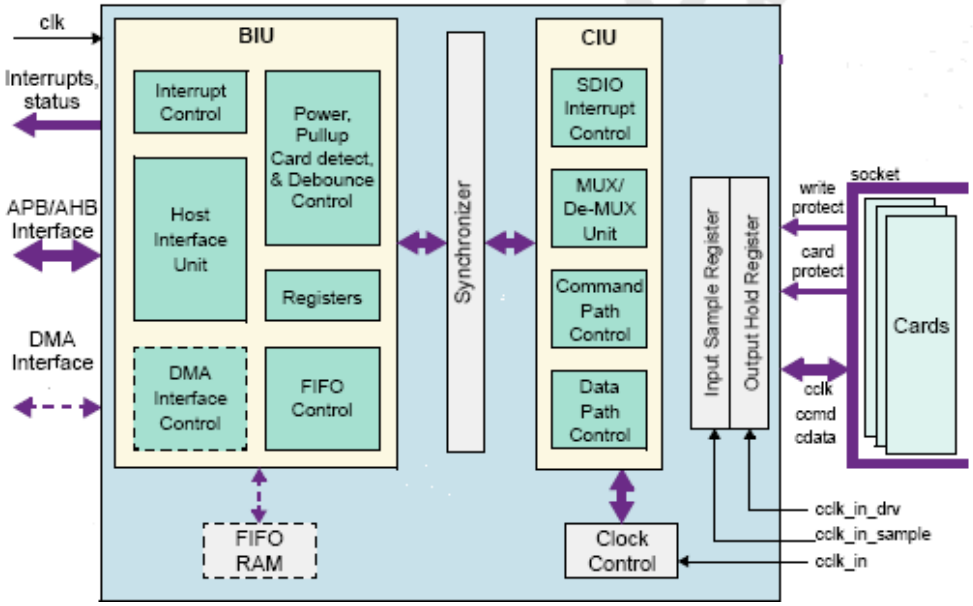


Fig 59. Block diagram

2.2 Interface description

2.2.1 Clock signals

Table 387. Clock signals of the MCI module

Clock name	Acronym	I/O	Source/ destination	Description
SD_MMC_HCLK	CLK	I	CGU	AHB interface clock of the module. The AHB interface logic in this module runs on this clock. The maximum frequency of AHB_CLK is 75MHz. This clock is synchronous to cclk_in.
SD_MMC_CCLK_IN	CCLK_IN	I	CGU	Card interface input clock. The card interface input clock and CLK frequencies should meet a “CLK>=1/10*CCLK_IN” requirement. The maximum operating frequency of a SD card is 25 MHz, and a MMC-Ver4.0 card is 26 MHz or 52 MHz. This clock is synchronous to clk.
SD_MMC_CCLK_IN_SAMPLE	CCLK_IN_SAMPLE	I	MCI_CLK (PAD)	Delayed CCLK_IN to sample card inputs. Card input sampling clock is used for clocking card inputs. Since card inputs are guaranteed to meet only setup-and-hold time with regard to card output clock on its PAD, delayed version of cclk_in is needed to sample all card inputs. The cclk_in to cclk_in_sample delay should match cclk_in to cclk_out delay of core, plus cclk_out PAD delay, plus data/cmd pad input PAD delay. (MMC cards required to provide 3ns setup and 3ns hold on inputs; SD cards required to provide 5ns setup and 5ns hold on all inputs). Therefore the cclk_in_sample is derived from the cclk_out. The delay in clock is configured using SYSCREG_MCI_DELAYMODES register SYSCREG block.
SD_MMC_CCLK_IN_DRV	CCLK_IN_DRV	I	MCI_CLK (pad)	Delayed CCLK_OUT to drive card outputs. Card outputs driving clock is used for clocking optional hold-time registers. The delay should be ~5ns of cclk_out for SD cards, 2 ns for high-speed SD cards and 3 ns for (H)MMC and CE-ATA. (Cards need x ns holds on all inputs, so all outputs clocked out of cclk_in are re-clocked by cclk_in_drv to meet card hold-time requirement.). The cclk_in_drv is derived from the cclk_out.
SD_MMC_CCLK_OUT	CCLK_OUT	O	MCI_CLK (PAD)	Card clock. This clock is the input clock of the cards, SD, (H)MMC and CE-ATA.

2.2.2 Bus interface

The MCI is connected to the AHB bus.

2.2.3 Pin connections

Table 388. External signals of the MCI module

Pin name	MCI pin name[1]	Type	Reset value	Description
mGPIO5	MCI_CLK	I/O		Card clock and used as input for cclk_in_sample and cclk_in_drv.
mGPIO6	MCI_CMD	I/O	-	Card command in-/output.
mGPIO7	MCI_DAT_0	I/O	-	Card data in-/output.
mGPIO8	MCI_DAT_1	I/O	-	Card data in-/output.
mGPIO9	MCI_DAT_2	I/O	-	Card data in-/output.
mGPIO10	MCI_DAT_3	I/O	-	Card data in-/output.
mNAND_RYBN0	MCI_DAT_4	I/O	-	Card data in-/output.
mNAND_RYBN1	MCI_DAT_5	I/O	-	Card data in-/output.
mNAND_RYBN2	MCI_DAT_6	I/O	-	Card data in-/output.
mNAND_RYBN3	MCI_DAT_7	I/O	-	Card data in-/output.

[1] The MCI pins are multiplexed with GPIO and NAND flash pins.

2.2.4 Interrupt request signals

Table 389. Interrupt request signals of the MCI module

Name	Type	Description
INTRRUPT	0	Combined active-high, level-sensitive CPU interrupt

2.2.5 Reset signals

The CGU provides a system active-low reset pin; synchronous to clk. Should be kept active at least 12 clocks or cclk_in, whichever is lower frequency. The reset_n is the result of OR function of sd_mmc_pnres and sd_mmc_nres_cclk_in. The reset signal sd_mmc_pnres is a low active synchronous reset to clk. The reset signal sd_mmc_nres_cclk_in is a low active synchronous reset to cclk_in.

2.2.6 DMA transfer signals

Table 390. DMA signals of the MCI module

Name	Type	Description
SD_MMC__DMA_REQ	O	DMA

2.2.7 System control register (Syscreg) signals

Table 391. Number of delay cells in the MCI (delay) module (see Table 26–523)

Name	Type	Description
MCI_DELAYMODES[4:0]	I	This bus-signal specifies the number of delay cells to obtain the needed delay for cclk_in_drv. The delay should be ~5ns in comparison to cclk_out for SD cards, 2 ns for high-speed SD cards and 3 ns for (H)MMC and CE-ATA. (Cards need x ns holds on all inputs, so all outputs clocked out of cclk_in are re-clocked by cclk_in_drv to meet card hold-time requirement.).

Table 392. Configuration signals of MCI module (see Table 26–522)

Name	Type	Description
CARD_WRITE_PRT	I	Card write protect signal for SD cards. A 1 represents write is protected. Default is zero. Software must honor write-protect. This signal is set by setting the BIT[0] in SYSCREG_SD_MMC_CFG configuration register in SYSCREG module. Software should program this bit by detecting the event from unused external GPIO pin. So that the SD_MMC module responds to the event.
CARD_DETECT_N	I	Card detect signal. A 0 represents presence of card. Default is one. Any change in this signal will cause card_detect interrupt, if enabled. This signal is set by setting the BIT[1] in SYSCREG_SD_MMC_CFG configuration register in SYSCREG module. Software should program this bit by detecting the event from unused external GPIO pin. So that the SD_MMC module responds to the event.

3. Register overview

Table 393. Register overview: MCI (register base address 0x1800 0000)

Name	R/W	Address offset	Description	Reset value
CTRL	R/W	0x000	Control register.	0x0
PWREN	R/W	0X004	reserved	-
CLKDIV	R/W	0X008	Clock-divider register	0x0
CLKSRC	R/W	0x00C	Clock-source register	0x0
CLKENA	R/W	0x010	Clock-enable register	0x0
TMOUT	R/W	0x014	Time-out register (number of card clock output clocks – cclk_out)	0xFFFFFF40
CTYPE	R/W	0x018	Card-type register	0x0
BLKSIZ	R/W	0x01C	Block-size register	0x200
BYTCNT	R/W	0x020	Byte-count register	0x200
INTMASK	R/W	0x024	Interrupt-mask register	0x0
CMDARG	R/W	0x028	Command-argument register	0x0
CMD	R/W	0x02C	Command register	0x0
RESP0	R	0x030	Response-0 register	0x0
RESP1	R	0x034	Response-1register	0x0
RESP2	R	0x038	Response-2 register	0x0
RESP3	R	0x03C	Response-3 register	0x0
MINTSTS	R	0x040	Masked interrupt-status register	0x0
RINTSTS	R/W	0x44	Raw interrupt-status register	0x0
STATUS	R	0x48	Status register; mainly for debug purposes	{20'h00000, 4'b00xx, 8'h06}
FIFOTH	R/W	0x4C	FIFO threshold register	{4'h0, bits[27:16] = FIFO_DEPTH -1 bits[15:0] = 0}
CDETECT	R	0x50	Card-detect register	Value in card_detect signal
WRTPRT	R	0x54	Write-protect register	Value in card_write_prt signal
-	R/W	0x58	reserved	-

Table 393. Register overview: MCI (register base address 0x1800 0000)

Name	R/W	Address offset	Description	Reset value
TCBCNT	R	0x5C	Transferred CIU card byte count	0x0
TBBCNT	R	0x60	Transferred cpu/DMA to/from BIU-FIFO byte count	0x0
Reserved	-	0x64 - 0xFF	-	-
DATA	R/W	>= 0x100	Data FIFO read/write; if address is equal or greater than 0x100, then FIFO is selected as long as device is selected (hsel active)[1]	32'hx

[1] Address 0x100 and above are mapped to data FIFO. More than one address is mapped to data FIFO so that FIFO can be accessed using AMBA bursts.

4. Register description

Table 394. Control register (CTRL, address 0x1800 0000)

Bit	Symbol	Access	Reset value	Description
31:12	-	-	-	Reserved
11	ceata_device_interrupt_status		0	0 – Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register) 1 – Interrupts are enabled in CE-ATA device (nIEN = 0 in ATA control register) Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled (nIEN = 1). If the cpu enables CE-ATA device interrupt, then software should set this bit.
10	send_auto_stop_ccsd		0	0 – Do not send internally generated STOP command (CMD12) after sending Command Completion Signal Disable (CCSD) to CE-ATA device. 1 – Send internally generated STOP after sending CCSD to CE-ATA device. Always set send_auto_stop_ccsd and send_ccsd bits together; send_auto_stop_ccsd should not be set independent of send_ccsd. When set, the module automatically sends internally generated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in RINTSTS is set and generates interrupt to cpu if Auto Command Done interrupt is not masked. After sending the CCSD, the module automatically clears send_auto_stop_ccsd bit.
9	send_ccsd		0	0 – Do not send Command Completion Signal Disable (CCSD) to CE-ATA device. 1 – Send Command Completion Signal Disable (CCSD) to CE-ATA device When set, the module sends CCSD to CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device. Once the CCSD pattern is sent to device, the module automatically clears send_ccsd bit. It also sets Command Done (CD) bit in RINTSTS register and generates interrupt to cpu if Command Done interrupt is not masked.
8	Abort_read_data		0	0 – No change 1 – After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data state machine resets to idle. Used in SDIO card suspend sequence.

Table 394. Control register (CTRL, address 0x1800 0000) *…continued*

Bit	Symbol	Access	Reset value	Description
7	Send_irq_reponse		0	0 – No change 1 – Send auto IRQ response Bit automatically clears once response is sent. To wait for MMC card interrupts, cpu issues CMD40, and the module waits for interrupt response from MMC card(s). In meantime, if cpu wants the module to exit waiting for interrupt state, it can set this bit, at which time the module command state-machine sends CMD40 response on bus and returns to idle state.
6	Read_wait		0	0-- Clear read wait 1 – Assert read wait For sending read-wait to SDIO cards.
5	Dma_enable		0	0 – Disable DMA transfer mode 1 – Enable DMA transfer mode Even when DMA mode is enabled, cpu can still push/pop data into/from FIFO. If there is simultaneous FIFO push from CIU, DMA, and cpu (which should not happen in normal operation), priority is as follows: CIU, DMA, and cpu. Same priority given for FIFO pop.
4	Int_enable		0	Global interrupt enable/disable bit: 0 – Disable interrupts 1 – Enable interrupts The interrupt port is 1 only when this bit is 1 and one or more unmasked interrupts are set.
3	-	-	-	Reserved

Table 394. Control register (CTRL, address 0x1800 0000) *…continued*

Bit	Symbol	Access	Reset value	Description
2	Dma_reset		0	0 – No change 1 – Reset internal DMA interface control logic To reset DMA interface, firmware should set bit to 1. This bit is auto-cleared after two AHB clocks.
1	Fifo_reset		0	0 – No change 1 – Reset to data FIFO To reset FIFO pointers To reset FIFO, firmware should set bit to 1. This bit is auto-cleared after completion of reset operation.
0	Controller_reset		0	0 – No change 1 – Reset Module controller To reset controller, firmware should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets: * BIU/CIU interface * CIU and state machines * abort_read_data, send_irq_response, and read_wait bits of Control register * start_cmd bit of Command register Does not affect any registers or DMA interface, or FIFO or cpu

Table 395. Clock divider register (CLKDIV, address 0x1800 0008)

Bit	Symbol	Access	Reset value	Description
31:8	-	-	-	Reserved
7:0	Clk_divider	R/W	0	Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2*0 = 0 (no division, bypass), value of 1 means divide by 2*1 = 2, value of “ff” means divide by 2*255 = 510, and so on.

Table 396. Clock source register (CLKSRC, address 0x1800 000C)

Bit	Symbol	Access	Reset value	Description
31:2	-	-	-	Reserved
1:0	Clk_source	R/W	0x0	The value of this register has to be kept at zero (0x0).

Table 397. Clock enable register (CLKENA, address 0x1800 0010)

Bit	Symbol	Access	Reset value	Description
31:17	-	-	-	Reserved
16	Cclk_low_power	R/W	0x0	Low-power control for the output card clock: 0 – Non-low-power mode 1 – Low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).
15:1	-	-	-	Reserved
0	Cclk_enable	R/W	0	Clock-enable control for the output card clock: 0 – Clock disabled 1 – Clock enabled

Table 398. Timeout register (TMOUT, address 0x1800 0014)

Bit	Symbol	Access	Reset value	Description
31:8	Data_timeout	R/W	0xFFFFFF	Value for card Data Read Timeout; same value also used for Data Starvation by Cpu timeout. Value is in number of card output clocks – cclk_out of selected card.
7:0	Response_timeout	R/W	0x40	Response timeout value. Value is in number of card output clocks – cclk_out.

Table 399. Card type register (CTYPE, address 0x1800 0018)

Bit	Symbol	Access	Reset value	Description
31:17	-	-	-	Reserved
16	Card_width	R/W	0x0	This bit indicates if the card is 8-bit: 0 – Non 8-bit mode 1 – 8-bit mode
15:1	-	-	-	Reserved
0	Card_width	R/W	0	This bit indicates if the card is 1-bit or 4-bit: 0 – 1-bit mode 1 – 4-bit mode

The following examples use values for CTYPE[16]:

- If CTYPE[16] = 1, the card is in 8-bit mode. Note that the CTYPE[0] value is ignored; it is recommended to keep this set to 0.

- If CTYPE[16] = 0, the card is in either 1-bit or 4-bit mode, depending upon the value of CTYPE[0]; that is, if CTYPE[0] = 1 - 4-bit, CTYPE[0] = 0 - 1-bit.

Table 400. Blocksize register (BLKSIZ, address 0x1800 001C)

Bit	Symbol	Access	Reset value	Description
31:16	-	-	-	Reserved
15:0	Block_size	R/W	0x200	Block size

Table 401. Byte count register (BYCNT, address 0x1800 0020)

Bit	Symbol	Access	Reset value	Description
31:0	Byte_count	R/W	0x200	Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of cpu to explicitly send stop/abort command to terminate data transfer.

Table 402. Interrupt mask register (INTMASK, address 0x1800 0024)

Bit	Symbol	Access	Reset value	Description
31:17	-	-	-	Reserved
16	SDIO	R/W	0	Mask SDIO interrupt When masked, SDIO interrupt detection for the card is disabled. A 0 masks an interrupt, and 1 enables an interrupt.
15	EBE	R/W	0	End-bit error (read)/Write no CRC (EBE). A 0 masks an interrupt, and 1 enables an interrupt.
14	ACD	R/W	0	Auto command done (ACD). A 0 masks an interrupt, and 1 enables an interrupt.
13	SBE	R/W	0	Start-bit error (SBE). A 0 masks an interrupt, and 1 enables an interrupt.
12	HLE	R/W	0	Hardware locked write error (HLE). A 0 masks an interrupt, and 1 enables an interrupt.
11	FRUN	R/W	0	FIFO underrun/overrun error (FRUN). A 0 masks an interrupt, and 1 enables an interrupt.
10	HTO	R/W	0	Data starvation-by-cpu timeout (HTO). A 0 masks an interrupt, and 1 enables an interrupt.
9	DRTO	R/W	0	Data read timeout (DRTO). A 0 masks an interrupt, and 1 enables an interrupt.
8	RTO	R/W	0	Response timeout (RTO). A 0 masks an interrupt, and 1 enables an interrupt.
7	DCRC	R/W	0	Data CRC error (DCRC). A 0 masks an interrupt, and 1 enables an interrupt.
6	RCRC	R/W	0	Response CRC error (RCRC). A 0 masks an interrupt, and 1 enables an interrupt.
5	RXDR	R/W	0	Receive FIFO data request (RXDR). A 0 masks an interrupt, and 1 enables an interrupt.

Table 402. Interrupt mask register (INTMASK, address 0x1800 0024) *…continued*

Bit	Symbol	Access	Reset value	Description
4	TXDR	R/W	0	Transmit FIFO data request (TXDR). A 0 masks an interrupt, and 1 enables an interrupt.
3	DTO	R/W	0	Data transfer over (DTO). A 0 masks an interrupt, and 1 enables an interrupt.
2	CD	R/W	0	Command done (CD). A 0 masks an interrupt, and 1 enables an interrupt.
1	RE	R/W	0	Response error (RE). A 0 masks an interrupt, and 1 enables an interrupt.
0	CD	R/W	0	Card detect (CD). A 0 masks an interrupt, and 1 enables an interrupt.

Table 403. Command argument register (CMDARG, address 0x1800 0028)

Bit	Symbol	Access	Reset value	Description
31:0	cmd_arg	R/W	0	Value indicates command argument to be passed to card.

Table 404. Command register (CMD, address 0x1800 002C)

Bit	Symbol	Access	Reset value	Description
31	start_cmd	R/W	0	Start command. Once command is taken by CIU, bit is cleared. When bit is set, cpu should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register. Once command is sent and response is received from SD/MMC/CE-ATA cards, Command Done bit is set in raw interrupt register.
30:24	Reserved			
23	ccs_expected	R/W	0	0 – Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register), or command does not expect CCS from device 1 – Interrupts are enabled in CE-ATA device (nIEN = 0), and RW_BLK command expects command completion signal from CE-ATA device If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. The module sets Data Transfer Over (DTO) bit in RINTSTS register and generates interrupt to cpu if Data Transfer Over interrupt is not masked.
22	read_ceata_device	R/W	0	0 – Cpu is not performing read access (RW_REG or RW_BLK) towards CE-ATA device 1 – Module is performing read access (RW_REG or RW_BLK) towards CE-ATA device Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data timeout indication while performing CE-ATA read transfers. Maximum value of I/O transmission delay can be no less than 10 seconds. The module should not indicate read data timeout while waiting for data from CE-ATA device.

Table 404. Command register (CMD, address 0x1800 002C) *…continued*

Bit	Symbol	Access	Reset value	Description
21	update_clock_ registers_only	R/W	0	0 – Normal command sequence 1 – Do not send commands, just update clock register value into card clock domain following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when update_clock_registers_only = 0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card(s). When bit is set, there are no Command Done interrupts because no command is sent to SD/MMC/CE-ATA cards.
20:16	-	R/W	0	reserved
15	send_initialization	R/W	0	0 – Do not send initialization sequence (80 clocks of 1) before sending this command. 1 – Send initialization sequence before sending this command. After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card.
14	stop_abort_cmd	R/W	0	0 – Neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0. 1 – Stop or abort command intended to stop current data transfer in progress. When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state.
13	wait_prvdata_ complete	R/W	0	0 – Send command at once, even if previous data transfer has not completed. 1 – Wait for previous data transfer completion before sending command. The wait_prvdata_complete = 0 option typically used to query status of card during data transfer or to stop current data transfer.
12	send_auto_stop	R/W	0	0 – No stop command sent at end of data transfer. 1 – Send stop command at end of data transfer. Refer to Table 20–419 to determine: - when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands. - open-ended transfers that software should explicitly send to stop command. Additionally, when “resume” is sent to resume – suspended memory access of SD-Combo card – bit should be set correctly if suspended data transfer needs send_auto_stop. Don't care if no data expected from card

Table 404. Command register (CMD, address 0x1800 002C) *…continued*

Bit	Symbol	Access	Reset value	Description
11	Transfer_mode	R/W	0	0 – Block data transfer command 1 – Stream data transfer command Don't care if no data expected.
10	Read/write	R/W	0	0 – Read from card. 1 – Write to card. Don't care if no data expected from card.
9	Data_transfer_expected	R/W	0	0 – No data transfer expected (read/write). 1 – Data transfer expected (read/write).
8	Check_response_crc	R/W	0	0 – Do not check response CRC. 1 – Check response CRC. Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller
7	Response_length	R/W		0 – Short response expected from card 1 – Long response expected from card
6	Response_expect	R/W		0 – No response expected from card 1 – Response expected from card
5:0	Cmd_index	R/W	0	Command index

Table 405. Response 0 register (RESPO, address 0x1800 0030)

Bit	Name	Default	Description
31:0	response 0	0	Bit[31:0] of response

Table 406. Response 1 register (RESP1, address 0x1800 0034)

Bit	Name	Default	Description
31:0	response 1	0	Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in this register. Response for previous command sent by module is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always "short" for them. For information on when CIU sends auto-stop commands, refer to "Functional description.

Table 407. Response 2 register (RESP2, address 0x1800 0038)

Bit	Name	Default	Description
31:0	response 2	0	Bit[95:64] of long response

Table 408. Response 3 register (RESP3, address 0x1800 003C)

Bit	Name	Default	Description
31:0	response 3	0	Bit[127:96] of long response

Table 409. Masked interrupt status register (MINTSTS, address 0x1800 0040)

Bits	Name	Access	Reset value	Description
31:17	-	-		Reserved
16	sdio_interrupt	R/W		Interrupt from SDIO card. SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register (mask bit 1 enables interrupt; 0 masks interrupt). 0 – No SDIO interrupt from card 1 – SDIO interrupt from card
15	EBE	R/W	0	End-bit error (read)/Write no CRC (EBE).
14	ACD	R/W	0	Auto command done (ACD).
13	SBE	R/W	0	Start-bit error (SBE).
12	HLE	R/W	0	Hardware locked write error (HLE).
11	FRUN	R/W	0	FIFO underrun/overrun error (FRUN).
10	HTO	R/W	0	Data starvation-by-cpu timeout (HTO).
9	DRTO	R/W	0	Data read timeout (DRTO).
8	RTO	R/W	0	Response timeout (RTO).
7	DCRC	R/W	0	Data CRC error (DCRC).
6	RCRC	R/W	0	Response CRC error (RCRC).
5	RXDR	R/W	0	Receive FIFO data request (RXDR).
4	TXDR	R/W	0	Transmit FIFO data request (TXDR).
3	DTO	R/W	0	Data transfer over (DTO).
2	CD	R/W	0	Command done (CD).
1	RE	R/W	0	Response error (RE).
0	CD	R/W	0	Card detect (CD).

Table 410. Raw interrupt status register (RINTSTS, address 0x1800 0044)

Bit	Name	Access		Description
31:17	-	-		Reserved
16	sdio_interrupt	R/W	0	Interrupt from SDIO card. Writes to these bits clear them. Writing a value of 1 clears bit and 0 leaves bit intact. 0 – No SDIO interrupt from card 1 – SDIO interrupt from card Bits are logged regardless of interrupt-mask status.
15	EBE	R/W	0	End-bit error (read)/Write no CRC (EBE). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
14	ACD	R/W	0	Auto command done (ACD). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
13	SBE	R/W	0	Start-bit error (SBE). Writing a writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
12	HLE	R/W	0	Hardware locked write error (HLE). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.

Table 410. Raw interrupt status register (RINTSTS, address 0x1800 0044)

Bit	Name	Access		Description
11	FRUN	R/W	0	FIFO underrun/overrun error (FRUN). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
10	HTO	R/W	0	Data starvation-by-cpu timeout (HTO). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
9	DRTO	R/W	0	Data read timeout (DRTO). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
8	RTO	R/W	0	Response timeout (RTO). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
7	DCRC	R/W	0	Data CRC error (DCRC). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
6	RCRC	R/W	0	Response CRC error (RCRC). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
5	RXDR	R/W	0	Receive FIFO data request (RXDR). Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
4	TXDR	R/W	0	Transmit FIFO data request (TXDR). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
3	DTO	R/W	0	Data transfer over (DTO). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
2	CD	R/W	0	Command done (CD). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
1	RE	R/W	0	Response error (RE). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.
0	CD	R/W	0	Card detect (CD). Writing a value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.

Table 411. Status register (STATUS, address 0x1800 0048)

Bit	Name	Default	Description
31	Dma_req	0	DMA request signal state
30	dma_ack	0	DMA acknowledge signal state
29:17	fifo_count	0	FIFO count – Number of filled locations in FIFO
16:11	response_index	0	Index of previous response, including any auto-stop sent by core
10	data_state_mc_busy	0	Data transmit or receive state-machine is busy
9	data_busy	1 or 0;	Inverted version of MCI_DATA_0 pin 0 – card data not busy 1 – card data busy
8	data_3_status	1 or 0;	Raw state of MCI_DATA_3 pin; checks whether card is present 0 – card not present 1 – card present
7:4	command fsm states	0	Command FSM states: 0 – Idle 1 – Send init sequence 2 – Tx cmd start bit 3 – Tx cmd tx bit 4 – Tx cmd index + arg 5 – Tx cmd crc7 6 – Tx cmd end bit 7 – Rx resp start bit 8 – Rx resp IRQ response 9 – Rx resp tx bit 10 – Rx resp cmd idx 11 – Rx resp data 12 – Rx resp crc7 13 – Rx resp end bit 14 – Cmd path wait NCC 15 – Wait; CMD-to-response turnaround
3	fifo_full	0	FIFO is full status
2	fifo_empty	1	FIFO is empty status
1	fifo_tx_watermark	1	FIFO reached Transmit watermark level; not qualified with data transfer.
0	fifo_rx_watermark	0	FIFO reached Receive watermark level; not qualified with data transfer.

Table 412. FIFO threshold register, bits 31:28 (FIFOTH, address 0x1800 004C)

Bit	Name	Default	Description
31	-	-	Reserved
30:28	dma_multiple_transaction_size	3'b000	Burst size of multiple transaction; should be programmed same as DMA controller multiple-transaction-size SRC/DEST_MSIZE. 000 – 1 transfers 001 – 4 010 – 8 011 – 16 100 – 32 101 – 64 110 – 128 111 – 256 Value should be sub-multiple of (RX_WMark + 1)* (F_DATA_WIDTH/H_DATA_WIDTH) and (FIFO_DEPTH - TX_WMark)* (F_DATA_WIDTH/ H_DATA_WIDTH) For example, if FIFO_DEPTH = 16, FDATA_WIDTH == H_DATA_WIDTH Allowed combinations for MSize and TX_WMark are: MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMark = 8 MSize = 4, TX_WMark = 4 MSize = 4, TX_WMark = 12 MSize = 8, TX_WMark = 8 MSize = 8, TX_WMark = 4 Allowed combinations for MSize and RX_WMark are: MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMark = 3 MSize = 4, RX_WMark = 7 MSize = 4, RX_WMark = 11 MSize = 8, RX_WMark = 7 MSize = 8, RX_WMark = 11 Recommended: MSize = 8, TX_WMark = 8, RX_WMark = 7

Table 413. FIFO threshold register, bits 27:16 (FIFOTH, address 0x1800 004C)

Bit	Name	Default	Description
27:16	RX_WMark	0x1F (= FIFO_DEPTH-1)	FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of cpu to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. 12 bits – 1 bit less than FIFO-count of status register, which is 13 bits. Limitation: RX_WMark <= FIFO_DEPTH-2 Recommended: (FIFO_DEPTH/2) - 1; (means greater than (FIFO_DEPTH/2) - 1)

Table 414. FIFO threshold register, bits 15:0 (FIFOTH, address 0x1800 004C)

Bit	Name	Default	Description
15:12	-	-	Reserved
11:0	TX_WMark	0	FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If Interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated, regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, cpu is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. 12 bits – 1 bit less than FIFO-count of status register, which is 13 bits. Limitation: TX_WMark >= 1; Recommended: FIFO_DEPTH/2; (means less than or equal to FIFO_DEPTH/2)

Table 415. Card detect register (CDETECT, address 0x1800 0050)

Bits	Name	Default	Description
31:1	-	-	Reserved
0	Card_detect_n	0	0 represents presence of card. This bit is influenced by setting the BIT[1] in SYSCREG_SD_MMC_CFG configuration register in SYSCREG module.

Table 416. Write protect register (WRTPRT, address 0x1800 0054)

Bits	Name	Default	Description
31:1	-	-	Reserved
0	Write_protect	0	1 represents write protection. This bit is influenced by setting the BIT[0] in SYSCREG_SD_MMC_CFG configuration register in SYSCREG module.

Table 417. Transferred CIU card byte count (TCBINT, address 0x1800 005C)

Bits	Name	Default	Description
31:0	Trans_card_byte_count	0	Number of bytes transferred by CIU unit to card. The register should be read only after data transfer completes; during data transfer, register returns 0.

Table 418. Transferred cpu/DMA to/from BIU-FIFO byte count (TBBINT, address 0x1800 0060)

Bits	Name	Default	Description
31:0	Trans_fifo_byte_count	0	Number of bytes transferred between AHB/DMA memory and BIU FIFO.

5. Functional description

5.1 Auto-Stop

The auto-stop command helps to send an exact number of data bytes using a stream read or write for the MMC, and a multiple-block read or write for SD memory transfer for SD cards. The module internally generates a stop command and is loaded in the command path when the send_auto_stop bit is set in the Command register.

The software should set the send_auto_stop bit according to details listed in table below:

Table 419. send_auto_stop bit

Card Type	Transfer Type	Byte Count	send_auto_stop bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	>0	No	Byte count = 0 is illegal
MMC	Single-block write	>0	No	Byte count = 0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	>0	Yes**	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	>0	Yes**	Pre-defined multiple block

Table 419. send_auto_stop bit

Card Type	Transfer Type	Byte Count	send_auto_stop bit set	Comments
SDMEM	Single-block read	>0	No	Byte count = 0 is illegal
SDMEM	Single-block write	>0	No	Byte count = 0 is illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	>0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block
SDMEM	Multiple-block write	>0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	>0	No	Byte count = 0 is illegal
SDIO	Single-block write	>0	No	Byte count = 0 is illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block
SDIO	Multiple-block read	>0	No	Pre-defined multiple block
SDIO	Multiple-block write	0	No	Open-ended multiple block
SDIO	Multiple-block write	>0	No	Pre-defined multiple block

** The condition under which the transfer mode is set to block transfer and byte_count is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If byte_count = n*block_size (n = 2, 3, …), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the cpu software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue CMD18/CMD25 commands without setting the send_auto_stop bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the send_auto_stop bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop

command after the programmed byte count.

The following list conditions for the auto-stop command.

- Stream read for MMC card with byte count greater than 0 - The Module generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 - The Module generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.

- Multiple-block read memory for SD card with byte count greater than 0 - If the block size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.
- Multiple-block write memory for SD card with byte count greater than 0 - If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.
- Precaution for cpu software during auto-stop - Whenever an auto-stop command is issued, the cpu software should not issue a new command to the Module until the auto-stop is sent by the Module and the data transfer is complete. If the cpu issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the cpu wants to terminate the data transfer before the data transfer is complete, it can issue a stop or abort command, in which case the Module does not generate an auto-stop command.

6. Power optimization

Not applicable.

7. Programming guide

7.1 Software/hardware restrictions

Only one data transfer command should be issued at one time. For CE-ATA devices, if CE-ATA device interrupts are enabled (nIEN=0), only one RW_MULTIPLE_BLOCK command (RW_BLK) should be issued; no other commands (including a new RW_BLK) should be issued before the Data Transfer. Over status is set for the outstanding RW_BLK.

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

To avoid glitches in the card clock outputs (cclk_out), the software should use the following steps when changing the card clock frequency:

1. Update the Clock Enable register to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:
   - start_cmd bit
   - "update clock registers only" bits
   - "wait_previous data complete" bit

Wait for the CIU to take the command by polling for 0 on the start_cmd bit.

2. Set the start_cmd bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.
3. Set start_cmd to update the Clock Enable register in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the Data Transfer Over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX_WMark interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. In DMA mode while reading from a card, the DTO interrupt occurs only after all the FIFO data is flushed to memory by the DMA Interface unit.

While writing to a card in DMA mode, if an undefined-length transfer is selected by setting the Byte Count register to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a controller_reset command by setting control register bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the cpu. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly
- DMA data read is ignored
- Write data is unknown (x)

Additionally, if dma_reset is also issued, any pending DMA transfer is abruptly terminated. The DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.

One data-transfer requirement between the FIFO and cpu is that the number of transfers should be a multiple of the FIFO data width (F_DATA_WIDTH), which is 32. So if you want to write only 15 bytes to an SD/MMC/CE-ATA card (BYTCNT), the cpu should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers, if DMA mode is enabled. The software can still program the Byte Count register to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the cpu should still read all 16 bytes from the FIFO.

It is recommended that you do not change the FIFO threshold register in the middle of data transfers.

7.2 Programming sequence

7.2.1 Initialization

Once the power and clocks are stable, reset_n should be asserted (active-low) for at least two clocks of clk or cclk_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

1. After power on reset SD/MMC pins (mGPIO5 -mGPIO9) on this chip are configured as GPIO input pins. So set SYS_MUX_GPIO_MCI (0x13002894) to 1 and also update EBI_MCI-related registers in IOCONFIG to change mGPIO5-mGPIO9 pins from inputs to 'driven by IP' state.
2. Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register @0x024. Set the global int_enable bit of the Control register @0x00. It is recommended that you write 0xffff_ffff to the Raw Interrupt register @0x044 in order to clear any pending interrupts before setting the int_enable bit.
3. Enumerate card stack - Each card is enumerated according to card type; for details, refer to "Enumerated Card Stack". For enumeration, you should restrict the clock frequency to 400 KHz in accordance with SD_MMC/CE-ATA standards.
4. Changing clock. The cards operate at a maximum of 26 MHz (at maximum of 52 MHz in high-speed mode).
5. Set other IP parameters, which normally do not need to be changed with every command, with a typical value such as timeout values in cclk_out according to SD_MMC/CE-ATA specifications.

   ResponseTimeOut = 0x40

   DataTimeOut = highest of one of the following:

   – (10 * ((TAAC*Fop) + (100*NSAC))
   – Cpu FIFO read/write latency from FIFO empty/full

   FIFO threshold value in bytes in the FIFOTH register @0x04C. Typically, the threshold value can be set to half the FIFO depth (=32); that is:

   – RX_WMark = (FIFO_DEPTH/2) - 1;
   – TX_WMark = FIFO_DEPTH/2
6. If the software decides to handle the interrupts provided by the IP core, you should create another thread to handle interrupts.

7.2.2 Enumerated Card Stack

The card stack does the following:

- Enumerates all connected cards
- Sets the RCA for the connected cards
- Reads card-specific information
- Stores card-specific information locally

Enumerate_Card_Stack - Enumerates the card connected on the module. The card can be of the type MMC, CE-ATA, SD, or SDIO. All types of SDIO cards are supported; that is, SDIO_IO_ONLY, SDIO_MEM_ONLY, and SDIO_COMBO cards. The enumeration sequence includes the following steps:

1. Check if the card is connected.
2. Clear the bits in the card_type register. Clear the register bit for a 1-bit, 4-bit, or 8-bit bus width.
3. Identify the card type; that is, SD, MMC, or SDIO.
   - Send CMD5 first. If a response is received, then the card is SDIO
   - If not, send ACMD41; if a response is received, then the card is SD.
   - Otherwise, the card is an MMC or CE-ATA
4. Enumerate the card according to the card type.

   Use a clock source with a frequency = Fod (that is, 400 KHz) and use the following enumeration command sequence:
   - SD card - Send CMD0, ACMD41, CMD2, CMD3.
   - SDHC card - send CMD0, SDCMD8, ACMD41, CMD2, CMD3
   - SDIO - Send CMD5; if the function count is valid, CMD3. For the SDIO memory section, follow the same commands as for the SD card.
   - MMC - Send CMD0, CMD1, CMD2, CMD3
5. Identify the MMC/CE-ATA device.
   - Selecting ATA mode for a CE-ATA device.
   - Cpu should query the byte 504 (S_CMD_SET) of EXT_CSD register by sending CMD8. If bit 4 is set to "1," then the device supports ATA mode.
   - If ATA mode is supported, the cpu should select the ATA mode by setting the ATA bit (bit 4) of the EXT_CSD register slice 191(CMD_SET) to activate the ATA command set for use. The cpu selects the command set using the SWITCH (CMD6) command.
   - The current mode selected is shown in byte 191 of the EXT_CSD register.

     If the device does not support ATA mode, then the device can be an MMC device or a CE-ATA v1.0 device.
   - Send RW_REG; if a response is received and the response data contains CE-ATA signature, the device is a CE-ATA device.
   - Otherwise the device is an MMC card.
6. You can change the card clock frequency after enumeration.

7.2.3 Clock Programming

The clock programming has to be done in the CGU. The cclk_in has to be equal to the cclk_out. Therefore the registers that support this have to be:

- CLKDIV @0x08 = 0x0 (bypass of clock divider).
- CLKSRC @0x0C = 0x0
- CLKENA @0x10 =0x0 or 0x1. This register enables or disables clock for the card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The Module loads each of these registers only when the start_cmd bit and the Update_clk_regs_only bit in the CMD register are set. When a command is successfully loaded, the Module clears this bit, unless the Module already has another command in the queue, at which point it gives an HLE (Hardware Locked Error); for details on HLEs, refer to "Error Handling".

Software should look for the start_cmd and the Update_clk_regs_only bits, and should also set the wait_prvdata_complete bit to ensure that clock parameters do not change during data transfer. Note that even though start_cmd is set for updating clock registers, the Module does not raise a command_done signal upon command completion.

7.2.4 No-Data Command With or Without Response Sequence

To send any non-data command, the software needs to program the CMD register @0x2C and the CMDARG register @0x28 with appropriate parameters. Using these two registers, the Module forms the command and sends it to the command bus. The Module reflects the errors in the command response through the error bits of the RINTSTS register.

When a response is received - either erroneous or valid - the Module sets the command_done bit in the RINTSTS register. A short response is copied in Response Register0, while a long response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3 register bit 31 represents the MSB, and the Response0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

1. Program the Command register @0x28 with the appropriate command argument parameter.
2. Program the Command register @0x2C with the settings in Table 20–420.
3. Wait for command acceptance by cpu. The following happens when the command is loaded into the Module:
   - Module accepts the command for execution and clears the start_cmd bit in the CMD register, unless one command is in process, at which point the Module can load and keep the second command in the buffer.
   - If the Module is unable to load the command - that is, a command is already in progress, a second command is in the buffer, and a third command is attempted - then it generates an HLE (hardware-locked error).
   - Check if there is an HLE.
   - Wait for command execution to complete. After receiving either a response from a card or response timeout, the Module sets the command_done bit in the RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.
   - Check if response_timeout error, response_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the RINTSTS register @0x44. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers @0x30-0x3C.

Software should not modify clock parameters while a command is being executed.

Table 420. CMD register settings for No-Data Command

Name	Value	Comment
start_cmd	1	
update_clock_ registers_only	0	No clock parameters update command
card_number	0	Card number in use. Only zero is possible because one card is support.
Data_expected	0	No data command.
Send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
Cmd_index	Command index	
Response_length	0	Can be 1 for R2 (long) response
Response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
Wait_prvdata_complete	1	Before sending command on command line, cpu should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
Check_response_crc	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

7.2.5 Data Transfer Commands

Data transfer commands transfer data between the memory card and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively.

For the data transfer commands, it is important that the same bus width that is programmed in the card should be set in the card type register @0x18. Therefore, in order to change the bus width, you should always use the following supplied APIs as appropriate for the type of card:

- Set_SD_Mode() - SD/SDIO card
- Set_HSmodeSettings() - HSMMC card

The Module generates an interrupt for different conditions during data transfer, which are reflected in the RINTSTS register @0x44 as:

1. Data_Transfer_Over (bit 3) - When data transfer is over or terminated. If there is a response timeout error, then the Module does not attempt any data transfer and the "Data Transfer Over" bit is never set.
2. Transmit_FIFO_Data_request (bit 4) - FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO.
3. Receive_FIFO_Data_request (bit 5) - FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
4. Data starvation by Cpu timeout (bit 10) - FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the Module cannot continue with data transfer. The clock to the card has been stopped.
5. Data read timeout error (bit 9) - Card has not sent data within the timeout period.
6. Data CRC error (bit 7) - CRC error occurred during data reception.
7. Start bit error (bit 13) - Start bit was not received during data reception.
8. End bit error (bit 15) - End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

   Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response timeout, then no data transfer occurred.

7.2.6 Single-Block or Multiple-Block Read

Steps involved in a single-block or multiple-block read are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C. The Module expects data from the card in blocks of size BLKSIZ each.
3. Program the CMDARG register @0x28 with the data address of the beginning of a data read. Program the Command register with the parameters listed in Table 20–421. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

   After writing to the CMD register, the Module starts executing the command; when the command is sent to the bus, the command_done interrupt is generated.
4. Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
5. Software should look for Receive_FIFO_Data_request and/or data starvation by cpu timeout conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
6. When a Data_Transfer_Over interrupt is received, the software should read the remaining data from the FIFO.

Table 421. CMD register settings for Single-block or Multiple-block Read

Name	Value	Comment
start_cmd	1	
update_clock_ registers_only	0	No clock parameters update command
card_number	0	Card number in use. Only zero is possible because one card is support.
Data_expected	1	
Send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
Send_auto_stop	0/1	Set according to <tbd>
Transfer_mode	0	Block transfer
Read_write	0	Read from card
Cmd_index	Command index	
Response_length	0	Can be 1 for R2 (long) response
Response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
Wait_prvdata_complete	1	Before sending command on command line, cpu should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
Check_response_crc	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

7.2.7 Single-Block or Multiple-Block Write

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C; the Module sends data in blocks of size BLKSIZ each.
3. Program CMDARG register @0x28 with the data address to which data should be written.
4. Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.

5. Program the Command register with the parameters listed in Table 20–422. For SD and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

   After writing to the CMD register, Module starts executing a command; when the command is sent to the bus, a command_done interrupt is generated.
6. Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending the STOP command.
7. Software should look for Transmit_FIFO_Data_request and/or timeout conditions from data starvation by the cpu. In both cases, the software should write data into the FIFO.
8. When a Data_Transfer_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the Module should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto_command_done interrupt - bit 14 of the RINTSTS register. A response to AUTO_STOP is stored in RESP1 @0x34.

Table 422. CMD register settings for Single-block or Multiple-block write

Name	Value	Comments
start_cmd	1	
update_clock_ registers_only	0	No clock parameters update command
card_number	0	Card number in use. Only zero is possible because one card is support.
Data_expected	1	
Send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
Send_auto_stop	0/1	Set according to XXXXXXXXX
Transfer_mode	0	Block transfer
Read_write	1	Write to card
Cmd_index	Command index	
Response_length	0	Can be 1 for R2 (long) response
Response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on

Table 422. CMD register settings for Single-block or Multiple-block write

Name	Value	Comments
User-selectable		
Wait_prvdata_complete	1	Before sending command on command line, cpu should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
Check_response_crc	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

7.2.8 Stream Read

A stream read is like the block read mentioned in "Single-Block or Multiple-Block Read", except for the following bits in the Command register:

transfer_mode = 1; //Stream transfer

cmd_index = CMD20;

A stream transfer is allowed for only a single-bit bus width.

7.2.9 Stream Write

A stream write is exactly like the block write mentioned in "Single-Block or Multiple-Block Write", except for the following bits in the Command register:

- transfer_mode = 1;//Stream transfer
- cmd_index = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a transfer, the Module sends the STOP command. Completion of this AUTO_STOP command is reflected by the Auto_command_done interrupt. A response to an AUTO_STOP is stored in the RESP1 register @0x34.

A stream transfer is allowed for only a single-bit bus width.

7.2.10 Sending Stop or Abort in Middle of Transfer

The STOP command can terminate a data transfer between a memory card and the Module, while the ABORT command can terminate an I/O data transfer for only the SDIO_IOONLY and SDIO_COMBO cards.

- Send STOP command - Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer. For information on sending this command, refer to "No-Data Command With or Without Response Sequence".

You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (stop_abort_cmd) to

1. If stop_abort_cmd is not set to 1, the Module does not know that the user stopped a data transfer. Reset bit 13 of the Command register (wait_prvdata_complete) to 0 in order to make the Module send the command at once, even though there is a data transfer in progress.
   - Send ABORT command - Can be used with only an SDIO_IOONLY or SDIO_COMBO card. To abort the function that is transferring data, program the function number in ASx bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

     This is a non-data command. For information on sending this command, refer to "No-Data Command With or Without Response Sequence".

     Program the CMDARG register @0x28 with the appropriate command argument parameters listed in Table 20–423.
   - Program the Command register using the command index as CMD52. Similar to the STOPcommand, set bit 14 of the Command register (stop_abort_cmd) to 1, which must be done in order to inform the Module that the user aborted the data transfer. Reset bit 13 (wait_prvdata_complete) of the Command register to 0 in order to make the Module send the command at once, even though a data transfer is in progress.

     Wait for command_transfer_over

     Check response (R5) for errors

Table 423. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1
30-28	Function number	0, for CCCR access
27	RAW flag	1, if needed to read after write
26	Don’t care	-
25-9	Register address	0x06
8	Don’t care	-
7-0	Write data	Function number to be aborted

7.3 Suspend or Resume Sequence

In an SDIO card, the data transfer between an I/O function and the Module can be temporarily halted using the SUSPEND command; this may be required in order to perform a high-priority data transfer with another function. When desired, the data transfer can be resumed using the RESUME command.

The following functions can be implemented by programming the appropriate bits in the CCCR register (Function 0) of the SDIO card. To read from or write to the CCCR register, use the CMD52 command.

1. SUSPEND data transfer - Non-data command.
   - Check if the SDIO card supports the SUSPEND/RESUME protocol; this can be done through the SBS bit in the CCCR register @0x08 of the card.

     Check if the data transfer for the required function number is in process; the function number that is currently active is reflected in bits 0-3 of the CCCR register @0x0D. Note that if the BS bit (address 0xc::bit 0) is 1, then only the function number given by the FSx bits is valid.

     To suspend the transfer, set BR (bit 2) of the CCCR register @0x0C.

     Poll for clear status of bits BR (bit 1) and BS (bit 0) of the CCCR @0x0C. The BS (Bus Status) bit is 1 when the currently-selected function is using the data bus; the BR (Bus Release) bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function has been suspended.

     During a read-data transfer, the Module can be waiting for the data from the card. If the data transfer is a read from a card, then the Module must be informed after the successful completion of the SUSPEND command. The Module then resets the data state machine and comes out of the wait state. To accomplish this, set abort_read_data (bit 8) in the Control register.

     Wait for data completion. Get pending bytes to transfer by reading the TCBCNT register @0x5C.
2. RESUME data transfer - This is a data command.
   - Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.

     If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.

     Check that a function to be resumed is ready for data transfer; this can be confirmed by reading the RFx flag in CCCR @0x0F. If RF = 1, then the function is ready for data transfer.

     To resume transfer, use CMD52 to write the function number at FSx bits (0-3) in the CCCR register @0x0D. Form the command argument for CMD52 and write it in CMDARG @0x28; bit values are listed in Table 20–424.
   - Write the block size in the BLKSIZ register @0x1C; data will be transferred in units of this block size.

     Write the byte count in the BYTCNT register @0x20. This is the total size of the data; that is, the remaining bytes to be transferred. It is the responsibility of the software to handle the data.

     Program Command register; similar to a block transfer. For details, refer to "Single-Block or Multiple-Block Read" and "Single-Block or Multiple-Block Write".

     When the Command register is programmed, the command is sent and the function resumes data transfer. Read the DF flag (Resume Data Flag). If it is 1, then the function has data for the transfer and will begin a data transfer as soon as the function or memory is resumed. If it is 0, then the function has no data for the transfer.

     If the DF flag is 0, then in case of a read, the Module waits for data. After the data timeout period, it gives a data timeout error.

Table 424. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1
30-28	Function number	0, for CCCR access
27	RAW flag	1, read after write
26	Don't care	-
25-9	Register address	0x0D
8	Don't care	-
7-0	Write data	Function number to be aborted

7.3.1 Read_Wait Sequence

Read_wait is used with only the SDIO card and can temporarily stall the data transfer-either from function or memory-and allow the cpu to send commands to any function within the SDIO device. The cpu can stall this transfer for as long as required. The Module provides the facility to signal this stall transfer to the card. The steps for doing this are:

1. Check if the card supports the read_wait facility; read SRW (bit 2) of the CCCR register @0x08. If this bit is 1, then all functions in the card support the read_wait facility. Use CMD52 to read this bit.
2. If the card supports the read_wait signal, then assert it by setting the read_wait (bit 6) in the CTRL register @0x00.
3. Clear the read_wait bit in the CTRL register.

7.3.2 CE-ATA Data Transfer Commands

This section describes the CE-ATA data transfer commands. For information on the basic settings and interrupts generated for different conditions, refer to "Data Transfer Commands".

Reset and Device Recovery

Before starting CE-ATA operations, the cpu should perform an MMC reset and initialization procedure. The cpu and device should negotiate the MMC TRAN state (defined by the MultiMedia Card System Specification) before the device enters the MMC TRAN state. The cpu should follow the existing MMC Card enumeration procedure in order to negotiate the MMC

TRAN state. After completing normal MMC reset and initialization procedures, the cpu should query the initial ATA Task File values using RW_REG/CMD39.

By default, the MMC block size is 512 bytes-indicated by bits 1:0 of the srcControl register inside the CE-ATA device. The cpu can negotiate the use of a 1KB or 4KB MMC block size. The device indicates MMC block sizes that it can support through the srcCapabilities register; the cpu reads this register in order to negotiate the MMC block size. Negotiation is complete when the cpu controller writes the MMC block size into the srcControl register bits 1:0 of the device.

ATA Task File Transfer

ATA task file registers are mapped to addresses 0x00h-0x10h in the MMC register space. RW_REG is used to issue the ATA command, and the ATA task file is transmitted in a single RW_REG MMC command sequence.

The cpu software stack should write the task file image to the FIFO before setting the CMDARG and CMD registers. The cpu processor then sets the address and byte count in the CMDARG-offset 0x28 in the BIU register space-before setting the CMD (offset 0x2C) register bits.

For RW_REG, there is no command completion signal from the CE-ATA device

ATA Task File Transfer Using RW_MULTIPLE_REGISTER (RW_REG)

This command involves data transfer between the CE-ATA device and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Steps involved in an ATA Task file transfer (read or write) are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C; the Module expects a single block transfer.
3. Program the CMDARG register @0x28 with the beginning register address.

You should program the CMDARG, CMD, BLKSIZ, and BYTCNT registers according to the following tables.

- Program the Command Argument (CMDARG) register as shown below.

Table 425. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1 (write) or 0 (read)
30-24	Reserved	0
23:18	Starting register address for read/write; Dword aligned	0
17:16	Register address; Dword aligned	0
15-8	Reserved; bits cleared to 0 by CPU	0
7:2	Number of bytes to read/write; integral number of Dwords	16
1:0	Byte count in integral number of Dwords	0

- Program the Command (CMD) register as shown below.

Table 426. CMD register settings

Name	Value	Comment
start_cmd	1	
Css_expect	0	Command Completion Signal is not expected
Read_ceata_device	0/1	1 – If RW_BLK or RW_REG read
update_clock_ registers_only	0	No clock parameters update command
card_number	0	Card number in use. Only zero is possible because one card is support.
Data_expected	1	
Send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	
Send_auto_stop	0	
Transfer_mode	0	Block transfer
Read_write	0/1	0 read from card, 1 - Write to card
Cmd_index	Command index	
Response_length	0	
Response_expect	1	
User-selectable		
Wait_prvdata_complete	1	0 – Sends command immediately 1 – Sends command after previous data transfer over
Check_response_crc	1	0 – Do not check response CRC 1 – Check response CRC

- Program the block size (BLKSIZ) register as shown below.

Table 427. BLKSIZ register

Bits	Value	Comment
31:16	0	Reserved bits as zeroes (0)
15:0	16	For accessing entire task file (16, 8-bit registers); block size of 16 bytes

- Program the Byte Count (BYTCNT) register as shown below.

Table 428. BYTCNT register

Bits	Value	Comment
31:0	16	For accessing entire task file(16, 8 bit registers); byte count value of 16 is used with the block size set to 16

ATA Payload Transfer Using RW_MULTIPLE_BLOCK (RW_BLK)

This command involves data transfer between the CE-ATA device and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Steps involved in an ATA payload transfer (read or write) are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C. The Module expects a single/multiple block transfer.
3. Program the CMDARG register @0x28 to indicate the Data Unit Count.

   You should program the CMDARG, CMD, BLKSIZ, and BYTCNT registers according to the following tables.

   – Program the Command Argument (CMDARG) register as shown below.

Table 429. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1 (write) or 0 (read)
30-24	Reserved	0
23:16	Reserved	0
15:8	Data Count Unit [15:8]	Data count
1:0	Data Count Unit [7:0]	Data count

- Program the Command (CMD) register as shown below.

Table 430. CMD register settings

Name	Value	Comment
start_cmd	1	-
Css_expect	1	Command Completion Signal is expected; set for RW_BLK if interrupts are enabled in CE-ATA device, nIEN = 0
Read_ceata_device	0/1	1 – If RW_BLK or RW_REG read
update_clock_ registers_only	0	No clock parameters update command
card_number	0	Card number in use. Only zero is possible because one card is support.
Data_expected	1	
Send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	
Send_auto_stop	0	
Transfer_mode	0	Block transfer
Read_write	0/1	0 read from card, 1 - Write to card
Cmd_index	Command index	
Response_length	0	
Response_expect	1	
User-selectable		
Wait_prvdata_complete	1	0 – Sends command immediately 1 – Sends command after previous data transfer over
Check_response_crc	1	0 – Do not check response CRC 1 – Check response CRC

- Program the block size (BLKSIZ) register as shown below.

Table 431. BLKSIZ register

Bits	Value	Comment
31:16	0	Reserved bits as zeroes (0)
15:0	512, 1024, 4096	MMC block size can be 512, 1024, or 4096 bytes as negotiated by CPU

- Program the Byte Count (BYTCNT) register as shown below.

Table 432. BYTCNT register

Bits	Value	Comment
31:0	N*block_size	byte_count should be integral multiple of block size; for ATA media access commands, byte count should be multiple of 4KB. (N*block_size = X*4KB, where N and X are integers)

Sending Command Completion Signal Disable

While waiting for the Command Completion Signal (CCS) for an outstanding RW_BLK, the cpu can send a Command Completion Signal Disable (CCSD).

- Send CCSD - Module sends CCSD to the CE-ATA device if the send_ccsd bit is set in the CTRL register; this bit is set only after a response is received for the RW_BLK.
- Send internal Stop command - Send internally generated STOP (CMD12) command after sending the CCSD pattern. If send_auto_stop_ccsd bit is also set when the controller is programmed to send the CCSD pattern, the Module sends the internally generated STOP command on the CMD line. After sending the STOP command, the Module sets the Auto Command Done bit in the RINTSTS register.

Recovery after Command Completion Signal Timeout

If timeout happened while waiting for Command Completion Signal (CCS), the cpu needs to send Command Completion Signal Disable (CCSD) followed by a STOP command to abort the pending ATA command. The cpu can program the Module to send internally generated STOP command after sending the CCSD pattern

- Send CCSD - Set the send_ccsd bit in the CTRL register.
- Reset bit 13 of the Command register (wait_prvdata_complete) to 0 in order to make the Module send the command at once, even though there is a data transfer in progress.
- Send internal STOP command - Set send_auto_stop_ccsd bit in the CTRL register, which programs the cpu controller to send the internally generated STOP command. After sending the STOP command, the Module sets the Auto Command Done bit in the RINTSTS register.

Reduced ATA Command Set

It is necessary for the CE-ATA device to support the reduced ATA command subset. The following details discuss this reduced command set.

- IDENTIFY DEVICE - Returns 512-byte data structure to the cpu that describes device-specific information and capabilities. The cpu issues the IDENTIFY DEVICE command only if the MMC block size is set to 512 bytes; any other MMC block size has indeterminate results.

The cpu issues RW_REG for the ATA command, and the data is retrieved through RW_BLK.

The cpu controller uses the following settings while sending RW_REG for the IDENTIFY DEVICE ATA command. The following lists the primary bit

- CMD register setting - data_expected field set to 0
- CMDARG register settings:
  - Bit [31] set to 0
  - Bits [7:2] set to 128.
- Task file settings:
  - Command field of the ATA task file set to ECh
  - Reserved fields of the task file cleared to 0
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16

  The cpu controller uses the following settings for data retrieval (RW_BLK):\
- CMD register settings:
  - ccs_expect set to 1
  - data_expected set to 1
- CMDARG register settings:
  - Bit [31] set to 0 (Read operation)

    Data Count set to 1 (16'h0001)
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 512 IDENTIFY DEVICE can be aborted as a result of the CPU issued CMD12.
  - READ DMA EXT - Reads a number of logical blocks of data from the device using the Data-In data transfer protocol. The cpu uses RW_REG to issue the ATA command and RW_BLK for the data transfer.
  - WRITE DMA EXT - Writes a number of logical blocks of data to the device using the Data-Out data transfer protocol. The cpu uses RW_REG to issue the ATA command and RW_BLK for the data transfer.
  - STANDBY IMMEDIATE - No data transfer (RW_BLK) is expected for this ATA command, which causes the device to immediately enter the most aggressive power management mode that still retains internal device context.
- CMD Register setting - data_expected field set to 0

  CMDARG register settings:
  - Bit [31] set to 1
  - Bits [7:2] set to 4
- Task file settings:

- Command field of the ATA task file set to E0h
- Reserved fields of the task file cleared to 0

- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16
  - FLUSH CACHE EXT - No data transfer (RW_BLK) is expected for this ATA command. For devices that buffer/cache written data, the FLUSH CACHE EXT command ensures that buffered data is written to the device media. For devices that do not buffer written data, FLUSH CACHE EXT returns a success status. The cpu issues RW_REG for the ATA command, and the status is retrieved through CMD39/RW_REG; there can be error status for this ATA command, in which case fields other than the status field of the ATA task file are valid.

- The CPU uses the following settings while sending the RW_REG for STANDBY IMMEDIATE ATA command:
- CMD register setting - data_expected field set to 0
- CMDARG register settings:
- Bit [31] set to 1
- Bits [7:2] set to 4
- Task file settings:
- Command field of the ATA task file set to EAh
- Reserved fields of the task file cleared to 0
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16

7.3.3 Controller/DMA/FIFO Reset Usage

Communication with the card involves the following:

- Controller - Controls all functions of the Module.
- FIFO - Holds data to be sent or received.
- DMA - If DMA transfer mode is enabled, then transfers data between system memory and the FIFO.
- Controller reset - Resets the controller by setting the controller_reset bit (bit 0) in the CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset - Resets the FIFO by setting the fifo_reset bit (bit 1) in the CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

  DMA reset - Resets the internal DMA controller logic by setting the dma_reset bit (bit 2) in the CTRL register, which abruptly terminates any DMA transfer in process. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

The following are recommended methods for issuing reset commands:

- Non-DMA transfer mode - Simultaneously sets controller_reset and fifo_reset; clears the RAWINTS register @0x44 using another write in order to clear any resultant interrupt.

- Generic DMA mode - Simultaneously sets controller_reset, fifo_reset, and dma_reset; clears the RAWINTS register @0x44 by using another write in order to clear any resultant interrupt. If a "graceful" completion of the DMA is required, then it is recommended to poll the status register to see whether the dma request is 0 before resetting the DMA interface control and issuing an additional FIFO reset.
- DW_DMA mode - Sets controller_reset and fifo_reset; waits until dw_dma_req goes inactive (the Status register indicates the value of this signal). Resets the FIFO again. Clears the interrupts by clearing the RAWINTS register @0x44 using another write in order to clear any resultant interrupt. You also need to reset and reprogram the channel(s) of the DW_ahb_dmac controller that are interfaced to the Module.
- In DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the RAWINTS register caused by the DMA transfers after the FIFO was reset.

7.3.4 Error Handling

The Module implements error checking; errors are reflected in the RAWINTS register @0x44 and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (int_enable in the CTRL register is 0), and all the interrupts are masked (bits 0-31 of the INTMASK register; default is 0). Error handling:

- Response and data timeout errors - For response timeout, software can retry the command. For data timeout, the Module has not received the data start bit - either for the first block or the intermediate block - within the timeout period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the TCBCNT later, the software can decide how many bytes remain to be copied.
- Response errors - Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors - Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.
- Hardware locked error - Set when the Module cannot load a command issued by software. When software sets the start_cmd bit in the CMD register, the Module tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
- FIFO underrun/overrun error - If the FIFO is full and software tries to write data in the FIFO, then an overrun error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read
- the fifo_empty or fifo_full bits in the Status register.

- Data starvation by cpu timeout - Raised when the Module is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated timeout period. Under this condition and when a read transfer is in process, the software
- Should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command - If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response timeout is expected from the Module. The ATA layer is notified that an MMC transport layer error occurred.
- Write operation - Any MMC Transport layer error known to the device causes an outstanding ATA command to be terminated. The ERR bits are set in the ATA status registers and the appropriate error code is sent to the ATA Error register.
- If nIEN=0, then the Command Completion Signal (CCS) is sent to the cpu.

If device interrupts are not enabled (nIEN=1), then the device completes the entire Data Unit Count if the cpu controller does not abort the ongoing transfer.

During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register. It then continues further data transmission until all the bytes are transmitted.

- Read operation - If MMC transport layer errors are detected by the cpu controller, the cpu completes the ATA command with an error status.

The cpu controller can issue a Command Completion Signal Disable (CCSD) followed by a STOP TRANSMISSION (CMD12) to abort the read transfer. The cpu can also transfer the entire Data Unit Count bytes without aborting the data transfer.

UM10314

Chapter 21: LPC3130/31 Universal Asynchronous Receiver/Transmitter (UART)

Rev. 1 — 4 March 2009 User manual

1. Introduction

Many applications need a simple method of communicating data between equipment. The Universal Asynchronous Receiver Transmitter (UART) protocol is standard for such communications. The UART supports the industry standard serial interface and can be used for connecting a modem, blue tooth IC, or a terminal emulator. The term asynchronous is used because it is not necessary to send clocking information with the data that is sent. The UART interface is fully compliant with industry standard 16C750 from various manufacturers. The UART can also function as an IrDA (Infra-Red Data Exchange) modem by setting a register bit in the UART configuration register bank.

1.1 Features

This module has the following features:

- Programmable baud rate with a maximum of 1049 kbaud.
- Programmable data length (5-8 bits).
- Implements only asynchronous UART.
- Transmit break character length indication.
- Programmable one to two stop bits in transmission.
- Odd/Even/Force parity check/generation.
- Frame error, overrun error and break detection.
- Automatic modem flow control.
- Independent control of transmit, receive, line status, data set interrupts and FIFO’s.
- SIR-IrDA encoder/decoder (from 2400 to 115 kbaud).
- Supports interrupts.
- Supports DMA transfers.

2. General description

2.1 About UART

UART links are character oriented (the smallest unit of data that can be correctly received or transmitted is a character). Typical applications of asynchronous links are connections between terminals and computer equipment. Two UARTs can communicate using a system like this if parameters, such as the parity scheme and character length, are the same for both transmitter and receiver. The character format of the UART protocol is illustrated in the following figure:

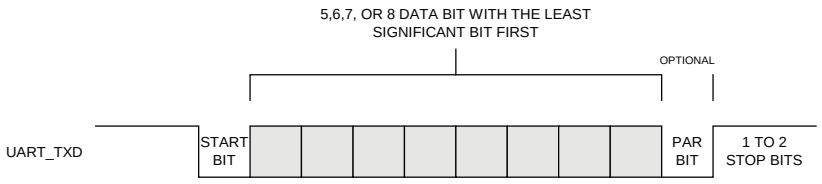


Fig 60. UART Character Format

When data is not transmitted in the UART protocol, a continuous stream of ones is transmitted, called the idle condition. Since the start bit is always a zero, the receiver can detect when real data is once again present on the line. UART also specifies an all-zeros character (start, data, parity, stop), which is used to abort a character transfer sequence.

2.2 About IrDA

IrDA stands for infrared (IR) Data Exchange. It's a common name for a suit of protocols for infrared exchange of data between two devices, up to 1 or 2 meters apart (20 to 30 cm for low-power devices). IrDA devices typically have throughput of up to either 115.2 Kbps (Serial IR, SIR). IrDA protocols are implemented in many (smart) mobile phones, PDAs and portable devices, printers and laptop computers. The Infrared Data Association, the industry body that specifies IrDA protocols, was originally founded by Hewlett-Packard and others.

2.3 Block diagram

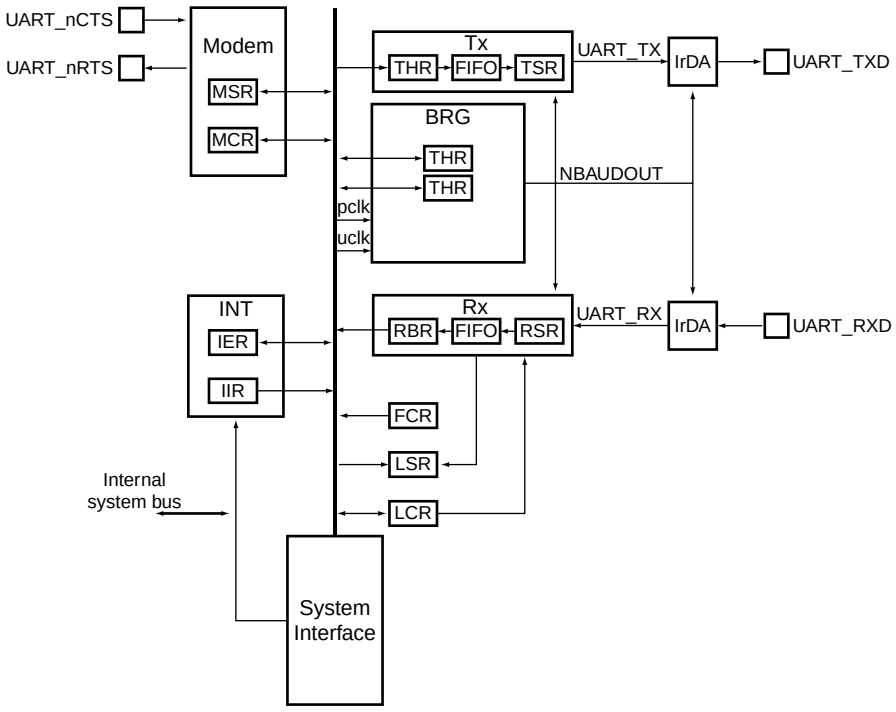


Fig 61. UART block diagram

The UART internal architecture consists of the following modules:

- A APB bus transceiver (APB SYSTEM INTERFACE)
- A UART modem block (MODEM)
- A Baud Rate Generator block (BRG)
- Receive and transmit buffer blocks (respectively RX, TX)
- An IrDA block (can be enable/disabled according to the UART register settings).

The UART has two clock domains, an APB clock domain to interface to the rest of the internal system, and a clock domain dedicated to the UART serial interface.

2.4 Interface description

2.4.1 Clock signals

The UART has two clock inputs:

- The APB_CLK operates the APB interface and the registers in the software view.
- The U_CLK is used for UART baud-rate generation and operates most of the UART data-path including the optional IrDA.

The clocks can be asynchronous i.e. need not have a frequency or phase relation. The APB_CLK frequency can be smaller than, equal to, or higher than the U_CLK frequency.

Table 433. Clock Signals of the UART

Clock Name	Acronym	I/O	Source/ Destination	Description
UART_APB_CLK	APB_CLK	I	CGU	The APB_CLK can be switched off by an external clock gate. In this case the UART registers are not accessible from the APB bus. The UART can still receive and transmit data but will not forward interrupt and DMA status to the CPU and the DMA controller. Frequency should be between 10-50 MHz.
UART_U_CLK	U_CLK	I	CGU	The U_CLK can be switched off by an external clock gate. In this case the UART cannot receive and transmit characters; the APB interface is still operational and can be accessed without deadlocking the APB bus. Frequency should be between 10-50 MHz.

2.4.2 Pin connections

Table 434. UART Signals to pins of the IC for the UART

Name	Type	Reset	Description
UART_RXD	I	-	Serial Data Receive
UART_TXD	O	-	Serial Data Transmit
mUART_CTS_N	I	-	Modem flow control: Clear to Send
mUART_RTS_N	O	-	Modem flow control: Request to Send

2.4.3 Interrupt Request Signals

The UART creates one interrupt signal. The UART has two interrupt modes. If the UART is configured as NHP interrupt mode (see register overview), it will comply to the NHP interrupt standard. If the MODE.NHP bit is cleared, the UART is in standard 'x50 mode and interrupts should be handled via the IIR/IER registers.

2.4.4 Reset Signals

The CGU provides one UART global asynchronous reset signal to the UART block.

2.4.5 DMA Transfer Signals

The UART module implements DMA flow control using following signals.

Table 435. DMA signals of the UART

Name	Type	Description
UART_RX_RDY_N	O	UART rx receive FIFO request information.
UART_TX_RDY_N	O	UART tx transmit FIFO request information.

3. Register overview

Table 436. Register overview: UART (register base address 0x1500 1000)

Name	R/W	Address Offset	Description
RBR	R	0x000	Receiver Buffer Register
THR	W	0x000	Transmitter Holding Register
DLL	R/W	0x000	Divisor Latch LSB
DLM	R/W	0x004	Divisor Latch MSB
IER	R/W	0x004	Interrupt Enable Register
IIR	R	0x008	Interrupt Identification Register
FCR	W	0x008	FIFO Control Register
LCR	R/W	0x00C	Line Control Register
MCR	R/W	0x010	Modem Control Register
LSR	R	0x014	Line Status Register
MSR	R	0x018	Modem status Register
SCR	R/W	0x01C	Scratch Register
ACR	R/W	0x020	Auto-baud Control Register
ICR	R/W	0x024	IrDA Control Register
FDR	R/W	0x028	Fractional Divider Register
-		0x02C	Reserved:
POP	W	0x030	NHP Pop Register
MODE	R/W	0x034	NHP Mode Selection Register
-		0x038-0xFD0	Reserved:
CFG	R	0xFD4	Configuration Register
INTCE	W	0xFD8	Interrupt Clear Enable Register
INTSE	W	0xFDC	Interrupt Set Enable Register
INTS	R	0xFE0	Interrupt Status Register

Table 436. Register overview: UART (register base address 0x1500 1000) *…continued*

Name	R/W	Address Offset	Description
INTE	R	0xFE4	Interrupt Enable Register
INTCS	W	0xFE8	Interrupt Clear Status Register
INTSS	W	0xFEC	Interrupt Set Status Register
-		0xFF0-0xFF8	Reserved:

4. Register description

4.1 Receive Buffer Register

Table 437. Receive Buffer Register (RBR, address 0x1500 1000)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7.0	RBRFirst	R	NA	In FIFO mode (FCR[0] = 1), top of Receiver FIFO; in non FIFO mode(FCR[0] = 0), value of buffer register

The Receiver Buffer Register (RBR) can only be accessed if the LCR, DLAB = 0, otherwise the DLL register will be accessed. The RBR register is read-only; writing this address will access the THR register.

In non FIFO mode (i.e. when FCR[0] is not set), the RBR can store one received character.The value of the RBR can be read via this register. After reading the register the value is undefined until the next character is received.

In FIFO mode i.e. when FCR[0] is set, the value in the register represents the first character in the UART FIFO. In this mode reading a character from the RBR will pop the character from the RBR FIFO. After the read operation the RBR will have the value of the next character in the FIFO. If the FIFO is empty the value of the register will be undefined until the next character is received.

If the UART is configured as Nexperia Home Platform (NHP) compliant by setting the MODE.NHP bit then the RBR register of the UART is protected against speculative read options and reading RBR will not pop the character from the top of the FIFO. Instead the top FIFO character will be popped by writing 0x1 to the POP register.

Bit 0 is the least significant bit and is the first bit serially received. If the UART transmit/receive word length as defined in LCR[1:0] is less than 8 bits per character then the upper bits of the RBR will be 0.

4.2 Transmitter Holding Register (THR)

Table 438. Transmitter Holding Register (THR, address 0x1500 1000)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7.0	THRLast	R	NA	In FIFO mode (FCR[0] = 1), top of Receiver FIFO; in non FIFO mode (FCR[0] = 0), value of buffer register

The Transmitter Holding Register (THR) can only be accessed if LCR, DLAB = 0, otherwise the DLL register will be accessed. The THR register is write-only; reading this address will access the RBR register.

In non FIFO mode (i.e. when FCR[0] is not set), the THR can store one transmit character (In an 'x50 configuration in non FIFO mode the full Tx FIFO depth can still be filled). The value for the THR can be written via this register. After writing the register the value will be transmitted on the UART serial out port (sout). New data can only be written in the register after old data has been transmitted. If new data is written before old data has been transmitted then the new data will be discarded.

In FIFO mode i.e. when FCR[0] is set, the value in the register represents the tail the UART's transmit FIFO/queue. In this mode writing a character to the THR will queue a character for transmission. If the FIFO is full any writes to the THR will be discarded.

Bit 0 is the least significant bit and is the first bit serially transmitted. If the UART transmit/receive word length as defined in LCR[1:0] is less than 8 bits per character then the upper bits of the THR will be ignored.

4.3 Divisor Latch register LSB (DLL)

Table 439. Divisor register Latch LSB (DLL, address 0x1500 1000)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7.0		R/W	0x1	Least significant byte of the divisor latch value

The Divisor Latch LSB register (DLL) can only be accessed if LCR.DLAB = 1. If LCR.DLAB = 0 reading from and writing to this address will access RBR and THR. The divisor value is 16 bits of which the most significant bits are stored in the DLM register and the least significant bits are defined in the DLL register.

For an optimal baud-rate the minimum value of the divisor should be set to 2.

The value of the DLL should not be modified while transmitting/receiving data or data may be lost or corrupted.

4.4 Divisor latch register MSB

The Divisor Latch MSB register (DLM) can only be accessed if LCR, DLAB = 1. If LCR.DLAB = 0 reading and writing this address will access the IER register.

The value of the DLM should not be modified while transmitting/receiving data or data may be lost or corrupted.

Table 440. Divisor latch register MSB (DLM, address 0x1500 1004)

Bit	Symbol	R/W	Reset Value	Description
31:8	reserved	R	0x0	reserved for future use
7:0	DLLVal	R/W	0x1	Most significant byte of the divisor latch value

4.5 Interrupt Enable Register (IER)

Table 441. Interrupt Enable Register (IER, address 0x1500 1004)

Bit	Symbol	R/W	Reset Value	Description
31:10	-			Reserved
9	ABTOIntEn	R/W	0x0	Auto-baud Time-Out Interrupt Enable.
8	ABEOIntEn	R/W	0x0	End of Auto-Baud Interrupt Enable.
7	CTSIntEn	R/W	0x0	If auto-cts mode is enabled this enables/disables themodem status interrupt generation on a cts_ansignal transition. If auto-cts mode is disabled acts_an transition will generate an interrupt ifMSIntEn is set.
6:4				Reserved
3	MSIntEn	R/W	0x0	Modem Status interrupt enable
2	RLSIntEn	R/W	0x0	Receiver Line Status interrupt enable
1	THREIntEn	R/W	0x0	Transmitter Holding Register Empty interrupt enable
0	RDAIntEn	R/W	0x0	Receive Data Available interrupt enable,

The IER can only be accessed if LCAR, DLAB = 0, otherwise the DLM register will be accessed. The IER register can be read and written.

The IER masks the interrupts from receiver ready, transmitter empty, line status and modem status registers. These interrupts would normally be seen in the IIR register and on the interrupt request output pin (uart_intreq).

In normal operation a cts_an signal transition will generate a Modem Status interrupt unless the interrupt has been disabled by clearing the MSIntEn bit in the IER register. In auto-cts mode a transition on the cts_an bit will trigger an interrupt only if both the MSIntEn and the CTSIntEn bits are set.

In Nexperia Home Platform compliant systems SW should use the INTSS and INTSE registers for implementing interrupt service routines.

4.6 Interrupt Identification Register (IIR)

Table 442. Interrupt Identification Register (IIR, address 0x1500 1008)

Bit	Symbol	R/W	Reset Value	Description
31:10	-			Reserved
9	ABTOInt	R	0x0	Auto-baud time-out interrupt True if auto-baud hastimed out and interrupt is enabled.
8	ABEOInt	R	0x0	End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.
7:6	FIFOEn	R	0x0	Copies of FCR[0]
5:4				Reserved
3:1	IntId	R	0x0	Interrupt identification
0	IntStatus	R	0x0	Interrupt status. If 0 then interrupt is pending; if 1 no interrupt is pending

Bit [9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

Bits [7:6] of this register are copies of the FCR.FIFOEn bit.

If the IntStatus bit is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt according to Table 21–443. If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero.

Table 443. Interrupt Identification and Priorities

INTD	Priority	Type	SET	CLEAR
0x3	1 (highest)	Receiver Line Status	Set on an overrun error, parity error, framing error or break indication	Reading the LSR
0x2	2	Received Data Available	Receiver data available in '450 mode or trigger level reached in FIFO mode	Reading the RBR in '450mode; in non '450 mode the FIFO level drops below the trigger level
0x6	2	Character time-out; only generated if the FIFO is enabled	Set on an overrun error, parity error, framing error or break indication	Reading the RBR
0x1	3	Transmitter Holding Register empty	Transmitter Holding Register empty (THRE)	Reading the IIR if IIR value is 0x1, or writing to THR
0x0	4 (lowest)	Modem status	Set on transition of cts_an	Reading the MSR

The IIR only indicates an interrupt if the corresponding bit in the IER register is set.

If multiple interrupts are pending only the highest priority interrupt will be indicated in the IntId bits of the IIR. Only after clearing a higher priority interrupt a low priority interrupt will be indicated. The UART provides four prioritized levels of interrupts:

Priority 1 - Receiver line status (highest priority)

Priority 2 - Receiver data available or receiver character time-out

Priority 3 - Transmitter holding register empty

Priority 4 - Modem status (lowest priority)

Interrupts can be cleared by reading the register causing the interrupt. The LSR, THRE can be cleared either by reading the IIR or writing to the THR register. The LSR, THRE interrupt will only be cleared by an IIR read if the LSR, THRE interrupt is the highest pending interrupt.

This interrupt is activated when THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the THR FIFO a chance to fill up with data to eliminate many LSR, THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE=1and there have not been at least two characters in the LSR, THR at one time since the last LSR, THRE=1 event.

This delay is provided to give CPU time to write data to THR without a THRE interrupt to decode and service. A LSR, THRE interrupt is set immediately if the THR FIFO has held two or more characters at one time and currently, the THR is empty.

Note: the interrupt register can indicate multiple interrupts concurrently: the time-out and end of auto-baud interrupts and one of the non auto-baud interrupts.

In Nexperia Home Platform compliant systems SW should use the INTSS and INTSE register for implementing interrupt service routines since the LSR, THRE bit is not protected for speculative read operations

4.7 FIFO Control Register (FCR)

Table 444. FIFO Control Register (FCR, address 0x1500 1008)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7:6	RxTrigLevel	W	0x0	Receiver trigger level selection; The Rx FIFO can store 32 characters in total. RxTrigLevel = 0x0: trigger point at character 1 RxTrigLevel = 0x1: trigger point at character 16 RxTrigLevel = 0x2: trigger point at character 32 Rx TrigLevel = 0x3: trigger point at character 56
5:4				Reserved
3	DMAMode	W	0x0	DMA mode select. When IIR, FIFOEn is set, setting DMAMode causes the rx_rdy_n, and tx_rdy_n to change from mode 0 to mode 1.
2	TxFIFORst	W	0x0	Transmitter FIFO reset. When set clears all bytes in the transmit FIFO and resets its counter to 0. The TSR is not cleared. The logic 1 that is written to this bit position is self-clearing.
1	RxFIFORst	W	0x0	Receiver FIFO reset. When set clears all bytes in the receiver FIFO and resets its counter.The RSR is not cleared. The logic 1 that is written to this bit position is self-clearing.
0	FIFOEnable	W	0x0	Transmit and receive FIFO enable. 0: UART Rx and Tx FIFOs disabled. 1: UART Rx and Tx FIFOs enabled and other FCR bits activated.

The FIFO Control Register (FCR) is write-only; reading this address will access the IIR register. The value of the FCR should not be modified while receiving/transmitting data or data might get lost or corrupted.

4.8 Line Control Register (LCR)

Table 445. Line Control Register (LCR, address 0x1500 100C)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7	DLAB	R/W	0x0	Divisor Latch Access bit. If set enables access to the divisor latch registers, if cleared enables access to the RBR, THR, IER registers.
6	BrkCtrl	R/W	0x0	Break control bit. BrkCtrl is set to force a break condition; i.e., a condition where sout is forced to the spacing (low) state. WhenBrkCtrl is cleared, the break condition is disabled and has no affect on the transmitter logic; it only affects the serial output.
5	ParStick	R/W	0x0	If parity is enabled by setting the ParEn bit the ParStick bit enables the stick parity mode.
4	ParEven	R/W	0x0	If parity is enabled by setting the ParEn bit the ParEvenSel bit selects between even, odd, stick 0 and stick 1 parity.
3	ParEn	R/W	0x0	Parity enable. Setting this bit appends a parity bit to each transmission and the receiver checks each word for parity errors. Clearing this bit disables parity transmission and receiver parity checking.
2	StopBitNum	R/W	0x0	Number of stop bits selector. The number of stopbits depends on the value of WdLenSel and StopBitNum.
1:0	WdLenSel	R/W	0x0	Word length selector

Table 446. UART Parity Generation Options

LCR.PAREN	LCR.PAREVEN	LCR.PARSTICK	DESCRIPTION
0	x	x	Parity is disabled. No parity bit will be transmitted and no parity will be checked during receive.
1	0	0	Odd parity mode. Data+parity bit will have an odd number of 1s
1	1	0	Even parity mode. Data+parity bit will have an even number of 1s
1	0	1	Stick 1 mode: parity is always 1
1	1	1	Stick 0 mode: parity is always 0

The next table lists the word length and stop bit options available in the UART. Apart from the BrkCtrl bit the value of the LCR should not be modified while transmitting/receiving data or data might get lost or corrupted.

Table 447. UART Character Length and Stop Bits Generation Options

LCR.WDLENSEL	LCR.STOPBIT	CHARACTER LENGTH	NUMBER OF STOP BITS
0x0	0	5	1
0x1	0	6	1
0x2	0	7	1
0x3	0	8	1
0x0	1	5	1.5

Table 447. UART Character Length and Stop Bits Generation Options

LCR.WDLENSEL	LCR.STOPBIT	CHARACTER LENGTH	NUMBER OF STOP BITS
0x1	1	6	2
0x2	1	7	2
0x3	1	8	2

4.9 MCR (Modem Control Register)

Table 448. Modem Control Register (MCR, address 0x1500 1010)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7	AutoCTSEn	R/W	0x0	Auto-cts flow control enable
6	AutoRTSEn	R/W	0x0	Auto-rts flow control enable
5				Reserved
4	LoopEn	R/W	0x0	Loop-back mode enable. When LoopEn is set, registers MCR[3:0] are forced to "0000"
3	OUT2	R/W	0x0	Inverse control for the out2_n output
2	OUT1	R/W	0x0	Inverse control for the out1_n output
1	RTS	R/W	0x0	Request To Send. If the AutoRTSEn bit is set the RTS bit is read-only and will reflect the current state of the rts_n output. If the AutoRTSEn is cleared then the RTS bit is the inverse control for the rts_n output.
0	DTR	R/W	0x0	Inverse control for the Data Terminal Ready output

4.10 LSR (Line Status Register)

Table 449. Line Status Register) (LSR, address 0x1500 1014)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7	RxEr	R	0x0	Error in receiver FIFO.Logic 0 = No error (normal default condition).Logic 1 = At least one parity error, framing error or break indication is in the current FIFO data. This bit is cleared when LSR register is read.
6	TEMT	R	0x1	Transmitter empty: TSR and THR are empty. This bit is set to a logic 1 whenever the transmit holding register and the transmit shift register are both empty. It is reset to logic 0 whenever either the THRor TSR contains a data character. In the FIFO mode, this bit is set to '1' whenever the transmit FIFO and transmit shift register are both empty.

Table 449. Line Status Register) (LSR, address 0x1500 1014) *…continued*

Bit	Symbol	R/W	Reset Value	Description
5	THRE	R	Undefined	Transmitter Holding Register empty. This bit indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to CPU when the THR interrupt enable is set. The THRE bit is set to a logic 1 when a character is transferred from the transmit holding register into the transmitter shift register. The bit is reset to a logic 0 concurrently with the loading of the transmitter holding register by the CPU. In the FIFO mode, this bit is set when the transmit FIFO is empty; it is cleared when at least 1 byte is written to the transmit FIFO.
4	BI	R*	Undefined	Break indication.Logic 0 = No break condition (normal default condition).Logic 1 = The receiver received a break signal (RX was a logic 0 for one character frame time). In the FIFO mode, only one break character is loaded into the FIFO. This bit is cleared when the LSR register is read.
3	FE	R*	Undefined	Framing error.Logic 0 = No framing error (normal default condition).Logic 1 = Framing error. The receive character didn't have a valid stop bit(s).n the FIFO mode, this error is associated with the character at the top of the FIFO.
2	PE	R*	Undefined	Parity error.Logic 0 = No parity error (normal default condition).Logic 1 = Parity error. The receive character does not have correct parity information and is suspect.In the FIFO mode, this error is associated with the character at the top of the FIFO.This bit is cleared when the LSR register is read.
1	OE	R*	Undefined	Overrun error. Logic 0 = No overrun error (normal default condition).Logic 1 = Overrun error. A data overrun error occurred in the receive shift register. This happens when additional data arrives while the FIFO is full. In this case, the previous data in the shift register is overwritten.Note that under this condition, the data byte in the receive shift register is not transferred into the FIFO; therefore the data in the FIFO is not corrupted by the error.This bit is cleared when the LSR register is read
0	DR	R	0x0	Data readyLogic 0 = No data in receive holding register or FIFO (normal default condition).Logic 1 = Data has been received and is saved in the receive holding register or FIFO.

Note that in some industry standard UARTs the MSR register is read/writable.

For these UARTs writing is only intended to be used for factory testing which is why the APB UART does not support writing this register.

Bytes are transferred from THR to TSR as soon as 50% of the start bit has been transmitted by the TSR. The LSR.THRE bit is updated as soon as a byte has been transferred from THR to TSR or if a byte is written into the THR. The LSR.TEMT bit is updated as soon as 50% of the first stop bit has been transmitted or if a byte is written into the THR.

* In Nexperia Home Platform compliant systems SW should not use the BI/FE/PE/OE bits in the LSR since these are sensitive to speculative read operations. The corresponding INTSS bits should be used instead. Reading LSR, IIR or INTSS will not have side effects on the value of the bits in INTSS.

4.11 MSR (Modem Status Register)

Table 450. Modem Status Register (MSR, address 0x1500 1018)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7	DCD	R	0x0	Data Carrier Detect. Normally this bit is the complement of the dcd_an input. In the loop-back mode this bit is equivalent to the OUT2 bit in the MCR register.
6	RI	R	0x0	Ring Indicator. Normally this bit is the complement of the ri_an input. In the loop-back mode this bit is equivalent to the OUT1 bit in the MCR register.
5	DSR	R	0x0	Data Set Ready. Normally this bit is the complement of the dsr_an input. In loop-back mode this bit is equivalent to the DTR bit in the MCR register.
4	CTS	R*	0x0	Clear To SendCTS functions as a modem flow control signal input if it is enabled. Flow control (when enabled) allows starting and stopping the transmissions based on the external modem CTS signal. A logic 1 at the CTS pin will stop the UART transmissions as soon as current character has finished transmission.Normally CTS is the complement of the cts_an input. However, in the loop-back mode, this bit is equivalent to the RTS bit in the MCR register.
3	DDCD	R*	0x0	Delta Data Carrier Detect. This bit is set as soon as DCD changes its value.The bit is cleared by a MSR read.
2	TERI	R*	0x0	Trailing Edge Ring Indicator. This bit is set as soon as RI transitions from a HIGH to a LOW value. The bit is cleared by a MSR read.
1	DDSR	R*	0x0	Delta Data Set Ready. This bit is set as soon as DSR changes it value. The bit is cleared by a MSR read.
0	DCTS	R*	0x0	Delta Clear To Send. This bit is set as soon as CTS changes its value. The bit is cleared by a MSR read.

Note that in some industry standard UARTs the MSR register is read/writable. For these UARTs writing is only intended to be used for factory testing which is why the APB UART does not support writing this register.

In Nexperia Home Platform compliant systems SW should not use the DDCD/TERI/DDSR/DCTS bits in the MSR since these are sensitive to speculative read operations. The corresponding INTSS bits should be used instead. Reading MSR does not affect the value of the bits in INTSS.

4.12 SCR (Scratch Register)

Table 451. Scratch Register (SCR, address 0x1500 101C)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7.0	SCRVal	R/W	0x0	Scratch Value

The scratch register is not used by the UART it can be used by software as a temporary storage.

4.13 ACR (Auto-Baud Control Register)

Table 452. Auto-Baud Control Register (ACR, address 0x1500 1020)

Bit	Symbol	R/W	Reset Value	Description
31:10	-			Reserved
9	ABTOIntCl	W	0x0	Auto-baud time-out interrupt clear. Writing a 1 will clear the corresponding interrupt in the IIR.Should be explicitly written with '0' to reset the bit, after clearing the interrupt
8	ABEOIntClr	W	0x0	End of auto-baud interrupt clear. Writing a 1 will clear the corresponding interrupt in the IIR.
7:0				Reserved

4.14 ICR (IrDA Control Register)

Table 453. IrDA Control Register (ICR, address 0x1500 1024)

Bit	Symbol	R/W	Reset Value	Description
31:6	-			Reserved
5:3	PulseDiv	R/W	0x0	Configures of pulse in fixed pulse width mode. Only relevant if FixPulseEn is set.
2	FixPulseEn	R/W	0x0	Enables IrDA fixed pulse width mode
1	IrDAInv	R/W	0x0	If true the serial input is inverted; if false the input is not inverted. The serial output is not affected by the value of this bit.
0	IrDAEn	R/W	0x0	If true, enable IrDA; if false disable IrDA and pass UART sin/sout transparently.

The value of the ICR should not be modified while transmitting/receiving data or data may be lost or corrupted.

The IrDA.PulseDiv bits are used for configuring the pulse width of the fixed pulse width mode of the UART. The value of these bits should be configured such that the resulting pulse width is at least 1.63 µs T_{U_CLK} is the period time of clock U_CLK.

Table 454. IrDA Pulse Width

IRDA.FIXPULSEEN	IRDA.PULSEDIV	IRDA TRANSMITTERPULSE WIDTH [US]
0	x	3 / (16 * baud-rate)
1	0	2 * T_{u_clk}
1	1	4 * T_{u_clk}
1	2	8 * T_{u_clk}
1	3	16 * T_{u_clk}
1	4	32 * T_{u_clk}
1	5	64 * T_{u_clk}
1	6	128 * T_{u_clk}
1	7	256 * T_{u_clk}

4.15 FDR (Fractional Divider Register)

Table 455. Fractional Divider Register (FDR, address 0x1500 1028)

Bit	Symbol	R/W	Reset Value	Description
31:8	-			Reserved
7:4	MulVal	R/W	0x1	Baud-rate pre-scaler multiplier value
3:0	DivAddVal	R/W	0x0	Baud-rate generation pre-scaler divisor value

The FDR register controls the clock pre-scaler for the baud rate generation. The clock can be pre-scaled by a value of MulVal/(MulVal+DivAddVal).

The value of MulVal and DivAddVal should comply to the following expressions:

- 0 < MulVal <= 15
- 0 <= DivAddVal <= 15

If the register value does not comply to the above expression, then the fractional divider output is undefined.

If DivAddVal is zero then the fractional divider is disabled and the clock will not be divided.

The value of the FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

4.16 NHP POP Register

Table 456. NHP POP Register (POP, address 0x1500 1030)

Bit	Symbol	R/W	Reset Value	Description
31:1	-			Reserved
0	PopRBR	W	0x0	Setting this bit will pop the first item from the Receiver Buffer Register's FIFO as if RBR were read in non NHP mode. The bit will clear automatically.

The POP register is a write-only register. Reading the register will return 0x0.

In 'NHP mode' which protects the UART from speculative reads the PopRBR bit in the POP register can be used to pop the first data element from the RBR FIFO as if the RBR register were read in non 'NHP mode'.

Typically in a NHP mode application each RBR read operation is followed by writing the PopRBR bit to remove the top read data from the FIFO.

4.17 Mode Selection Register

Table 457. Mode Selection Register (Mode, 0x1500 1034)

Bit	Symbol	R/W	Reset Value	Description
31:1	-			Reserved
0	NHP	R/W	0x0	Setting this bit will switch the UART in 'NHP mode' and protect the UART RBR from speculative reads; RBR needs to be popped explicitly via the POP register. Intreq is derived from INTS instead of IIR. Clearing the bit will switch the UART in 'x50 mode.

After reset the UART will be in non NHP compliant, normal 'x50 compliant mode in which read operations will have side effects. After setting the NHP bit the UART will be in NHP compliant mode which will have two implications:

- RBR read operations will have no side effects. LSR, IIR and MSR reads will still have side effects, in NHP mode the speculative read sensitive bits in the LSR/IIR/MSR registers should not be used by SW. Instead SW should use the INTS register for determining the state of modem, line and interrupt.
- In NHP mode the uart_intreq output will be derived from the INTS register instead of the IIR register.

4.18 CFG (Configuration Register)

Table 458. Configuration Register (CFG, address 0x1500 1FD4)

Bit	Symbol	R/W	Reset Value	Description
31:13	-			Reserved
12	HasIrDA	R	0x1	0x0 = IrDA module not included 0x1 = IrDA module included
11:10	-			Reserved
9	HasLevel	R	0x0	0x0 = FIFO level interface not included 0x1 = FIFO level interface included
8	HasDMA	R	<tbd>	0x0 = ARM DMA interface not included 0x1 = ARM DMA interface included
7:6	-			Reserved
5:4	Modem	R	0x1	0x0 = Modem interface not included 0x1 = cts and rts modem interface included 0x2 = All modem interface signals included
3:2	-			Reserved
1:0	Type	R	<tbd>	0x0 = UART type '450. 0x1 = UART type '550 0x2 = UART type '650 0x3 = UART type '750

Typically the register is used for run-time configuration of the software driver.

4.19 INTCE (Interrupt Clear Enable Register)

Table 459. Interrupt Clear Enable Register (INTCE, address 0x1500 1FD8)

Bit	Symbol	R/W	Reset Value	Description
31:16	-			Reserved
15	OEIntEnClr	W	0x0	Overrun Error Interrupt Enable Clear
14	PEIntEnClr	W	0x0	Parity Error Interrupt Enable Clear
13	FEIntEnClr	W	0x0	Frame Error Interrupt Enable Clear
12	BIIntEnClr	W	0x0	Break Indication Interrupt Enable Clear
11-10	reserved	W	0x0	Reserved
9	ABTOIntEnClr	W	0x0	Auto-Baud Time-Out Interrupt Enable Clear
8	ABEOIntEnClr	W	0x0	End of Auto-Baud Interrupt Enable Clear
7	-			Reserved
6	RxDAIntEnClr	W	0x0	Receiver Data Available Interrupt. Enable Clear
4	THREIntEnClr	W	0x0	Transmitter Holding Register Empty Interrupt Enable Clear
3	DDCDIntEnClr	W	0x0	Delta Data Carrier Detect Interrupt Enable Clear
2	TERIIntEnClr	W	0x0	Trailing Edge Ring Indicator Interrupt Enable Clear
1	DDSRIntEnClr	W	0x0	Delta Data Set Ready Interrupt Enable Clear
0	DCTSIntEnClr	W	0x0	Delta Clear To Send Interrupt Enable Clear

The register bits are one-shot registers and automatically cleared.

4.20 INTSE (Interrupt Set Enable Register)

Table 460. Interrupt Set Enable Register (INTSE, address 0x1500 1FDC)

Bit	Symbol	R/W	Reset Value	Description
31:16	-			Reserved
15	OEIntEnSet	W	0x0	Overrun Error Interrupt Enable Set
14	PEIntEnSet	W	0x0	Parity Error Interrupt Enable Set
13	FEIntEnSet	W	0x0	Frame Error Interrupt Enable Set
12	BIIntEnSet	W	0x0	Break Indication Interrupt Enable Set
11-10	-			Reserved
9	ABTOIntEnSet	W	0x0	Auto-Baud Time-Out Interrupt Enable Set
8	ABEOIntEnSet	W	0x0	End of Auto-Baud Interrupt Enable Set
7	-			Reserved
6	RxDAIntEnSet	W	0x0	Receiver Data Available Interrupt. Enable Set
5	RxTOIntEnSet	W	0x0	Receiver Time-Out Interrupt Enable Set
4	THREIntEnSet	W	0x0	Transmitter Holding Register Empty Interrupt Enable Set
3	DDCDIntEnSet	W	0x0	Delta Data Carrier Detect Interrupt Enable
2	TERIIntEnSet	W	0x0	Trailing Edge Ring Indicator Interrupt Enable Set
1	DDSRIntEnSet	W	0x0	Delta Data Set Ready Interrupt Enable Set
0	DCTSIntEnSet	W	0x0	Delta Clear To Send Interrupt Enable Set

The register bits are one-shot registers and automatically cleared.

4.21 INTS (Interrupt Status Register)

Table 461. Interrupt Status Register (INTS, address 0x1500 1FE0)

Bit	Symbol	R/W	Reset Value	Description
31:16	-			Reserved
15	OEInt	R	0x0	Overrun Error Interrupt. Set if RBR overrun. Cleared by setting INTCS.OEIntClr
14	PEInt	R	0x0	Parity Error Interrupt. Set if top of RBR has parity error. Cleared by popping RBR.
13	FEInt	R	0x0	Frame Error Interrupt. Set if top of RBR has framing error. Cleared by popping RBR.
12	BIInt	R	0x0	Break Indication Interrupt. Set if top of RBR has break indication. Cleared by popping RBR.
11-10	-			Reserved
9	ABTOInt	R	0x0	Auto-Baud Time-Out Interrupt. Set on auto-baud time-out. Cleared by setting INTCS.ABTOIntClr
8	ABEOInt	R	0x0	End of Auto-Baud Interrupt. Set at end of auto-baud. Cleared by setting INTCS.ABEOIntClr.
7	-			Reserved
6	RxDAInt	R	0x0	Receiver Data Available Interrupt. Cleared by popping RBR below FIFO level.
5	RxTOInt	R	0x0	Receiver Time-Out Interrupt. Cleared by popping RBR, receiving new character or setting the INTCS.RxTOIntClr bit.
4	THREInt	R	0x0	Transmitter Holding Register Empty Interrupt. Set if THR is empty. Cleared by writing THR or setting INTCS.THREIntClr
3	DDCDInt	R	0x0	Delta Data Carrier Detect Interrupt. Set on change of DCD. Cleared by setting INTCS.DDCSIntClr.
2	TERIInt	R	0x0	Trailing Edge Ring Indicator Interrupt. Set on falling edge of RI. Cleared by setting INTC.TERIIntClr.
1	DDSRInt	R	0x0	Delta Data Set Ready Interrupt. Set on change of DSR. Cleared by setting INTCS.DDSRIntClr.
0	DCTSInt	R	0x0	Delta Clear To Send Interrupt. Set on change of CTS. Cleared by setting INTCS.DCTSIntClr.

Only a limited number of bits from the NHP Interrupt Status Register can be set and cleared by software. The INTS.RxDAInt/PEInt/FEInt/BIInt bits cannot be set/cleared from SW. These bits are controlled by HW.

4.22 INTE (Interrupt Enable Register)

Table 462. Interrupt Enable Register (INTE, address 0x1500 1FE4)

Bit	Symbol	R/W	Reset Value	Description
31:16	-			Reserved
15	OEIntEn	R	0x0	Overrun Error Interrupt Enable
14	PEIntEn	R	0x0	Parity Error Interrupt Enable

Table 462. Interrupt Enable Register (INTE, address 0x1500 1FE4) *…continued*

Bit	Symbol	R/W	Reset Value	Description
13	FEIntEn	R	0x0	Frame Error Interrupt Enable
12	BIIntEn	R	0x0	Break Indication Interrupt Enable
11-10	-			Reserved
9	ABTOIntEn	R	0x0	Auto-Baud Time-Out Interrupt Enable
8	ABEOIntEn	R	0x0	End of Auto-Baud Interrupt Enable
7	-			Reserved
6	RxDAIntEn	R	0x0	Receiver Data Available Interrupt. Enable
5	RxTOIntEn	R	0x0	Receiver Time-Out Interrupt Enable
4	THREIntEn	R	0x0	Transmitter Holding Register Empty Interrupt Enable
3	DDCDIntEn	R	0x0	Delta Data Carrier Detect Interrupt Enable
2	TERIIntEn	R	0x0	Trailing Edge Ring Indicator Interrupt Enable
1	DDSRIntEn	R	0x0	Delta Data Set Ready Interrupt Enable
0	DCTSIntEn	R	0x0	Delta Clear To Send Interrupt Enable

4.23 INTCS (Interrupt Clear Status Register)

Table 463. Interrupt Clear Status Register (INTCS, address 0x1500 1FE8)

Bit	Symbol	R/W	Reset Value	Description
31:16	-			Reserved
15	OEIntClr	W	0x0	Overrun Error Interrupt Clear
14:10	-			Reserved
9	ABTOIntClr	W	0x0	Auto-Baud Time-Out Interrupt Clear. Alias of the ACR.ABTOIntClr bit
8	ABEOIntClr	W	0x0	End of Auto-Baud Interrupt Clear.Alias of the ACR.ABEOIntClr bit
7:6	-			Reserved
5	RxTOIntClr	W	0x0	Receiver Time-Out Interrupt Clear
4	THREIntClr	W	0x0	Transmitter Holding Register Empty Interrupt Clear
3	DDCDIntClr	W	0x0	Delta Carrier Detect Interrupt Clear
2	TERIIntClr	W	0x0	Trailing Edge Ring Indicator Interrupt Clear
1	DDSRIntClr	W	0x0	Delta Data Set Ready Interrupt Clear
0	DCTSIntClr	W	0x0	Delta Clear To Send Interrupt Clear

The register bits are one-shot registers and automatically cleared.

4.24 INTSS (Interrupt Set Status Register)

Table 464. Interrupt Set Status Register (INTSS, address 0x1500 FEC)

Bit	Symbol	R/W	Reset Value	Description
31:16	-			Reserved
15	OEIntSet	W	0x0	Overrun Error Interrupt Set
14:10	reserved	W	0x0	reserved for future use
9	ABTOIntSet	W	0x0	Auto-Baud Time-Out Interrupt Set
8	ABEOIntSet	W	0x1	End of Auto-Baud Interrupt Set
7:6	-			Reserved
5	RxTOIntSet	W	0x0	Receiver Time-Out Interrupt Set
4	THREIntSet	W	0x0	Transmitter Holding Register Empty Interrupt Set
3	DDCDIntSet	W	0x0	Delta Data Carrier Detect Interrupt Set
2	TERIIntSet	W	0x0	Trailing Edge Ring Indicator Interrupt Set
1	DDSRIntSet	W	0x0	Delta Data Set Ready Interrupt Set
0	DCTSIntSet	W	0x0	Delta Clear To Send Interrupt Set

The register bits are one-shot registers and automatically cleared.

5. Functional Description

5.1 Internal Architecture

Figure 21–62 shows the internal architecture of the UART.

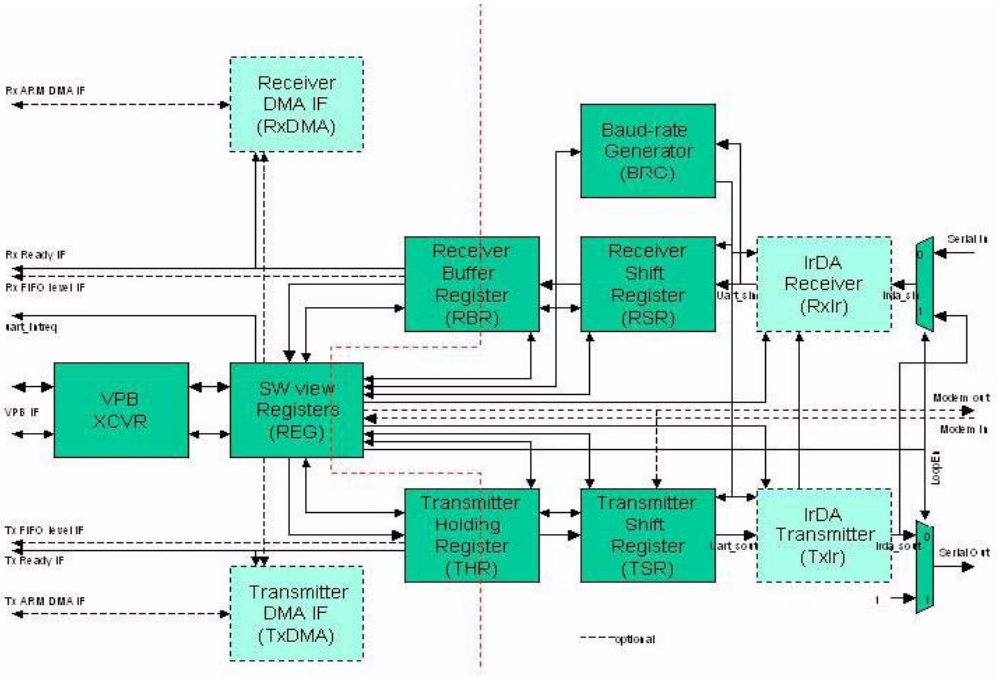


Fig 62. UART internal architecture

The internal architecture of the UART consists of the following modules:

- A APB transceiver
- The register (REG) module implementing the SW view of the UART and generating the control signals for the UART's data-path; the register module controls and observes the optional modem interface inputs and outputs
- The receiver buffer register (RBR) module implementing the receiver buffer and the receiver FIFO
- The receiver shift register (RSR) module converting the serial input stream into bytes while taking care of start, stop and parity bits
- The optional receiver IrDA interface (RxIr) converts a serial IrDA stream into a serial UART stream. The RxIr has a bypass mode to directly forward the serial input to the RSR
- The transmit holding register (THR) implementing the transmitter buffer and transmitter FIFO
- The transmit shift register (TSR) converting bytes from the THR into a serial output stream while taking care of start, stop and parity bits
- The optional transmitter IrDA interface (TxIr) converts a serial UART stream into a serial IrDA stream. The TxIr has a bypass mode to directly forward the TSR output to the serial output pin.
- The baud-rate generator generating the baud-rate for the serial interface.

The dashed line in Figure 21–62 shows the clock domain crossing in the design. Only the SW register module, the transmit holding register and the receive buffer register modules have dual clocks and implement clock crossings.

5.2 Serial Interface

Figure 21–63 shows the protocol on the serial interfaces (sin_a, sout) of the UART.

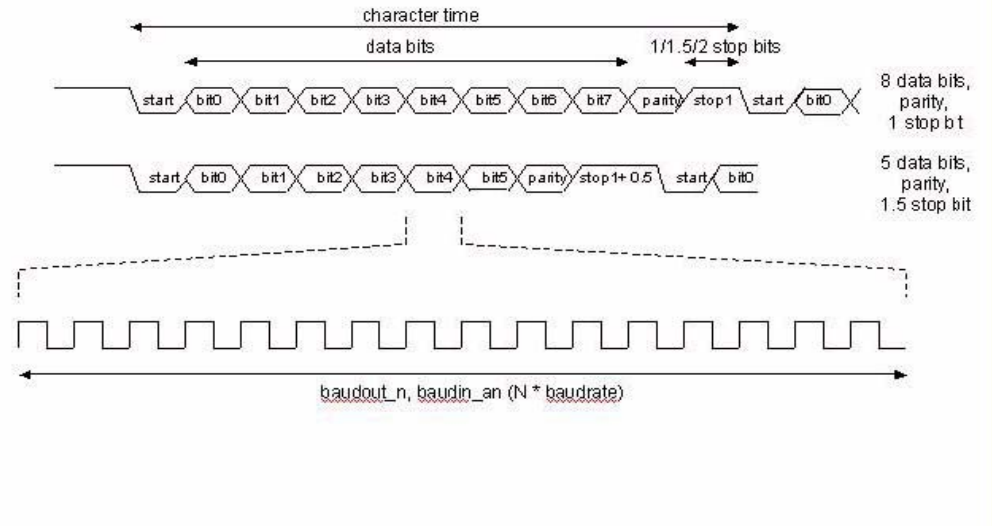


Fig 63. UART Serial Interface Protocol

The UART line control register configures the following:

- The number of bits in a receive and transmit character

  The parity type: no/even/odd/stick-high/stick-low parity

  The number of stop bits: 1, 1.5 or 2.

The length of each bit on the serial interface corresponds to 16*baudout_n pulses for transmit and 16*baudin_an pulses for receive

The character time of a UART is defined as the time it takes to transmit or receive a single character including start bit, optional parity bit and all stop bits.

5.3 Basic receive and transmit

Figure 21–64 illustrates a basic UART transmission and reception.

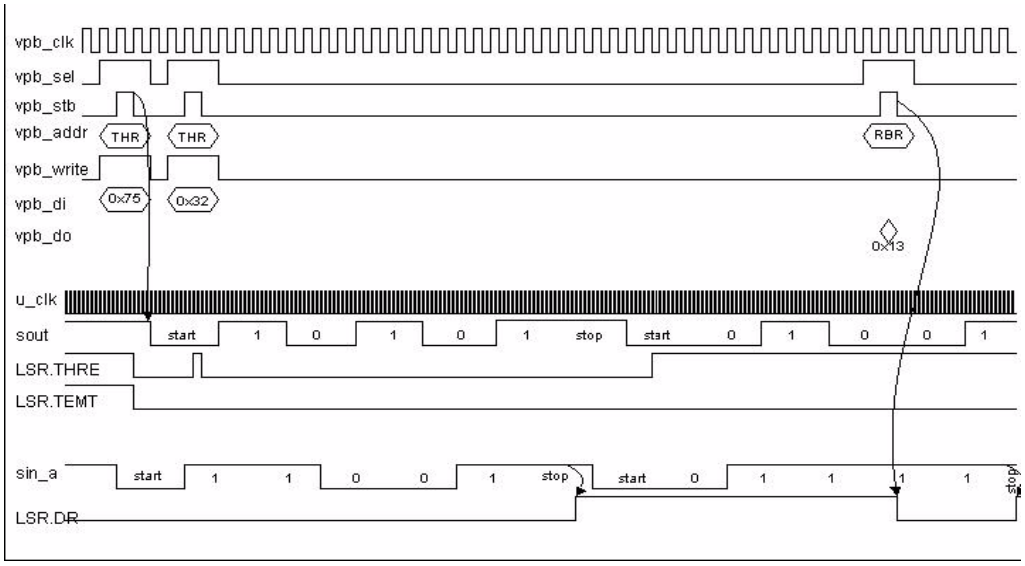


Fig 64. Basic data transmission and reception

In this example two characters are written via the APB interface in the THR FIFO. After some clock domain synchronization delays the TSR will read the character from the FIFO and start transmission.

After writing the THR the THRE (Transmitter Holding Register Empty) bit in the Line status Register will be de-asserted together will the TEMT (Transmitter Empty) bit in the same register. The THRE bit will get asserted as soon as the character is transferred from the THR to the TSR i.e. as soon as THR is empty again. The TEMT bit will remain de-asserted as long as there is a character in the THR or TSR.

In this example only the five least significant bits in the byte get transmitted i.e. the LCR.WdLenSel bits are set to 0x0.

While transmitting data the UART receives data on its sin_a input. After receiving the stop bit the data is written in the RBR and the Data Ready bit in the Line Status Register is set (LSR.DR = 1). The CPU reads the character from the RBR which clears the LSR.DR bit.

5.4 Loop-back mode

The UART's loop-back mode is activated by setting the MCR.LoopEn bit. In loop-back mode the UART's serial output is looped back to the serial input. Activating the modem loop-back will also loop-back the modem output signals to the modem inputs signals as depicted in Figure 21–64.These loop-backs are realized internally in the UART. In loop-back mode the UART's output pins will be inactive (high). In loop-back mode the UART's input will be disconnected.

Table 465. Loop-back mode UART's internal input to output relation

Output	Input
sout_n	sin_an
rts_an	cts_an

Note, Figure 21–64 only depicts the loop-back logic for the sin_a/sout signals, the modem interface has similar loop-back logic. In case of a partial (cts/rts only) modem interface or a UART configured without a modem interface the missing modem interface signals will still be looped back in loop-back mode internally; in functional mode the missing modem interface inputs are tied to a high, inactive value internally.

Also note that loop-back can be used in IrDA mode on the IrDA signals.

5.5 Break mode

Transmission of a break condition is controlled by the LCR.BrkCtrl bit. When enabled (setting the bit), the Break control bit causes a break condition to be transmitted. While break control is enabled the UART's sout output is forced to a low (logic 0) state. This condition exists until disabled by setting LCR.BrkCtrl to a logic 0. On the receiver side break conditions are detected in the LSR.BI bit. In the FIFO mode, only one break character is loaded into the FIFO. Receiving a break condition will trigger a Receiver Line Status interrupt if the interrupt is enabled.

5.6 Baud-rate generation

The UART baud-rate generator generates the UART baud-rate for transmission. A single character bit is transmitted in N*baudout_n clock cycles, where N = 6, 6.125, ...,16. The UART receiver uses N*baudin_an to capture a single character bit. When the transmitter and receiver use the same baud-rate, an external loop-back can connect the baudout_n output to the baudin_an input. Figure 21–65 illustrates the UART baud-rate generator architecture.

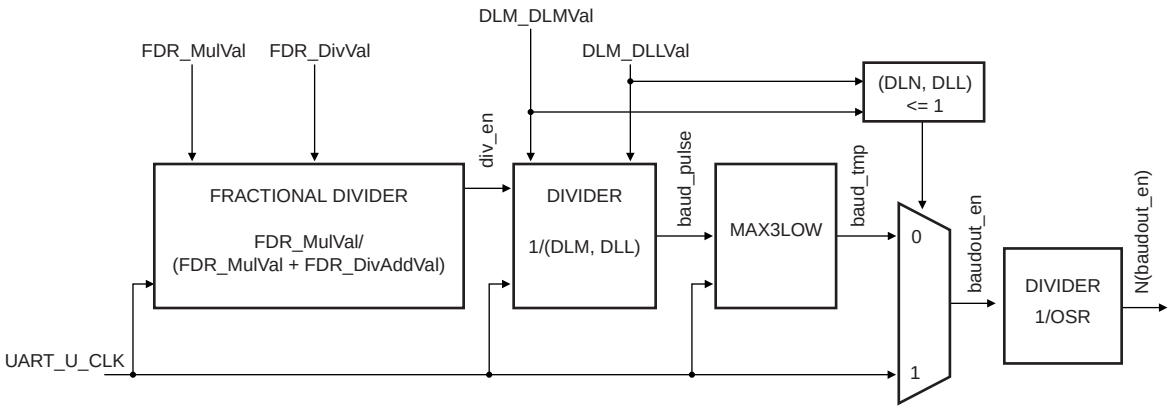


Fig 65. Baud-rate generation architecture

The baud-rate generator consists of the following:

- A MulVal/(MulVal+DivAddVal) fraction divider. The fractional divider can multiply the u_clk frequency by an MulVal/(MulVal+DivAddVal) ratio, where MulVal and DivAddVal both are 4 bits values. The MulVal/(MulVal+DivAddVal) ratio can be programmed through the Fractional Divider Register (FDR). The fractional divider can be bypassed by defining DivAddVal to zero

- A 1/X divider. The 1/X can divide the clock generated by the fractional divider by another 16 bits value X. The 16 bits X divisor value consists of two parts: a most significant part (Divisor Latch MSB) and a least significant part (Divisor Latch LSB). DLL and DLM can be programmed via the corresponding registers
- A max2low circuit which limits the low time of the baudout_n to a maximum of two u_clk cycles.
- A multiplexer and control logic passes the u_clk to the baudout_n output in scan test mode or if the divisor value is smaller than or equal to one otherwise the baudout_n will be generated by the dividers.

Figure 21–66 illustrates three examples of the baud-rate generator.

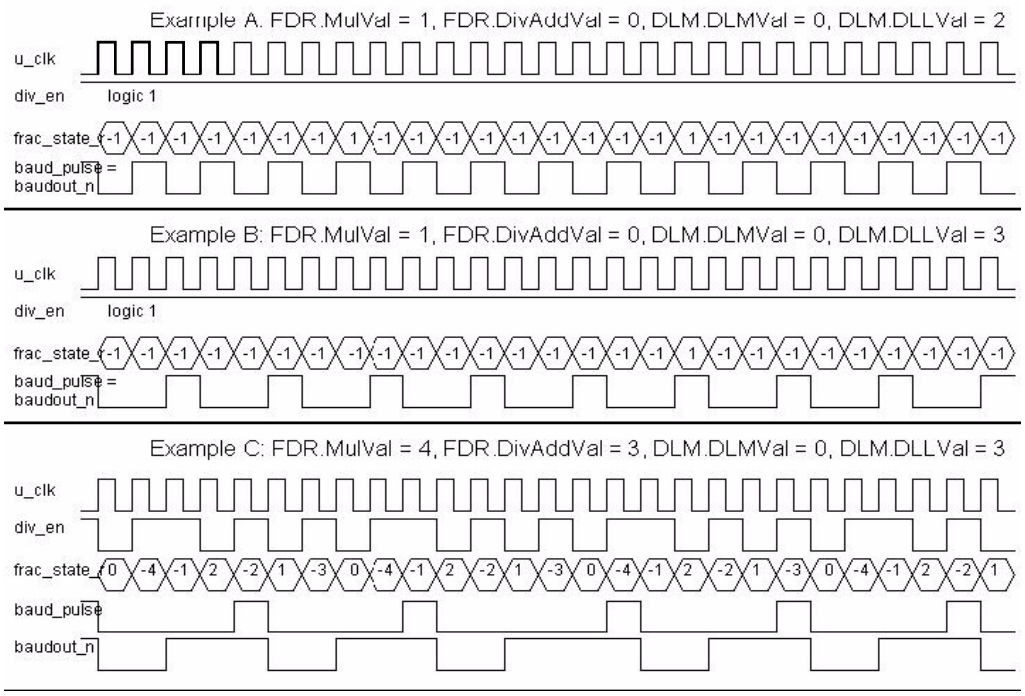


Fig 66. Baud-rate generation examples

In example A the fractional divider is bypassed while the second divider is dividing by 2. In example B the fractional divider is bypassed while the second divider is dividing by 3. In example C fractional divider is multiplying the clock by 4/(3+4)=4/7 while the second divider divides by 3. Table 21–466 lists the baud-rates in case the U_CLK is 48 MHz.

Table 466. Baud-rates using 48MHz 'u_clk' clock

FDR.MULVAL != 0, FDR.DIVADDVAL = 0			Optimal FDR.MULVAL, FDR.DIVADDVAL			
Desired Baud-rate [Hz]	{DLM, DLL} divisor used to generate 16X Baud-rate	Percent error between desired and actual Baud-rate	{DLM, DLL} divisor used to generate 16X Baud-rate	Fractional pre-scaler values MULDIV/(MULDIV+DIVADDVAL)		Percent error between desired and actual Baud-rate
				MULDIV	DIVADDVAL	
50.0	60000	0.0000%	60000	1	0	0.0000%
75.0	40000	0.0000%	40000	1	0	0.0000%
110.0	27273	0.0010%	25000	11	1	0.0000%
134.5	22305	0.0007%	14194	7	4	0.0001%
150.0	20000	0.0000%	20000	1	0	0.0000%
300.0	10000	0.0000%	10000	1	0	0.0000%
600.0	5000	0.0000%	5000	1	0	0.0000%
1200.0	2500	0.0000%	2500	1	0	0.0000%
1800.0	1667	0.0200%	1250	3	1	0.0000%
2000.0	1500	0.0000%	1500	1	0	0.0000%
2400.0	1250	0.0000%	1250	1	0	0.0000%
3600.0	833	0.0400%	625	3	1	0.0000%
4800.0	625	0.0000%	625	1	0	0.0000%
7200.0	417	0.0799%	250	3	2	0.0000%
9600.0	312	0.1603%	125	2	3	0.0000%
19200.0	156	0.1603%	125	4	1	0.0000%
38400.0	78	0.1603%	37	9	10	0.0178%
56000.0	54	0.7937%	25	7	8	0.0000%
57600.0	52	0.1603%	25	12	13	0.0000%
112000.0	27	0.7937%	25	14	1	0.0000%
115200.0	26	0.1603%	23	15	2	0.0959%
224000.0	13	3.0220%	7	12	11	0.1775%
320400.0	9	4.0366%	5	8	7	0.1248%
448000.0	7	4.3367%	5	3	1	0.4464%
460800.0	7	6.9940%	2	4	9	0.1603%
921600.0	3	8.5069%	2	8	5	0.1603%
1048576.0	3	4.6326%	2	7	3	0.1358%

The second and third columns in the table list the DLM, DLL value and the error when bypassing the fractional divider. Columns five, six and seven list the DLL, DLM, fractional divider parameters and error when enabling the fractional divider. Note that the error between actual baud-rate and desired baud-rate is always smaller when using the fractional divider clock pre-scaling.

Table 21–467 lists the fractional pre-scaling and divider values for achieving higher baud rates at 24 MHz U_CLK frequency. Note that the maximum baud rate at an U_CLK frequency of 24 MHz is limited to 460800 Hz, at higher baud rates the error between desired and actual baud rate will become too large.

Table 467. Baud-rates using 24 MHz 'u_clk' clock

FDR.MULVAL != 0, FDR.DIVADDVAL = 0			Optimal FDR.MULVAL, FDR.DIVADDVAL			
Desired Baud-rate [Hz]	{DLM, DLL} divisor used to generate 16X Baud-rate	Percent error between desired and actual Baud-rate	{DLM, DLL} divisor used to generate 16X Baud-rate	Fractional pre-scaler values MULDIV/(MULDIV+DIVADDVAL)		Percent error between desired and actual Baud-rate
				MULDIV	DIVADDVAL	
50.0	30000	0.0000%	30000	1	0	0.0000%
75.0	20000	0.0000%	20000	1	0	0.0000%
110.0	13636	0.0027%	12500	11	1	0.0000%
134.5	11152	0.0037%	7097	7	4	0.0001%
150.0	10000	0.0000%	10000	1	0	0.0000%
300.0	5000	0.0000%	5000	1	0	0.0000%
600.0	2500	0.0000%	2500	1	0	0.0000%
1200.0	1250	0.0000%	1250	1	0	0.0000%
1800.0	833	0.0400%	625	3	1	0.0000%
2000.0	750	0.0000%	750	1	0	0.0000%
2400.0	625	0.0000%	625	1	0	0.0000%
3600.0	417	0.0799%	250	3	2	0.0000%
4800.0	312	0.1603%	125	2	3	0.0000%
7200.0	208	0.1603%	125	3	2	0.0000%
9600.0	156	0.1603%	125	4	1	0.0000%
19200.0	78	0.1603%	37	9	10	0.0178%
38400.0	39	0.1603%	23	10	7	0.0959%
56000.0	27	0.7937%	25	14	1	0.0000%
57600.0	26	0.1603%	23	15	2	0.0959%
112000.0	13	3.0220%	7	12	11	0.1775%
115200.0	13	0.1603%	13	1	0	0.1603%
224000.0	7	4.3367%	5	3	1	0.4464%
320400.0	5	6.3670%	2	3	4	0.3210%
448000.0	3	11.6071%	2	3	2	0.4464%
460800.0	3	8.5069%	2	8	5	0.1603%
921600.0	2	18.6198%	2	1	0	18.6198%
1048576.0	1	43.0511%	2	1	0	1000000%

6. Power optimization

The UART module has an asynchronous clock domain crossing, allowing the APB clock frequency to be independent from the Baud Generator clock frequency. This allows power saving by lowering the APB bus frequency while receiving and transmitting on the serial interface, with the same unchanged Baud Generator clock frequency.

Furthermore, when using the UART try to comply with the following guidelines:

- Operate at a low baud rate, when possible.

- When communicating at a low baud rate, you must reduce the U_CLK accordingly.
- Independently from the baud rate, you must reduce the APB_CLK if possible.
- Switch off the clocks when the device is not in use.
- Reduce the amount of interrupt routines.
- Use the FIFO's for a more efficient data transfer.

UM10314

Chapter 22: LPC3130/31 LCD interface

Rev. 1 — 4 March 2009 User manual

1. Introduction

The LCD interface contains logic to interface to 6800 (Motorola) and 8080 (Intel) compatible LCD controllers with 4/8/16 bit modes. This module also supports a serial interface mode. The speed of the interface can be adjusted in software to match the speed of the connected LCD display.

This module has the following features:

- 4/8/16 bit parallel interface mode: 6800-series, 8080-series.
- Serial interface mode.
- Multiple frequencies for the 6800/8080 bus selectable to support high and low speed controllers.
- Supports polling the busy flag from LCD controller to off load the CPU from polling.
- Contains a 16 byte FIFO for sending control and data information to the LCD controller.
- Supports maskable interrupts.
- Supports DMA transfers.

2. General description

The LCD interface is compatible with the 6800 bus standard and the 8080 bus standard, with one address pin (RS) for selecting the data or instruction register.

The LCD interface makes use of a configurable clock (programmed in the CGU) to adjust the speed of the 6800/8080 bus to the speed of the connected peripheral.

2.1 Block diagram

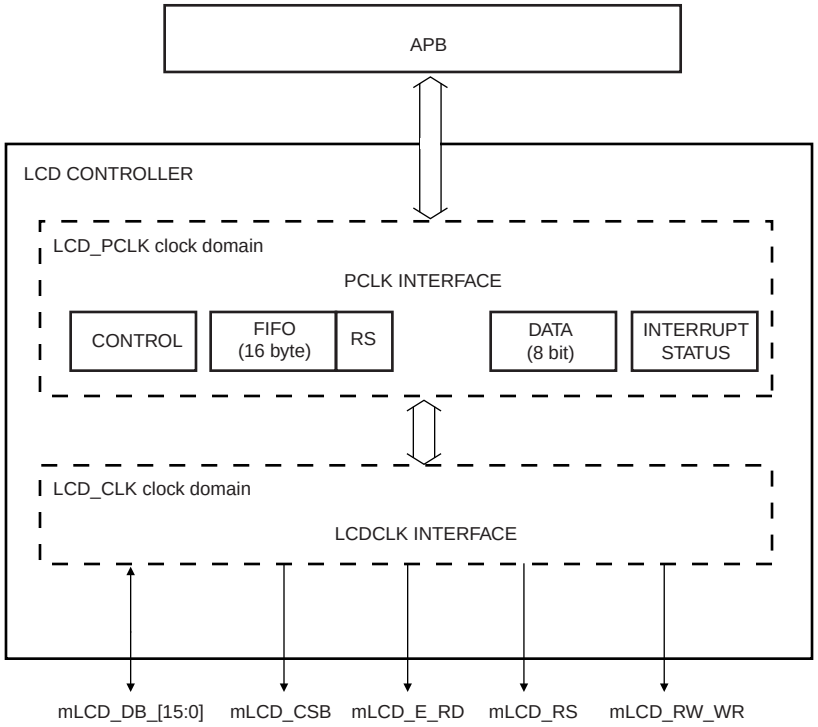


Fig 67. LCD block diagram

2.2 System Interface

2.2.1 Clock signals

Table 468. Clock signals of the LCD interface module

Clock name	Acronym	I/O	Source	Description
LCD_PCLK	PCLK	I	CGU	APB CLK
LCD_CLK	LCDCLK	I	CGU	Clock used for data and control flow towards the external LCD Controller

The LCD_CLK is configurable to match the clock frequency of external LCD Controller.

2.2.2 External pin connections

Table 469. External signals of the LCD interface module

Name	Type (func)	Description
mLCD_CSB	O	Chip Select for external LCD-Controller. Default active HIGH.
mLCD_RS	O	Register Select (also seen as A0) '1' = Data, '0' = Instruction

Table 469. External signals of the LCD interface module *…continued*

Name	Type (func)	Description
mLCD_RW_WR	O	Read Write / WRITE: Read/write in 6800 mode, Write in 8080 mode
mLCD_E_RD	O	Enable / Read: Enable in 6800 mode, Read in 8080 mode
mLCD_DB_[15:0]	I/O	Bi-directional data bus

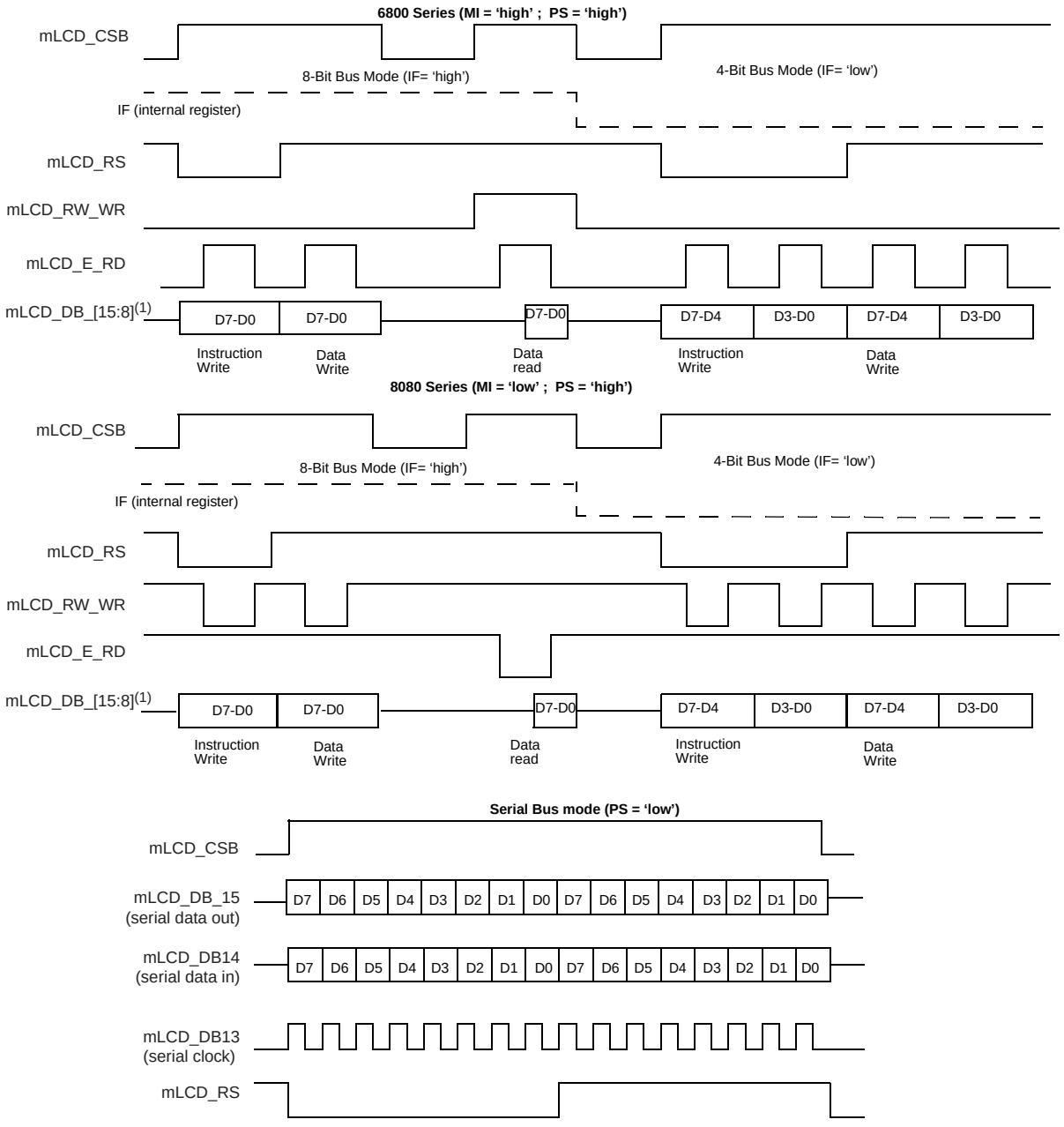
2.2.3 Interrupt request signals

The LCD controller block provides one interrupt output. This interrupt has four possible functions: LCD FIFO EMPTY, LCD FIFO HALF EMPTY, LCD FIFO OVERRUN, LCD READ VALID (see Section 22–5.1). The valid function at any give time can be read from the LCD interface status register (Table 22–471).

2.2.4 Reset signals

The LCD interface is reset by a APB bus Reset.

2.2.5 LCD timing



(1) In 8-bit mode, the data D[7:0] is output on LCD_DB[15:8] pins.

Fig 68. Timing diagram

3. Register overview

Table 470. Register overview: LCD (register base address 0x1500 0400)

Name	R/W	Address offset	Description	Reset Value
LCD_STATUS	R	0x000	Status register	0x0
LCD_CONTROL	R/W	0X004	Control register	0x0000 3CF0
LCD_INT_RAW	R	0X008	Interrupt Raw register	0X0000 0003
LCD_INT_CLEAR	W	0x00C	Interrupt Clear register	0x0
LCD_INT_MASK	R/W	0x010	Interrupt Mask Register	0x0000 000F
LCD_READ_CMD	W	0x014	Read Command register	0x0
LCD_INST_BYTE	R/W	0x020	Instruction Byte Register	0x0
LCD_DATA_BYTE	R/W	0x030	Data Byte Register	0x0
LCD_INST_WORD	W	0x040	Instruction Word register	0x0
LCD_DATA_WORD	W	0x080	Data Word register	0x0

4. Register description

4.1 LCD interface Status Register

Read only. This register stores the interrupt status; the busy bit and the FIFO counter value.

Table 471. LCD interface Status Register (LCD_STATUS, address 0x1500 0400)

Bit	Symbol	Access	Reset Value	Description
31:10	Reserved	-	-	Reserved
9:5	LCD_COUNTER	R	0x00000	current value of the FIFO counter. 0x00 means empty.
4	LCD_INTERFACE_BUSY	R	0x1	LCD interface is still reading the value from the controller
3	LCD_INT_READ_VALID	R	0x0	value read from the LCD controller is valid and not masked in the LCD_INT_MASK register
2	LCD_INT_FIFO_OVERRUN	R	0x0	value being written is larger that the FIFO can hold and not masked in the LCD_INT_MASK register
1	LCD_INT_FIFO_HALF_EMPTY	R	0x0	FIFO is less then half full (when LCD_counter < 8) and not masked in the LCD_INT_MASK register
0	LCD_INT_FIFO_EMPTY	R	0x0	FIFO is empty (LCD_counter = 0) and not masked in the LCD_INT_MASK register

4.2 LCD interface Control register

Read/write. This register stores the control bits used by the LCD interface.

Table 472. LCD interface Control register (LCD_CONTROL, address 0x1500 404)

Bit	Symbol	Access	Reset Value	Description
31:21	Reserved	-	-	Reserved
20	BYASYNC_RELCLK	R/W	0x0	Bypass the logic which assumes asynchronous relation between PCLK & LCDCLK
19	IF_16	R/W	0x0	Interface to 16 bit LCD-Controller. If set, overrides the IF (bit 3)
18	LOOPBACK	R/W	0x0	'0' = Normal Operation '1' = LCD Interface in Loopback mode
17	MSB_FIRST	R/W	0x0	8-bit mode: Don't care. 4-bit mode: `0' means that bits 3-0 will be transmitted first. `1' means that bits 7-4 will be transmitted first. Serial mode: `0' means that bit 0 will be transmitted first. `1' means that bit 7 will be transmitted first.
16	INVERT_E_RD	R/W	0x0	Intel 8080 mode (MI = 0): '0' = E_RD output pin will be active low '1' = E_RD output pin will be active high Motorola 6800 mode (MI = 1): '0' = E_RD output pin will be active high '1' = E_RD output pin will be active low
15	INVERT_CS	R/W	0x0	'1' = Chip select output will be active low '0' = Chip select output will be active high
14	BUSY_RS_VALUE	R/W	0x0	'0' = busy check will happen on RS = '0'. '1' = busy check will happen on RS = '1'. Don't care if BUSY_FLAG_CHECK = `0'
13:10	BUSY_BIT_NR	R/W	0xF	This 4 bit value stores the bit of the 6800/8080 bus which represents the busy flag. Don't care if BUSY_FLAG_CHECK = '0'
9	BUSY_VALUE	R/W	0x0	'0' = if the checked bit equals to '0' that the LCD controller is not busy. '1' = if the checked bit equals to `1' that the LCD controller is busy. Don't care if BUSY_FLAG_CHECK = `0'
8	BUSY_FLAG_CHECK	R/W	0x0	'0' will disable the busy-flag checking '1' will enable the busy-flag-checking
7:6	SERIAL_READ_POS	R/W	0x3	Parallel mode (PS = 0): Dont care Serial Mode (PS=1): '00xx' = Sampling done at the beginning of the cycle '01xx' -- Sampling done at 0.25 * cycle '10xx' -- Sampling done at 0.5 * cycle '11xx' -- Sampling done at 0.75 * cycle See Figure 22–69.

Table 472. LCD interface Control register (LCD_CONTROL, address 0x1500 404) …continued

Bit	Symbol	Access	Reset Value	Description
5:4	SERIAL_CLK_SHIFT	R/W	0x3	Parallel mode (PS = 0): Don't care Serial Mode (PS=1): ‘xx00' -- clock mode 0 ‘xx01' -- clock mode 1 ‘xx10' -- clock mode 2 ‘xx11' -- clock mode 3
3	IF (8bit/4bit)	R/W	0x0	‘0' = put the LCD interface in 8 bit mode ‘1' = put the LCD interface in 4 bit mode. Do not care if PS = '1' Do not care if IF_16 = ’1’
2	MI (Motorola 6800/Intel 8080)	R/W	0x0	‘0' = put the LCD interface in 8080 mode. ‘1' = put the LCD interface in 6800 mode. Do not care if PS='1'
1	PS (Parallel/Serial)	R/W	0x0	‘0' = put LCD interface in parallel mode ‘1' = put the LCD interface in serial mode
0	Reserved	R/W	0x0	Reserved

4.3 LCD interface Interrupt raw register

This register contains the status of the interrupts, without any masking.

Table 473. LCD interface Interrupt Raw register (LCD_INT_RAW, address 0x1500 0408)

Bit	Symbol	Access	Reset Value	Description
31:4	Reserved	-	-	Reserved
3	LCD_INT_READ_VALID_RAW	R	0x0	Is set when the value that has been read from the LCD controller is valid.
2	LCD_INT_OVERRUN_RAW	R	0x0	Is set when FIFO overrun occurs.
1	LCD_INT_FIFO_HALF_EMPTY_RAW	R	0x0	Is set when the FIFO is less then half full (when LCD_counter < 8)
0	LCD_INT_FIFO_EMPTY_RAW	R	0x0	Is set when the FIFO is empty (LCD_counter = 0)

4.4 LCD interface Interrupt Clear register

Write only. Writing to this register clears the selected interrupts.

Table 474. LCD interface Interrupt Clear register (LCD_INT_CLEAR, address 0x1500 040C)

Bit	Symbol	Access	Reset Value	Description
31:4	Reserved	-	0x0	Reserved
3	LCD_INT_READ_VALID_CLR	W	0x0	Clear Interrupt caused by Valid Read

Table 474. LCD interface Interrupt Clear register (LCD_INT_CLEAR, address 0x1500 040C)

Bit	Symbol	Access	Reset Value	Description
2	LCD_FIFO_OVERRUN _CLR	W	0x0	Clear Interrupt caused by FIFO Overrun
1	LCD_INT_FIFO_HALF_EMPTY _CLR	W	0x0	Clear Interrupt caused by FIFO Half Empty
0	LCD_INT_FIFO_EMPTY_CLR	W	0x0	Clear Interrupt caused by FIFO Empty

4.5 LCD interface Interrupt Mask register

Read/write. This register contains the masking information for the interrupt. If a mask bit equals to `1' than that specific interrupt won't be used as a source for the IRQ to the CPU.

Table 475. LCD interface Interrupt Mask register (LCD_INT_MASK, address 0x1500 0410)

Bit	Symbol	Access	Reset Value	Description
31:4	Reserved	-	0x0	Reserved
3	LCD_READ_VALID_MASK	R/W	0x1	Interrupt mask for Valid Read
2	LCD_FIFO_OVERRUN_MASK	R/W	0x1	Interrupt mask for FIFO Overrun
1	LCD_FIFO_HALF_EMPTY_MASK	R/W	0x1	Interrupt mask for FIFO Half Empty
0	LCD_FIFO_EMPTY_MASK	R/W	0x1	Interrupt mask for FIFO Empty

4.6 LCD interface Read Command register

Write only. Writing to this register will result in a read operation on the LCD Interface bus. A write to this register during a read operation will trigger a new read, and will discard the old.

Writing `0x00' will result in a read on the INST_BYTE (see Table 22–477).

Writing `0x01' will result in a read in DATA_BYTE (see Table 22–478).

If a read is finished (valid) the byte can be read in either LCD_INST_BYTE or LCD_DATA_BYTE.

Table 476. LCD interface Read Command register (LCD_READ_CMD, address 0x1500 0414)

Bit	Symbol	Access	Reset Value	Description
31:1	Reserved	-	-	Reserved
0	LCD_READ_COMMAND	W	0x0	‘1’ = read on the INST_BYTE ‘0’ = read in DATA_BYTE

4.7 LCD interface Instruction Byte register

Read/write. Writing to this register will write one byte into the FIFO, tagged as instruction.

Reading from this register will return the data read from the LCD controller. The data can be considered valid if the interrupt: LCD_INT_DATA_VALID occurred, or if bit `LCD_INTERFACE_BUSY' reads `0'.

Table 477. LCD interface Instruction Byte register (LCD_INST_BYTE, address 0x1500 0420)

Bit	Symbol	Access	Reset Value	Description
31:8 / 31:16	Reserved	-	-	16 bit mode = 31:16 Reserved 8 bit mode = 31:8 Reserved
15:0 / 7:0	INST_BYTE	R/W	0x00	16 bit mode = 15:0 Instruction 8 bit mode = 7:0 Instruction

4.8 LCD interface Data Byte register

Read/write. Writing to this register will write one byte into the FIFO, tagged as data.

Reading from this register will return the data read from the LCD controller. The data can be considered valid if the interrupt LCD_INT_DATA_VALID occurred, or if bit `LCD_INTERFACE_BUSY' reads `0'

Table 478. LCD interface Data Byte register (LCD_DATA_BYTE, address 0x1500 0430)

Bit	Symbol	Access	Reset Value	Description
31:8 / 31:16	Reserved	-	-	16 bit mode = 31:16 Reserved 8 bit mode = 31:8 Reserved
15:0 / 7:0	DATA_BYTE	R/W	0x00	16 bit mode = 15:0 Data 8 bit mode = 7:0 Data

4.9 LCD interface Instruction Word register

Word write only. Writing to this register writes the 32-bit value into the FIFO, tagged as instruction. Burst writes are allowed. The LSB byte of the word will be transmitted out of the FIFO first.

Table 479. LCD interface Instruction Word register (LCD_INST_WORD, address 0x1500 0440)

Bit	Symbol	Access	Reset Value	Description
31: 0	INST_WORD	W	0x0	32 bit Instruction Word

4.10 LCD interface Data Word register

Word write only. Writing to this register writes the 32-bit value into the FIFO, tagged as data. Burst writes are allowed. The LSB byte of the word will be transmitted out of the FIFO first.

Table 480. LCD interface Data Word Register (LCD_DATA_WORD, address 0x1500 0480)

Bit	Symbol	Access	Reset Value	Description
31: 0	DATA_WORD	W	0x0	32 bit Data Word

5. Functional description

5.1 Interrupt generation

An interrupt is generated on the following occasions:

- When the FIFO is empty (LCD_FIFO_EMPTY).
- When the FIFO is half empty (LCD_FIFO_HALF_EMPTY).
- When the FIFO is overrun (LCD_FIFO_OVERRUN).
- When the requested instruction/data register is valid (LCD_READ_VALID).

Any of these interrupts can be masked individually to keep them from generating an interrupt to the CPU, by using the LCD_INT_MASK register. The interrupts after masking can be read in the LCD_STATUS register. Writing a `1' in the mask register will mask the interrupt. The status of the interrupts without masking can be read in the `LCD_INT_RAW' register.

5.2 Clearing the interrupts

An interrupt can be cleared by writing a `1' to the respective bit in the LCD_INT_CLR register. If the interrupt has not been solved, for instance the FIFO is still empty, this will re-set the interrupt, when not masked.

5.3 Using DMA flow control

All data transfers towards the LCD interface can be done using the DMA and the corresponding flow control, reducing CPU interrupting.

The DMA has the ability to transfer blocks of data from memory and the LCD FIFO while checking the FIFOLEVEL flow control before sending data. This way data will only be send when the LCD interface has space left in its local FIFO.

This construction allows much larger blocks of data to be transported to the LCD interface than the maximal 16 bytes of the FIFO at a time.

If the external LCD controller supports it, a single enable command to the DMA controller may refresh the complete LCD screen, making the LCD-controller a very low CPU-intensive piece of hardware.

It requires a single DMA channel to transfer data to the LCD interface. If a linked-list must be supported, then two sequential channels are required. See the DMA chapter of this document for more info.

LCD-reading is not supported with DMA flow control.

6. Power optimization

To reduce the power consumption, this module uses a gated clock. The clock coming from the CGU will be disabled when the module is not in use.

7. Programming guide

7.1 Resetting the external LCD controller

A GPIO pin or the system reset can be used to act as a reset signal for the external LCD controller. In some cases a simple instruction to the controller is enough to perform the reset.

7.2 FIFO

A 16 byte write FIFO (First In First Out memory) is implemented in the LCD interface which will store both instructions and data. This way the CPU can write multiple bytes or words to the LCD interface instead of just one and doesn't have to wait for the LCD controller to finish for a new write action.

Information to the LCD controller can be written in the LCD_INST_WORD or LCD_DATA_WORD register. If the total amount of data written is bigger then the FIFO can store, the `FIFO_overrun' interrupt will be set, and the last byte or word will not be written into the FIFO.

If a word is written into the FIFO, the LSB byte (bits 7-0) will be put on the data pins first.

A status pin indicating that the FIFO is at least half empty is connected at top-level, so the Simple DMA controller can decide when it is allowed to transfer data and when it is not allowed.

7.3 Operational modes

The LCD Interface has four modes for outputting data: 16-bit mode, 8-bit mode, 4-bit mode and serial-mode.

7.3.1 16-bit mode

The LCD_CONTROL register also has IF_16 bit to enable 16-bit mode. When this bit is set, the IF bit becomes don't care. When IF_16 bit is reset, then IF bit decides between 8-bit or 4-bit mode.

The FIFO remains 8-bit wide. The LDCCLK interface state machine consecutively reads 2 bytes from the FIFO. To announce the consecutive read to the PCLK interface state machine, the ’continue FIFO’ signal was added. This way the 16-bit functionality is added to the hardware. It is backwards compatible. So in 8-bit mode, the LCD interface writes (to the outside world) 1 byte, and in 16-bit mode it writes 2 bytes. Similarly on a READ action (LCD-Interface reading from external LCD-Controller), 8-bit data is read in 8-bit mode, and 16-bit data is read in 16-bit mode.

7.3.2 8-bit mode

The most significant byte of the databus is used in 8-bit mode. It means the [7:0] bits of the databus of the external controller should be connected to the [15:8] bits of the databus interface of the LCD interface.

At each shift of the FIFO, the last byte inside the FIFO will be put on the data pins, and pin LCD_RS will indicate if the data is an instruction or data value.

In read mode the data on pins LCD_DB[15:8] will be sampled by the LCD interface.

7.3.3 4-bit mode

The most significant nibble of the databus is used in 4-bit mode. It means the [3:0] bits of the databus of external controller should be connected to the [15:12] bits of the databus interface of the LCD interface.

At each shift of the FIFO, the last byte from the FIFO will be split, where the order depends on the `MSB_first' from the control register. When set to `1', bit 7-4 from the FIFO byte will be put first, or read first, at the data pins, and then bit 3-0. When set to `0' bits 3-0 will be written or read first, and then bits 7-4.

7.3.4 Serial mode

At each shift of the FIFO, the last FIFO byte will be split in 8 separate bits and be put on data pin LCD_DB14 (serial data in) and LCD_DB15 (serial data out), where the bit-order depends on the `MSB_first' from the control register.

When set to `0', then first bit 0 and last bit 7 will be written/read first, else the order is from 7 down to 0.

Signal LCD_RS is included for each 8 bits and indicates an instruction or data. Not all controllers require this signal in serial mode, but can be used if required.

7.4 Writing data

There are 2 modes by which information can be sent out on the 6800/8080 bus. These can be addressed, by using the following registers:

LCD_INST_BYTE/LCD_INST_WORD

Writing to one of these registers will add the contents to the FIFO, tagged as instruction.

- To write a word into the FIFO: write the word to register LCD_INST_WORD. The LSB byte of the word will be transmitted first out of the FIFO.
- To write a byte (8-bit mode) or half-word (16-bit mode) into the FIFO: write the byte(8-bit mode) or half-word(16-bit mode) to register LCD_INST_BYTE.

Remark: The separation of bytes and words is necessary, because the APB assumes all data to be 32 bit wide.

LCD_DATA_BYTE / LCD_DATA_WORD

Writing to these registers will add the contents to the FIFO, tagged as data.

To write a word into the FIFO: write the word to address: LCD_DATA_WORD. The LSB byte of the word will be transmitted first out of the FIFO.

To write a byte(8-bit mode) or half-word(16-bit mode) into the FIFO: write the byte(8-bit mode) or half-word(16-bit mode) to register LCD_DATA_BYTE.

7.5 Reading data

Reads from the LCD controller are quite rare. Normally only writes are performed to a LCD controller. Therefore a simple double byte register is used to store the byte / half-word that has been read from the 6800/8080 bus.

The LCD controller is a slow peripheral. When the CPU requests information from the 6800/8080 bus, it will not be available directly. To read the data register or the instruction register, the CPU has to wait a while before the data becomes valid.

The read procedure is as follows:

1. Write the following in the LCD_READ_CMD register:
   - 0x0' for initiating a read on INST_BYTE (RS=0)
   - 0x1' for initiating a read on DATA_BYTE (RS=1)
2. Wait until an IRQ arrives, and check the `LCD_read_valid' bit of the STATUS register

   or

   keep polling the LCD_INTERFACE_BUSY bit of the STATUS register. `0' means valid.
3. If the returned value is valid, the byte can be read from the LCD_INST_BYTE or LCD_DATA_BYTE register. A write to LCD_READ_CMD will initiate a new read on the LCD bus.

If the LCD_INST_BYTE or LCD_DATA_BYTE register is read, the LCD_INTERFACE_BUSY bit of the CONTROL register stays logical `0' until a new read is started.

7.6 Combined Writing and reading data

A read operation can be performed at any time. The read-request is stored (queued), and as soon as the write FIFO is empty, the read operation will be executed. Only one read request is stored, so if there were multiple read requests (which is considered invalid), the last one will be executed.

Note: A write operation after a read operation, which is still queued or has not completed its cycle, is considered invalid. The read operation will be discarded or aborted and the LCD interface will enter or stay (if the LCD interface did not start reading yet) in write mode. The written value is stored in the FIFO as being a regular write transfer, and the LCD interface will proceed normally, without executing the read command.

7.7 Using wait states

The LCD controller does not support waitstates. If the LCD bus needs to be made slower, it is best to reduce the LCD_CLK speed in the CGU.

7.8 Serial mode

The LCD interface can be put in serial mode by setting bit `PS' of LCD_CONTROL register to `1'.

7.8.1 Serial writes

In this mode, the FIFO will be used to shift the data on a bit-by-bit basis on pin LCD_DB15 of the LCD interface. The `MSB_first' bit of the control register controls if the MSB or LSB bit is transmitted first:

- `0' to select LSB first (first bit 0, then 1,2,3,4,5,6,7).
- `1' to select if the serial data is transmitted with MSB first (so first bit 7, then 6,5,4,3,2,1,0)

This is depending on how the LCD controller accepts serial information.

If the FIFO runs empty, the LCD_FIFO_empty bit of the raw-status interrupt register will be set, and the serial clock output will be disabled, until new data is available in the FIFO.

The LCD_RS output can be used to select the data or instruction register of the LCD controller.

7.8.2 Serial reads

Serial reading uses the same commands as parallel reading.

Reading the serial data is done on pin LCD_DB14 of the 6800/8080 bus. The `MSB_first' bit controls which bit is sampled first:

- `0' to select that the first sampled bit will be stored in the LSB bit
- `1' to select that the first sampled bit will be stored in the MSB bit.

Eight clock cycles are generated on the serial clock, and the serial input will be sampled for 8 cycles.

Now the information in the LCD_DATA_BYTE register is valid and can be read.

7.8.3 Serial clock timing

The serial output clock `SCL' can be set to four modes to comply to the specifications of the LCD controller. Each mode is a 25% shift of the previous mode.The clock mode is set by writing to the `serial_clk_shift' location of the control register.

For reading, register location `serial_read_pos' of the control register determines at which position the data is sampled by the LCD interface. See Figure 22–69.

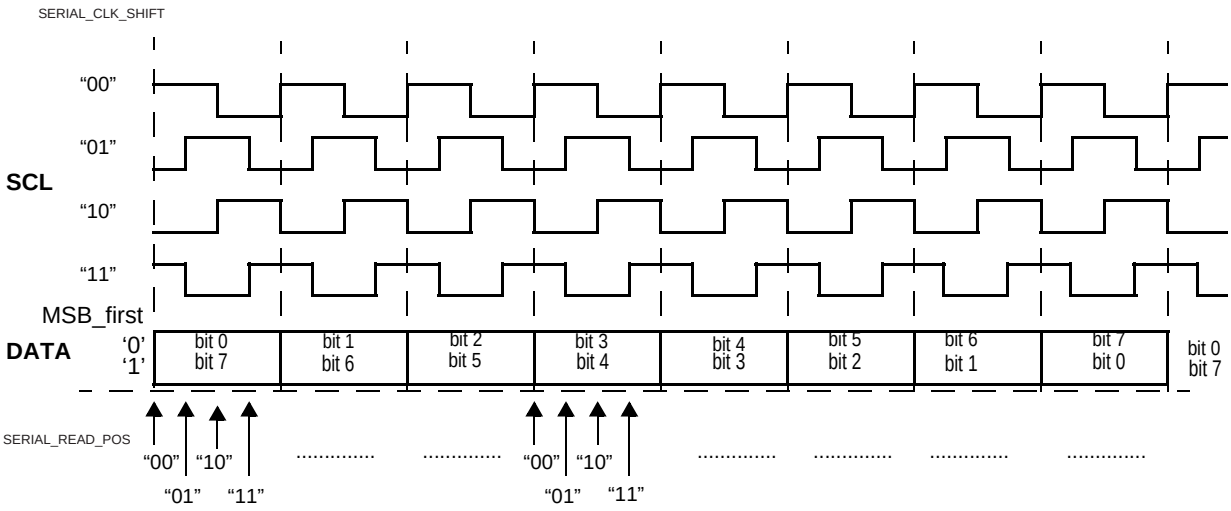


Fig 69. Serial mode timing

7.9 Checking the busy flag of the LCD controller

Most LCD controllers contain a status register with a bit that represents the busy flag. If available, this flag has to be polled before a read or write can be performed from or to the LCD controller.

To off-load the CPU from polling the LCD controller before each access, an option is included that the LCD interface takes over this polling for data read, data write and instruction write accesses.

To enable the busy-flag checking, set the following bits in the CONTROL register:

- BUSY_FLAG_CHECK': set to `1' to enable the busy-flag-checking
- BUSY_BIT_NR': Set which bit number has to be checked for the busy flag. Don't care if BUSY_FLAG_CHECK = `0'
- BUSY_RS_VALUE': Set which address (RS value) has to be checked for the busy flag.
  - when `0' then register belonging to RS='0' will be checked.
  - when `1' the register belonging to RS='1' will be checked.

  Don't care if BUSY_FLAG_CHECK = `0'
- BUSY_VALUE': Set if a logic `0' or `1' represents the busy flag. Don't care if BUSY_FLAG_CHECK = `0'

Note: Reading the register from the LCD controller set by BUSY_RS_VALUE will return the value of that register, without checking for the busy-flag first.

The busy-flag-checking can only be used if one of the two LCD controller registers have a bit that represents the `busy' value. If this bit is not available, the busy-flag-checking feature has to be disabled and a slower clock has to be used to ensure reading or writing a valid value.

7.10 Loop back mode

Setting the register `LOOPBACK' of the CONTROL register to `1', will set the LCD interface in loop back-mode.

Internally, the LCD data output is connected to the LCD data input. The programmer can test correct behavior of the LCD interface, by doing the following:

- Place the LCD interface in parallel, 8-bit mode
- Write a single byte to the LCD_DATA_BYTE register
- Write `0x01' to the LCD_READ_CMD register to request a bus read
- Poll the status bit, or wait for the `valid' interrupt (if MASK is cleared)
- If valid, read the byte from LCD_DATA_BYTE register
- Compare this value with the written value

To ensure correctness of the test, perform the above a couple of times, with different values.

7.11 Clock relation PCLK and LCDCLK

The LCDCLK and the PCLK (see Table 22–468) can be totally independent from each other. The PCLK logic samples the LCDLCK and uses this as a timing reference. Internally the LCDCLK is converted to pulses at each rising edge of the PCLK.

The PCLK should run at least 2 times as fast as the LCDCLK. The LCDCLK should have the `clock_stretching' option enabled when it is configured.

Every 5 LCDCLK cycles means one LCD bus cycle.

If Busy flag checking is enabled each write will take 10 cycles, since each write is accompanied by a read.

7.12 MSB_FIRST bit of Control Register

In 8-bit mode, value of MSB_FIRST is do not care. In 8-bit mode, it always works as LSB_FIRST. This feature was there because the fifo is 8-bit wide, and while sending data out, it can be programmed which bit / nibble of that 8 bit should go out first. Now although the databus was extended to 16-bit wide, the fifo width was kept as it was = 8-bit. Hence in 8-bit mode, there is no extra facility. Whatever is available at the end of FIFO, will come out first. And the FIFO is fed from its input side as LSB_FIRST protocol. (LS-Byte of the 32-bit word written to IP via APB bus.).

UM10314

Chapter 23: LPC3130/31 I²C-bus interface

Rev. 1 — 4 March 2009 **User manual**

1. Introduction

The LPC3130/31 contains two I2C Master / Slave interfaces (I²C).

1.1 Features

This module has the following features:

- **I²C0-bus interface:** I2C0 is a standard I2C-compliant bus interface with open-drain pins. This interface supports functions described in the I2C-bus specification for speeds up to 400 kHz. This includes multi-master operation and allows powering off this device in a working system while leaving the I2C-bus functional.
- **I²C1-bus interface:** The I2C1-bus interface uses standard I/O pins and is intended for use with a single-master I2C-bus and does not support powering off of this device. Standard I/Os also do not support multi-master I2C implementations.
- Supports normal mode (100 kHz SCL) and fast mode (400 kHz SCL).
- Interrupt support.
- Supports DMA transfers (single).
- Four modes of operation:
  - Master transmitter
  - Master receiver
  - Slave transmitter
  - Slave receiver

2. General description

2.1 Description

There are two I²C interfaces in the LPC3130/31. These I²C blocks can be configured as a master, multimaster or slave supporting clock rates up to 400 kHz. The I2C blocks also support 7 or 10 bit addressing. Each has a four word FIFO for both transmit and receive. An interrupt signal is available from each block.

There is a separate slave transmit FIFO. The slave transmit FIFO (TXS) and its level are only available when the controller is configured as a Master/Slave device and is operating in a multi-master environment. Separate TX FIFOs are needed in a multi-master because a controller might have a message queued for transmission when an external master addresses it to be come a slave-transmitter, a second source of data is needed.

A typical I²C-bus configuration is shown in Figure 23–70. Depending on the state of the direction bit (r/w), two types of data transfers are possible on the I²C-bus:

Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.

Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I2C-bus will not be released.

Each of the I2C interfaces on the LPC3130/31 contains a four byte FIFO, allowing more data to be transferred before additional software attention is needed.

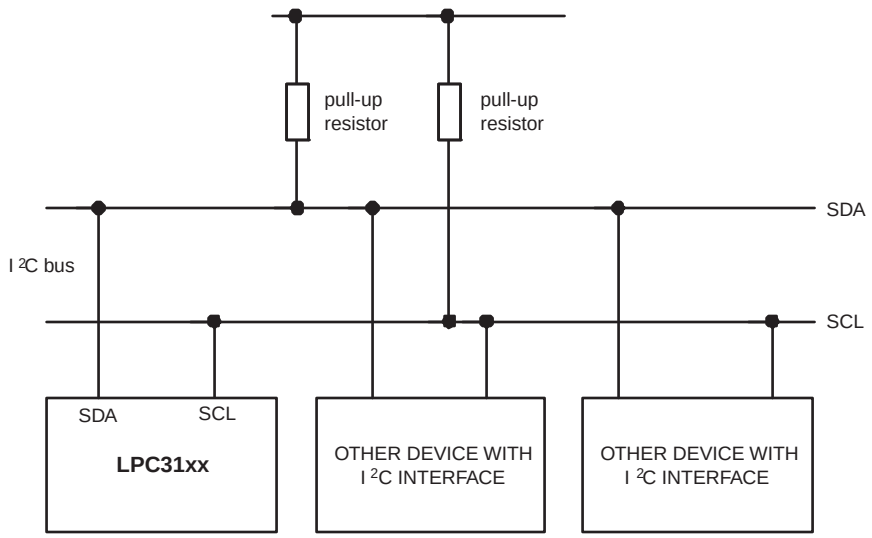


Fig 70. I2C-bus configuration

2.2 Block diagram

The ARM Peripheral Bus interface (APB) provides a communication link between the CPU and the I2C controller. The shift register handles serializing/de-serializing data and counting bits of a byte. The Rx and Tx Control blocks count bytes and handling byte acknowledgement. The Master and Slave control blocks can each enable or disable the Rx and Tx control and also handle transferring data between the shift register and the Rx and Tx FIFOs. SDAOUT is driven either by shift register data, acknowledge signals, or the Master control for creating START or STOP conditions. SCLOUT is driven by the Master control generating a clock or by the Slave control extending the clock of an external master.

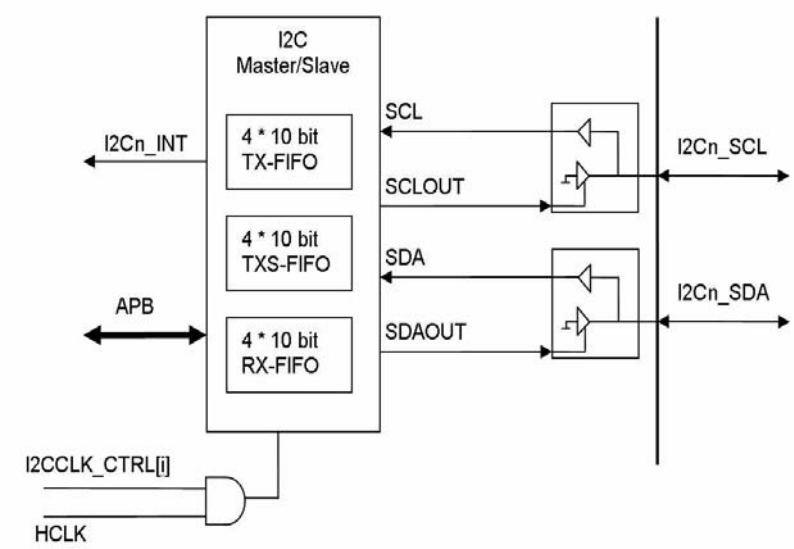


Fig 71. I2C Block diagram

2.3 Interface description

2.3.1 Clock signals

Table 481. Clock signals of the I2C Master/Slave Interface

Clock name	Acronym	I/O	Source/ destination	Description
I2C_SCL0/1	SCL	I/O	Pin	Serial Clock. This is the current value of the I2C serial clock input from a pad.
I2C0/1_PCLK	HCLK	I	CGU	APB Clock. The single clock domain reference for the I2C. This signal is un-buffered.

2.3.2 Pin connections

Table 482. Signals to pins of the IC for the I2C Master/Slave Interface

Name	Type	Reset value	Description
I2C_SCL0/1	I/O	-	Serial Clock. This is the current value of the I2C serial clock input from a pad.
I2C_SDA0/1	I/O	-	Serial Data. This is the current value of the I2C serial data line input from a pad.

Table 483. Auxiliary pin signals of the I2C Master/Slave interface

Name	Type	Reset value	Description
SCLOUT	O	1	Transmitter Clock. This drives the external serial clock line low or allows it to float high.
SDAOUT	O	1	Transmitter Data. This drives the external serial data line low or allows it to float high.
SCANTESTMODE	I	0	Scan test enabled. This signal is active during scan testing to avoid bus contention on the PD bus.

2.3.3 Interrupt requests

The I2C0/1 interfaces produce one (active LOW) interrupt each. The reason for the interrupt is encoded in the status register (STS). There are several possible interrupt types: transfer completed, arbitration failure, missing acknowledge, need more data, Tx FIFO has room for more data, or data has been received. The interrupt signal NINTR is a combination of the status register bits and the enables in the control register.

2.3.4 Reset signals

The I2C0/1 internal registers and FIFOs are reset by a PNRES signal from the CGU.

2.3.5 DMA flow control transfer signals

The I2C Master/Slave interface supports single DMA transfers. The DMA request line is connected with the interrupt line (NINTR) when DMA is used. For DMA operations, the proper interrupts should be unmasked in the control register (CTL).

3. Register overview

Table 484. Register overview: I2C0 registers (base address 0x1300 A000)

Name	Access	Address offset	Description
I2C0_RX	RO	0x00	I2C0 RX Data FIFO
I2C0_TX	WO	0x00	I2C0 TX Data FIFO
I2C0_STAT	RO	0x04	I2C0 Status Register
I2C0_CTRL	R/W	0x08	I2C0 Control Register
I2C0_CLK_HI	R/W	0x0C	I2C0 Clock Divider high
I2C0_CLK_LO	R/W	0x10	I2C0 Clock Divider low
I2C0_ADR	R/W	0x14	I2C0 Slave Address
I2C0_RXFL	RO	0x18	I2C0 Rx FIFO level
I2C0_TXFL	RO	0x1C	I2C0 Tx FIFO level
I2C0_RXB	RO	0x20	I2C0 Number of bytes received
I2C0_TXB	RO	0x24	I2C0 Number of bytes transmitted
I2C0_S_TX	WO	0x28	Slave Transmit FIFO
I2C0_S_TXFL	RO	0x2C	Slave Transmit FIFO level

Table 485. Register overview: I2C1 registers (base address 0x1300 A400)

Name	Access	Address offset	Description
I2C1_RX	RO	0x00	I2C1 RX Data FIFO
I2C1_TX	WO	0x00	I2C1 TX Data FIFO
I2C1_STAT	RO	0x04	I2C1 Status Register
I2C1_CTRL	R/W	0x08	I2C1 Control Register
I2C1_CLK_HI	R/W	0x0C	I2C1 Clock Divider high
I2C1_CLK_LO	R/W	0x10	I2C1 Clock Divider low
I2C1_ADR	R/W	0x14	I2C1 Slave Address
I2C1_RXFL	RO	0x18	I2C1 Rx FIFO level
I2C1_TXFL	RO	0x1C	I2C1 Tx FIFO level
I2C1_RXB	RO	0x20	I2C1 Number of bytes received
I2C1_TXB	RO	0x24	I2C1 Number of bytes transmitted
I2C1_S_TX	WO	0x28	Slave Transmit FIFO
I2C1_S_TXFL	RO	0x2C	Slave Transmit FIFO level

4. Register description

4.1 I2Cn RX Data FIFO (I2Cn_RX - 0x1300 A000, 0x1300 A400)

The RX FIFO may be cleared via a soft reset, by setting bit 8 in the I2Cn_CTRL register.

When operating as a master-receiver, the TX_FIFO must be written for proper operation. When operating as a master-receiver the controller ignores bits [7:0] in the TX_FIFO register. The master-receiver must write a (dummy) byte to the TX_FIFO for each byte it expects to receive in the RX_FIFO. The first dummy byte triggers the clock sequence to transfer data from the slave transmitter. Each dummy byte that follows has a dual purpose, it acknowledges reception of the previous byte and triggers the clock sequence to transfer the next byte form the slaver transmitter. If the master-receiver sets the STOP_bit (9) (to signal the end of a receive) or sets the START_bit (8) (to cause a RESTART condition), then the last byte read from the slave is not acknowledged. That is, the last byte of a master-receiver does not acknowledge by writing a dummy value to the TX_FIFO register.

If the RX FIFO is read while empty, a DATA ABORT exception is generated.

Table 486. I2Cn RX Data FIFO (I2Cn_RX - 0x1300 A000, 0x1300 A400)

I2Cn_RX	Function	Description	Reset value
7:0	RxData	Receive FIFO data bits 7:0	N/A

4.2 I2Cn TX Data FIFO (I2Cn_TX - 0x1300 A000, 0x1300 A400)

The TX FIFO may be cleared via a soft reset, by setting bit 8 in the I2Cn_CTRL register.

If the controller is configured as a Master/Slave and is operating in a multi-master environment, then only master-transmit data should be written to I2Cn_TX, slave transmit data should be written to I2Cn_S_TX.

If the TX FIFO is written to while full a DATA ABORT exception is generated.

Table 487. I2Cn TX Data FIFO (I2Cn_TX - 0x1300 A000, 0x1300 A400)

I2Cn_TX	Function	Description	Reset value
9	STOP	0 = Do not issue a STOP condition after transmitting this byte 1 = Issue a STOP condition after transmitting this byte.	NA
8	START	0 = Do not Issue a START condition before transmitting this byte 1 = Issue a START condition before transmitting this byte.	NA
7:0	TxData	Transmit FIFO data bits 7:0	NA

4.3 I2Cn Status register (I2Cn_STAT - 0x1300 A004, 0x1300 A404)

The status is a read-only register that provides status information on the TX and RX blocks as well as the current state of the external buses. A soft reset will clear the status register with the exception of the TFE and RFE bits, which will be set, and the SCL and SDA bits, which continue to reflect the state of the bus pins.

Table 488. I2Cn Status register (I2Cn_STAT - 0x1300 A004, 0x1300 A404)

I2Cn_STAT	Function	Description	Reset value
13	TFES	Slave Transmit FIFO Empty. Slave TFE is set when the slave TX FIFO is empty and is cleared when the slave TX FIFO contains valid data. 0 = TX FIFO is not empty. 1 = TX FIFO is empty	1
12	TFFS	Slave Transmit FIFO Full Slave TFF is set when the slave TX FIFO is full and is cleared when the slave TX FIFO is not full. 0 = Slave TX FIFO is not full. 1 = SlaveTX FIFO is full	0
11	TFE	Transmit FIFO Empty. TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data. 0 = TX FIFO contains valid data. 1 = TX FIFO is empty	1
10	TFF	Transmit FIFO Full. TFF is set when the TX FIFO is full and is cleared when the TX FIFO is not full. 0 = TX FIFO is not full. 1 = TX FIFO is full	0
9	RFE	Receive FIFO Empty. RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data. 0 = RX FIFO contains data. 1 = RX FIFO is empty	1

Table 488. I2Cn Status register (I2Cn_STAT - 0x1300 A004, 0x1300 A404) *…continued*

I2Cn_STAT	Function	Description	Reset value
8	RFF	Receive FIFO Full. This bit is set when the RX FIFO is full and cannot accept any more data. It is cleared when the RX FIFO is not full. If a byte arrives when the Receive FIFO is full, the SCL is held low until the ARM reads the RX FIFO and makes room for it. 0 = RX FIFO is not full 1 = RX FIFO is full	0
7	SDA	The current value of the SDA signal.	NA
6	SCL	The current value of the SCL signal.	NA
5	ACTIVE	Indicates whether the bus is busy. This bit is set when a START condition has been seen. It is cleared when a STOP condition is seen.	0
4	DRSI	Slave Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a stop condition or it will hold SCL low until more data is available. The Slave Data Request bit is set when the slave transmitter is data-starved. If the slave TX FIFO is empty and the last byte transmitted was acknowledged, then SCL is held low until the ARM core writes another byte to transmit. This bit is cleared when a byte is written to the slave Tx FIFO. 0 = Slave transmitter does not need data. 1 = Slave transmitter needs data.	NA
3	DRMI	Master Data Request Interrupt. Once a transmission is started, the transmitter must have Data to transmit as long as it isn't followed by a stop condition or it will hold SCL low until more data is available. The Master Data Request bit is set when the master transmitter is data-starved. If the master TX FIFO is empty and the last byte did not have a STOP condition flag, then SCL is held low until the ARM core writes another byte to transmit. This bit is cleared when a byte is written to the master TX FIFO. 0 = Master transmitter does not need data. 1 = Master transmitter needs data.	0

Table 488. I2Cn Status register (I2Cn_STAT - 0x1300 A004, 0x1300 A404) …continued

I2Cn_STAT	Function	Description	Reset value
2	NAI	No Acknowledge Interrupt. After every byte of data is sent, the transmitter expects an acknowledge from the receiver. This bit is set if the acknowledge is not received. It is cleared when a byte is written to the master TX FIFO. 0 = Last transmission received an acknowledge. 1 = Last transmission did not receive an acknowledge.	0
1	AFI	Arbitration Failure Interrupt. When transmitting, if the SDA is low when SDAOUT is high, then this I2C has lost the arbitration to another device on the bus. The Arbitration Failure bit is set when this happens. It is cleared by writing a '1' to bit 1 of the status register. 0 = No arbitration failure on last transmission. 1 = Arbitration failure occurred on last transmission.	NA
0	TDI	Transaction Done Interrupt. This flag is set if a transaction completes successfully. It is cleared by writing a '1' to bit 0 of the status register. It is unaffected by slave transactions. 0 = Transaction has not completed. 1 = Transaction completed.	0

4.4 I2Cn Control Register (I2Cn_CTRL - 0x1300 A008, 0x1300 A408)

The CTL register is used to enable interrupts and reset the I2C state machine.

Note that the DMA request line is connected with the interrupt line when DMA is used. For DMA operations, the proper interrupts should be unmasked in the control register.

Table 489. I2Cn Control Register (I2Cn_CTRL - 0x1300 A008, 0x1300 A408)

I2Cn_CTRL	Function	Description	Reset value
10	TFFSIE	Slave Transmit FIFO Not Full Interrupt Enable. This enables the Slave Transmit FIFO Not Full interrupt to indicate that the more data can be written to the slave transmit FIFO. Note that this is not full. 0 = Disable the TFFSI. 1 = Enable the TFFSI.	0
9	SEVEN	Seven-bit slave address. This bit is selects 7-bit or 10-bit slave address operation. 0 = Use 7-bit slave addressing, bits [6:0] in address register. 1 = Use 10-bit slave address. Note: Performing a Soft Reset clears this bit to 0. Use 7-bit slave addressing.	0
8	RESET	Soft Reset. On a soft reset, the TX and RX FIFOs are flushed, STS register is cleared, and all internal state machines are reset to appear idle, and clears bit 9 forcing 7-bit slave addressing. The I2Cn_CLK_LO, I2Cn_CLK_HI, I2Cn_CTRL, and I2Cn_ADR registers are NOT modified by a soft reset. 0 = No effect 1 = Reset the I2C to idle state. Self clearing.	0

Table 489. I2Cn Control Register (I2Cn_CTRL - 0x1300 A008, 0x1300 A408) *…continued*

I2Cn_CTRL	Function	Description	Reset value
7	TFFIE	Transmit FIFO Not Full Interrupt Enable. This enables the Transmit FIFO Not Full interrupt to indicate that more data can be written to the transmit FIFO. Note that this is not full. It is intended help the ARM Write to the I2C block only when there is room in the FIFO to accept it and do this without polling the status register. 0 = Disable the TFFI. 1 = Enable the TFFI.	0
6	RFDAIE	Receive FIFO Data Available Interrupt Enable. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty). 0 = Disable the DAI. 1 = Enable the DAI.	0
5	RFFIE	Receive FIFO Full Interrupt Enable. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data. 0 = Disable the RFFI. 1 = Enable the RFFI.	0
4	DRSIE	Data Request Slave Transmitter Interrupt Enable. This enables the DRSI interrupt which signals that the slave transmitter has run out of data and the last byte was acknowledged, so the SCL line is being held low. 0 = Disable the DRSI interrupt. 1 = Enable the DRSI interrupt.	NA
3	DRMIE	Data Request Master Transmitter Interrupt Enable. This enables the DRMI interrupt which signals that the master transmitter has run out of data, has not issued a Stop, and is holding the SCL line low. 0 = Disable the DRMI interrupt. 1 = Enable the DRMI interrupt.	0
2	NAIE	Transmitter No Acknowledge Interrupt Enable. This enables the NAI interrupt signalling that transmitted byte was not acknowledged. 0 = Disable the NAI. 1 = Enable the NAI.	0
1	AFIE	Transmitter Arbitration Failure Interrupt Enable. This enables the AFI interrupt which is asserted during transmission when trying to set SDA high, but the bus is driven low by another device. 0 = Disable the AFI. 1 = Enable the AFI.	NA
0	TDIE	Transmit Done Interrupt Enable. This enables the TDI interrupt signalling that this I2C issued a stop condition. 0 = Disable the TDI interrupt. 1 = Enable the TDI interrupt.	0

4.5 I2Cn Clock Divider High (I2Cn_CLK_HI - 0x1300 A00C, 0x1300 A40C)

The I2Cn_CLK_HI register holds a terminal count for counting I2Cn_PCLK cycles to create the high period of the slower I2C serial clock, SCL. When reset, the clock divider will be set to run at its reset value.

Table 490. I2Cn Clock Divider High (I2Cn_CLK_HI - 0x1300 A00C, 0x1300 A40C)

I2Cn_CLK_HI	Function	Description	Reset value
9:0	CLK_DIV_HI	Clock Divisor High. This value sets the number of cycles SCL will be high. $F_{SCL}=F_{I2Cn\_PCLK}/(CLK\_DIV\_HI + CLK\_DIV\_LO)$ This means that in order to get F_{SCL} =100 kHz set CLK_DIV_HIGH = CLK_DIV_LOW = 520. (F_{I2Cn_PCLK} =104 MHz). The lowest operating frequency is about 50 kHz.	0x2BA

4.6 I2Cn Clock Divider Low (I2Cn_CLK_LO - 0x1300 A010, 0x1300 A410)

The I2Cn_CLK_LO register holds a terminal count for counting I2Cn_PCLK cycles to create the low period of the slower I2C serial clock, SCL. When reset, the clock divider will be set to run at its reset value frequency.

Table 491. I2Cn Clock Divider Low (I2Cn_CLK_LO - 0x1300 A010, 0x1300 A410)

I2cN_CLK_LO	Function	Description	Reset value
9:0	CLK_DIV_LO	Clock Divisor Low. This value sets the number of I2Cn_PCLK cycles SCL will be low. $FSCL=F_{I2Cn\_PCLK}/(CLK\_DIV\_HI + CLK\_DIV\_LO)$ This means that in order to get FSCL =100 kHz set CLK_DIV_HIGH = CLK_DIV_LOW = 520. (F_{I2Cn_PCLK} =104 MHz). The lowest operating frequency is about 50 kHz.	0x2BA

4.7 I2Cn Slave Address (I2Cn_ADR - 0x1300 A014, 0x1300 A414)

The I2Cn_ADR register holds the I2C bus slave address.

Table 492. I2Cn Slave Address (I2Cn_ADR - 0x1300 A014, 0x1300 A414)

I2Cn_ADR	Function	Description	Reset value
9:0	ADR	ADR is the I2C bus slave address. Bits 6:0 are enabled if the I2C is configured for slave mode. Bits 9:7 are enabled only if it is configured for slave mode and '10-bit addressing'. Note: A soft-reset disables Bits 9:7, see the description of Bits 8 and 9 in the I2Cn_CTRL registers.	0x06E

4.8 I2Cn Receive FIFO level (I2Cn_RXFL - 0x1300 A018, 0x1300 A418)

The I2Cn_RXFL is a read only register that contains the number of bytes in the RX FIFO.

Table 493. I2Cn RX FIFO level (I2Cn_RXFL - 0x1300 A018, 0x1300 A418)

I2Cn_RXFL	Function	Description	Reset value
1:0	RxFL	Receive FIFO level	0

4.9 I2Cn Transmit FIFO level (I2Cn_TXFL - 0x1300 A01C, 0x1300 A41C)

The I2Cn_TXFL is a read only register that contains the number of bytes in the TX FIFO.

Table 494. I2Cn TX FIFO level (I2Cn_TXFL - 0x1300 A01C, 0x1300 A41C)

I2Cn_TXFL	Function	Description	Reset value
1:0	TxFL	Transmit FIFO level	0

4.10 I2Cn Receive byte count (I2Cn_RXB - 0x1300 A020, 0x1300 A420)

The I2Cn_RXB contains the number of bytes received. The register is reset when I2C transitions from inactive to active.

Table 495. I2Cn RX byte count (I2Cn_RXB - 0x1300 A020, 0x1300 A420)

I2Cn_RXB	Function	Description	Reset value
15:0	RxB	Number of bytes received	N/A

4.11 I2Cn Transmit Byte count (I2Cn_TXB - 0x1300 A024, 0x1300 A424)

The I2Cn_TXB contains the number of bytes transmitted. The register is reset when I2C transitions from inactive to active.

Table 496. I2Cn TX byte count (I2Cn_TXB - 0x400A 0024, 0x1300 A424)

I2Cn_TXB	Function	Description	Reset value
15:0	TxB	Number of bytes sent	N/A

4.12 I2Cn Slave Transmit FIFO (I2Cn_S_TX - 0x1300 A028, 0x1300 A428)

The I2Cn_S_TX FIFO may be cleared via a soft reset, by setting bit 8 in the I2Cn_CTRL register.

If the controller is configured as a Master/Slave and is operating in a multi-master environment, then only master-transmit data should be written to I2Cn_TX, slave transmit data should be written to I2Cn_S_TX.

If the TX FIFO is written to while full a DATA ABORT exception is generated.

Table 497. I2Cn Slave TX Data FIFO (I2Cn_S_TX - 0x1300 A028, 0x1300 A428)

I2Cn_S_TX	Function	Description	Reset value
7:0	TXS	Slave Transmit FIFO data bits 7:0	NA

4.13 I2Cn Slave Transmit FIFO level (I2Cn_S_TXFL - 0x1300 A02C, 0x1300 A42C)

The I2Cn_S_TXFL is a read only register that contains the number of bytes in the Slave TX FIFO.

Table 498. I2Cn Slave TX FIFO level (I2Cn_S_TXFL - 0x1300 A02C, 0x1300 A42C)

I2Cn_S_TXFL	Function	Description	Reset value
1:0	TxFL	Slave Transmit FIFO level	0

5. Functional description

5.1 Overview

I2C is a two-wire interface made up of a clock (SCL) and data (SDA). Each of these two wires has a pull-up (or current source). Any device on the bus can pull it to a logic zero or let it float to a logic one (Wire-ANDed). A bus master drives the clock line and is responsible for driving an address on the data wire. A bus slave is any device being addressed by a master. A bus transmitter writes data to the bus by driving the data line. A bus receiver reads data from the bus. A bus master can be either a transmitter or a receiver; likewise for a bus slave.

5.2 I2C clock settings

Table 23–499 shows some examples of clock settings for various I2Cn_PCLK and I2C frequencies.

Table 499. Example I2C rate settings

I2C0/1_PCLK frequency (MHz)	I2C clock rate (kHz)	I2Cn_CLK_HI + I2Cn_CLK_LO	I2Cn_CLK_HI	I2Cn_CLK_LO	Comment
104	100	1040	520	520	Symmetric clock (standard for 100 kHz I2C)
52	100	520	260	260	Symmetric clock (standard for 100 kHz I2C)
12	100	120	60	60	Symmetric clock (standard for 100 kHz I2C)
104	400	260	94	166	Asymmetric clock (per 400 kHz I2C spec)
52	400	130	47	83	Asymmetric clock (per 400 kHz I2C spec)
12	400	30 (rounded up)	12	18	Asymmetric clock (per 400 kHz I2C spec). Actual rate will be 393.9 kHz.

5.3 I2C Data

When sending data, the data line, SDA, must change when the clock line, SCL, is low. Data is sampled at the rising edge of SCL. Figure 23–72 shows the relative timing of SDA to SCL

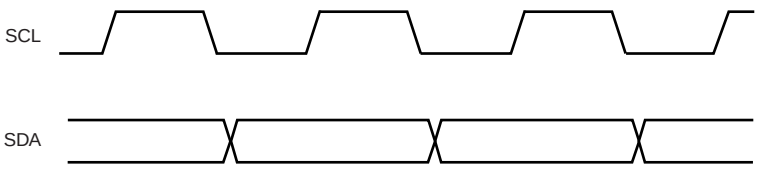


Fig 72. I2C Data

5.4 Start Condition

A start condition is signaled by a high-to-low transition of SDA while SCL is high. Figure 23–73 illustrates a start condition. A bus master can issue a start when the bus is idle or if it already has control of the bus.

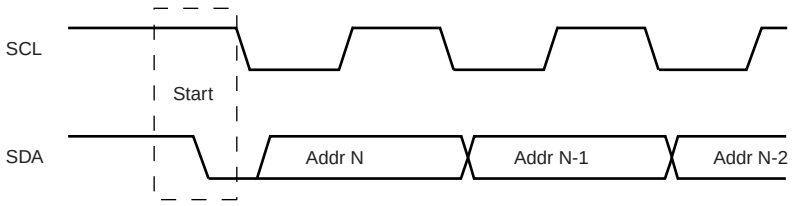


Fig 73. I2C Start Condition

5.5 Stop Condition

A stop condition is signaled by a low-to-high transition of SDA while SCL is high. Figure 23–74 illustrates a stop condition.

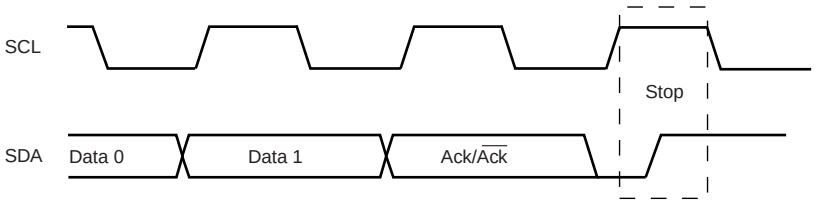


Fig 74. I2C Stop Condition

5.6 Acknowledge

After every byte is transferred the receiver must acknowledge receipt of the byte through an acknowledge bit. This is done by pulling SDA low for one cycle. The acknowledge is just like a data bit in that SDA must be changed when SCL is low and the acknowledge must be stable for the rising edge of SCL. If a bus master doesn't receive an acknowledge after sending a byte, the master issues a stop condition and the transfer is aborted.

When operating as a master-receiver, the slave-transmitter must release the bus at the end of the transfer so the master can generate a stop condition. To force the slave-transmitter to release, the master receiver does not acknowledge the last byte received. See Section 23–4.1 for additional information.

5.7 I2C Addresses

I2C devices can use 7-bit addressing or 10-bit addressing. The bus master must send out an address after issuing a start condition. The address is sent in MSB to LSB order followed by a read (not write) bit which controls the direction of the following transfer. An example of a 7-bit address is shown in Figure 23–75 where S is a start condition, Ax is bit x of the address, r/w is the read (not write) bit, Ack is the acknowledge, and P is a stop condition.

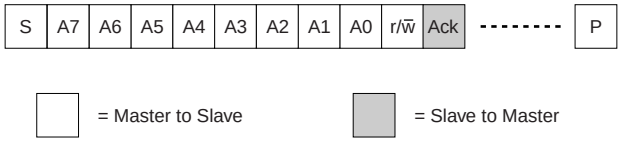


Fig 75. I²C 7-bit device addressing

10-bit addressing is accomplished by sending the address in two bytes. The First byte is 11110xx followed by the r/w bit where xx are the two MSBs of the address. The second byte is the eight LSBs of the address. Figure 23–76 shows 10-bit addressing. Slaves matching the first byte of the address must acknowledge. Slaves matching the second address byte also acknowledge that byte. Writing to a 10-bit address issues the two address bytes followed by the data.

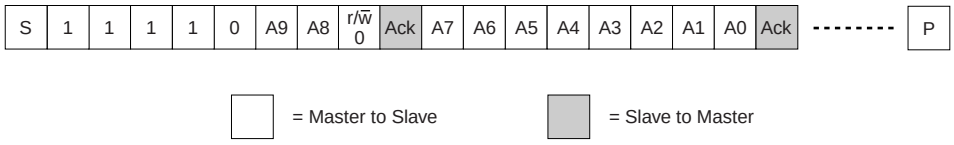


Fig 76. I²C 10-bit device addressing

5.8 I²C Write

To write data on the I2C bus using 7-bit addressing, the master sends out a slave address, a write bit, receives an acknowledge, then sends data bytes in MSB to LSB order receiving an acknowledge after each byte, and finally issues a stop. Figure 23–77 illustrates a write operation.

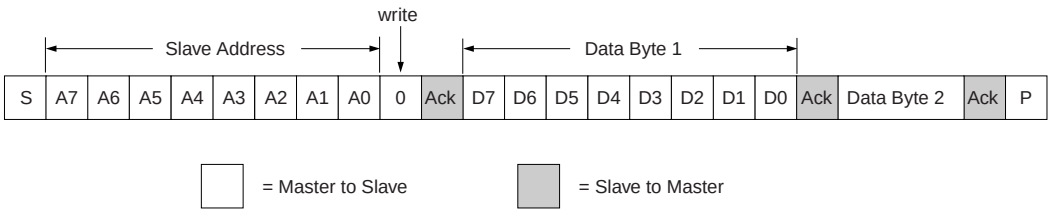


Fig 77. I²C 7-bit slave address write operation

To write data on the I2C bus using 10-bit addressing, the master sends out a the high slave address, a write bit, receives an acknowledge, then sends the low slave address, receives an acknowledge, then sends data bytes in MSB to LSB order receiving an acknowledge after each byte, and finally issues a stop. Figure 23–78 illustrates a 10-bit slave address write operation.

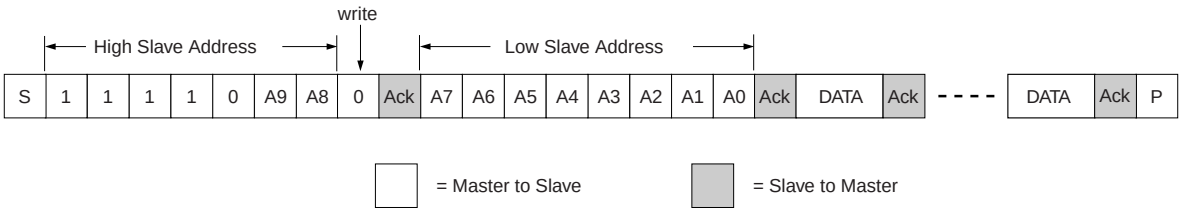


Fig 78. I2C 10-bit slave address write operation

5.9 I2C Read

To read data on the I2C bus, the master send out a slave address, a read bit, receives an acknowledge, then receives data bytes in MSB to LSB order sending an acknowledge after each byte, and finally issues a stop. Figure 23–79 illustrates a read operation. The master drives SCL for the entire operation.

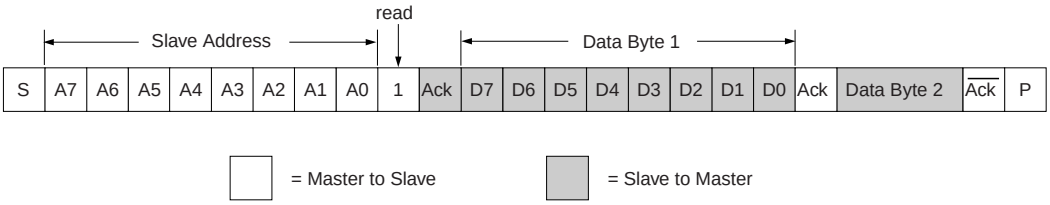


Fig 79. I2C Read Operation from a 7-bit slave address device

Reading from a 10-bit address is somewhat different; The master must write to the slave before restarting and reading from the same slave. The master must send the first byte with the r/w bit low (write) then second address byte, followed by a restart, then send only the first address byte with the r/w bit high (read), then read bytes from the slave. The last byte read from the slave is not acknowledged to signal the end of the read operation.

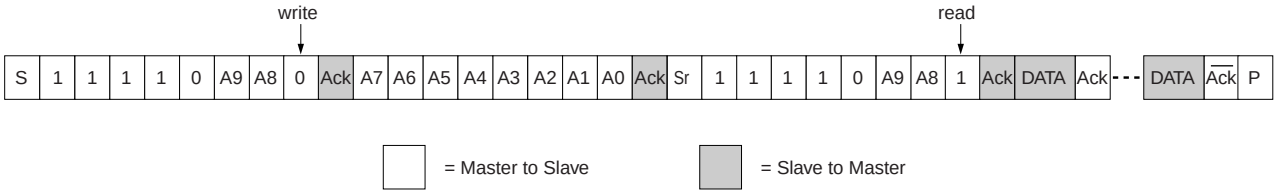


Fig 80. I2C Reading from a 10-bit slave address device

5.10 I2C Write/Read

A repeated start condition allows the bus master to reissue an address with the option of changing the r/w bit. The master can issue any number of start conditions before issuing a stop. Figure 23–81 is an example of a write followed by a read where Sr is a repeated start condition.

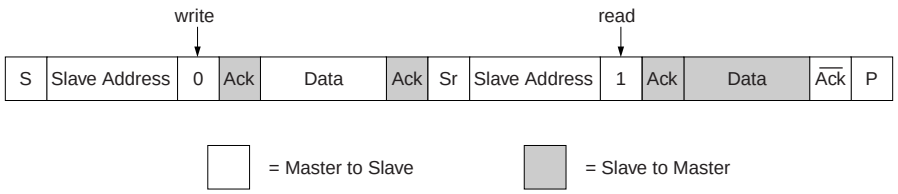


Fig 81. I²C 7-bit slave address write then read operation

5.11 Bus Arbitration

The I2C allows any bus master to start a transfer when the bus is idle. In a multi-master system, it is possible to have more than one master start a transfer at the same time. To arbitrate between masters, all I2C masters must monitor the state of the bus while they are driving it. If a master is trying to put a “1” on the bus while another is driving a “0”, the bus will be low (wire-AND) and the master trying to put a “1” on the bus must abort its operation. The master driving a “0” continues its operation unaware that another master aborted a transfer.

5.12 Clock Synchronization

Since different devices can drive SCL at different frequencies, masters must account for this by starting their clock-high timer when SCL actually goes high rather than when it tries to drive it high. Likewise, the master starts counting the clock-low time when SCL actually goes low. An example is shown in Figure 23–82.

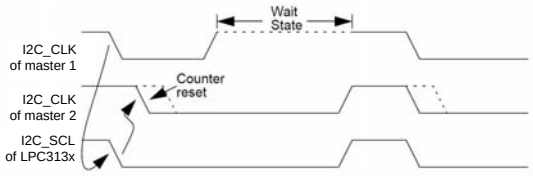


Fig 82. I²C Clock Synchronization

The frequency of the I2C serial clock can be generated with the values of the CLKHI and CLKLO registers. Note that the frequency of the I2C serial clock is dependent of the pull-up resistance Rd and the load. So this frequency is board dependent and will have a value which is in between the maximal possible frequency and the minimal frequency which is possible for certain values of CLKHI and CLKLO. The de-glitch length is 7 I2Cn_PCLK cycles in this case. An indication of the maximal frequency of the I2C serial clock can be calculated and determined with the following figure:

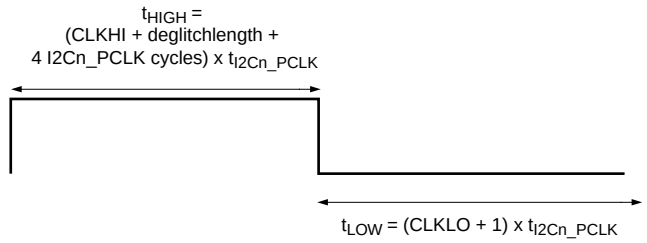


Fig 83. Maximum frequency of the I2C serial clock

The time for the rising and falling edges are neglected for calculation of the maximum possible frequency. This because the time of the edges are low in comparison with the total period time when Rd is low and when the load is low. The 4 extra APB cycles added for the high time, results from internal timing and synchronization. The minimum frequency of the I2C serial clock, by certain values of CLKLO and CLKHI can be calculated and determined with the following figure, which shows 1 period of SCL. Maximum and minimum values of tr, tHIGH, tf and tLOW are shown in the I2C specification.

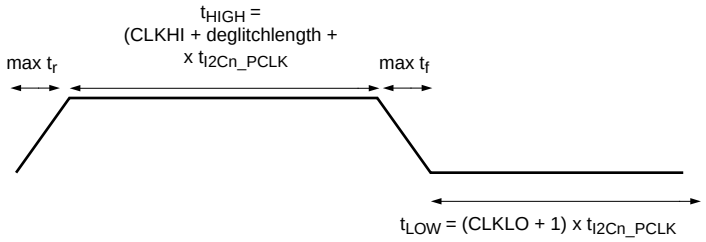


Fig 84. Minimum frequency of the I2C serial clock

The time of the rising edge and falling edge are incorporated in the calculation of the minimal frequency by certain values of CLKHI and CLKLO and with low values of resistance Rd and higher load. This is done because these times are significant in comparison with the total period time. Logic ’1’ is defined as > 0.7 VDD en the logic ’0’ is defined as < 0.4 V. The following timing figure shows the constraint of the minimum I2Cn_PCLK frequency for a certain I2C clock:

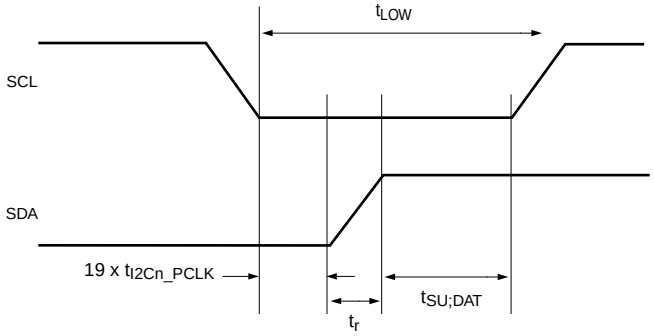


Fig 85. Minimum I2Cn_PCLK frequency for a certain I2C serial clock

An internal counter of 19 I2Cn_PCLK cycles is used before SDA may change after the falling edge of SCL. The following formula is used for calculating minimal period time of the I2Cn_PCLK, by a given value of tLOW of the I2C clock SCL.

$$\left(T_{I2CnPCLK} < \frac{(t_{LOW} - t_r - t_{SU;DAT})}{19}\right)$$

CLKHI and CLKLO can be calculated when T_{I2Cn_PCLK} is known:

$$CLKHI = \frac{t_{HIGH}}{T_{I2CnPCLK}} - deglitchlength$$

$$CLKLO = \frac{t_{LOW}}{T_{I2CnPCLK}} - 1$$

Note that the constraints for t_{HIGH}, t_{LOW}, t_r and $t_{SU;STO}$ can be found in the I2C specification. These values are written down in Table 23–500:

Table 500. Constraints for t_{HIGH}, t_{LOW}, t_r and $t_{SU:STO}$

Parameter	Symbol	Standard mode		Fast mode		Unit
		Min	Max	Min	Max	
SCL clock frequency	f_{SCL}	0	100	0	400	kHz
LOW period of the SCL clock	t_{LOW}	4.7	-	1.3	-	µs
HIGH period of the SCL clock	t_{HIGH}	4.0	-	0.6	-	µs
Rise time of both SDA and SCL signals	t_r	-	1000	20 + 0.1Cb	300	ns
Fall time of both SDA and SCL signals	t_f	-	300	20 + 0.1Cb	300	ns
Set-up time for STOP condition	$t_{SU;STO}$	4.0	-	0.6	-	µs
Data set-up time	$t_{SU;DAT}$	250	-	100	-	ns
I2Cn_PCLK clock frequency (with min. tLOW, max. tr and min. tSU;DAT) for reaching max fSCL	f_{I2Cn_PCLK}	5.5	-	21.1	-	MHz

1. Introduction

The LPC3130/31 contains four fully independent timer modules that can be used to generate interrupts after a pre-set time interval has elapsed.

1.1 Features

- Four fully independent timers.

  Each timer has:

  - Individual clock and select input.
  - A 32 bit wide down-counter with selectable pre-scalers (0, 4 or 8 stages of pre-scale) allowing the system clock division by a factor of 1, 16 or 256.
- Support for two modes of operation:
  - Free-running timer: The timer generates an interrupt when the counter reaches zero. The timer wraps around to 0xFFFFFFFF and continues counting down.
  - Periodic timer: The timer generates an interrupt when the counter reaches zero. It reloads the value from a load register and continues counting down from that value. An interrupt will be generated every time the counter reaches zero. This effectively gives a repeated interrupt at a regular interval.
- At any time the current timer value can be read.
- At any time the value in the load register may be re-written, causing the timer to restart.

2. General description

2.1 Clock Signals

The timer speed depends on the system clock PCLK for each timer.

Table 501. Timer module clock signals

Clock Name	I/O	Source/Destination	Description
TIMER0/1/2/3_PCLK	I	CGU	APB clock

Remark: The clock is asynchronous to the AHB Clock.

2.2 Interface description

2.2.1 Interrupt Requests

The Timer module has four independent interrupt request signals (TIMER0_INT, TIMER1_INT, TIMER2_INT & TIMER3_INT) to the interrupt controller.

2.2.2 Reset Signals

The CGU provides one asynchronous, active LOw reset signal (PRESETn) to each timer.

3. Register overview

Table 502. Register overview: Timer0/1/2/3 (register base address 0x1300 8000 (Timer0), 0x1300 8400 (Timer1), 0x1300 8800 (Timer2), and 0x1300 8C00 (Timer3))

Name	R/W	Address offset	Description
TimerLoad	R/W	0x00	Contains the initial 32 bit value of the timer and is also used as the reload value in periodic timer mode.
TimerValue	R	0x04	The value at this location gives the current 32 bit value of the timer.
TimerCtrl	R/W	0x08	Provides enable/disable mode and pre-scale configurations for the timer. This is explained in more detail in the next section.
TimerClear	W	0x0C	Writing to this location clears the interrupt generated by the counter timer.

4. Register description

4.1 Timer Load register

Table 503. Timer Load register (TimerLoad, address 0x1300 8000 (Timer0), 0x1300 8400 (Timer1), 0x1300 8800 (Timer2), and 0x1300 8C00 (Timer3))

Bit	Symbol	Access	Reset Value	Description
31:0	LOADVALUE	R/W	-	Contains the initial 32 bit value of the timer and is also used as the reload value in periodic timer mode.

4.2 Timer Value register

Table 504. Timer Value register (TimerValue, address 0x1300 8004 (Timer0), 0x1300 8404 (Timer1), 0x1300 8804 (Timer2), and 0x1300 8C04 (Timer3))

Bit	Symbol	Access	Reset Value	Description
31:0	VALUE	R	-	Gives the current 32 bit value of the timer.

4.3 Timer Control register

Table 505. Timer Control register (TimerCtrl, address 0x1300 8008 (Timer0), 0x1300 8408 (Timer1), 0x1300 8808 (Timer2), and 0x1300 8C08 (Timer3))

Bit	Symbol	R/W	Reset Value	Description
31-8	-	-	-	Undefined.Bits must be written as zero and read as undefined.
7	Enable	R/W	0	0-Timer Disabled 1-Timer Enabled
6	Mode	R/W	0	0-Free running Mode 1-Periodic Timer Mode

Table 505. Timer Control register (TimerCtrl, address 0x1300 8008 (Timer0), 0x1300 8408 (Timer1), 0x1300 8808 (Timer2), and 0x1300 8C08 (Timer3)) *…continued*

Bit	Symbol	R/W	Reset Value	Description
5-4	-	-	-	Undefined. Bits must be written as zero and read as undefined.
3-2	PreScale	R/W	0	See Table 24–506
1-0	-	-	-	Undefined. Bits must be written as zero and read as undefined.

Table 506. Pre Scale Bits (Bit 3,2)

Bit 3	Bit 2	Clock divided by	Stages of Pre-Scale
0	0	1	0
0	1	16	4
1	0	256	8
1	1	Undefined	n/a

4.4 Timer Clear register

Table 507. Timer Clear register (TimerClear, address 0x1300 800C (Timer0), 0x1300 840C (Timer1), 0x1300 880C (Timer2), and 0x1300 8C0C (Timer3))

Bit	Symbol	Access	Reset Value	Description
31:0	CLEAR	W	-	Writing to this location clears the interrupt generated by the counter timer.

5. Functional description

The timer is loaded by writing to the Load register and then, if enabled, the timer will count down to zero. On reaching a count of zero an interrupt will be generated. The interrupt may be cleared by writing to the Clear register.

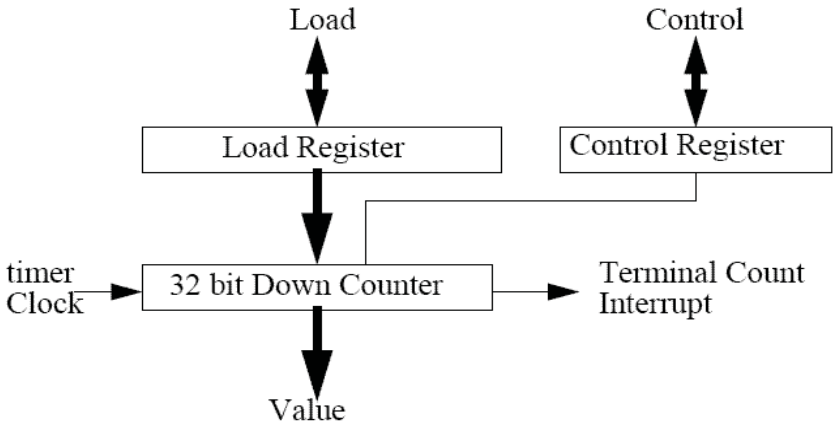


Fig 86. Timer Block Diagram

After reaching a zero count, if the timer is operating in free-running mode then the timer will continue to decrement from its maximum value. If periodic timer mode is selected then the timer will reload from the Load register and continue to decrement. In this mode the timer will effectively generate a periodic interrupt. The mode is selected by a bit in the Control register.

For example, in periodic mode if timer requires to generate an interrupt every 1ms and clock frequency is 1MHz then value 0x3E8 need to be programmed in the Load Register. As counter counts down every clock cycle it will reach 0 after 1ms.

At any point the current timer value may be read from the Value register.

At any point the timer_load may be re-written. This will cause the timer to restart to the timer_load value.

The timer is enabled by a bit in the control register. At reset the timer will be disabled, the interrupt will be cleared and the Load Register will be undefined. The mode and pre-scale value will also be undefined.

The timer clock is generated by a pre-scale unit. The timer clock may be the system clock, the system clock divided by 16, which is generated by 4 bits of pre-scale, or the system clock divided by 256, which is generated by a total of 8 bits of pre-scale.

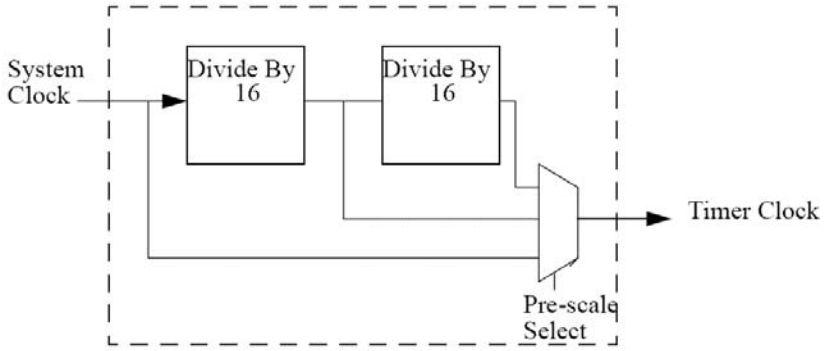


Fig 87. Timer Pre-Scale Unit

6. Power optimization

Power saving can be achieved by switching off the clock to this module by clock gating when the module is not in use. Clock gating can be enabled by setting the RUN bit in PCR register for a particular clock in CGU.

7. Programming guide

To set-up a normal operation following registers has to be programmed:

- Register TimerLoad should be written with the count value. Note that when a new value is written to the load register, the timer will start.
- Register TimerControl should be programmed.

1. Introduction

This pulse width modulation module (PWM) can be used to convert a digital signal to an analogue value by simple external low-pass filtering. The duration of the PWM output being at a high level, is programmed by software. This module also supports pulse density modulation (PDM). A PDM signal is a stream of constant-width pulses, having a density (rate of occurrence) proportional to a corresponding digital value. The PWM is intended to be used for backlighting.

1.1 Features

This module has the following features:

- Programmable Pulse Width Modulation (PWM).
- Supports Pulse Density Modulation (PDM).
- Output can be set to fixed high.
- Output frequency can be adjusted.
- Loop mode.

2. General description

In the PWM mode, a free running 12-bit counter is compared against a value programmed in PWM_TMR[11:0]. The PWM output is high if the value of the counter is lower than the value programmed in a register, and low otherwise. (see Figure 25–88 “PWM output”) Additional features like PDM and LOOP-mode are also supported.

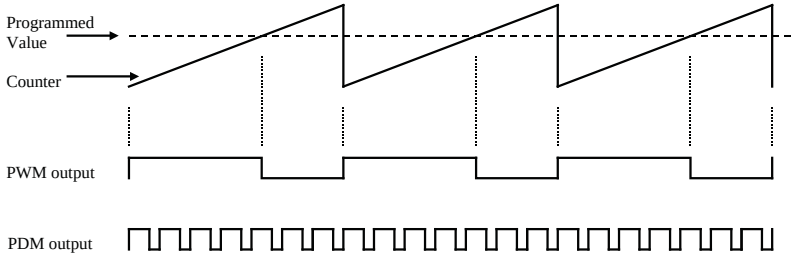


Fig 88. PWM output

2.1 Interface description

2.1.1 Clock Signals

Table 508. Clock Signals of the PWM Module

Clock name	Acronym	I/O	Source /Destination	Description
PWM_PCLK	pclk	I	CGU	APB bus clock. This is the normal APB bus clock.
PWM_PCLK_REGS	pclk_regs	I	CGU	APB bus clock. Gated APB clock, used for register access.
PWM_CLK	pwm_clk	I	CGU	PWM output clock. Clock used for generating the output of the PWM. Adjustable and derived from the same clock base, as the other clocks.

2.1.2 Pin Connections

Table 509. Pin connections of the PWM block

Name	I/O	Description
PWM_DATA	O	PWM modulated output signal to be used for backlighting. The value of PWM_DATA is tri-stated during reset until the desired function is programmed.

2.1.3 Reset Signals

The CGU provides two reset signals to the PWM: An APB reset signal (PNRES) and the PWM functional reset, an active low asynchronous reset signal (PWM_RES_AN).

3. Register overview

Table 510. Register overview: PWM (register base address 0x1300 9000)

Name	R/W	Address Offset	Description
tmr	R/W	0x000	Timer Register
cntl	R/W	0x004	Control Register

4. Register description

PWM contains registers (R/W access is handled through APB bus):

- tmr register only includes a timer value, used for pulse width or pulse density.
- cntl register is divided into 4 fields: cntl.clk, cntl.hi, cntl.loop and cntl.pdm.

Table 511. Timer register (tmr, address 0x1300 9000)

Bit	Symbol	R/W	Reset Value	Description
11.0	MR	R/W	NA	Timer used for PWM and PDM

Table 512. Control register (cntl, address 0x1300 9004)

Bit	Symbol	R/W	Reset Value	Description
1:0	CLK	R/W	0	This will define how the pwm_clk is used for generating the output pulses: 00 = pwm_clk 01 = pwm_clk/2 10 = pwm_clk/4 11 = pwm_clk/8
4	HI	R/W	0	If HI, is set to ‘1’, the pwm output will be forced high.
6	LOOP	R/W	0	If loop is set to ‘1’, the output is inverted with a repetition of the top4 tmr bits
7	PDM	R/W	0	PDM set to ‘1’ will select PDM mode (PWM is standard: PDM = ‘0’)

5. Functional description

In PWM mode, a free running12-bit counter is compared against a value programmed in PMR_TMR [11:0]. The PWM output is high if the value of the counter is lower than the value programmed in a register and low otherwise. (see Figure 25–88 “PWM output”)

In PDM mode, the programmed value TMR [11:0] determines the number of clock cycles to be counted before a PWM pulse is delivered. The total number of pulses that will be generated is defined by the TMR value.

In both modes, the LOOP bit of the Control Register can be used to allow switching the output level after a programmed number (M) of output pulses. This number M is defined by the top 4 bits of the tmr register.

The PWM output clock is derived from the PWM_CLK, and it is selected through the cntl register. Supported values are PWM_CLK, PWM_CLK/2, PWM_CLK/4 and PWM_CLK/8. This is internally not a direct clock divider but is only used for the output frequency.

6. Programming guide

The PWM is initialized with a tri-stated output. The output will be enabled automatically after setting the TMR value. Depending on the implementation of the backlighting implementation one can choose to initialize the output high by selection the HI mode for an active-low backlight or, when the backlight is active-high, the PWM can be initialized by setting the TMR value to '0'.

For driving the LCD backlight, only the PWM and HI modes will be necessary. The internal timer will count up to 4095 (12-bit). The PWM output will have the frequency of the (PWM_CLK frequency) / 4095, with a duty-cycle of ((PWM_TMR) / 4095) x 100%.

1. Introduction

The System Control Registers (SysCReg) module provides a register interface for some of the high-level settings in the system such as multiplexers and mode settings.

1.1 Features

- The SysCReg module contains registers for generic LPC3130/31 configuration:
  - PAD multiplexing and configuration settings
  - USB PLL settings
  - SDRAM refresh configuration
  - ISROM and ISRAM configuration
  - MPMC configuration
  - ADC configuration
  - RNG configuration
  - SD/MMC configuration
  - AHB priority configuration
  - Shadow memory configuration
  - EBI configuration
- The module only consumes power when data is written to it.

2. Interface description

2.1 Clock Signals

Table 513. Clock Signals of the SysCReg Module

Clock Name	I/O	Source/Destination	Description
SYSCREG_PCLK	I	CGU	Main Clock of the module; The logic in this module runs on this clock.

2.2 Reset Signals

The system control block is reset by an APB reset.

3. Register overview

Table 514. Register overview: SysCReg block (register base address 0x1300 2800)

Name	R/W	Address offset	Description
-	R/W	0x000 to 0x004	Reserved
Miscellaneous system configuration registers, part1			
SYSCREG_EBI_MPMC_PRIO	R/W	0X008	Priority of MPMC channel for EBI interface
SYSCREG_EBI_NANDC_PRIO	R/W	0x00C	Priority of NAND controller channel for EBI interface
SYSCREG_EBI_UNUSED_PRIO	R/W	0x010	Priority of unused channel
SYSCREG_RING_OSC_CFG	R/W	0x014	RING oscillator configuration register
SYSCREG_ADC_PD_ADC10BITS	R/W	0x018	Powerdown register of ADC 10bits
SYSCREG_CGU_DYN_HP0	R/W	0x01C	reserved
SYSCREG_CGU_DYN_HP1	R/W	0x020	reserved
SYSCREG_ABC_CFG	R/W	0x024	AHB burst control register
SYSCREG_SD_MMC_CFG	R/W	0x028	SD_MMC (MCI) configuration register
SYSCREG_MCI_DELAYMODES	R/W	0x02C	Delay register for the SD_MMC (MCI) clocks
USB configuration registers			
SYSCREG_USB_ATX_PLL_PD_REG	R/W	0x030	Power down register of USB ATX PLL
SYSCREG_USB_OTG_CFG	R/W	0x034	USB OTG configuration register
SYSCREG_USB_OTG_PORT_IND_CTL	R	0x038	USB OTG port indicator LED control outputs
-	R/W	0x03C	reserved
SYSCREG_USB_PLL_NDEC	R/W	0x040	USB OTG PLL configuration register NOEC
SYSCREG_USB_PLL_MDEC	R/W	0x044	USB OTG PLL configuration register MDEC
SYSCREG_USB_PLL_PDEC	R/W	0x048	USB OTG PLL configuration register PDEC
SYSCREG_USB_PLL_SELR	R/W	0x04C	USB OTG PLL configuration register SELR
SYSCREG_USB_PLL_SELI	R/W	0x050	USB OTG PLL configuration register SELI
SYSCREG_USB_PLL_SELP	R/W	0x054	USB OTG PLL configuration register SELP
ISRAM/ISROM configuration registers			
SYSCREG_ISRAM0_LATENCY_CFG	R/W	0x058	Internal SRAM 0 latency configuration register
SYSCREG_ISRAM1_LATENCY_CFG	R/W	0x05C	Internal SRAM 1 latency configuration register (LPC3131 only)
SYSCREG_ISROM_LATENCY_CFG	R/W	0X060	Internal SROM latency configuration register
MPMC configuration registers			
SYSCREG_AHB_MPMC_MISC	R/W	0x064	Configuration register of MPMC
SYSCREG_MPMP_DELAYMODES	R/W	0x068	Configuration of MPMC clock delay
SYSCREG_MPMC_WAITREAD_DELAY0	R/W	0x06C	Configuration of the wait cycles for read transfers
SYSCREG_MPMC_WAITREAD_DELAY1	R/W	0x070	Configuration of the wait cycles for read transfers
SYSCREG_WIRE_EBI_MSIZE_INIT	R/W	0x074	Configuration of the memory width for MPMC
SYSCREG_MPMC_TESTMODE0	R/W	0x078	Configuration for refresh generation of MPMC
SYSCREG_MPMC_TESTMODE1	R/W	0x07C	Configuration for refresh generation of MPMC
Miscellaneous system configuration registers, part 2			
SYSCREG_AHB0_EXTPRIO	R/W	0x080	Priority of the AHB masters

Table 514. Register overview: SysCReg block (register base address 0x1300 2800)

Name	R/W	Address offset	Description
SYSCREG_ARM926_SHADOW_POINTER	R/W	0x084	Memory mapping
-	R/W	0x088	reserved
-	R	0x08C	reserved
Pin multiplexing control registers			
SYSCREG_MUX_LCD_EBI_SEL	R/W	0x090	Selects between lcd_interface and EBI pins
SYSCREG_MUX_GPIO_MCI_SEL	R/W	0x094	Selects between GPIO and MCI pins
SYSCREG_MUX_NAND_MCI_SEL	R/W	0x098	Selects between NAND flash controller and MCI pins
SYSCREG_MUX_UART_SPI_SEL	R/W	0x09C	Selects between UART and SPI pins
SYSCREG_MUX_I2STX_PCM_SEL	R/W	0x0A0	Selects between I2STX and PCM pins
Pad configuration registers			
SYSCREG_EBI_D_9_PCTRL	R/W	0x0A4	Provides the input to the programmable section of the pad, EBI_D_9
SYSCREG_EBI_D_10_PCTRL	R/W	0x0A8	Provides the input to the programmable section of the pad, EBI_D_10
SYSCREG_EBI_D_11_PCTRL	R/W	0x0AC	Provides the input to the programmable section of the pad, EBI_D_11
SYSCREG_EBI_D_12_PCTRL	R/W	0x0B0	Provides the input to the programmable section of the pad, EBI_D_12
SYSCREG_EBI_D_13_PCTRL	R/W	0x0B4	Provides the input to the programmable section of the pad, EBI_D_13
SYSCREG_EBI_D_14_PCTRL	R/W	0x0B8	Provides the input to the programmable section of the pad, EBI_D_14
SYSCREG_I2SRX_BCK0_PCTRL	R/W	0x0BC	Provides the input to the programmable section of the pad, I2SRX_BCK0
SYSCREG_MGPIO9_PCTRL	R/W	0x0C0	Provides the input to the programmable section of the pad, MGPIO9
SYSCREG_MGPIO6_PCTRL	R/W	0x0C4	Provides the input to the programmable section of the pad, MGPIO6
SYSCREG_MLCD_DB_7_PCTRL	R/W	0x0C8	Provides the input to the programmable section of the pad, MLCD_DB_7
SYSCREG_MLCD_DB_4_PCTRL	R/W	0x0CC	Provides the input to the programmable section of the pad, MLCD_DB_4
SYSCREG_MLCD_DB_2_PCTRL	R/W	0x0D0	Provides the input to the programmable section of the pad, MLCD_DB_2
SYSCREG_MNAND_RYBN0_PCTRL	R/W	0x0D4	Provides the input to the programmable section of the pad, MNAND_RYBN0
SYSCREG_GPIO1_PCTRL	R/W	0x0D8	Provides the input to the programmable section of the pad, GPIO1
SYSCREG_EBI_D_4_PCTRL	R/W	0x0DC	Provides the input to the programmable section of the pad, EBI_D_4
SYSCREG_MI2STX_CLK0_PCTRL	R/W	0x0E0	Provides the input to the programmable section of the pad, MI2STX_CLK0
SYSCREG_MI2STX_BCK0_PCTRL	R/W	0x0E4	Provides the input to the programmable section of the pad, MI2STX_BCK0

Table 514. Register overview: SysCReg block (register base address 0x1300 2800)

Name	R/W	Address offset	Description
SYSCREG_EBI_A_1_CLE_PCTRL	R/W	0x0E8	Provides the input to the programmable section of the pad, EBI_A_1_CLE
SYSCREG_EBI_NCAS_BLOUT_0_PCTRL	R/W	0x0EC	Provides the input to the programmable section of the pad, EBI_NCAS_BLOUT_0
SYSCREG_NAND_NCS_3_PCTRL	R/W	0x0F0	Provides the input to the programmable section of the pad, NAND_NCS_3
SYSCREG_MLCD_DB_0_PCTRL	R/W	0x0F4	Provides the input to the programmable section of the pad, MLCD_DB_0
SYSCREG_EBI_DQM_0_NOE_PCTRL	R/W	0x0F8	Provides the input to the programmable section of the pad, EBI_DQM_0_NOE
SYSCREG_EBI_D_0_PCTRL	R/W	0x0FC	Provides the input to the programmable section of the pad, EBI_D_0
SYSCREG_EBI_D_1_PCTRL	R/W	0x100	Provides the input to the programmable section of the pad, EBI_D_1
SYSCREG_EBI_D_2_PCTRL	R/W	0x104	Provides the input to the programmable section of the pad, EBI_D_2
SYSCREG_EBI_D_3_PCTRL	R/W	0x108	Provides the input to the programmable section of the pad, EBI_D_3
SYSCREG_EBI_D_5_PCTRL	R/W	0x10C	Provides the input to the programmable section of the pad, EBI_D_5
SYSCREG_EBI_D_6_PCTRL	R/W	0x110	Provides the input to the programmable section of the pad, EBI_D_6
SYSCREG_EBI_D_7_PCTRL	R/W	0x114	Provides the input to the programmable section of the pad, EBI_D_7
SYSCREG_EBI_D_8_PCTRL	R/W	0x118	Provides the input to the programmable section of the pad, EBI_D_8
SYSCREG_EBI_D_15_PCTRL	R/W	0x11C	Provides the input to the programmable section of the pad, EBI_D_15
SYSCREG_I2STX_DATA1_PCTRL	R/W	0x120	Provides the input to the programmable section of the pad, I2STX_DATA1
SYSCREG_I2STX_BCK1_PCTRL	R/W	0x124	Provides the input to the programmable section of the pad, I2STX_BCK1
SYSCREG_I2STX_WS1_PCTRL	R/W	0x128	Provides the input to the programmable section of the pad, I2STX_WS1
SYSCREG_I2SRX_DATA0_PCTRL	R/W	0x12C	Provides the input to the programmable section of the pad, I2SRX_DATA0
SYSCREG_I2SRX_WS0_PCTRL	R/W	0x130	Provides the input to the programmable section of the pad, I2SRX_WS0
SYSCREG_I2SRX_DATA1_PCTRL	R/W	0x134	Provides the input to the programmable section of the pad, I2SRX_DATA1
SYSCREG_I2SRX_BCK1_PCTRL	R/W	0x138	Provides the input to the programmable section of the pad, I2SRX_BCK1
SYSCREG_I2SRX_WS1_PCTRL	R/W	0x13C	Provides the input to the programmable section of the pad, I2SRX_WS1
SYSCREG_SYSCLK_O_PCTRL	R/W	0x140	Provides the input to the programmable section of the pad, SYSCLK_O

Table 514. Register overview: SysCReg block (register base address 0x1300 2800)

Name	R/W	Address offset	Description
SYSCREG_PWM_DATA_PCTRL	R/W	0x144	Provides the input to the programmable section of the pad, PWM_DATA
SYSCREG_UART_RXD_PCTRL	R/W	0x148	Provides the input to the programmable section of the pad, UART_RXD
SYSCREG_UART_TXD_PCTRL	R/W	0x14C	Provides the input to the programmable section of the pad, UART_TXD
SYSCREG_I2C_SDA1_PCTRL	R/W	0x150	Provides the input to the programmable section of the pad, I2C_SDA1
SYSCREG_I2C_SCL1_PCTRL	R/W	0x154	Provides the input to the programmable section of the pad, I2C_SCL1
SYSCREG_CLK_256FS_O_PCTRL	R/W	0x158	Provides the input to the programmable section of the pad, CLK_256FS_O
SYSCREG_GPIO0_PCTRL	R/W	0x15C	Provides the input to the programmable section of the pad, GPIO0
SYSCREG_GPIO2_PCTRL	R/W	0x160	Provides the input to the programmable section of the pad, GPIO2
SYSCREG_GPIO3_PCTRL	R/W	0x164	Provides the input to the programmable section of the pad, GPIO3
SYSCREG_GPIO4_PCTRL	R/W	0x168	Provides the input to the programmable section of the pad, GPIO4
SYSCREG_GPIO11_PCTRL	R/W	0x16C	Provides the input to the programmable section of the pad, GPIO11
SYSCREG_GPIO12_PCTRL	R/W	0x170	Provides the input to the programmable section of the pad, GPIO12
SYSCREG_GPIO13_PCTRL	R/W	0x174	Provides the input to the programmable section of the pad, GPIO13
SYSCREG_GPIO14_PCTRL	R/W	0x178	Provides the input to the programmable section of the pad, GPIO14
SYSCREG_GPIO15_PCTRL	R/W	0x17C	Provides the input to the programmable section of the pad, GPIO15
SYSCREG_GPIO16_PCTRL	R/W	0x180	Provides the input to the programmable section of the pad, GPIO16
SYSCREG_GPIO17_PCTRL	R/W	0x184	Provides the input to the programmable section of the pad, GPIO17
SYSCREG_GPIO18_PCTRL	R/W	0x188	Provides the input to the programmable section of the pad, GPIO18
SYSCREG_GPIO19_PCTRL	R/W	0x18C	Provides the input to the programmable section of the pad, GPIO19
SYSCREG_GPIO20_PCTRL	R/W	0x190	Provides the input to the programmable section of the pad, GPIO20
SYSCREG_SPI_MISO_PCTRL	R/W	0x194	Provides the input to the programmable section of the pad, SPI_MISO
SYSCREG_SPI_MOSI_PCTRL	R/W	0x198	Provides the input to the programmable section of the pad, SPI_MOSI
SYSCREG_SPI_CS_IN_PCTRL	R/W	0x19C	Provides the input to the programmable section of the pad, SPI_CS_IN

Table 514. Register overview: SysCReg block (register base address 0x1300 2800)

Name	R/W	Address offset	Description
SYSCREG_SPI_SCK_PCTRL	R/W	0x1A0	Provides the input to the programmable section of the pad, SPI_SCK
SYSCREG_SPI_CS_OUT0_PCTRL	R/W	0x1A4	Provides the input to the programmable section of the pad, SPI_CS_OUT0
SYSCREG_NAND_NCS_0_PCTRL	R/W	0x1A8	Provides the input to the programmable section of the pad, NAND_NCS_0
SYSCREG_NAND_NCS_1_PCTRL	R/W	0x1AC	Provides the input to the programmable section of the pad, NAND_NCS_1
SYSCREG_NAND_NCS_2_PCTRL	R/W	0x1B0	Provides the input to the programmable section of the pad, NAND_NCS_2
SYSCREG_MLCD_CSB_PCTRL	R/W	0x1B4	Provides the input to the programmable section of the pad, MLCD_CSB
SYSCREG_MLCD_DB_1_PCTRL	R/W	0x1B8	Provides the input to the programmable section of the pad, MLCD_DB_1
SYSCREG_MLCD_E_RD_PCTRL	R/W	0x1BC	Provides the input to the programmable section of the pad, MLCD_E_RD
SYSCREG_MLCD_RS_PCTRL	R/W	0x1C0	Provides the input to the programmable section of the pad, MLCD_RS
SYSCREG_MLCD_RW_WR_PCTRL	R/W	0x1C4	Provides the input to the programmable section of the pad, MLCD_RW_WR
SYSCREG_MLCD_DB_3_PCTRL	R/W	0x1C8	Provides the input to the programmable section of the pad, MLCD_DB_3
SYSCREG_MLCD_DB_5_PCTRL	R/W	0x1CC	Provides the input to the programmable section of the pad, MLCD_DB_5
SYSCREG_MLCD_DB_6_PCTRL	R/W	0x1D0	Provides the input to the programmable section of the pad, MLCD_DB_6
SYSCREG_MLCD_DB_8_PCTRL	R/W	0x1D4	Provides the input to the programmable section of the pad, MLCD_DB_8
SYSCREG_MLCD_DB_9_PCTRL	R/W	0x1D8	Provides the input to the programmable section of the pad, MLCD_DB_9
SYSCREG_MLCD_DB_10_PCTRL	R/W	0x1DC	Provides the input to the programmable section of the pad, MLCD_DB_10
SYSCREG_MLCD_DB_11_PCTRL	R/W	0x1E0	Provides the input to the programmable section of the pad, MLCD_DB_11
SYSCREG_MLCD_DB_12_PCTRL	R/W	0x1E4	Provides the input to the programmable section of the pad, MLCD_DB_12
SYSCREG_MLCD_DB_13_PCTRL	R/W	0x1E8	Provides the input to the programmable section of the pad, MLCD_DB_13
SYSCREG_MLCD_DB_14_PCTRL	R/W	0x1EC	Provides the input to the programmable section of the pad, MLCD_DB_14
SYSCREG_MLCD_DB_15_PCTRL	R/W	0x1F0	Provides the input to the programmable section of the pad, MLCD_DB_15
SYSCREG_MGPIO5_PCTRL	R/W	0x1F4	Provides the input to the programmable section of the pad, MGPIO5
SYSCREG_MGPIO7_PCTRL	R/W	0x1F8	Provides the input to the programmable section of the pad, MGPIO5

Table 514. Register overview: SysCReg block (register base address 0x1300 2800)

Name	R/W	Address offset	Description
SYSCREG_MGPIO8_PCTRL	R/W	0x1FC	Provides the input to the programmable section of the pad, MGPIO8
SYSCREG_MGPIO10_PCTRL	R/W	0x200	Provides the input to the programmable section of the pad, MGPIO10
SYSCREG_MNAND_RYBN1_PCTRL	R/W	0x204	Provides the input to the programmable section of the pad, MNAND_RYBN1
SYSCREG_MNAND_RYBN2_PCTRL	R/W	0x208	Provides the input to the programmable section of the pad, MNAND_RYBN2
SYSCREG_MNAND_RYBN3_PCTRL	R/W	0x20C	Provides the input to the programmable section of the pad, MNAND_RYBN3
SYSCREG_MUART_CTS_N_PCTRL	R/W	0x210	Provides the input to the programmable section of the pad, MUART_CTS_N
SYSCREG_MI2STX_DATA0_PCTRL	R/W	0x218	This register, SYSCREG_MI2STX_DATA0_PCTRL, provides the input to the programmable section of the pad, MI2STX_DATA0
SYSCREG_MI2STX_WS0_PCTRL	R/W	0x21C	Provides the input to the programmable section of the pad, MI2STX_WS0
SYSCREG_EBI_NRAS_BLOUT_1_PCTRL	R/W	0x220	Provides the input to the programmable section of the pad, EBI_NRAS_BLOUT_1
SYSCREG_EBI_A_0_ALE_PCTRL	R/W	0x224	Provides the input to the programmable section of the pad, EBI_A_0_ALE
SYSCREG_EBI_NWE_PCTRL	R/W	0x228	Provides the input to the programmable section of the pad, EBI_NWE
SYSCREG_ESHCTRL_SUP4	R/W	0x22C	Provides the input to control the performance of the pad at 1.8 and 2.8 V (Nandflash/EBI pads)
SYSCREG_ESHCTRL_SUP8	R/W	0x230	Provides the input to control the performance of the pad at 1.8 and 2.8 V (LCD interface/SDRAM pads)

4. Register description

4.1 Miscellaneous system control registers, part 1

4.1.1 EBI timeout registers

Table 515. SYSCREG_EBI_MPMC_PRIO (address 0x1300 2808)

Bit	Symbol	R/W	Reset Value	Description
31:10	-	-	-	Reserved
9:0	TIMEOUTVALUE	R/W	0x0	Time out value of the MPMC channel. The higher the time out value the lower the priority is.

Table 516. SYSCREG_EBI_NANDC_PRIO (address 0x1300 280C)

Bit	Symbol	R/W	Reset Value	Description
31:10	-	-	-	Reserved
9:0	TIMEOUTVALUE	R/W	0xF	Time out value of the NAND controller channel. The higher the time out value the lower the priority is.

Table 517. SYSCREG_EBI_UNUSED_PRIO (address 0x1300 2810)

Bit	Symbol	R/W	Reset Value	Description
31:10	-	-	-	Reserved
9:0	TIMEOUTVALUE	R/W	0xF	Time out value of unused channel. Program 0x3F to set this channel as lowest priority.

4.1.2 Ring oscillator enable register

Table 518. RING_OSC_CFG (address 0x1300 2814)

Bit	Symbol	R/W	Reset Value	Description
1	ring_osc_cfg_osc1_en	R/W	0x1	Enable of the ring oscillator 1
0	ring_osc_cfg_osc0_en	R/W	0x1	Enable of the ring oscillator 0

4.1.3 ADC power-down register

Table 519. SYSCREG _ADC_PD_ADC10BITS (address 0x1300 2818)

Bit	Variable	R/W	Reset Value	Description
0	adc_pd_adc10bits	R/W	0x0	Powerdown bit 10 bits ADC. '0' activates the 10 bits ADC. See also Section 16–6.

4.1.4 AHB master configuration register

Table 520. SYSCREG_ABC_CFG (address 0x1300 2824)

Bit	AHB MASTER	R/W	Reset Value	Description
31:12	Reserved	R/W	0x0	Reserved
11:9	Usb_otg	R/W	0x0	3 bits for the AHB master USB_OTG to control its AHB bus bandwidth
8:6	arm926ejs_i	R/W	0x0	3 bits for the AHB master ARM926EJS instruction port to control its AHB bus bandwidth
5:3	arm926ejs_d	R/W	0x0	3 bits for the AHB master ARM926EJS data port to control its AHB bus bandwidth
2:0	Simple dma	R/W	0x0	3 bits for the AHB master dma to control its AHB bus bandwidth

For each AHB master, the 3 control bits correspond to the register bits as listed in Table 26–521.

Table 521. AHB master control bits

Bit	MODE	Description
000	Normal mode	The default setting. This setting does not impact normal operation.No manipulations will be performed on the AHB signals
001	Make any burst a non-sequential access	When the selected master performs a burst operation, it will be treated like single transfers.This allows the AHB multilayer to re-arbitrate after each single access.Note: It will make the data-transfers very inefficient on the AHB bus, slowing down the selected master more than the other modes.
010	SPlit to 4-beat	This setting will break an INCR8 or INCR16 burst to a 4-burst INCR. If the selected master puts an 8-beat or 16-beat incrementing burst on the bus, it will be split after every 4 words, allowing the AHB multilayer to re-arbitrate. This setting improves the access speed of the other masters.
011	SPlit to 8-beat	This setting will break any INCR burst to a 8-burst INCR. If the selected master puts a 16-beat incrementing burst on the bus, it will be split at the 8-th word, allowing the AHB multilayer to re-arbitrate in the middle.This setting improves the access speed of the other masters (goal: YUV).
100	eXTend to 8-beat	This setting will extend two sequential 4-beat incr to an '8-beat'. This only works if the master really has 2 sequential words on the bus, without delays. The hardware checks if the second word is sequential to the first word and will extend the transfer to 8 words.This will give the selected master more bandwidth compared to other masters if bus-arbitration is required.
101	eXTend to 16-beat	This setting will extend two sequential 8-beat incr to a '16-beat'. This only works if the master really has 2 sequential words on the bus, without delays. The hardware checks if the second word is sequential to the first word and will extend the transfer to 16 words.This will give the selected master more bandwidth compared to other masters if bus-arbitration is required.
110	SPlit to 4-beat	This setting will break an WRAP8 to WRAP4. This will break a wrapping 8-beat burst to two 4-beat wrapping words.This will only be done if the transfer is aligned with the beginning of an 8-beat burst.
111	eXTend to 32-beat	This setting will extend four sequential 8-beat incr (or multi 4-beats or 16-beats) to a '32-beat'. This only works if the master really sets 2 sequential words on the bus, without delays. The hardware checks if the second word is sequential to the first word and will extend the transfer to 32 words.This will give the selected master more bandwidth compared to other masters if bus-arbitration is required.

4.2 SD/MMC configuration registers

Table 522. SYSCREG_SD_MMC_CFG (address 0x1300 2828)

Bit	Symbol	R/W	Reset Value	Description
31:2	-	-	0x0	reserved
1	card_detect_n	R/W	0x1	Card detect signal. A 0 represents presence of card. Default is one. Any change in this signal will cause card_detect interrupt in SD_MMC module, if enabled. Software should program this bit by detecting the event from unused external GPIO pin, so that the SD_MMC module responds to the event.
0	card_write_prt	R/W	0x0	Card write protect signal for SD cards. A 1 represents write is protected. Default is zero. Software should program this bit by detecting the event from unused external GPIO pin, so that the SD_MMC module responds to the event.

Table 523. SYSCREG_MCI_DELAYMODES (address 0x1300 282C)

Bit	Symbol	R/W	Reset Value	Description
31:5	-	-	0x0	reserved
4	delay_enable	R/W	0x0	Enable delay cells.
3:0	delay_cells	R/W	0x0	This bus-signal specifies the number of delay cells to obtain the needed delay for cclk_in_drv. The delay should be ~5ns in comparison to cclk_out for SD cards, 2 ns for high-speed SD cards and 3 ns for (H)MMC and CE-ATA. (Cards need x ns holds on all inputs, so all outputs clocked out of cclk_in are re-clocked by cclk_in_drv to meet card hold-time requirement.). See SD_MMC chapter for more details.

4.3 USB registers

Table 524. USB_ATX_PLL_PD_REG (address 0x1300 2830)

Bit	Symbol	R/W	Reset Value	Description
0	USB_ATX_PLL_PD_REG	R/W	0x1	Powerdown bit of the USB pll. 0x0: in powerdown mode 0x1: in active mode

Table 525. USB_OTG_CFG (address 0x1300 2834)

Bit	Symbol	R/W	Reset Value	Description
4	Reserved	R/W	0x0	-
3	usb_otg_vbus_pwr_fault	R/W	0x0	Indication of the charge pump overcurrent.Software should set this bit when it detects external wake-up event (from external GPIO pin) in host mode so that USB_OTG IP responds to the event. For more explanation see USB chapter.
2	usb_otg_dev_wakeup_n	R/W	0x0	External wakeup signal for device mode. Software should set this bit when it detects external wake-up event (from external GPIO pin) in host mode so that USB_OTG block responds to the event. For more explanation see Section 8–9.2.
1	usb_otg_host_wakeup_n	R/W	0x1	External wake-up signal for host mode. Software should set this bit when it detects external wake-up event (from external GPIO pin) in host mode so that USB_OTG block responds to the event. For more explanation see Section 8–9.3.
0	-	R/W	0x0	Reserved

Table 526. USB_OTG_PORT_IND_CTL (address 0x1300 2838)

Bit	Symbol	R/W	Reset Value	Description
1:0	USB_OTG_PORT_IND_CTL	R	0x0	Status bits for USB connector LEDs, 00=off, 01=amber, 10=green, 11=undefined (more information in USB chapter). Software should read this register and drive USB LEDs (if present on board) appropriately.

4.3.1 USB PLL configuration registers

Table 527. USB_PLL_NDEC (address 0x1300 2840)

Bit	Symbol	R/W	Reset Value	Description
9:0	USB_PLL_NDEC	R/W	0x0	Pre-divider for the USB pll. The default value should not be modified.

Table 528. USB_PLL_MDEC (address 0x1300 2844)

Bit	Symbol	R/W	Reset Value	Description
16:0	USB_PLL_MDEC	R/W	0x7FFA	Feedback-divider for the USB pll. The default value should not be modified.

Table 529. USB_PLL_PDEC (address 0x1300 2848)

Bit	Symbol	R/W	Reset Value	Description
3:0	USB_PLL_PDEC	R/W	0x0	Feedback-divider for the USB pll. The default value should not be modified.

Table 530. USB_PLL_SELR (address 0x1300 284C)

Bit	Symbol	R/W	Reset Value	Description
3:0	USB_PLL_SELR	R/W	0x0	Bandwidth selection selr. This should not be modified.

Table 531. USB_PLL_SELI (address 0x1300 2850)

Bit	Symbol	R/W	Reset Value	Description
3:0	USB_PLL_SELI	R/W	0x18	Bandwidth selection seli. This should not be modified.

Table 532. USB_PLL_SELP (address 0x1300 2854)

Bit	Symbol	R/W	Reset Value	Description
3:0	USB_PLL_SELP	R/W	0xB	Bandwidth selection selp. This should not be modified.

4.4 ISRAM configuration registers

These registers contain the waitstates programming for the internal Ram (ISRAM0/1).

Table 533. SYSCREG_ISRAM0_LATENCY_CFG (address 0x1300 2858)

Bit	Symbol	R/W	Reset Value	Description
1:0	Isram0_latency_cfg	R/W	0x0	Number of waitstates. 00=0 waitstates 01=1 waitstate 11=2 waitstates

Table 534. SYSCREG_ISRAM1_LATENCY_CFG (address 0x1300 285C)

Bit	Symbol	R/W	Reset Value	Description
1:0	Isram1_latency_cfg	R/W	0x0	Number of waitstates. 00=0 waitstates 01=1 waitstate 11=2 waitstates

4.5 ISROM configuration registers

This register contains the waitstates programming for the internal Ram (ISRROM).

Table 535. SYSCREG_ISROM_LATENCY_CFG (address 0x1300 2860)

Bit	Symbol	R/W	Reset Value	Description
1:0	Isrom_latency_cfg	R/W	0x0	Number of waitstates. 00=0 waitstates 01=1 waitstate 11=2 waitstates

4.6 MPMC configuration registers

4.6.1 Static memory chip and address select modes

Table 536. SYSCREG_AHB_MPMC_MISC (address 0x1300 2864)

Bit	Symbol	R/W	Reset Value	Description
0	ahb_mpmc_misc_srefreq	R/W	0x0	Self refresh request, when '1' it indicates a self refresh request from the CGU.
1	Reserved	-	0x0	Reserved
2	Reserved	-	0x0	Reserved
3	ahb_mpmc_misc_stcs0pol	R/W	0x0	Polarity of static memory CS0. This power on reset value can be over written through the register interface. When '1', it indicates active HIGH chip select and when '0', it indicates an active LOW chip select.
4	ahb_mpmc_misc_stcs1pol	R/W	0x0	Polarity of static memory CS1. This power on reset value can be over written through the register interface. When '1', it indicates active HIGH chip select and when '0', it indicates an active LOW chip select.
5	-	-	0x0	reserved
6	-	-	0x0	reserved
7	ahb_mpmc_misc_stcs1pb	R/W	0x0	Polarity of byte lane select for static memory CS1. This power on reset value can be over written through the register interface. When '1', for reads and write, the respective active bits of nMPMCBLSOUT[3:0] are LOW. When '0', for reads, all the bits of nMPMCBLSOUT[3:0] are HIGH and for writes, the respective active bits of nMPMCBLSOUT[3:0] are LOW.
8	ahb_mpmc_misc_rel1config	R/W	0x0	Static memory address mode select (more information see below).

Static memory address mode select:

When LOW, it indicates that the static memory addresses should be connected as follows:

When external memory width is 8-bits or 16-bits, EBI_A[15:0] should be connected to the A[15:0] pins of the static memory device. When memory width (MW field) in MPMCStaticConfigx register is set as 16-bit, LPC313x automatically shifts AHB address to EBI address (EBI_A[15:0] = AHB[16:1]).

When HIGH, the static memory address should be connected as follows:

- When external memory width is 8-bits, EBI_A[15:0] should be connected to the A[15:0] pins of the static memory device.

- When external memory width is 16-bits, EBI_A[15:1] should be connected to the A[14:0] pins of the static memory device and EBI_A[0] is not used. LPC313x does not automatically shift AHB address to EBI address even when memory width (MW field) in MPMCStaticConfigx register is set as 16-bit.

4.6.2 Static memory delay modes

Table 537. SYSCREG_MPMP_DELAYMODES (address 0x1300 2868)

Bit	Symbol	R/W	Reset Value	Description
5:0	MPMC_delaymodes	R/W	110010 (binary)	Configures the amount of delay cells between MPMCCLK and MPMCFBCLKIN. MPMCFBCLKIN is the feedback clock for SDRAM. See for the relation between the programmed value and the amount of delay cells.
11:6	MPMC_delaymodes	R/W	110010 (binary)	Configures the amount of delay cells, between MPMCCLK and MPMCCLKDELAY. This clock is used in Command Delayed strategy. See for the relation between the programmed value and the amount of delay cells.
17:12	MPMC_delaymodes	R/W	110010 (binary)	Configures the amount of delay cells, used for delaying MPMCCLKOUT. This 'clock out' goes to the external SDRAM. See for relation between the programmed value and the amount of delay cells.

See for relation between programmed value and amount of delay cells in Table 26–538.

Table 538. MPMC Delay line settings

Setting	WC Delay	BC Delay	Setting	WC Delay	BC Delay
00	0.44	0.16			
20	5.12	2.61	30	24.15	10.40
21	6.43	3.14	31	25.46	10.93
22	7.60	3.59	32	26.63	11.38
23	8.91	4.12	33	27.94	11.91
24	10.01	4.61	34	29.04	12.40
25	11.32	5.14	35	30.35	12.93
26	12.49	5.59	36	31.52	13.38
27	13.80	6.12	37	32.83	13.91
28	14.75	6.56	38	33.78	14.35
29	16.06	7.09	39	35.09	14.88
2A	17.23	7.54	3A	36.26	15.33
2B	18.54	8.07	3B	37.57	15.86
2C	19.64	8.56	3C	38.67	16.35

Table 538. MPMC Delay line settings

Setting	WC Delay	BC Delay	Setting	WC Delay	BC Delay
2D	20.95	9.09	3D	39.98	16.88
2E	22.12	9.54	3E	41.15	17.33
2F	23.43	10.07	3F	42.46	17.86

This register is used for the static device0 of the MPMC. It provides that the output enable signal for the static device0 (OE) is split up into two equal portions, with one inactive cycle in the middle. This is needed because some memories do no detect consecutive reads within one OE period.

Table 539. SYSCREG_MPMC_WAITREAD_DELAY0 (address 0x1300 286C)

Bit	Symbol	R/W	Reset Value	Description
[5]	Enable_extra_OE_inactive_cycle	R/W	0x0	Enable the extra inactive OE cycle if bit is 1.
[4:0]	Static read wait counter	R/W	0x1	Program the value that you have programmed in MPMCStaticWaitRd0

This register is used for the static device1 of the MPMC. It provides that the output enable signal for the static device1 (OE) is split up into two equal portions, with one inactive cycle in the middle. This is needed because some memories do no detect consecutive reads within one OE period.

Table 540. SYSCREG_MPMC_WAITREAD_DELAY1 (address 0x1300 2870)

Bit	Symbol	R/W	Reset Value	Description
[5]	Enable_extra_OE_inactive_cycle	R/W	0x0	Enable the extra inactive OE cycle if bit is 1.
[4:0]	Static read wait counter	R/W	0x1	Program the value that you have programmed in MPMCStaticWaitRd1

4.6.3 MPMC CS1 memory width register

Table 541. SYSCREG_WIRE_EBI_MSIZE_INIT (address 0x1300 2874)

Bit	Symbol	R/W	Reset Value	Description
[1:0]	wire_ebi_msize_init	R/W	0x1	memory width of CS1, ahb_mpmc_misc_stcs1mw[1:0]. This power on reset value can be over written through the register interface. ’00’ indicates 8-bits, ’01’ indicates 16-bits and ’10’ & ’11’ are reserved. Do not change this register during normal operation.

4.6.4 MPMC testmode registers

Table 542. MPMC_TESTMODE0 (address 0x1300 2878)

Bit	Symbol	R/W	Reset Value	Description
11:0	External refresh counter value	R/W	0x0	The value programmed here times 16 times 'base_clk cycle time' is the period of every external refresh. For more information see below this table.
12:0	External refresh enable	R/W	0x0	When the External refresh bit is '1', then the external refresh generator will take over the refresh generation of the MPMC. '0' Normal MPMC refresh method '1' The external refresh generator generates the refresh for the MPMC

The value programmed in 'external refresh counter value' will be the timing of the refresh. The clock of the external refresh generator should always be the base_clock, making the refresh AHB clock independent.

This value can be calculated like this:

- refresh_time/(base_clk_period*16)
- For example, the refresh time is 15us and the base_clock of the SYS_BASE domain is 60 MHz (16.6 ns period time)
- Calculated: 15000/(16.6 * 16) = 56 (always round downward)

Table 543. MPMC_TESTMODE1 (address 0x1300 287C)

Bit	Symbol	R/W	Reset Value	Description
[7:0]	high speed enable counter for the external refresh generator.	R/W	0x0	The value programmed in this register will determine the amount of clock cycles The 'hi_speed_enable' towards the CGU will be active at the moment of refresh request. This allows the AHB clock to temporarily run faster while refreshing, which might improve SDRAM power consumption.

4.7 Miscellaneous system configuration registers, part2

4.7.1 AHB external priority settings

Table 544. AHB0_EXTPRIO (address 0x1300 2880)

Bit	Symbol	R/W	Reset Value	Description
0	DMA	R/W	0x0	If this bit =1 then DMA has higher priority on the AHB bus then the other AHB masters of which this bit is not set.
1	ARM926 Instruction bus	R/W	0x0	If this bit =1 then ARM926 Instruction has higher priority on the AHB bus then the other AHB masters of which this bit is not set.
2	ARM926 Data bus	R/W	0x0	If this bit =1 then ARM926 Data has higher priority on the AHB bus then the other AHB masters of which this bit is not set.
3	USB OTG	R/W	0x0	If this bit =1 then USB OTG has higher priority on the AHB bus then the other AHB masters of which this bit is not set.

The rules for determining which master is granted for slave x are described in Section 11–1.2.

4.7.2 Shadow memory control register

Table 545. SYSCREG_ARM926_SHADOW_POINTER (address 0x1300 2884)

Bit	Symbol	R/W	Reset Value	Description
31:0	ARM926EJS_shadow_pointer	R/W	0x1200 0000	This register is provided to be able to change the memory mapping. The first 4KB address space of the 32 bit address value programmed in this register is mirrored/shadowed at address 0x0 for ARM926EJS bus master. Note, other bus masters on AHB matrix do not have this re-map logic. The lower 10bits of the address value should always be zeros. It is freely programmable in increments of 1 kByte.

For more information see also Section 11–1.2.

4.8 Pin multiplexing registers

Table 546. SYSCREG_MUX_LCD_EBI_SEL(address 0x1300 2890)

Bit	Symbol	R/W	Reset Value	Description
0	Mux_LCD_EBI_sel	R/W	0x0	Selects between lcd_interface and EBI/MPMC pins. 0 - lcd interface; 1 - EBI/MPMC The pins are LCD_CSB - MPMC_NSTCS_0 LCD_DB_1 - MPMC_NSTCS_1 LCD_DB_0 - MPMC_CLKOUT LCD_E_RD - MPMC_CKE LCD_RS - MPMC_NDYCS LCD_RW_WR - MPMC_DQM_1 LCD_DB_2 - EBI_A_2 LCD_DB_3 - EBI_A_3 l LCD_DB_4 - EBI_A_4 l LCD_DB_5 - EBI_A_5 l LCD_DB_6 - EBI_A_6 LCD_DB_7 - EBI_A_7 LCD_DB_8 - EBI_A_8 LCD_DB_9 - EBI_A_9 LCD_DB_10 - EBI_A_10 LCD_DB_11 - EBI_A_11 LCD_DB_12 - EBI_A_12 LCD_DB_13 - EBI_A_13 LCD_DB_14 - EBI_A_14 LCD_DB_15 - EBI_A_15

Table 547. SYSCREG_MUX_GPIO_MCI_SEL (address 0x1300 2894)

Bit	Symbol	R/W	Reset Value	Description
0	Mux_GPIO_MCI_sel	R/W	0x0	Selects between GPIO and MCI pins. 0 - GPIO; 1 - MCI The pins are GPIO5 - MCI_CLK GPIO6 - MCI_CMD GPIO7 - MCI_DAT_0 GPIO8 - MCI_DAT_1 GPIO9 - MCI_DAT_2 GPIO10 - MCI_DAT_3

Table 548. SYSCREG_MUX_NAND_MCI_SEL (address 0x1300 2898)

Bit	Symbol	R/W	Reset Value	Description
0	Mux_NAND_MCI_sel	R/W	0x0	Selects between NANDI and MCI pins. 0 - NAND; 1 - MCI The pins are NAND_RYBN0-MCI_DAT4 NAND_RYBN1-MCI_DAT5 NAND_RYBN2-MCI_DAT6 NAND_RYBN3-MCI_DAT7

Table 549. SYSCREG_MUX_UART_SPI_SEL (address 0x1300 289C)

Bit	Symbol	R/W	Reset Value	Description
0	Mux_UART_SPI_sel	R/W	0x0	Selects between SPI and UART pins. 0 - UART; 1 - SPI The pins are UART_CTS_N - SPI_CS_OUT1 UART_RTS_N -SPI_CS_OUT2

Table 550. SYSCREG_MUX_I2STX_IPINT_SEL (address 0x1300 28A0)

Bit	Symbol	R/W	Reset Value	Description
0	Mux_I2STX_0_PCM_sel	R/W	0x0	Selects between I2STX_0 and IPINT_1 pins. 0 - I2STX_0; 1 - PCM The pins are I2STX_CLK0 - PCM_DB I2STX_DATA0 - PCM_DA I2STX_WS0 - PCM_DCK I2STX_BCK0 -PCM_FSC.

4.9 Pad configuration registers

Table 551. SYSCREG_padname_PCTRL (addresses 0x1300 28A4 to 0x1300 2A28)

Bit	Symbol	R/W	Reset Value	Description
1	SYSCREG_<padname>_PCTRL_<padname>_P2	R/W	0x0/0x1	The logic pin p2 of the pad. The reset value depends on the pad.The reset value is 0x1 for all pads except for I2C_SDA1 and I2C_SCL1.
0	SYSCREG_<padname>_PCTRL_<padname>_P1	R/W	0x0/0x1	The logic pin p1 of the pad. The reset value depends on the pad (see table below). The reset value is 0x0 for all pads except from GPIO0 and GPIO1.

See also Table 26–552 for explanation of the P1 and P2.

Table 552. Logic Behaviour of Input Cell

Input				Output		Mode
IO[1]	EN	P1	P2	IO	Z1	
Bond pad input state	EN = disable output driver			Bond pad output	Input to chip core logic	
L[2]	H	-[4]	-	L	L	Receiving
H	H	-	-	H	H	Receiving
Z	H	L	L	h[3]	H	Pull-up
Z	H	L	H	Z	[5]	Plain input
Z	H	H	L	rpt	RPT	Repeater
Z	H	H	H	I	L	Weak pull-down

[1] Externally driven.

[2] Capital letters indicate strong signal

[3] Lower case letters indicate a weak signal

[4] Dash (-) Indicates any or do not care.

[5] ZI is driven to the same logic state as IO.

Table 553. SYSCREG_ESHCTRL_SUP4 (address 0x1300 2A2C)

Bit	Symbol	R/W	Reset	Description
0	SYSCREG_ESHCTRL_SUP4	R/W	0x1	This bit controls the performance of all pads which belong to the supply domain SUP4 (Nandflash and EBI pads). SUP4 has a typical supply voltage of 1.8V or 2.8V. When this bit is: 0x0 : high speed-performance 0x1: less switching noise. (To obtain the same speed-performance at the supply voltage of 1.8V as 2.8V, this bit has to be set at 0x0.)

Table 554. SYSCREG_ESHCTRL_SUP8 (address 0x1300 2A30)

Bit	Symbol	R/W	Reset	Description
0	SYSCREG_ESHCTRL_SUP8	R/W	0x1	This bit controls the performance of all pads which belong to the supply domain SUP8 (LCD interface/SDRAM pads). SUP8 has a typical supply voltage of 1.8V or 2.8V. When this bit is: 0x0 : high speed-performance 0x1: less switching noise (To obtain the same speed-performance at the supply voltage of 1.8V as 2.8V, this bit has to be set at 0x0.)

UM10314

Chapter 27: LPC3130/31 Pulse Code Modulation (PCM)/ISDN Oriented Modular (IOM) interface

Rev. 1 — 4 March 2009 User manual

1. Introduction

PCM (Pulse Code Modulation) is a very common method used for transmitting analog data in digital format. Most common applications of PCM are Digital audio as in Audio CD and computers, digital telephony and digital videos.

The IOM (ISDN Oriented Modular) interface is primarily used to interconnect telecommunications ICs providing ISDN compatibility. It delivers a symmetrical full-duplex communication link containing user data, control/programming, and status channels.

The combined PCM/IOM interface is also called IPINT in the following text.

1.1 Features

- Four wire serial interface.
- Can function in both Master and Slave modes.
- Supports:
  - PCM: Pulse code modulation. Single clocking physical format.
  - MP PCM: Multi-Protocol PCM. Configurable direction per slot.
  - IOM-2: Extended ISDN-Oriented modular. Double-clocking physical format.
- Twelve eight bit slots in a frame with enabling control per slot.
- Internal frame clock generation in master mode.
- Receive (RX) and Transmit (TX) DMA handshaking using a request/clear protocol.
- Interrupt generation per frame.

2. General description

2.1 Interface description

2.1.1 Clock signals

Table 555. Clock signals of the PCM

Clock name	I/O	Source/ destination	Description
PCM_PCLK	I	CGU	PCM Clock. Used to synchronize the DMA handshake signals, needs to be continuously running.
PCM_APB_PCLK	I	CGU	APB Interface clock. Used to perform register accesses and is gated with psel.
PCM_CLK_IP	I	CGU	Clock for Timing.
PCM_FCS_IN	I	on pin mI2STX_BCK0/ PCM_FSC	8 kHz frame sync signal in slave mode

Table 555. Clock signals of the PCM

Clock name	I/O	Source/ destination	Description
PCM_FCS_OUT	O	on pin mI2STX_BCK0/ PCM_FSC	8 kHz frame sync signal in master mode
PCM_DCLK_IN	I	on pin mI2STX_WS0/ PCM_DCLK	Data clock input in slave mode
PCM_DCLK_OUT	O	on pin mI2STX_WS0/ PCM_DCLK	Data clock output in master mode
CLK_IP_ENABLE	O	CGU	Enable for PCM_PCLK. This signal is connected internally to CGU module to tell CGU when to gate & not gate PCM_CLK_IP clock.

2.1.2 Pin connections

Table 556. Pins of the PCM interface

Name	Type	Description
mI2STX_DATA0\ PCM_DA[2]	I/O	Serial data input/output
mI2STX_CLK0\ PCM_DB[2]	I/O	Serial data input/output
mI2STX_WS0\ PCM_DCLK[2]	I/O	Data clock input when in slave mode or output when in master mode
mI2STX_BCK0\ PCM_FSC[2]	I/O	8 KHz clock synchronization input when in slave mode or output when in master mode.

[1] Directions of PCM_DA and PCM_DB are configured using TYP_DO_IP in CNTL0 register (see Table 27–560). PCM_DCLK and PCM_FSC are configured using TYP_OD in CNTL0 register (see Table 27–560).

[2] The PCM pins are multiplexed with the I2S-bus pins.

2.1.3 Interrupt requests

The IPINT provides an interrupt through event router module to the ARM processor to allow voice processing. It is generated when all new data from the PCM/IOM ports are available in the associated registers.

2.1.4 Reset signals

The CGU provides a synchronous reset for the PCM_PCLK clock domain (PNRES) and an asynchronous reset (RESET_ASYNC) and synchronous reset (RESET) for the PCM_CLK_IP clock domain.

Asynchronous reset signals go only to asynchronous reset on flip/flops. There are no internally generated asynchronous reset signals.

2.1.5 DMA transfer signals

Table 557. DMA signals of the IPINT

Name	Type	Description
DMAREQ_TX	O	Request to DMA for transmit data
DMAREQ_RX	O	Request to DMA for receive data

3. Register overview

Table 558. Register overview: PCM/IOM (register base address 0x1500 0000)

Name	R/W	Address offset	Description
GLOBAL	R/W	0x000	Global register
CNTL0	R/W	0x004	Control register 0
CNTL1	R/W	0x008	Control register 1
HPOUT[0]	W	0x00C	Transmit data register 0
HPOUT[1]	W	0x010	Transmit data register 1
HPOUT[2]	W	0x014	Transmit data register 2
HPOUT[3]	W	0x018	Transmit data register 3
HPOUT[4]	W	0x01C	Transmit data register 4
HPOUT[5]	W	0x020	Transmit data register 5
HPIN[0]	R	0x024	Receive data register 0
HPIN[1]	R	0x028	Receive data register 1
HPIN[2]	R	0x02C	Receive data register 2
HPIN[3]	R	0x030	Receive data register 3
HPIN[4]	R	0x034	Receive data register 4
HPIN[5]	R	0x038	Receive data register 5
CNTL2	R/W	0x03C	Control register 2

4. Register description

Table 559. GLOBAL register (address 0x1500 0000)

Bit	Symbol	R/W	Reset Value	Description
31-5	-	-	-	Reserved
4	DMARXENABLE	R/W	0	When true DMA rx enabled
3	DMATXENABLE	R/W	0	When true DMA tx enabled
2	NORMAL	R/W	0	0: Slave single 16bits slot mode[1] 1: Normal mode
1	-	-	-	Reserved
0	ON_OFF	R/W	0	When true IPINT is active

[1] In Slave0 single slot mode there is only one 16-bit PCM slot, and does only support frame sync LF mode as set in CNTL0 TYP_FRMSYNC. Data is available in hpin[3] (INSLOT[7:6]) and data is sent from hpout[0] (OUTSLOT[1:0]). There needs to be at least 16 PCM_DCLK clock cycles in one PCM frame and there should be no more then 31 clock cycles. The pcm_int occurs at the end of the 16 bits, this is after the last bit has been received.

Table 560. CNTL0 register (address 0x1500 0004)

Bit	Symbol	R/W	Reset Value	Description
31-15	-	-	-	Reserved
14	MASTER	R/W	0	PCM/IOM master mode. 0: PCM is slaved to the PCM/IOM port timing. PCM_DCLK and PCM_FCS are inputs. 1: PCM is master on the PCM/IOM port. PCM_DCLK and PCM_FCS are outputs.
13-12	-	-	-	Reserved
11	LOOPBACK	R/W	0	Internal loop-back mode 0: no loop-back selected. 1: data out is internally connected to data in. Data is not re-clocked.
10	TYP_OD	R/W	0	Type of PCM_FCS an PCM_DCLK output port. 0: PCM_FCS and PCM_DCLK use open-drain. 1: PCM_FCS and PCM_DCLK use push-pull.
9-8	TYP_DO_IP	R/W	0	Type of PCM/IOM data output ports. 00: PCM/IOM port outputs set to tri-state 01: Outputs use open-drain and are set to tri-state outside transmission. 10: Outputs use push-pull and are set to tri-state outside transmission 11: Outputs use push-pull and are always driven
7-6	TYP_FRMSYNC	R/W	0	Shape of frame synchronization signal. 00: Short frame-sync FR enclosing the first rising clock edge 01: Short frame-sync FF enclosing the first falling clock edge 10: Short frame-sync LF enclosing the last falling clock edge 11: Long frame-sync over the first slot (8 bit)
5-3	CLK_SPD	R/W	0	Port frequency selection PCM: 1 clock cycle per bit 000: 512kHz, 64 bits clocked per frame of 125 ms 001: 768kHz, 96 bits clocked per frame 010: 1.536 MHz, 192 bits clocked per frame 011: 2.048 MHz, 256 bits clocked per frame IOM: 2 clock cycles per bit 100: 512kHz, 32 bits clocked per frame 101: 768kHz, 48 bits clocked per frame 110: 1.536 MHz, 96 bits clocked per frame 111: 4.096 MHz, 256 bits clocked per frame
2-0	-	-	-	Reserved

Table 561. CNTL1 register (address 0x1500 0008)

Bit	Symbol	R/W	Reset Value	Description
31-12	-	-	-	Reserved
11-0	ENSLT	R/W	0	Enable PCM/IOM Slots, one control bit for each slot.

Table 562. HPOUT[5:0] registers (addresses 0x1500 000C to 0x1500 0020)

Bit	Symbol	R/W	Reset Value	Description
31-16	-	-	-	Reserved
15-0	HPOUT[5:0]	W	0	For each two slots a 16 bits transmit data register exists. { slot[i * 2 + 1], slot[i * 2] } = hpout[i]

Table 563. HPIN[5:0] registers (addresses 0x1500 0024 to 0x1500 0038)

Bit	Symbol	R/W	Reset Value	Description
31-16	-	-	-	Reserved
15-0	HPIN[5:0]	R	0	For each two slots a 16 bits receive data register exists. hpin[i] = { slot[i * 2 + 1], slot[i * 2] }

The PCM/IOM bus data bits are numbered from 0 (first, i.e. earliest bit) to N.

Table 564. CNTL2 register (address 0x1500 003C)

Bit	Symbol	R/W	Reset Value	Description
31-12	-	-	-	Reserved
11-0	SLOTDIRINV	R/W	0	PCM A/B port configuration One control bit for each slot. 0: A is an output, B is an input 1: A is an input, B is an output

5. Functional description

5.1 Architecture

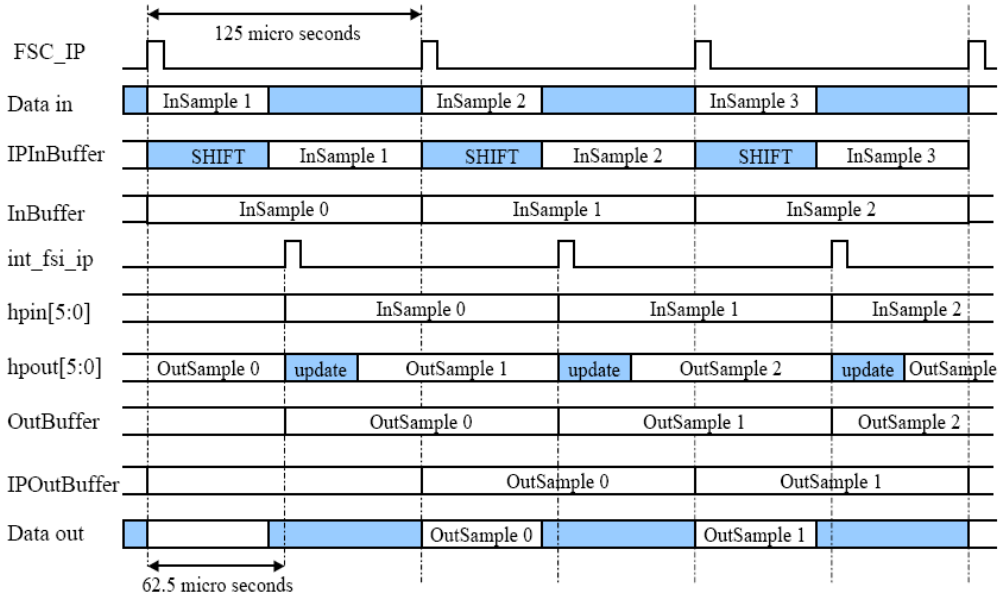


Fig 89. PCM architecture

The PCM provides an interrupt, pcm_int, to the ARM processor to allow voice processing. It is generated when all new data from the PCM/IOM ports are available in the associated registers.

Alternatively the RX and/or TX DMA request/clear handshake signals can be enabled. The request is then generated after the rising edge of pcm_int and is synchronized to pclk; it will remain high until a clear signal is given.

In the receive direction, the first 96-bits of each IP frame are clocked in by the IPINT and shifted into the IPInbuffer. There they are held until they are copied into the INBuf at the start of the next IP frame. From there, data is forwarded to the HPIN register. An interrupt, pcm_int, to the Host Processor is generated halfway the frame, thus ensuring stable data. At this time the processor may read and process PCM/IOM data from the HPIN register.

In the transmit direction the pcm_int interrupt to the Host Processor is generated allowing it to manually write data into the HPOUT register. With the next pcm_int pulse, the data in the HPOUT register is latched into the OutBuffer. From there, they are transferred to the IPOutBuffer shift register with the start of the next IP frame and clocked serially out of the IP port.

All transfers are synchronized to clk_ip, hence no violation is possible and data integrity is guaranteed. The IPINT control register configures the PCM/IOM port, including operation, master/slave mode, shape of synchronization signal, port frequency selection and PCM/IOM data length.

5.2 Operation

The PCM_DA and PCM_DB are bi-directional ports of PCM/IOM INTERFACEIPINT, which are connected to two 1 bit inout lines A and B. The A and B I/O lines may be jointly programmed as:

- 1.A input and B output
- 2.A output and B input

SLOTDIRINV bit in register CNTL2 controls this direction. The SLOTDIRINV configuration allows Multi Protocol-PCM transmission.

The PCM_FCS is an 8kHz-framing signal for synchronizing data transmission on DA and DB, i.e. frame period will always be 125 microseconds. The rising edge of PCM_FCS gives the time reference for the first bit transmitted in the first slot of a frame. The IPINT can work either in master mode (bus timing is provided by IPINT) or in slave mode (IPINT is synchronized to the timing received from the bus).

The number of slots per frame depends on the selected data rate, however each slot will always contain 8 data bits. The IPINT handles the first 96 bits of PCM/IOM frames. Processing 96 bits in a PCM/IOM frame allows accessing maximum of 12 channels (slots) to 8 bit devices. For longer frames all bits after the 96th are ignored.

Data is transmitted in 16-bit word order:

Data (A or B) send = {hpout[5], hpout[4], hpout[3], hpout[2], hpout[1], hpout[0]}

Where hpout[i] = {slot[i * 2 + 1], slot[i * 2]} are registers accessible via the APB bus.

Transmitted data in order (left to right, top to bottom) is.

{slot1, slot0}: hpout[0][15], hpout[0][14], ..., hpout[0][1], hpout[0][0].

{slot3, slot2}: hpout[1][15], hpout[1][14], ..., hpout[1][1], hpout[1][0]

...

{slot9, slot8}: hpout[4][15], hpout[4][14], ..., hpout[4][1], hpout[4][0]

{slot11, slot10}: hpout[5][15], hpout[5][14], ..., hpout[5][1], hpout[5][0]

Received data is stored into six 16-bit hpin registers in the same order as their hpout counterparts, i.e. hpout[i] => transmit => hpin[i]…

However, if the frame size is smaller than 12 slots then the lower hpout slots are not transmitted, i.e. hpout[i + (12 - frameslots) / 2] => transmit => hpin[i]. As an example, if a 64-bit frame is configured then transmission starts from hpout[2] up to hpout[5] and is received starting from hpin[0] up to hpin[3].

PCM_DCLK is a data clock. Its frequency is twice the selected data rate in IOM mode. In PCM mode, the PCM_DCLK frequency is equal to the data rate. Refer to CLK_SPD bits in CNTL0 register.

The IPINT can work at several bit rates; these are specified in the table below. The bit rate is selected by writing appropriate 3-bit code into the Control Register 0.

Table 565. Bit Rates and Modes

IOM (KBIT/SEC)	PCM (Kbit/sec)	MODE
	128 to 248	Slave only
256	512	Master and slave
384	768	Master and slave
768	1536	Master and slave
2048	2048	Master and slave

The availability data is signalled to the processor by a single interrupt, pcm_int, for every PCM/IOM frame. And by two sets of DMA handshaking signals, one for incoming data (RX) and one for outgoing data (TX), if enabled. DMA handshaking involves following steps.

- If data is available and/or required a dmareq_rx or dmareq_tx is set high
- When a data access is performed to the corresponding register set, i.e. hpout for TX and hpin for RX, it becomes possible to clear the DMA request
- dmaclr_rx/tx becomes high the request will be removed.

The IPINT supports the physical data formats for PCM (one clock cycle per bit) or IOM (two clock cycles per bit). Any potentially required data structuring (e.g. monitor channel for IOM) has to be performed by the processor. The connection between the IP port and the different blocks and channels needs to be handled by the Processor by copying data between the IPINT registers and the different registers associated with each block and channel.

5.3 PAD Configuration

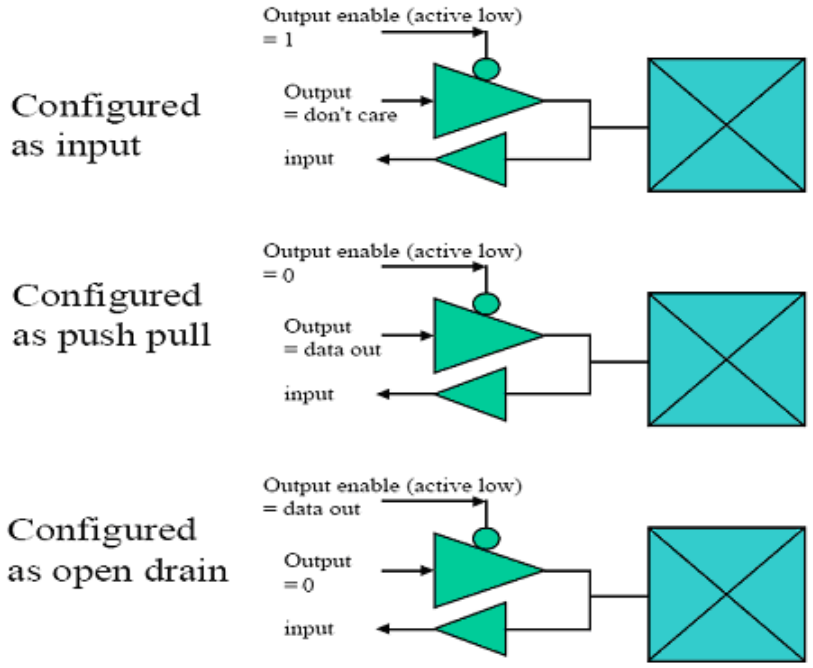


Fig 90. IPINT Pad Configuration

Pad configuration is controlled by CNTL0 register. Bits TYP_DO_IP (9:8) in CNTL0 control the type of data output port. Pads can be configured in tristate, push pull or open drain.

5.4 Bi-directional data line configuration

The IPINT allows configuration of data direction for both data lines on a slot-by-slot basis where one of the lines must be set as input and the other one as output.

- Time slots are activated by setting relevant bits of ENSLT in register cntl1. For inactive slots (ENSLT[i] = 0), both data lines will be set to HighZ. When switching the direction of the data lines at least one inactive slot must be taken into account as guard space.
- The direction of the data lines for each of the twelve slots is configured by bits of SLOTDIRINV in register cntl2.
- Bi-directional data line configuration is possible only in short frame sync.
  - cntl0 TYP_FRMSYNC = 10: LF, first bit transmitted on the cycle after Frame Sync (PCM_FCS) is active.
- The following must be taken into account when using the IPINT in bi-directional mode.
  - The SLOTDIRINV[i], ENSL[i] applies from the first cycle to the last cycle of a given slot i. The first slot starts, i.e. the first bit is transmitted, on the cycle occurring after Frame Sync active cycle. The data line (DA or DB) that is configured as Output is tri-stated during guard slots and during the idle frame period (see Figure 27–91).

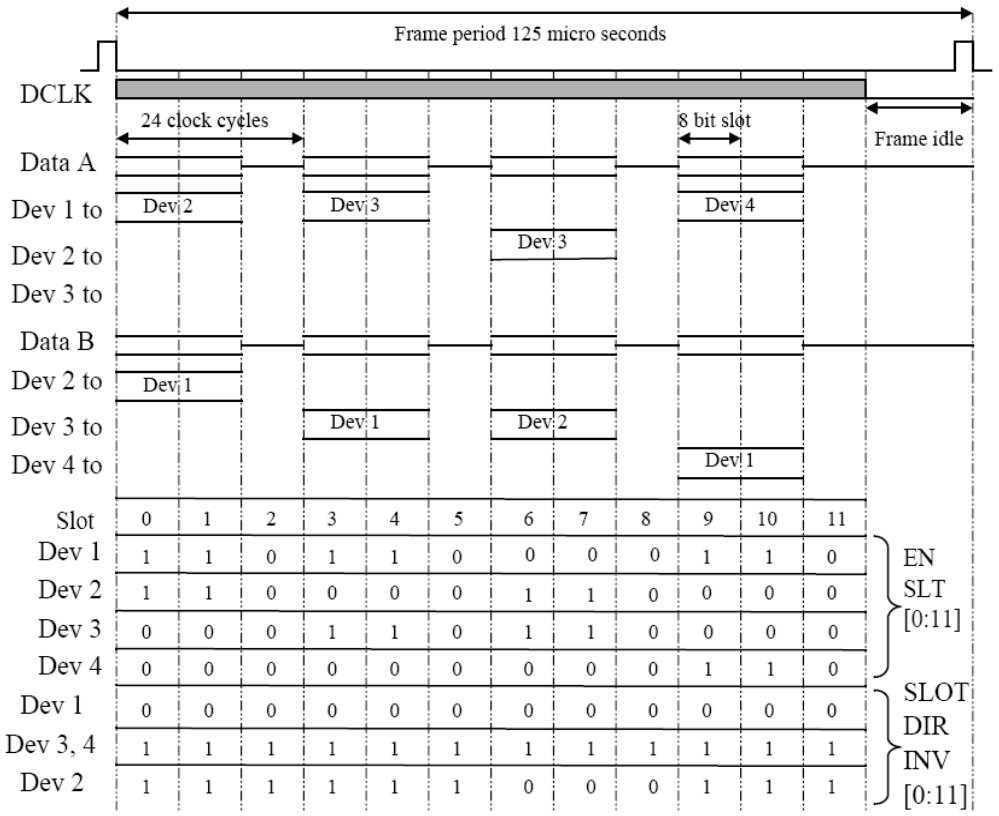
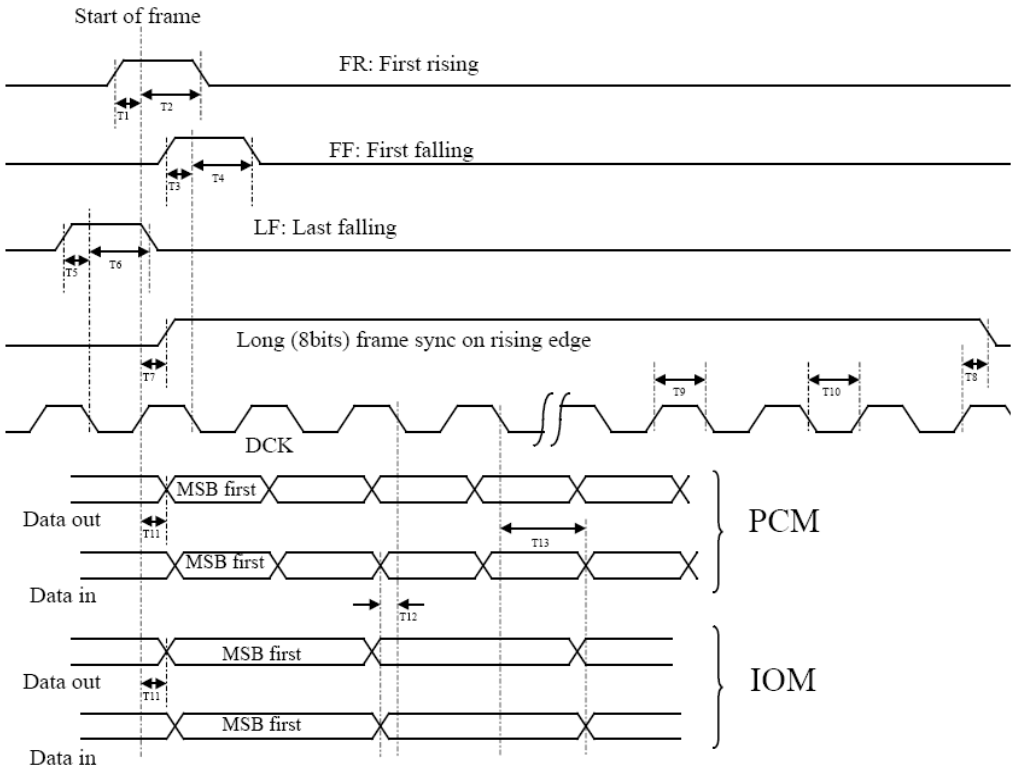


Fig 91. Frame Sync Active Cycle

Figure 27–91 shows an example of communication between 4 devices using bi-directional PCM data lines. Each device has its DA pin connected to a common A line and DB pin connected to a common B line. Data is transferred in words of 16-bit (2 slots per word). Between each data word one guard slot is used to switch the direction of the data lines. This slot scheduling is not exclusive; it depends on the number of devices that are connected to the data lines A and B and which pair of devices is communicating at each slot. Therefore, for a given device the DA and DB lines are able to switch from input to output and output to input depending with which devices it has to communicate during a certain slot. The user must ensure that only one device is transmitting on a PCM data line at the time.

5.5 Timing



Detailed external timing diagram for IOM/PCM

Table 566. PCM Port Timing

Symbol	Description	MIN (NS)	MAX (NS)
T1	Frame sync (short FR) to clock rising setup time	80	N.A
T2	Frame sync (short FR) to clock rising hold time	50	N.A
T3	Frame sync (short FF) to clock falling setup time	80	N.A
T4	Frame sync (short FF) to clock falling hold time	50	N.A
T5	Frame sync (short LF) to clock falling setup time	80	N.A
T6	Frame sync (short LF) to clock falling hold time	50	N.A
T7	Frame sync (long) rising edge delay	N.A	20
T8	Frame sync (long) falling edge hold time	N.A	20
T9	PCM Clock high time	80	N.A
T10	PCM Clock low time	80	N.A
T11	PCM Data out valid delay	N.A	60
T12	PCM Data in setup time	20	N.A
T13	PCM Data in hold time	50	N.A

Table 567. IOM Port Timing

Symbol	Description	MIN (NS)	MAX (NS)
T1	Frame sync (short FR) to clock rising setup time	70	N.A
T2	Frame sync (short FR) to clock rising hold time	40	N.A
T3	Frame sync (short FF) to clock falling setup time	70	N.A
T4	Frame sync (short FF) to clock falling hold time	40	N.A
T5	Frame sync (short LF) to clock falling setup time	70	N.A
T6	Frame sync (short LF) to clock falling hold time	40	N.A
T7	Frame sync (long) rising edge delay	N.A	20
T8	Frame sync (long) falling edge hold time	N.A	20
T9	IOM Clock high time	80	N.A
T10	IOM Clock low time	80	N.A
T11	IOM Data out valid delay	N.A	30
T12	IOM Data in setup time	30	N.A
T13	IOM Data in hold time	45	N.A

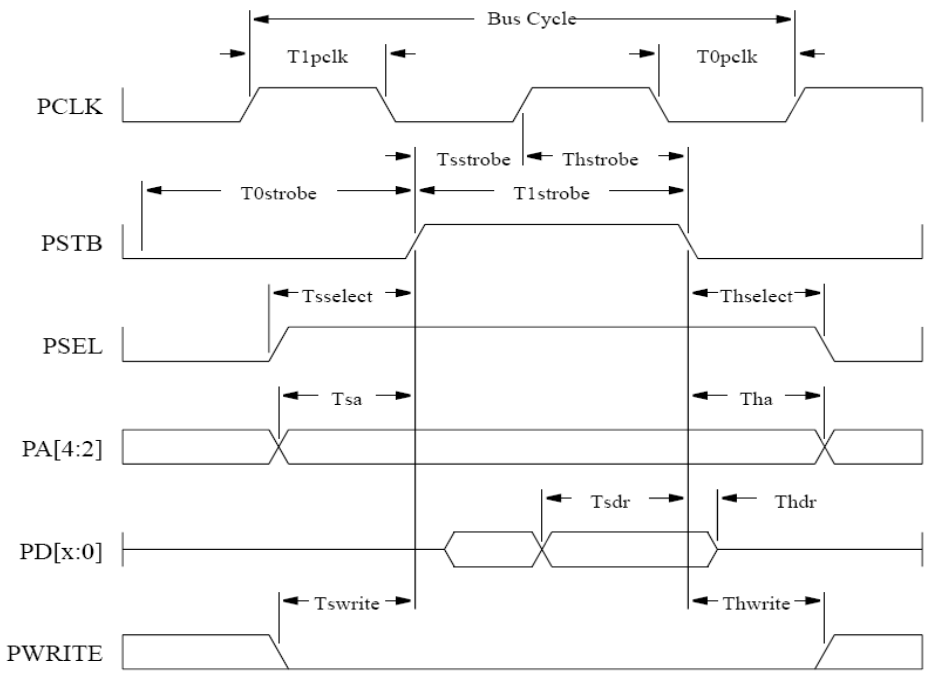


Fig 92. Read from the IPINT block

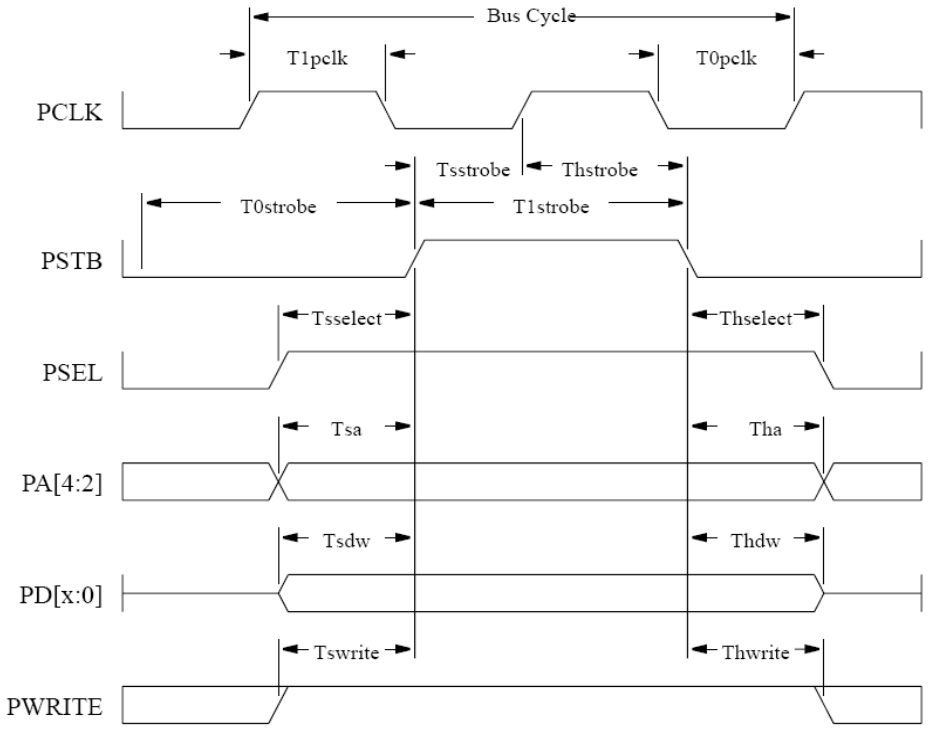


Fig 93. Write to the IPINT block

Table 568. Timing Parameters

Parameter Name	Description
T1pclk	Minimum required high time for PCLK.
T0pclk	Minimum required low time for PCLK.
T1strobe	Minimum required high time for PSTB.
T0strobe	Minimum required high time for PSTB.
Tsstrobe	Minimum required setup time for PSTB.
Thstrobe	Minimum required hold time for PSTB.
Tsselect	Minimum required setup time for PSEL.
Thselect	Minimum required hold time for PSEL.
Tsa	Minimum required setup time for PA[4:2].
Tha	Minimum required hold time for PA[4:2].
Tsdr	Minimum provided setup time for a PD[31:0] read.
Thdr	Minimum provided hold time for a PD[31:0] read.
Tsdw	Minimum required setup time for a PD[31:0] write.
Thdw	Minimum required hold time for a PD[31:0] write.
Tswrite	Minimum required setup time for PWRITE.
Thwrite	Minimum required hold time for PWRITE.

5.6 Timing controller

The IPINT may operate as timing master or slave on the PCM/IOM bus. When being configure as PCM/IOM slave, it is synchronized to the PCM_FCS signal received from the PCM/IOM bus. In the PCM/IOM master configuration, the IPINT generates the PCM_FCS signal.

The signal pcm_int is used to synchronize other blocks on the chip to the PCM/IOM timing. This signal is obtained by counting half the frame period to ensure data stability during data transfers towards/from the host processor and/or DMA controller.

Frame bit timing is governed by the internally generated signal dclk. In master mode the IPINT will generate dclk by dividing clk_ip towards the desired frequency using a fractional divider. CLK_IP must be equal to 24MHz. In slave mode DCK_IP from padcell will be used as dclk.

Table 569. IPINT clock division schemes

DCLK FREQUENCY	CLK_IP (24MHZ) DIVISION FACTOR	X	Y
512 kHz	375/8	375	16
768 kHz	125/4	125	8
1.536 MHz	125/8	125	16
2.048 MHz	375/32	375	64
4.096 MHz	375/64	375	128

6. Power optimization

The PCM_CLK_IP and PCM_APB_PCLK can be clock gated. For this, matching CGU registers have to be programmed.

UM10314

Chapter 28: LPC3130/31 I2S interface

Rev. 1 — 4 March 2009 **User manual**

1. Introduction

The I2S bus provides a standard communication interface for digital audio applications. The LPC3130/31 includes two I2S interfaces; I2S0 and I2S1.

The I2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select signal. The basic I2S connection has one master, which is always the master, and one slave. The I2S interface on the LPC3130/31 provides a separate transmit and receive channel.

1.1 Features

The I2S0/1 interfaces have the following features:

- Audio interfaces compatible with the I2S standard.
- The I^2S0/1 receive interfaces support master and slave mode.
- The I^2S0/1 transmit interfaces support master mode.
- Support LSB justified words of 16, 18, 20, and 24 bits.
- Support a configurable number of bit clock periods per Word Select period (up to 128 bit clock periods).
- Support DMA transfers.
- Transmit FIFO or receive FIFO of 4 stereo samples.
- Support single 16 bit transfers to/from the left or right FIFO.
- Support single 24 bit transfers to/from the left or right FIFO.
- Support 32-bit interleaved transfers, with the lower 16 bits representing the left audio sample, and the higher 16 bits representing the right audio sample.
- Support two 16-bit samples audio samples combined in a 32-bit word (2 left or 2 right samples) to reduce bus load.
- Provide maskable interrupts for audio status. (FIFO underrun / overrun / full / half_full / not empty for left and right channel separately).

2. General description

The I2S transfers serial data out via the transmit channel and serial data in via the receive channel. These blocks have a four word FIFO and support both DMA flow control and interrupts.

The I2S interfaces support the following formats: I2S, LSC16, LSB18, LSB20 and LSB24. The I2SRX interfaces generate a NEWSAM flag indicating that a new audio sample is valid. For the NEWSAM flag, a dedicated WS signal is provided which is connected to an edge detector that generates this NEWSAM. This NEWSAM is used for both I2STX outputs, which means that this system can only operate on a single sampling frequency.

2.1 Interface description

2.1.1 Clock signals

Table 570. Clock Signals of the I2S block

Clock Name	I/O	Source/Destination	Description
I2C_CFG_PCLK	I	CGU	APB related clock. All registers in the I2C_config block run on this clock.
EDGE_DET_PCLK	I	CGU	APB related clock. All registers in the edge_det block run on this clock.
I2STX_FIFO_0_PCLK	I	CGU	APB related clock. All registers in the I2STX_FIFO_0 block run on this clock.
I2STX_FIFO_1_PCLK	I	CGU	APB related clock. All registers in the I2STX_FIFO_1 block run on this clock.
I2SRX_FIFO_0_PCLK	I	CGU	APB related clock. All registers in the I2SRX_FIFO_0 block run on this clock.
I2SRX_FIFO_1_PCLK	I	CGU	APB related clock. All registers in the I2SRX_FIFO_1 block run on this clock.
I2STX_IF_0_PCLK	I	CGU	APB related clock. sysclk of the I2STX_IF_0 runs on this APB synchronous clock.
I2STX_IF_1_PCLK	I	CGU	APB related clock. sysclk of the I2STX_IF_1 runs on this APB synchronous clock.
I2SRX_IF_0_PCLK	I	CGU	APB related clock. sysclk of the I2SRX_IF_0 runs on this APB synchronous clock.
I2SRX_IF_1_PCLK	I	CGU	APB related clock. sysclk of the I2SRX_IF_1 runs on this APB synchronous clock.
I2S_EDGE_DETECT_CLK	I	CGU	Sampling frequency clock. Used to generate NEWSAM flag from edge_detection.
I2STX_BCK0_N	I	CGU	I2S Bit Clock. Indicates the number of bits per audio sample. Typically 48 x fs or 64 x fs.
I2STX_WS0	I	CGU	I2S Word Select. Same frequency as I2S_EDGE_DETECT_CLK. 1 stereo sample per WS cycle.
I2STX_CLK0	I	CGU	256fs system clock for external reference.
I2STX_BCK1_N	I	CGU	I2S Bit clock. Indicates the number of bits per audio sample. Typically 48 x fs or 64 x fs.
I2STX_WS1	I	CGU	I2S Word Select. Same frequency as I2STX_WS. One stereo sample per WS cycle.
CLK_256FS	I	CGU	256 fs system clock for external reference
I2SRX_BCK0	I	Pin	I2S Bit Clock. Input to CGU PLL.
I2SRX_WS0	I	Pin	I2S Word Select. Same frequency as I2S_EDGE_DETECT_CLK. Input to CGU PLL.
I2SRX_BCK1	I	Pin	I2S Bit Clock. PLL may lock to this clock.
I2SRX_WS1	I	Pin	I2S Word Select. Same frequency as I2S_EDGE_DETECT_CLK. Input to CGU PLL.

2.1.2 Pin connections

Table 571. I2S-bus pins

Name	I/O	Description
I2SRX_DATA0	I	I2S Data
I2SRX_BCK0	I	I2S Bitclock
I2SRX_WS0	I	I2S Word Select
I2STX_DATA0	O	I2S Data
I2STX_BCK0	O	I2S Bitclock
I2STX_WS0	O	I2S Word Select
I2STX_CLK0	O	256fs system clock for external reference
I2SRX_DATA1	I	I2S Data
I2SRX_BCK1	O	I2S Bitclock
I2SRX_WS1	O	I2S Word Select
I2STX_DATA1	O	I2S Data
I2STX_BCK1	O	I2S Bitclock
I2STX_WS1	O	I2S Word Select
CLK_256fs	O	256 fs system clock for external reference. Used as system clock for external reference of I2STX_1.

2.1.3 Interrupt requests

Relevant information about interrupt requests are generated by this module, except for register descriptions. The module describes the interrupt signals, interrupt reasons and how/when they are generated etc.

Table 572. Interrupt Request signals of the I^2S interface

Name	Type	Description
I2STX0_IRQ	O	Configurable I2STX0 interrupt request
I2STX1_IRQ	O	Configurable I2STX1 interrupt request
I2SRX0_IRQ	O	Configurable I2SRX0 interrupt request
I2SRX1_IRQ	O	Configurable I2SRX1 interrupt request

Table 573. Interrupt request sources for I2SRX

Name	Description
0	FIFO RIGHT underrun
1	FIFO LEFT underrun
2	FIFO RIGHT overrun
3	FIFO LEFT overrun
4	FIFO LEFT full
5	FIFO LEFT half_full
6	FIFO LEFT NOT empty
7	FIFO RIGHT full
8	FIFO RIGHT half_full
9	FIFO RIGHT NOT empty

Table 574. Interrupt request sources for I2STX

Number	Description
0	FIFO right underrun
1	FIFO left underrun
2	FIFO right overrun
3	FIFO left overrun
4	FIFO left full
5	FIFO left half_empty
6	FIFO left empty
7	FIFO right full
8	FIFO right half_empty
9	FIFO right empty

2.1.4 Reset signals

Table 575. Reset signals of the I2S sub modules

Name	Type	Description
PNRES_I2S_CFG_CONFIG	I	I2C_config APB domain reset (active low)
PRESETN	I	I2SRX_FIFO/I2STX_FIFO APB domain reset (active low)
RESET	I	I2SRX_IF/I2STX_IF reset (active high)
EDGE_DET_RST_N	I	Edge detection reset (active low)

2.1.5 DMA transfer signals

For each of the I2STX_FIFO and I2SRX_FIFO blocks in the I2S interface the following DMA signals are supplied:

Table 576. I2S DMA signals

Name	Type	Description
DMA_REQ_LEFT	O	Left channel FIFO DMA request
DMA_REQ_RIGHT	O	Right channel FIFO DMA request
DMA_CLR_LEFT	I	Left channel DMA acknowledge
DMA_CLR_RIGHT	I	Right channel DMA acknowledge

Use pin I2STX0_dma_req_left(slave number 6) when transferring interleaved data (INTERLEAVED_0 address is used as DMA channel destination) using DMA to I2STX0 block. Similarly use I2STX1_dma_req_left(8), I2SRX0_dma_req_left(10) and I2SRX_dma_req_left(12) for transferring interleaved data to I2STX1, I2SRX0 and I2SRX1 blocks.

3. Register overview

Table 577. Register overview: I2S configuration module (register base address 0x1600 0000)

Name	R/W	Address offset	Description
I2S_FORMAT_SETTINGS	R/W	0x00	I2S formats

Table 577. Register overview: I2S configuration module (register base address 0x1600 0000)

Name	R/W	Address offset	Description
I2S_CFG_MUX_SETTINGS	R/W	0x04	Misc controls
-	-	0x08 - 0x0C	reserved
N_SOF_COUNTER	R/W	0x10	NSOF counter control

Table 578. Register overview: I2STX0 interface (register base address: 0x1600 0080)

Name	R/W	Address offset	Description
LEFT_16BIT	R/W	0x00	16 bits left channel data
RIGHT_16BIT	R/W	0x04	16 bits right channel data
LEFT_24BIT	R/W	0x08	24 bits left channel data
RIGHT_24BIT	R/W	0x0C	24 bits right channel data
INT_STATUS	R/W	0x10	FIFO status register
INT_MASK	R/W	0x14	Interrupt Mask register
LEFT_32BIT_0 .. LEFT_32BIT_7	R/W	0x20-0x3C	2x16 bits left channel data; 16 LSB's represent the first sample
RIGHT_32BIT_0 .. RIGHT_32BIT_7	R/W	0x40-0x5C	2x16 bits right channel data; 16 LSB's represent the first sample
INTERLEAVED_0 .. INTERLEAVED_7	R/W	0x60-0x7C	Interleaved data; upper 16 bits is right

Table 579. Register overview: I2STX1 interface (register base address: 0x1600 0100)

Name	R/W	Address offset	Description
LEFT_16BIT	R/W	0x00	16 bits left channel data
RIGHT_16BIT	R/W	0x04	16 bits right channel data
LEFT_24BIT	R/W	0x08	24 bits left channel data
RIGHT_24BIT	R/W	0x0C	24 bits right channel data
INT_STATUS	R/W	0x10	FIFO status register
INT_MASK	R/W	0x14	Interrupt Mask register
LEFT_32BIT_0 .. LEFT_32BIT_7	R/W	0x20-0x3C	2x16 bits left channel data; 16 LSB's represent the first sample
RIGHT_32BIT_0 .. RIGHT_32BIT_7	R/W	0x40-0x5C	2x16 bits right channel data; 16 LSB's represent the first sample
INTERLEAVED_0 .. INTERLEAVED_7	R/W	0x60-0x7C	Interleaved data; upper 16 bits is right

Table 580. Register overview: I2SRX0 interface (register base address: 0x1600 0180)

Name	R/W	Address offset	Description
LEFT_16BIT	R/W	0x00	16 bits left channel data
RIGHT_16BIT	R/W	0x04	16 bits right channel data
LEFT_24BIT	R/W	0x08	24 bits left channel data
RIGHT_24BIT	R/W	0x0C	24 bits right channel data
INT_STATUS	R/W	0x10	FIFO status register
INT_MASK	R/W	0x14	Interrupt Mask register

Table 580. Register overview: I2SRX0 interface (register base address: 0x1600 0180)

Name	R/W	Address offset	Description
LEFT_32BIT_0 .. LEFT_32BIT_7	R/W	0x20-0x3C	2x16 bits left channel data; 16 LSB's represent the first sample
RIGHT_32BIT_0 .. RIGHT_32BIT_7	R/W	0x40-0x5C	2x16 bits right channel data; 16 LSB's represent the first sample
INTERLEAVED_0 .. INTERLEAVED_7	R/W	0x60-0x7C	Interleaved data; upper 16 bits is right

Table 581. Register overview: I2SRX1 interface (register base address: 0x1600 0200)

Name	R/W	Address offset	Description
LEFT_16BIT	R/W	0x00	16 bits left channel data
RIGHT_16BIT	R/W	0x04	16 bits right channel data
LEFT_24BIT	R/W	0x08	24 bits left channel data
RIGHT_24BIT	R/W	0x0C	24 bits right channel data
INT_STATUS	R/W	0x10	FIFO status register
INT_MASK	R/W	0x14	Interrupt Mask register
LEFT_32BIT_0 .. LEFT_32BIT_7	R/W	0x20-0x3C	2x16 bits left channel data; 16 LSB's represent the first sample
RIGHT_32BIT_0 .. RIGHT_32BIT_7	R/W	0x40-0x5C	2x16 bits right channel data; 16 LSB's represent the first sample
INTERLEAVED_0 .. INTERLEAVED_7	R/W	0x60-0x7C	Interleaved data; upper 16 bits is right

The following rules apply to the I2S registers:

For the I2STX:

Writing to the data registers puts the sample data in the corresponding FIFO's. The I2STX FIFO's are configured as four 32 bits wide words.

For the I2SRX:

Reading from the data registers gets the sample data from the corresponding FIFO's. The I2SRX FIFO's are configured as four 32 bits wide words.

4. Register descriptions

Table 582. I2S_FORMAT_SETTINGS (address 0x1600 0000)

Bit	Symbol	R/W	Reset Value	Description
2:0	I2STX0_format	R/W	0x3	I2STX0 I2S input format (see Table 28–583).
5:3	I2STX1_format	R/W	0x3	I2STX1 I2S input format (see Table 28–583).
8:6	I2SRX0_format	R/W	0x3	I2SRX0 I2S output format (see Table 28–583).
11:9	I2SRX1_format	R/W	0x3	I2SRX1 I2S output format (see Table 28–583).
31:12	reserved	R/W	0	-

Table 583. Format settings for I2S

Value	Description
0	undefined
1	undefined
2	undefined
3	I2S
4	LSB justified 16 bits
5	LSB justified 18 bits. Use this option when sample data is 24 bit wide and LPC3130/31 is interfacing with LSB justified 18 bit audio codec. Write sample data to LEFT_24BIT/RIGHT_24BIT registers.
6	LSB justified 20 bits. Use this option when sample data is 24 bit wide and LPC3130/31 is interfacing with LSB justified 20 bit audio codec. Write sample data to LEFT_24BIT/RIGHT_24BIT registers.
7	LSB justified 24 bits. This option can also be used when sample data is 18 bit and LPC3130/31 is interfacing with LSB justified 18 bit audio codec. Similarly when sample data is 20 bit and interfacing with 20 bit audio codec.

Table 584. I2C_CFG_MUX_SETTINGS (address 0x1600 0004)

Bit	Symbol	R/W	Reset Value	Description
0	-	-	0	reserved
1	I2SRX0_oe_n	R/W	1	selects mode for I2SRX0 0: slave 1: master
2	I2SRX1_oe_n	R/W	0	selects mode for I2SRX1 0: slave 1: master
31:3	reserved	R/W	0	-

5. Functional Description

- **I2SRX0/1 interface**: This part is an I2S decoder with a serial audio input interface to the APB. It generates a parallel data signal for the left and right channels, and a NEWSAM signal. The NEWSAM signal is a strobe/latch signal that indicates that a new audio sample is available on the data lines, by going high for one clock cycle on the same clock edge, as the new data becomes available. The input format can be I2S or LSB justified.
- **I2STX0/1 interface**: This part will generate an I2S output data stream and depending on the setting, it will be an I2S or LSB justified output format. The I2STX includes a serial audio output interface to the APB.

- **Edge Detector**: This module creates a NEWSAM signal (which is a latch enable signal) from the I2S_EDGE_DETECT_CLK. This word select is generated by the CGU. This I2S_EDGE_DETECT_CLK is not connected to the pads of the I2STX interfaces.
- **Configuration register block**: This block offers a APB slave interface to various configuration registers.

6. Power optimization

In order to save power, disable clocks when blocks are not used.

7. Programming guide

Setup and configure the AudioPLL (PLL0) and fractional dividers registers assuming an external 12MHz crystal connected to PLL0:

1. Choose the Sample Frequency Fs; for example Fs = 44.1KHz.
2. In Table 28–585, we can see that for Fs = 44.1KHz, the Fractional Divider 17 should be loaded with the value 256 (the column header established the value to be loaded into FracDiv17), while the Output Frequency of the PLL0 should be Fout = 11.2896 MHz (column Fout of the same row).
   a. In order to configure the PLL0 for Fout = 11.2896 MHz, the PLL0 parameters are: Ndec = 131, Mdec = 29784, Pdec = 7, SELR = 0, SELI = 8, and SELP = 31.
   b. The FracDiv17 register provides the frequency for the WS (Word Select) signal; that is the Fs. Then, for the BCLK (bit clock signal), if we choose 32 bit clocks per channel, multiplied by two channels (left and right), we will have 64 times the Fs. That means the FractDiv18 (for I2S_TX0 and I2S_RX0) will be FactDiv17/64. In our example, it would be 256/64 = 4. (The same applies for FracDiv20 which provides the bit clock for I2S_TX1 and I2S_RX1).
3. Enable I2S-related clocks in the CGU (Table 28–570).
4. Configure the interrupt (unmask desired interrupts) according Table 28–573 and Table 28–574, using INT_MASK register.
5. Configure I2S format in the I2S_FORMAT_SETTINGS register according Table 28–582 and Table 28–583. Considerations:
   – For LSB 16 bits format, set the LSB Justified 16 bits (value 4) format in the I2S_FORMAT_SETTINGS register and use LEFT_16BIT and RIGHT_16BIT registers for data.
   – For LSB 18, 20 or 24 bits formats, set the LSB Justified 24 bits (value 7) format in the I2S_FORMAT_SETTINGS register and use LEFT_24BIT and RIGHT_24BIT registers for data.
6. If I2S_RX is used, configure the module as Master or Slave using the I2S_CFG_MUX_SETTINGS register (Table 28–584).
7. If NSOF Counter is used, program the edge detection mode using the I2S_CFG_MUX_SETTINGS register (Table 28–584).
8. To transmit Left Channel data using I2S_TX, write to any of the following registers: LEFT_16BIT, LEFT_24BIT, LEFT_32BIT_0 .. LEFT_32BIT_7. INTERLEAVED_0 .. INTERLEAVED_7 (lower 16 bits).

9. To transmit Right Channel data using I2S_TX, write to any of the following registers: RIGHT_16BIT, RIGHT_24BIT, RIGHT_32BIT_0 .. RIGHT_32BIT_7. INTERLEAVED_0 .. INTERLEAVED_7 (upper 16 bits).
10. To receive Left Channel data using I2S_RX, read from any of the following registers: LEFT_16BIT, LEFT_24BIT, LEFT_32BIT_0 .. LEFT_32BIT_7. INTERLEAVED_0 .. INTERLEAVED_7 (lower 16 bits).
11. To receive Right Channel data using I2S_RX, read from any of the following registers: RIGHT_16BIT, RIGHT_24BIT, RIGHT_32BIT_0 .. RIGHT_32BIT_7. INTERLEAVED_0 .. INTERLEAVED_7 (upper 16 bits).

Table 585. I2S configuraion parameters

Fractional Divider 17 Value			Frequencies		PLL 0 parameter set-up					
256 Fs	**512 Fs**	**1024 Fs**	**Fout (MHz)**	**Fcco (MHz)**	**Ndec**	**Mdec**	**Pdec**	**SELR**	**SELI**	**SELP**
96 kHz	-	-	24.5760	491.5200	63	13523	14	0	8	31
88.2 kHz	-	-	22.5792	406.4256	131	29784	23	0	8	31
64 kHz	-	-	16.3840	491.5200	63	13523	24	0	8	31
48 kHz	24 kHz	-	12.2880	368.6400	63	2665	24	0	8	31
44.1 kHz	22.05 kHz	-	11.2896	406.4256	131	29784	7	0	8	31
32 kHz	16 kHz	8 kHz	8.1920	409.6000	62	30542	6	0	8	31
-	12 kHz	-	6.1440	307.2000	5	30580	6	0	56	31
-	11.025 kHz	-	5.6448	282.2400	63	31356	6	0	12	31

1. Introduction

The JTAG Controller (JTAGC) module supports debug access to the ARM926 Platform and tristate enable of the I/O pads. The overall strategy is to achieve good test and debug features without increasing the pin count and reducing the complexity of I/O muxing. The JTAG Controller is compatible with IEEE1149.1 Standard Test Access Port and Boundary Scan Architecture.

1.1 Features

The LPC3130/31 has the following features:

- ARM926 debug access.
- Boundary scan.
- No target resources are required by the software debugger in order to start the debugging session.
- Allows the software debugger to talk via a JTAG (Joint Test Action Group) port directly to the core.
- Inserts instructions directly in to the ARM core.
- The ARM core or the System state can be examined, saved or changed depending on the type of instruction inserted.
- Allows instruction to execute at a slow debug speed or at a fast system speed.

2. General description

The LPC3130/31 Debug Architecture uses a JTAG port as a method of accessing the core. The scan chains that are around the core for production test are reused in the debug state to capture information from the databus and to insert new information into the core or the memory. There are two Test Access Port (TAP) controllers within LPC3130/31.

- Scan TAP: A JTAG-style TAP Controller controls the scan chains. With TAP_ID 0x1541E02B.
- ARM TAP: TAP controller present within ARM926EJS core. With TAP_ID 0x07926F0F.

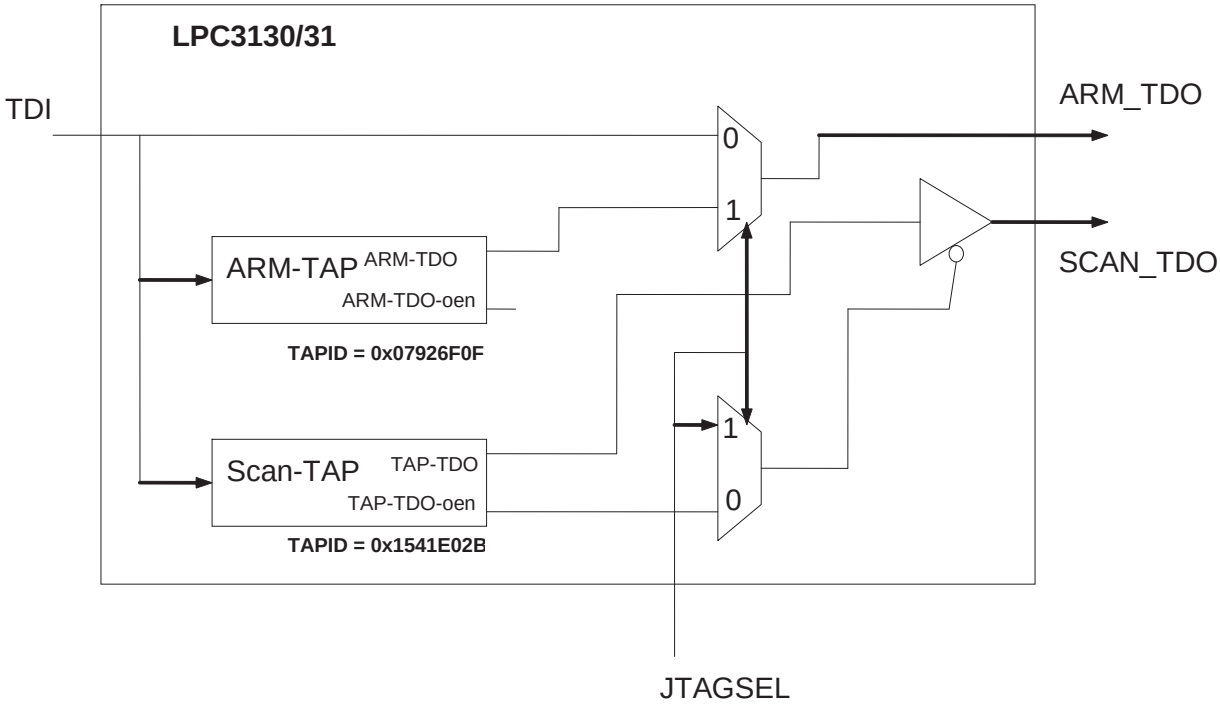


Fig 94. JTAG block diagram

As shown in the Figure 29–94, the “JTAGSEL” pin controls output function of SCAN_TDO and ARM_TDO signals.

When JTAGSEL = 0 (pull-down), ARM_TDO becomes buffered version of TDI and SCAN_TDO is enabled. SCAN_TDO should be connected to the TDO pin of JTAG port for boundary scan tests. In this mode TAP ID register reports 0x1541E02B.

When JTAGSEL = 1 (pull-up), ARM_TDO is enabled and should be connected to the TDO pin of JTAG port for ARM926 debug functions. SCAN_TDO is tri-stated in this mode. In this mode TAP ID register reports 0x07926F0F.

1. Introduction

This chapter contains the pinning information for the LPC3130/31. Because the TFBGA180 package does not have enough pins to allow for all signals for all peripherals to be pinned out, the LPC3130/31 employs a pin-multiplexing scheme to make an optimal selection of memories and peripherals possible.

2. Pinning information

2.1 Pinning

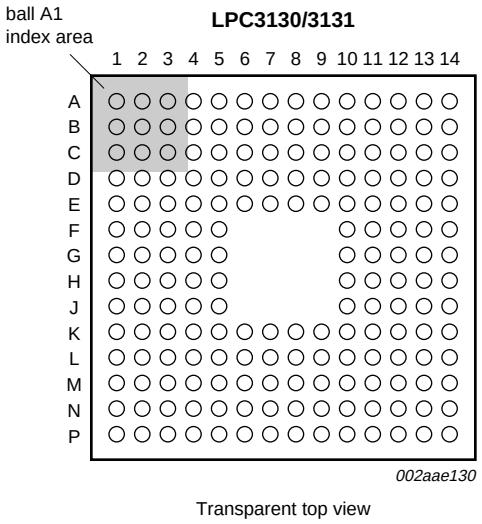


Fig 95. LPC3130/31 pinning TFBGA180 package

Table 586. Pin allocation table

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
Row A							
1	EBI_D_10	2	EBI_A_1_CLE	3	EBI_D_9	4	mGPIO10
5	mGPIO7	6	mGPIO6	7	SPI_CS_OUT0	8	SPI_SCK
9	VDDI	10	FFAST_IN	11	VSSI	12	ADC10B_GNDA
13	ADC10B_VDDA33	14	ADC10B_GPA1	-	-	-	-
Row B							
1	EBI_D_8	2	VDDE_IOA	3	EBI_A_0_ALE	4	mNAND_RYBN2
5	mGPIO8	6	mGPIO5	7	SPI_MOSI	8	SPI_CS_IN
9	PWM_DATA	10	FFAST_OUT	11	GPIO3	12	VSSE_IOC
13	ADC10B_GPA2	14	ADC10B_GPA0	-	-	-	-
Row C							

Table 586. Pin allocation table *…continued*

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	EBI_D_7	2	EBI_D_11	3	VSSE_IOA	4	VSSE_IOA
5	mGPIO9	6	VDDI	7	VSSI	8	SPI_MISO
9	VDDI	10	I2C_SDA0	11	GPIO4	12	VDDI
13	VDDE_IOC	14	ADC10B_GPA3	-	-	-	-
Row D							
1	EBI_D_5	2	EBI_D_6	3	EBI_D_13	4	mNAND_RYBN3
5	VDDE_IOC	6	VSSE_IOC	7	VDDE_IOC	8	VSSE_IOC
9	VSSE_IOC	10	I2C_SCL0	11	VDDA12	12	VSSI
13	BUF_TCK	14	BUF_TMS	-	-	-	-
Row E							
1	EBI_D_3	2	EBI_D_4	3	EBI_D_14	4	VSSE_IOA
5	VDDE_IOA	6	mNAND_RYBN0	7	mNAND_RYBN1	8	VDDE_IOC
9	VSSA12	10	VDDA12	11	ARM_TDO	12	I2C_SDA1
13	I2C_SCL1	14	I2STX_BCK1	-	-	-	-
Row F							
1	EBI_D_2	2	EBI_D_1	3	EBI_D_15	4	VSSE_IOA
5	VDDE_IOA	10	SCAN_TDO	11	BUF_TRST_N	12	I2STX_DATA1
13	I2SRX_WS1	14	I2SRX_BCK1	-	-	-	-
Row G							
1	EBI_NCAS_BLOUT_0	2	EBI_D_0	3	EBI_D_12	4	VSSI
5	VDDE_IOA	10	I2STX_WS1	11	VSSE_IOC	12	VDDE_IOC
13	SYSCLK_O	14	I2SRX_DATA1	-	-	-	-
Row H							
1	EBI_DQM_0_NOE	2	EBI_NRAS_BLOUT_1	3	VDDI	4	VSSE_IOA
5	VDDE_IOA	10	GPIO12	11	GPIO19	12	CLK_256FS_O
13	GPIO11	14	RSTIN_N	-	-	-	-
Row J							
1	NAND_NCS_0	2	EBI_NWE	3	NAND_NCS_1	4	CLOCK_OUT
5	USB_RREF	10	GPIO1	11	GPIO16	12	GPIO13
13	GPIO15	14	GPIO14	-	-	-	-
Row K							
1	NAND_NCS_2	2	NAND_NCS_3	3	VSSE_IOA	4	USB_VSSA_REF
5	mLCD_DB_12	6	mLCD_DB_6	7	mLCD_DB_10	8	mLCD_CSB
9	TDI	10	GPIO0	11	VDDE_ESD	12	GPIO17
13	GPIO20	14	GPIO18	-	-	-	-
Row L							
1	USB_VDDA12_PLL	2	USB_VBUS	3	USB_VSSA_TERM	4	VDDE_IOB
5	mLCD_DB_9	6	VSSI	7	VDDI	8	mLCD_E_RD
9	VSSE_IOC	10	VDDE_IOC	11	VSSI	12	VDDI
13	VSSE_IOC	14	GPIO2	-	-	-	-
Row M							

Table 586. Pin allocation table *…continued*

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	USB_ID	2	USB_VDDA33_DRV	3	VSSE_IOB	4	VSSE_IOB
5	VDDE_IOB	6	VSSE_IOB	7	VDDE_IOB	8	VSSE_IOB
9	VDDE_IOB	10	I2SRX_DATA0	11	mI2STX_WS0	12	mI2STX_BCK0
13	mI2STX_DATA0	14	TCK	-	-	-	-
Row N							
1	USB_GNDA	2	USB_DM	3	mLCD_DB_15	4	mLCD_DB_11
5	mLCD_DB_8	6	mLCD_DB_2	7	mLCD_DB_4	8	mLCD_DB_0
9	mLCD_RW_WR	10	I2SRX_BCK0	11	JTAGSEL	12	UART_TXD
13	mUART_CTS_N	14	mI2STX_CLK0	-	-	-	-
Row P							
1	USB_VDDA33	2	USB_DP	3	mLCD_DB_14	4	mLCD_DB_13
5	mLCD_DB_7	6	mLCD_DB_3	7	mLCD_DB_5	8	mLCD_RS
9	mLCD_DB_1	10	TMS	11	I2SRX_WS0	12	UART_RXD
13	TRST_N	14	mUART_RTS_N	-	-	-	-

Table 587. Pin description

Pin names with prefix m are multiplexed pins. See Table 30–590 for pin function selection of multiplexed pins.

Pin name	BGA Ball	Digital I/O level [1]	Application function	Pin state after reset	Cell Type [2]	Description
Clock Generation Unit						
FFAST_IN	A10	SUP1	AI	-	AIO2	12 MHz oscillator clock input
FFAST_OUT	B10	SUP1	AO		AIO2	12 MHz oscillator clock output
VDDA12	D11; E10	SUP1	Supply		PS3	12 MHz oscillator/PLLs Analog supply
VSSA12	E9		Ground	-	CG1	12 MHz oscillator/PLLs Analog ground
RSTIN_N	H14	SUP3	DI	I	DIO2	System Reset Input (active LOW)
CLK_256FS_O	H12	SUP3	DO	O	DIO1	Programmable clock output; fractionally derived from CLK1024FS_BASE clock domain. Generally used for Audio Codec master clock.
CLOCK_OUT	J4	SUP3	DO	O	DIO1	Programmable clock output; fractionally derived from SYS_BASE clock domain.
SYSCLK_O[3]	G13	SUP3	DO	O	DIO1	Programmable clock output. Output one of seven base/reference input clocks. No fractional divider.
10-bit ADC						
ADC10B_VDDA33	A13	SUP3	Supply	-	PS3	10-bit ADC Analog Supply
ADC10B_GNDA	A12		Ground	-	CG1	10-bit ADC Analog Ground
ADC10B_GPA0	B14	SUP3	AI	-	AIO1	10-bit ADC Analog Input
ADC10B_GPA1	A14	SUP3	AI	-	AIO1	10-bit ADC Analog Input
ADC10B_GPA2	B13	SUP3	AI	-	AIO1	10-bit ADC Analog Input
ADC10B_GPA3	C14	SUP3	AI	-	AIO1	10-bit ADC Analog Input

Table 587. Pin description

Pin names with prefix m are multiplexed pins. See Table 30–590 for pin function selection of multiplexed pins.

Pin name	BGA Ball	Digital I/O level [1]	Application function	Pin state after reset	Cell Type [2]	Description
USB HS 2.0 OTG						
USB_VBUS	L2	SUP5	AI	-	AIO3	USB supply detection line
USB_ID	M1	SUP3	AI	-	AIO1	Indicates to the USB transceiver whether in device (USB_ID HIGH) or host (USB_ID LOW) mode (contains internal pull-up resistor)
USB_RREF	J5	SUP3	AIO	-	AIO1	USB Connection for external reference resistor (12 kΩ ± 1 %) to analog ground supply
USB_DP	P2	SUP3	AIO	-	AIO1	USB D+ connection with integrated 45 Ω termination resistor
USB_DM	N2	SUP3	AIO	-	AIO1	USB D− connection with integrated 45 Ω termination resistor
USB_VDDA12_PLL	L1	SUP1	Supply	-	PS3	USB PLL supply
USB_VDDA33_DRV	M2	SUP3	Supply	-	PS3	USB Analog supply for driver
USB_VDDA33	P1	SUP3	Supply	-	PS3	USB Analog supply for PHY
USB_VSSA_TERM	L3		Ground	-	CG1	USB Analog ground for clean reference for on chip termination resistors
USB_GNDA	N1		Ground	-	CG1	USB Analog ground
USB_VSSA_REF	K4		Ground	-	CG1	USB Analog ground for clean reference
JTAG						
JTAGSEL	N11	SUP3	DI	I	DIO1	JTAG selection. Controls output function of SCAN_TDO and ARM_TDO signals.
TDI	K9	SUP3	DI	I	DIO1	JTAG Data Input
TRST_N	P13	SUP3	DI	I	DIO1	JTAG Reset Input
TCK	M14	SUP3	DI	I	DIO1	JTAG Clock Input
TMS	P10	SUP3	DI	I	DIO1	JTAG Mode Select Input
SCAN_TDO	F10	SUP3	DO	O/Z	DIO1	JTAG TDO signal from scan TAP controller. Pin state is controlled by JTAGSEL.
ARM_TDO	E11	SUP3	DO	O	DIO1	JTAG TDO signal from ARM926 TAP controller.
BUF_TRST_N	F11	SUP3	DO	O	DIO1	Buffered TRST_N out signal. Used for connecting an on board TAP controller (FPGA, DSP, etc.).
BUF_TCK	D13	SUP3	DO	O	DIO1	Buffered TCK out signal. Used for connecting an on board TAP controller (FPGA, DSP, etc.).
BUF_TMS	D14	SUP3	DO	O	DIO1	Buffered TMS out signal. Used for connecting an on board TAP controller (FPGA, DSP, etc.).
UART						
mUART_CTS_N[3][4]	N13	SUP3	DI / GPIO	I	DIO1	UART Clear To Send (active LOW)
mUART_RTS_N[3][4]	P14	SUP3	DO / GPIO	O	DIO1	UART Ready To Send (active LOW)
UART_RXD[3]	P12	SUP3	DI / GPIO	I	DIO1	UART Serial Input
UART_TXD[3]	N12	SUP3	DO / GPIO	O	DIO1	UART Serial Output

Table 587. Pin description

Pin names with prefix m are multiplexed pins. See Table 30–590 for pin function selection of multiplexed pins.

Pin name	BGA Ball	Digital I/O level [1]	Application function	Pin state after reset	Cell Type [2]	Description
I²C master/slave interface						
I2C_SDA0	C10	SUP3	DIO	I	IICD	I²C Data Line
I2C_SCL0	D10	SUP3	DIO	I	IICC	I²C Clock line
I2C_SDA1[3]	E12	SUP3	DIO	O	DIO1	I²C Data Line
I2C_SCL1[3]	E13	SUP3	DIO	O	DIO1	I²C Clock line
Serial Peripheral Interface						
SPI_CS_OUT0[3]	A7	SUP3	DO	O	DIO4	SPI Chip Select Output (Master)
SPI_SCK[3]	A8	SUP3	DIO	I	DIO4	SPI Clock Input (Slave) / Clock Output (Master)
SPI_MISO[3]	C8	SUP3	DIO	I	DIO4	SPI Data Input (Master) / Data Output (Slave)
SPI_MOSI[3]	B7	SUP3	DIO	I	DIO4	SPI Data Output (Master) / Data Input (Slave)
SPI_CS_IN[3]	B8	SUP3	DI	I	DIO4	SPI Chip Select Input (Slave)
Digital power supply						
VDDI	H3; L7; L12; C12; C6; A9; C9	SUP1	Supply	-	CS2	Digital Core Supply
VSSI	A11; C7; D12; G4; L6; L11		Ground	-	CG2	Digital Core Ground
Peripheral power supply						
VDDE_IOA	B2; E5; F5; G5; H5	SUP4	Supply	-	PS1	Peripheral supply for NAND flash interface
VDDE_IOB	L4; M5; M7; M9	SUP8	Supply	-	PS1	Peripheral supply for SDRAM/LCD
VDDE_IOC	C13; D5; D7; E8; G12; L10;	SUP3	Supply	-	PS1	Peripheral supply
VDDE_ESD	K11	SUP8	Supply	-	PS1	

Table 587. Pin description

Pin names with prefix m are multiplexed pins. See Table 30–590 for pin function selection of multiplexed pins.

Pin name	BGA Ball	Digital I/O level [1]	Application function	Pin state after reset	Cell Type [2]	Description
VSSE_IOA	C3; C4; E4; F4; H4; K3		Ground	-	PG1	
VSSE_IOB	M3; M4; M6; M8		Ground	-	PG1	
VSSE_IOC	B12; D6; D8; D9; G11; L9; L13		Ground	-	PG1	
LCD Interface						
mLCD_CSB[3]	K8	SUP8	DO	O	DIO4	LCD Chip Select (active LOW)
mLCD_E_RD[3]	L8	SUP8	DO	O	DIO4	LCD, 6800 Enable, 8080 Read Enable (active HIGH)
mLCD_RS[3]	P8	SUP8	DO	O	DIO4	LCD, Instruction Register (LOW)/ Data Register (HIGH) select
mLCD_RW_WR[3]	N9	SUP8	DO	O	DIO4	LCD, 6800 Read/write Select, 8080 Write Enable (active HIGH)
mLCD_DB_0[3]	N8	SUP8	DIO	O	DIO4	LCD Data 0
mLCD_DB_1[3]	P9	SUP8	DIO	O	DIO4	LCD Data 1
mLCD_DB_2[3]	N6	SUP8	DIO	O	DIO4	LCD Data 2
mLCD_DB_3[3]	P6	SUP8	DIO	O	DIO4	LCD Data 3
mLCD_DB_4[3]	N7	SUP8	DIO	O	DIO4	LCD Data 4
mLCD_DB_5[3]	P7	SUP8	DIO	O	DIO4	LCD Data 5
mLCD_DB_6[3]	K6	SUP8	DIO	O	DIO4	LCD Data 6
mLCD_DB_7[3]	P5	SUP8	DIO	O	DIO4	LCD Data 7
mLCD_DB_8[3]	N5	SUP8	DIO	O	DIO4	LCD Data 8 / 8-bit Data 0
mLCD_DB_9[3]	L5	SUP8	DIO	O	DIO4	LCD Data 9 / 8-bit Data 1
mLCD_DB_10[3]	K7	SUP8	DIO	O	DIO4	LCD Data 10 / 8-bit Data 2
mLCD_DB_11[3]	N4	SUP8	DIO	O	DIO4	LCD Data 11 / 8-bit Data 3
mLCD_DB_12[3]	K5	SUP8	DIO	O	DIO4	LCD Data 12 / 8-bit Data 4 / 4-bit Data 0
mLCD_DB_13[3]	P4	SUP8	DIO	O	DIO4	LCD Data 13 / 8-bit Data 5 / 4-bit Data 1 / Serial Clock Output
mLCD_DB_14[3]	P3	SUP8	DIO	O	DIO4	LCD Data 14 / 8-bit Data 6 / 4-bit Data 2 / Serial Data Input

Table 587. Pin description

Pin names with prefix m are multiplexed pins. See Table 30–590 for pin function selection of multiplexed pins.

Pin name	BGA Ball	Digital I/O level [1]	Application function	Pin state after reset	Cell Type [2]	Description
mLCD_DB_15[3]	N3	SUP8	DIO	O	DIO4	LCD Data 15 / 8-bit Data 7 / 4-bit Data 3 / Serial Data Output
I²S/Digital Audio Input						
I2SRX_DATA0[3]	M10	SUP3	DI / GPIO	I	DIO1	I²S Serial Data Receive Input
I2SRX_DATA1[3]	G14	SUP3	DI / GPIO	I	DIO1	I²S Serial Data Receive Input
I2SRX_BCK0[3]	N10	SUP3	DIO / GPIO	I	DIO1	I²S Bitclock
I2SRX_BCK1[3]	F14	SUP3	DIO / GPIO	I	DIO1	I²S Bitclock
I2SRX_WS0[3]	P11	SUP3	DIO / GPIO	I	DIO1	I²S Word select
I2SRX_WS1[3]	F13	SUP3	DIO / GPIO	I	DIO1	I²S Word select
I²S/Digital Audio Output						
mI2STX_DATA0[3]	M13	SUP3	DO / GPIO	O	DIO1	I²S Serial Data Transmit Output
mI2STX_BCK0[3]	M12	SUP3	DO / GPIO	O	DIO1	I²S Bitclock
mI2STX_WS0[3]	M11	SUP3	DO / GPIO	O	DIO1	I²S Word select
mI2STX_CLK0[3]	N14	SUP3	DO / GPIO	O	DIO1	I²S Serial Clock
I2STX_DATA1[3]	F12	SUP3	DO / GPIO	O	DIO1	I²S Serial Data Transmit Output
I2STX_BCK1[3]	E14	SUP3	DO / GPIO	O	DIO1	I²S Bitclock
I2STX_WS1[3]	G10	SUP3	DO / GPIO	O	DIO1	I²S Word select
General Purpose I/O (IOCONFIG module)						
GPIO0	K10	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 0 (Mode pin 0)
GPIO1	J10	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 1 (Mode pin 1)
GPIO2	L14	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 2 (Mode pin 2)
GPIO3	B11	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 3
GPIO4	C11	SUP3	GPI	I	DIO1	General Purpose Input Pin 4
mGPIO5[3]	B6	SUP3	GPIO	I	DIO4	General Purpose I/O Pin 5
mGPIO6[3]	A6	SUP3	GPIO	I	DIO4	General Purpose I/O Pin 6
mGPIO7[3]	A5	SUP3	GPIO	I	DIO4	General Purpose I/O Pin 7
mGPIO8[3]	B5	SUP3	GPIO	I	DIO4	General Purpose I/O Pin 8
mGPIO9[3]	C5	SUP3	GPIO	I	DIO4	General Purpose I/O Pin 9
mGPIO10[3]	A4	SUP3	GPIO	I	DIO4	General Purpose I/O Pin 10
GPIO11	H13	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 11
GPIO12	H10	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 12
GPIO13	J12	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 13
GPIO14	J14	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 14
GPIO15	J13	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 15
GPIO16	J11	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 16
GPIO17	K12	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 17
GPIO18	K14	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 18
GPIO19	H11	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 19

Table 587. Pin description

Pin names with prefix m are multiplexed pins. See Table 30–590 for pin function selection of multiplexed pins.

Pin name	BGA Ball	Digital I/O level [1]	Application function	Pin state after reset	Cell Type [2]	Description
GPIO20	K13	SUP3	GPIO	I	DIO1	General Purpose I/O Pin 20
External Bus Interface (NAND flash controller)						
EBI_A_0_ALE[3]	B3	SUP4	DO	O	DIO4	EBI Address Latch Enable
EBI_A_1_CLE[3]	A2	SUP4	DO	O	DIO4	EBI Command Latch Enable
EBI_D_0[3]	G2	SUP4	DIO	I	DIO4	EBI Data I/O 0
EBI_D_1[3]	F2	SUP4	DIO	I	DIO4	EBI Data I/O 1
EBI_D_2[3]	F1	SUP4	DIO	I	DIO4	EBI Data I/O 2
EBI_D_3[3]	E1	SUP4	DIO	I	DIO4	EBI Data I/O 3
EBI_D_4[3]	E2	SUP4	DIO	I	DIO4	EBI Data I/O 4
EBI_D_5[3]	D1	SUP4	DIO	I	DIO4	EBI Data I/O 5
EBI_D_6[3]	D2	SUP4	DIO	I	DIO4	EBI Data I/O 6
EBI_D_7[3]	C1	SUP4	DIO	I	DIO4	EBI Data I/O 7
EBI_D_8[3]	B1	SUP4	DIO	I	DIO4	EBI Data I/O 8
EBI_D_9[3]	A3	SUP4	DIO	I	DIO4	EBI Data I/O 9
EBI_D_10[3]	A1	SUP4	DIO	I	DIO4	EBI Data I/O 10
EBI_D_11[3]	C2	SUP4	DIO	I	DIO4	EBI Data I/O 11
EBI_D_12[3]	G3	SUP4	DIO	I	DIO4	EBI Data I/O 12
EBI_D_13[3]	D3	SUP4	DIO	I	DIO4	EBI Data I/O 13
EBI_D_14[3]	E3	SUP4	DIO	I	DIO4	EBI Data I/O 14
EBI_D_15[3]	F3	SUP4	DIO	I	DIO4	EBI Data I/O 15
EBI_DQM_0_NOE[3]	H1	SUP4	DO	O	DIO4	NAND Read Enable (active LOW)
EBI_NWE[3]	J2	SUP4	DO	O	DIO4	NAND Write Enable (active LOW)
NAND_NCS_0[3]	J1	SUP4	DO	O	DIO4	NAND Chip Enable 0
NAND_NCS_1[3]	J3	SUP4	DO	O	DIO4	NAND Chip Enable 1
NAND_NCS_2[3]	K1	SUP4	DO	O	DIO4	NAND Chip Enable 2
NAND_NCS_3[3]	K2	SUP4	DO	O	DIO4	NAND Chip Enable 3
mNAND_RYBN0[3]	E6	SUP4	DI	I	DIO4	NAND Ready/Busy 0
mNAND_RYBN1[3]	E7	SUP4	DI	I	DIO4	NAND Ready/Busy 1
mNAND_RYBN2[3]	B4	SUP4	DI	I	DIO4	NAND Ready/Busy 2
mNAND_RYBN3[3]	D4	SUP4	DI	I	DIO4	NAND Ready/Busy 3
EBI_NCAS_BLOUT_0[3]	G1	SUP4	DO	O	DIO4	EBI Lower lane byte select (7:0)
EBI_NRAS_BLOUT_1[3]	H2	SUP4	DO	O	DIO4	EBI Upper lane byte select (15:8)
Pulse Width Modulation module						
PWM_DATA[3]	B9	SUP3	DO / GPIO	O	DIO1	PWM Output

[1] Digital I/O levels are explained in Table 30–588.

[2] Cell types are explained in Table 30–589.

[3] Pin can be configured as GPIO pin in the IOCONFIG block.

[4] The UART flow control lines (mUART_CTS_N and mUART_RTS_N) are multiplexed. This means that if these balls are not required for UART flow control, they can also be selected to be used for an alternative function: SPI chip select signals (SPI_CS_OUT1 and SPI_CS_OUT2)

Table 588. Supply domains

Supply Domain	Voltage range	Related supply pins	Description
SUP1	1.0 V– 1.3 V	VDDI, VDDA12, USB_VDDA12_PLL,	Digital core supply
SUP3	2.7 V - 3.3 V	VDDE_IOC, ADC10B_VDDA33, USB_VDDA33_DRV, USB_VDDA33,	Peripheral supply
SUP4	1.65 V - 1.95 V (in 1.8 V mode) 2.5 V - 3.1 V (in 2.8 V mode)	VDDE_IOA	Peripheral supply for NAND flash interface
SUP5	4.5 V– 5.5 V	USB_VBUS	USB VBUS voltage
SUP8	1.65 V - 1.95 V (in 1.8 V mode) 2.5 V - 3.1 V (in 2.8 V mode)	VDDE_IOB	Peripheral supply for SDRAM/SRAM/bus-based LCD [1]

[1] When the SDRAM is used, the supply voltage of the NAND flash, SDRAM, and the LCD Interface must be the same, i.e. SUP4 and SUP8 should be connected to the same rail. (See also Section 30–3.)

Table 589: I/O pads

Cell type	Pad type	Function	Description
DIO1	bspts3chp	Digital Input/Output	Bidirectional 3.3 V; 3-state output; 3 ns slew rate control; plain input; CMOS with hysteresis; programmable pull-up, pull-down, repeater
DIO2	bpts5pcph	Digital Input/Output	Bidirectional 5 V; plain input; 3-state output; CMOS with programmable hysteresis; programmable pull-up, pull-down, repeater
DIO4	mem1 bsptz40pchp	Digital Input/Output	Bidirectional 1.8 V or 2.8 V; plain input; 3-state output; programmable hysteresis; programmable pull-up, pull-down, repeater
IICC	iic3m4scl	Digital Input/Output	I^2C-bus; clock signal
IICD	iic3mvsda	Digital Input/Output	I^2C-bus; data signal
AIO1	apio3v3	Analog Input/Output	Analog input/output; protection to external 3.3 V supply rail
AIO2	apio	Analog Input/Output	Analog input/output
AIO3	apiot5v	Analog Input/Output	Analog input/output; 5 V tolerant pad-based ESD protection
CS1	vddco	Core Supply	-
CS2	vddi	Core Supply	-
PS1	vdde3v3	Peripheral Supply	-
PS2	vdde	Peripheral Supply	-
CG1	vssco	Core Ground	-
CG2	vssis	Core Ground	-
PG1	vsse	Peripheral Ground	-

3. LCD/NAND flash/SDRAM multiplexing

The UM10314 contains a rich set of specialized hardware interfaces but the TFBGA package does not contain enough pins to allow use of all signals of all interfaces simultaneously. Therefore a pin-multiplexing scheme is created, which allows the selection of the right interface for the application.

Pin multiplexing is enabled between the following interfaces:

- between the dedicated LCD interface and the external bus interface.
- between the NAND flash controller and the memory card interface.
- between UART and SPI.
- between I2STX_0 output and the PCM interface.

The pin interface multiplexing is subdivided into five categories: storage, video, audio, external bus, and UART related pin multiplexing. Each category supports several modes, which can be selected by programming the corresponding registers in the SysCReg.

3.1 Pin connections

Table 590. Signals to pins for the pin interface multiplexing

Pin Name	Default Signal	Alternate Signal	Description
mLCD_CSB	LCD_CSB	EBI_NSTCS_0	Video related pin multiplexing, see Table 26–546.
mLCD_DB_1	LCD_DB_1	EBI_NSTCS_1	Video related pin multiplexing, see Table 26–546.
mLCD_DB_0	LCD_DB_0	EBI_CLKOUT	Video related pin multiplexing, see Table 26–546.
mLCD_E_RD	LCD_E_RD	EBI_CKE	Video related pin multiplexing, see Table 26–546.
mLCD_RS	LCD_RS	EBI_NDYCS	Video related pin multiplexing, see Table 26–546.
mLCD_RW_WR	LCD_RW_WR	EBI_DQM_1	Video related pin multiplexing, see Table 26–546.
mLCD_DB_2	LCD_DB_2	EBI_A_2	Video related pin multiplexing, see Table 26–546.
mLCD_DB_3	LCD_DB_3	EBI_A_3	Video related pin multiplexing, see Table 26–546.
mLCD_DB_4	LCD_DB_4	EBI_A_4	Video related pin multiplexing, see Table 26–546.
mLCD_DB_5	LCD_DB_5	EBI_A_5	Video related pin multiplexing, see Table 26–546.
mLCD_DB_6	LCD_DB_6	EBI_A_6	Video related pin multiplexing, see Table 26–546.
mLCD_DB_7	LCD_DB_7	EBI_A_7	Video related pin multiplexing, see Table 26–546.
mLCD_DB_8	LCD_DB_8	EBI_A_8	Video related pin multiplexing, see Table 26–546.

Table 590. Signals to pins for the pin interface multiplexing

Pin Name	Default Signal	Alternate Signal	Description
mLCD_DB_9	LCD_DB_9	EBI_A_9	Video related pin multiplexing, see Table 26–546.
mLCD_DB_10	LCD_DB_10	EBI_A_10	Video related pin multiplexing, see Table 26–546.
mLCD_DB_11	LCD_DB_11	EBI_A_11	Video related pin multiplexing, see Table 26–546.
mLCD_DB_12	LCD_DB_12	EBI_A_12	Video related pin multiplexing, see Table 26–546.
mLCD_DB_13	LCD_DB_13	EBI_A_13	Video related pin multiplexing, see Table 26–546.
mLCD_DB_14	LCD_DB_14	EBI_A_14	Video related pin multiplexing, see Table 26–546.
mLCD_DB_15	LCD_DB_15	EBI_A_15	Video related pin multiplexing, see Table 26–546.
mGPIO5	GPIO5	MCI_CLK	Storage related pin multiplexing, see Table 26–547.
mGPIO6	GPIO6	MCI_CMD	Storage related pin multiplexing, see Table 26–547.
mGPIO7	GPIO7	MCI_DAT_0	Storage related pin multiplexing, see Table 26–547.
mGPIO8	GPIO8	MCI_DAT_1	Storage related pin multiplexing, see Table 26–547.
mGPIO9	GPIO9	MCI_DAT_2	Storage related pin multiplexing, see Table 26–547.
mGPIO10	GPIO10	MCI_DAT_3	Storage related pin multiplexing, see Table 26–547.
mNAND_RYBN0	NAND_RYBN0	MCI_DAT_4	External bus related pin multiplexing, see Table 26–548.
mNAND_RYBN1	NAND_RYBN1	MCI_DAT_5	External bus related pin multiplexing, see Table 26–548.
mNAND_RYBN2	NAND_RYBN2	MCI_DAT_6	External bus related pin multiplexing, see Table 26–548.
mNAND_RYBN3	NAND_RYBN3	MCI_DAT_7	External bus related pin multiplexing, see Table 26–548.
mI2STX_DATA0	I2STX_DATA0	PCM_DA	Digital Audio related pin multiplexing, see Table 26–550.
mI2STX_BCK0	I2STX_BCK0	PCM_FSC	Digital Audio related pin multiplexing, see Table 26–550.
mI2STX_WS0	I2STX_WS0	PCM_DCK	Digital Audio related pin multiplexing, see Table 26–550.
mI2STX_CLK0	I2STX_CLK0	PCM_DB	Digital Audio related pin multiplexing, see Table 26–550.
mUART_CTS_N	UART_CTS_N	SPI_CS_OUT1	UART related pin multiplexing, see Table 26–549.
mUART_RTS_N	UART_RTS_N	SPI_CS_OUT2	UART related pin multiplexing, see Table 26–549.

3.2 Multiplexing between LCD and MPMC

The multiplexing between the LCD interface and MPMC allows for the following two modes of operation:

- MPMC-mode: SDRAM and bus-based LCD or SRAM.
- LCD-mode: Dedicated LCD-Interface.

The external NAND flash is accessible in both modes.

The block diagram Figure 30–96 gives a high level overview of the modules in the chip that are involved in the pin interface multiplexing between the EBI, NAND flash controller, MPMC, and RAM-based LCD interface.

Figure 30–96 only shows the signals that are involved in pad-muxing, so not all interface signals are visible.

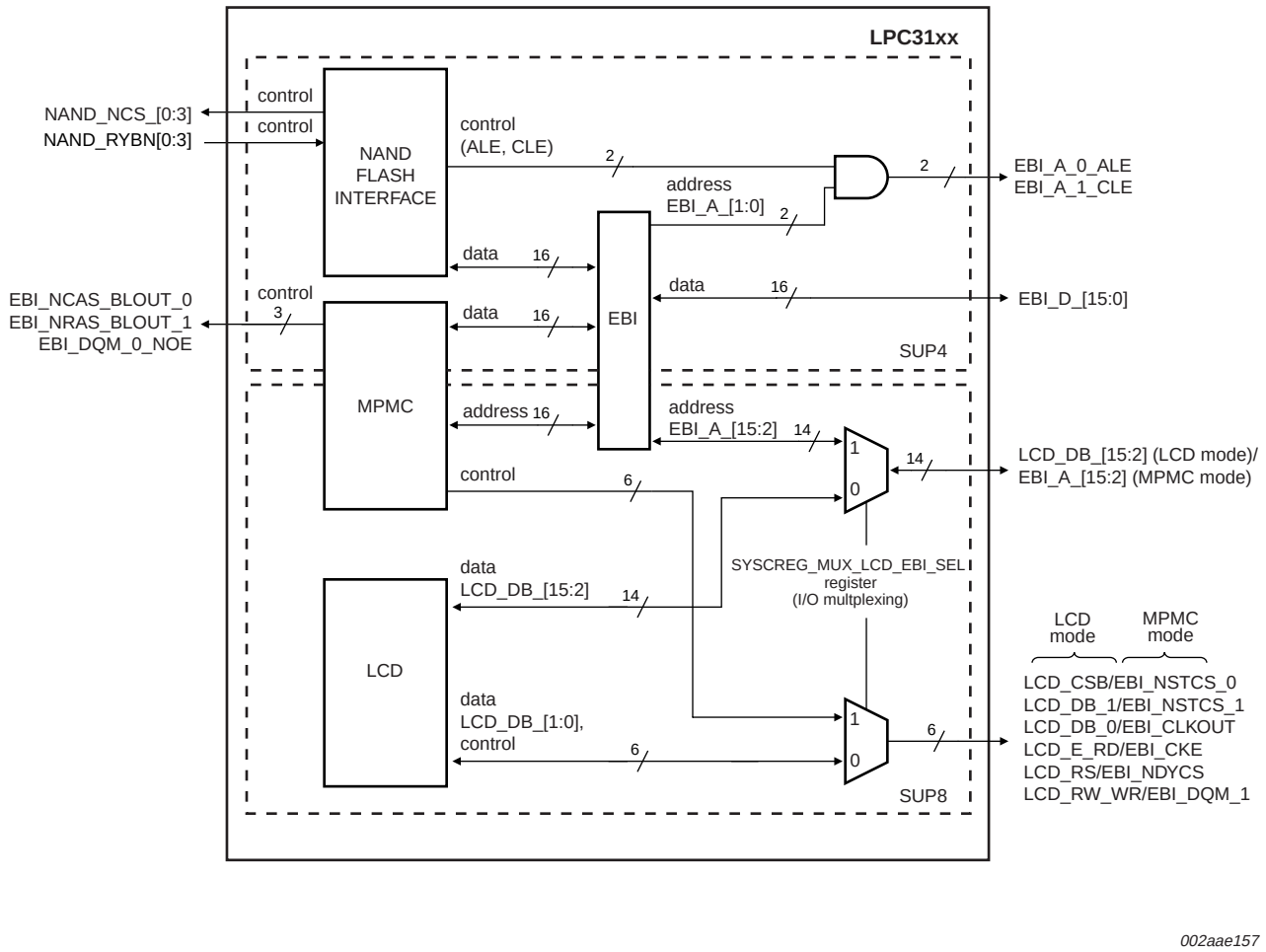


Fig 96. Muliplexing diagram of LCD and MPMC

The EBI unit between the NAND flash interface and the MPMC contains an arbiter that determines which interface is muxed to the outside world. Both NAND flash and SDRAM/SRAM initiate a request to the EBI unit. This request is granted using round-robin arbitration.

3.3 Supply domains

As is shown in Figure 30–96 the EBI (NAND flash/MPMC-control/data) is connected to a different supply domain than the LCD interface. The EBI control and address signals are muxed with the LCD interface signals and are part of supply domain SUP8. The SDRAM/SRAM data lines are shared with the NAND flash through the EBI and are part of supply domain SUP4. Therefore the following rules apply for connecting memories:

1. SDRAM and bus-based LCD or SRAM: This is the MPMC mode. The supply voltage for SDRAM/SRAM/bus-based LCD and NAND flash must be the same.The dedicated LCD interface is not available in this MPMC mode.
2. Dedicated LCD interface only: This is the LCD mode. The NAND flash supply voltage (SUP4) can be different from the LCD supply voltage (SUP8).

1. Abbreviations

Table 591: Abbreviations

Acronym	Description
ADC	Stereo Analog to Digital Converter for Audio
ADC10B	10 bit Analog to Digital Converter
ANVC	Analog Volume Control
BIU	Bus Interface Unit
CGU	Clock Generation Unit
DFU	Device Firmware Upgrade
DMA	Direct Memory Access Controller
DRM	Digital Rights Management
ECC	Error Correction Code
I^2C M/S	Inter IC Communication Master/Slave Interface
I^2S	Inter-IC Sound
INTC	Interrupt Controller
IOCONFIG	Input Output Configuration
ROM	ROM Memory
IrDA	Infrared Data Association
ISRAM	Internal Static RAM Memory
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
PHY	Physical Layer
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
RNG	Random Number Generator
SIR	Serial IrDA
SPI	Serial Peripheral Interface
SSI	Serial Synchronous Interface
SysCReg	System Control Registers
Timer	Timer module
UART	Universal Asynchronous Receiver Transmitter
USB 2.0 HS OTG	Universal Serial Bus 2.0 High Speed on the Go
WDT	WatchDog Timer

2. Legal information

2.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

2.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

2.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^2C-bus — logo is a trademark of NXP B.V.

3. Tables

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

4. Figures

continued >>

continued >>

5. Contents

Chapter 1: LPC3130/31 Introductory information

Chapter 2: LPC3130/31 NAND flash controller

Chapter 3: LPC3130/31 Multi-Port Memory Controller (MPMC)

continued >>

Chapter 4: LPC3130/31 External Bus Interface (EBI)

Chapter 5: LPC3130/31 memory map

Chapter 6: LPC3130/31 Internal Static Memory Controller ISROM/Boot ROM

continued >>

Chapter 7: LPC3130/31 Internal Static RAM (ISRAM) memory controller

Chapter 8: LPC3130/31 High-speed USB On-The-Go (OTG) controller

continued >>

continued >>

Chapter 9: LPC3130/31 DMA controller

Chapter 10: LPC3130/31 Interrupt controller

Chapter 11: LPC3130/31 AHB multi-layer matrix

Chapter 12: LPC3130/31 AHB-to-APB bridge

Chapter 13: LPC3130/31 Clock Generation Unit (CGU)

Chapter 14: LPC3130/31 WatchDog Timer (WDT)

Chapter 15: LPC3130/31 Input/Output configuration modules GPIO/IOCONFIG

Chapter 16: LPC3130/31 10-bit Analog-to-Digital Converter (ADC)

Chapter 17: LPC3130/31 event router

Chapter 18: LPC3130/31 Random Number Generator (RNG)

Chapter 19: LPC3130/31 SPI

Chapter 20: LPC3130/31 Memory Card Interface (MCI)

Chapter 21: LPC3130/31 Universal Asynchronous Receiver/Transmitter (UART)

Chapter 22: LPC3130/31 LCD interface

Chapter 23: LPC3130/31 I²C-bus interface

continued >>

Chapter 24: LPC3130/31 Timer 0/1/2/3

Chapter 25: LPC3130/31 Pulse Width Modulator (PWM)

Chapter 26: LPC3130/31 System control (SysCReg)

Chapter 27: LPC3130/31 Pulse Code Modulation (PCM)/ISDN Oriented Modular (IOM) interface

continued >>

Chapter 28: LPC3130/31 I2S interface

Chapter 29: LPC3130/31 JTAG

Chapter 30: LPC3130/31 Pin configuration and I/O muxing

Chapter 31: LPC3130/31 Supplementary information

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 4 March 2009
Document identifier: UM10314_1