Freescale Semiconductor
Data Sheet: Product Preview

Document Number: SCP220x
Rev. 2.1, 06/2015

SCP220x

SCP220x ICP Family Data Sheet

1 Introduction

The SCP220x is a family of highly-programmable Image Cognition Processors (ICP) enabling imaging and video applications for automotive smart cameras, video surveillance cameras and consumer devices such as personal media players. The ICPs of the SCP220x family are programmable system-on-chip (SoC) featuring CogniVue's patented APEX™ technology providing high computing performance at low power in a small package size.

The SCP220x family comprises:

- SCP2201 – Equipped with 128 Mbit (16 MB) of stacked Mobile DDR SDRAM in package
- SCP2207 – Equipped with 512 Mbit (64 MB) of stacked Mobile DDR SDRAM in package

1 Introduction . . . 1
1.1 The SCP220x function blocks . . . 2
1.2 SCP220x Features . . . 2
2 SCP220x Architecture Overview . . . 4
2.1 Voltage islands . . . 5
2.2 Blocks . . . 5
2.3 Buses and DMA . . . 6
2.4 Pin Configuration . . . 7
3 System Design Considerations . . . 8
3.1 Core and I/O Power . . . 8
3.2 PLL and Timing Generation . . . 10
3.3 Clock Configuration . . . 12
3.4 External Memory Interface . . . 16
3.5 Reset . . . 16
3.6 Boot-up . . . 17
3.7 Low Power Configurations . . . 20
4 Interconnect and Communication . . . 21
4.1 NAND Flash Interface . . . 21
4.2 UART . . . 24
4.3 SPI . . . 26
4.4 Sensor Interface (SIF) . . . 29
4.5 Display Sub-System (DSS) . . . 30
4.6 USB 2.0 HIGH SPEED . . . 35
4.7 Audio Interface . . . 36
4.8 Media Storage MMC and MMCPlus blocks (compatible SD/SDHC) . 42
4.9 I2C Interface . . . 45
4.10 Pulse Width Modulated Outputs . . . 50
4.11 KeyPad Scan Interface . . . 53
4.12 GPIOs and Alternate Functions . . . 54
4.13 Production Test and System Signals . . . 61
5 Registers . . . 61
5.1 Memory Map . . . 61
5.2 Clock Configuration Registers . . . 64
5.3 PAD and I/O registers . . . 71
5.4 Reset and Clock Gating . . . 78
5.5 Miscellaneous . . . 82
5.6 Memory Controller . . . 83
5.7 NAND Interface Registers Description . . . 95
5.8 UART Control Registers . . . 110
5.9 SPI Registers . . . 117
5.10 Audio Registers . . . 126
5.11 MMC/SD Control Registers . . . 136
5.12 MMCPlus Control Registers . . . 145
5.13 I2C Registers . . . 154
5.14 PWM Registers . . . 158
5.15 KeyScan Registers . . . 162
5.16 GPIO Registers . . . 165
6 Packaging . . . 168
6.1 SCP2201 . . . 168
6.2 SCP2207 . . . 169
6.3 SCP220x Pinout . . . 170
7 Electrical Specifications . . . 180
7.1 Absolute Maximum Rating . . . 180
7.2 Recommended Operating Ranges . . . 180
7.3 Thermal Characteristics . . . 181
7.4 DC Characteristics . . . 181
8 Revision History . . . 183

© Freescale Semiconductor, Inc., 2015

1.1 The SCP220x function blocks

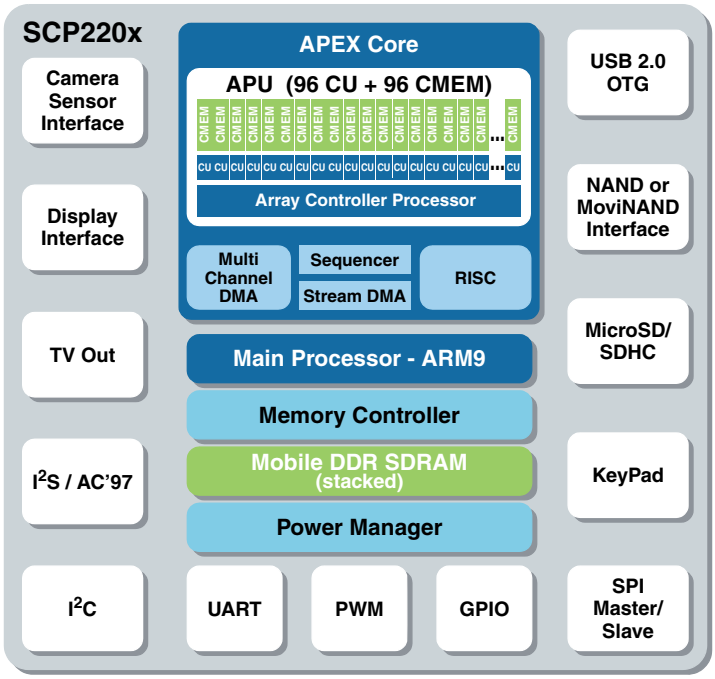


Figure 1. SCP220x Image Cognition Processors

1.2 SCP220x Features

1.2.1 SCP220x General Features

CogniVue APEX Processor – programmable 34Billion-Operations per second Vision Processor with patented massively parallel Array Processor Unit (APU) with 96 Computing Units (CUs) with dedicated memory, discreet RISC processor, H/W acceleration blocks, wide-bandwidth stream DMAs and internal 64-bit data buses

ARM926EJ-S™ RISC processor with 16 KB of instruction cache (I-cache) and 16 KB of data cache (D-cache)

Multiple power domains for different peripheral IOs

1.2.2 Interconnect and Communication

1.2.2.1 Video Processing

Fully-programmable Array Processor (APEX) for running video/image processing algorithms

Video codecs support diverse resolutions at 30 fps with 4 Mbps maximum bitrate

Supported video decoding standards:

MPEG-4 Simple Profile and Advanced Simple Profile supports 720x480 at 30 fps

For other standards consult factory

Supported video encoding standard is MPEG-4 Simple Profile, 720x480 at 30 fps

1.2.2.2 Audio Processing

Directly connects to I2S or AC97 compliant audio device

1.2.2.3 Graphics

True-color (24 bits per pixel) processing

2D graphics functions including: Bitblt, overlay, pixel-based alpha-blending, rotation, scaling, color space conversion, color depth expansion and reduction

1.2.2.4 Image Sensor Interface

Sensor Interface (SIF) supports 10-bit input, 8-bit YUV datapath up to 10 M-pixel resolution

Integrated YUV image enhancement functions such as scale-down

1.2.2.5 Display Sub-System

Independent dual output, one digital, one analog

Digital:

- LCD interface supports both TFT and buffered (CPU) LCDs
- Supports up to four CPU-like devices (for example dual 8/9/16/18bit LCD modules and two other devices with CPU-like interfaces) or supports up to WVGA TFT LCD up to 24 bits/pixel
- ITU-R 601/656 compatible digital video output

Analog: Integrated 10-bit DAC for analog composite video output to TV (PAL or NTSC)

1.2.2.6 USB 2.0 High Speed Controller

USB 2.0 HIGH SPEED compliant

USB 2.0 PHY integrated on-chip

USB 2.0 On-The-Go

1.2.2.7 Audio Interface

I2S and AC97 compliant Audio Interface

1.2.2.8 Media Storage Interface

Supports SD/SDHC removable memory cards

Compatible MMC Plus Interface

Supports 8-bit NAND flash devices

Supports FAT-16 and FAT-32 file system with long name support and international characters

1.2.2.9 Serial Interfaces

Two UART (1x 4-pin, 1x 2-pin) interfaces and two SPIs

1.2.2.10 Other Interfaces

General purpose I/O (GPIO) – selectable as alternative functions for various interface pins

Two PWM (Pulse width modulated) outputs with programmable frequency and duty cycle

JTAG test and debugging interface for the ARM926EJ-S processor

1.2.3 Reference Input Clock

Input clocks:

- Clocks supplied by either a crystal or oscillator
- 10-30 MHz (13 MHz, 19.5 MHz, 24 MHz or 27 MHz suggested).

5 on-chip PLLs generate clocks for system, array processor, display interface, other interfaces and memory

Programmable internal clock frequencies

1.2.4 Boot-Up Options

Boot from either serial SPI or NAND Flash

1.2.5 Integrated Memory

SCP2201 has 128 Mbit DDR SDRAM integrated in package

SCP2207 has 512 Mbit DDR SDRAM integrated in package

1.2.6 Package

SCP2201 and SCP2207 are both used in the 236 MAPBGA package (9 x 9 x 1.24 mm).

1.2.7 POWER Supply

1.0V core and 3.0V I/O power (1.8V or 3.0V Sensor I/O power)

1.8V memory power supply

3.0V PLL power supply

3.3V supplies for USB and internal DAC

Multiple power domain within the core for power management

1.2.8 Ambient Operating Temperature

SCP2201 and SCP2207 chips operate between -40°C to +105°C, Automotive Qualified

2 SCP220x Architecture Overview

The SCP220x are system-on-chip offering a large selection of computation and communication blocks in a single small form factor chip. The internal relationship between blocks within the chips can be represented by the following figure:

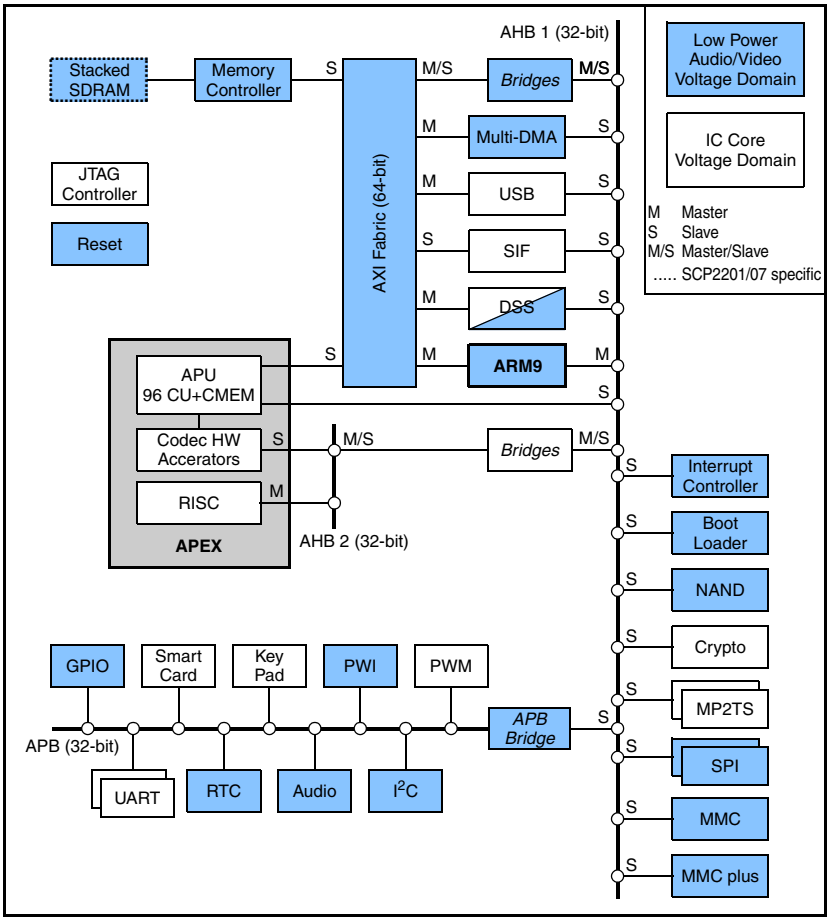


Figure 2. SCP220x Internal Architecture

2.1 Voltage islands

There are two voltage islands offering capabilities to lower power consumption when intensive computation is not required. The chip can boot in one or the other mode, through external configuration (see 3.6.2, Boot-up configuration), or can be switched by software. It is also possible to turn some blocks off for further power reduction (see 3.7, Low Power Configurations).

2.2 Blocks

2.2.1 APEX

APEX is a programmable Vision Processor capable of 34 Billion-Operations per second.

APEX is composed of:

- with patented massively parallel Array Processor Unit (APU) itself made of 96 Computing Units (CUs) with dedicated memory, wide-bandwidth stream DMAs and internal 64-bit data buses
- discreet RISC processor
- H/W acceleration blocks

APEX is a Single Instruction Multiple Data (SIMD) type of parallel processor. It is normally programmed in a proprietary SIMD Engine Language (SEL) to generate APU kernels. Custom APU kernels can be written with the

additional APEX toolkit. Standard and custom kernels can be combined with the automated APEX usage optimization tool ACF (APEX Core Framework).

APEX is then only used through supplied SDK, and no direct register accesses are required.

For advanced optimization needs, contact factory.

2.2.2 ARM926EJ-S RISC processor

The chip uses an ARM9 series as its main processor. It is a ARM926EJ-S™ RISC processor with 16 KB of instruction cache (I-cache) and 16 KB of data cache (D-cache).

All the software runs on this processor and it sets up all the other blocks including the APEX.

Operating Systems supported:

- Nucleus, embedded Real Time Operating System.

2.2.3 Reserved Use Blocks

We reserve the use of several blocks in the chip: Crypto, MP2TS.

2.2.4 Interconnect and Communication Blocks

Detailed description of the blocks can be found in 4, Interconnect and Communication.

2.3 Buses and DMA

There are four main buses in the chip:

- AXI Fabric (64-bit)
- AHB 1 and AHB 2 (both 32-bit)
- APB (32-bit)

2.3.1 AXI Fabric

The AXI fabric (Advanced eXtensible Interface) provides high performance data transfers. It is linked to the use of the APEX and so it is not recommended to access it directly. AXI provides an isolation from secondary data transfers from peripherals and offers maximized performance on the main data movements from and to memory, image input and output.

For your information, AXI provides a partial connectivity between the connected masters and slaves. The following table illustrates what slaves each master can access. An “x” indicates connectivity between the master and slave.

Table 1. AXI Master/Slave Connectivity

		apb3	Memory Controller	AHB 1 (data)	APEX	APEX CMEM	SIF
		S1	S2	S3	S4	S5	S6
AHB 1 (primary data)	M1	x	x			x	
USB	M2		x			x	
AHB 1 (instructions)	M3		x			x	
[reserved]	M4		x	x		x	

Table 1. AXI Master/Slave Connectivity

[reserved]	M5		x			x	
DSS Bitblt	M6		x		x	x	x
DSS Bitblt_mini	M7		x			x	
AHB 1 (secondary data)	M8	x	x			x	

2.3.2 AHB

There are two AHB (Advanced High-performance Bus) in the Chip. The first is the link between all the blocks. The second is a dedicated bus for video codec operations.

2.3.3 APB

APB (Advanced Peripheral Bus) provides connectivity to a large number of relatively slow peripheral interfaces blocs.

2.3.4 DMAs

There is an extensive number of DMAs (Direct Memory Access) in the chip. They play an important role in reaching high computing performance in video/image processing.

In general, the DMAs are handled directly through the SDK.

2.4 Pin Configuration

The SCP220x are designed to be small chip and so have constrains on the number of pins available. To offer maximum flexibility, the pins can have multiple functions selectable via software.

At most, a pin has a default function, a gpio use and an alternate function.

Also, the pin can have an internal Pull-Up (PU) or Pull-Down (PD) capability that could be activated by default.

Furthermore, the pin may have a direction and a configurable strength.

To illustrate, here is an example for the pin named audio_fsr.

4.12.1, GPIO and Alternate Function List tells us:

- the main function is audio_fsr (its name)
- if using as gpio it is the line 18
- the alternate function is pwm2_out
- the pin can have an internal Pull-Down
- the PAD type is A
- the power domain AUVDD

GPIO	Pin	Alternate	Power	PAD Type	PAD Resistor/Default
gpio18	audio_fsr_p	pwm2_out	AUVDD	A	PD/none

6.3, SCP220x Pinout tells us:

- the pin belongs to the AUDIO power domain
- the pin is capable of being an input or and output (bi-directional)
- the pad strength can be configured for 2 mA or for 4 mA
- the pin does NOT have its Pull-Down activated
- the pin is at position M9 on SCP2201 and SCP2207

Pin Name	Power Domain	PAD Type	Default PU/PD	SCP2201/07 Ball
audio_fsr	AUDIO	Bi-dir. 2 mA / 4 mA	none	M9

A pin is configured as gpio when the corresponding gpio enable bit is active.

A pin is configured as the alternate function when the gpio enable bit is inactive and that the alternate enable bit is active.

So a pin is configured as the primary function when the gpio enable bit is inactive and that the alternate enable bit is also inactive.

See 4.12.1, GPIO and Alternate Function List for details.

Note: most pins will be in tri-state during and after reset. The pins will start their primary function during the boot.

3 System Design Considerations

3.1 Core and I/O Power

The SCP220x are low power system-on-chip with a power consumption generally under <250 mW (active image processing with APEX).

They offer advanced power consumption reduction options see 3.7, Low Power Configurations.

In any case, all power need to be present at all times even if the chip is in power saving mode, undefined behavior may happen if some powers are not present.

The IO supply allows (3.0 V DC ± 10%).

The Sensor Interface (SIF) allows for 3.0VDC ± 10%, or 1.8VDC -5%, +10%.

The table below provides the SCP220x pin information for core and I/O power.

Table 2. SCP220x Power Supply

Power Supply	Pin Names	Pin Description
1.0 V	VDD_CORE	Power supply for IC core
	VDD_LP	Power supply for low power audio/video circuitry
3.0 V	VDDA_PLL	Analog supply voltage for PLL
*	VSSA_PLL	PLL power return (DO NOT CONNECT TO GROUND)
1.8 V	VDD_SDRAM	SDRAM core power
3.3 V	VDD_USB	Power supply for USB
	VDDA_DAC	Analog supply voltage for internal DAC

Table 2. SCP220x Power Supply

3.0 V or 1.8V	VDD_SENSOR	IO supply for Sensor Interface block and I2C
3.0 V	VDD_OSC	Power supply for crystal pad
	VDD_SCCARD	IO supply for Smart Card Interface
	VDD_GPIO	IO supply for GPIOs and KeyScan
	VDD_DIP	IO supply for DIP block
	VDD_MISCIF	IO supply for MP2TS, UART, SPI, and JTAG interfaces.
	VDD_SDMMC	IO supply for SD/SDHC/MMC Interface
	VDD_AUDIO	IO supply for Audio Interface block
	VDD_NAND	IO supply for NAND Interface block
GND	VSS	Common ground
	VSSA_DAC	Ground for internal analog DAC
	VSS_USB	Ground for USB
	VSS_OSC	Ground for crystal pad
* See Figure 4 for VSSA_PLL connectivity		

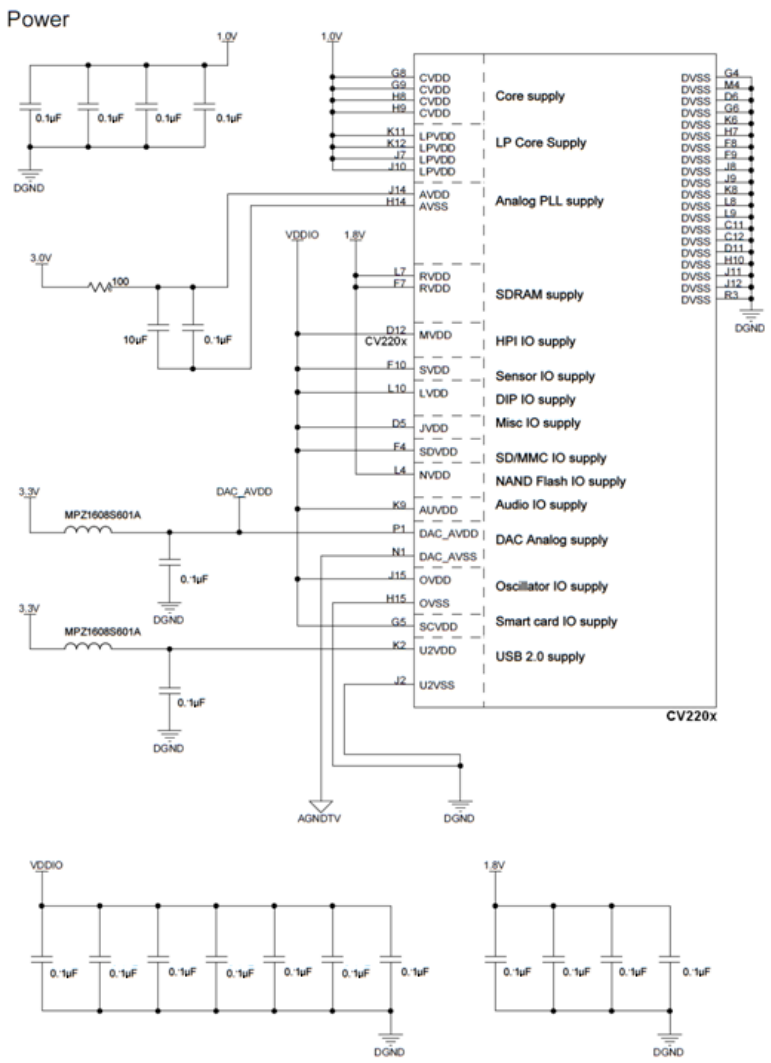


Figure 3. Powering the SCP220x

3.2 PLL and Timing Generation

The timing generation block provides and manages the clocks required by the internal logic and IP blocks. The clocks are produced from internal PLLs. An external crystal oscillator or clock provides the input clock to the PLLs. The following figure shows the connection between a crystal oscillator and the SCP220x. If the clock source is an oscillator, the clock output signals (VDDA_PLL, VSSA_PLL) are not connected.

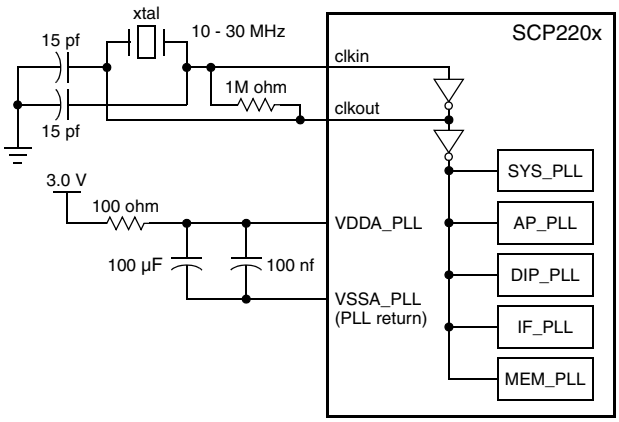


Figure 4. Crystal Connected to SCP220x

The input clock drives the five internal PLLs:

- SYS_PLL: System
- AP_PLL: Array Processor Unit
- DIP_PLL: Display Interface Port
- IF_PLL: Interfaces
- MEM_PLL: Memory

See 3.3, Clock Configuration for more details about the PLLs.

The SCP220x has different level of clock controls depending on the input oscillator or clock frequency:

- Input is 24 MHz: this is the default, all the clocks are set automatically, no external clock setting
- Input is 13 MHz, 19.2 MHz or 27 MHz: this is a pre-set, all clocks are set automatically through
- an external clock setting, see 3.6.2, Boot-up configuration
- Input is between 10 MHz and 30 MHz: this is a custom clock setting and so an external clock setting should be chosen and then all the PLLs and clock configuration need to be managed, see 3.3, Clock Configuration and then 5.2, Clock Configuration Registers.

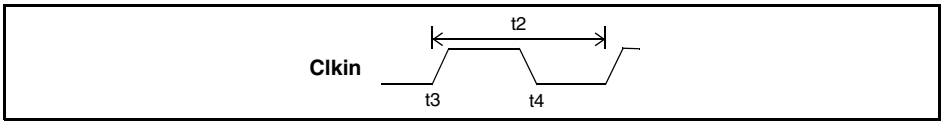


Figure 5. Input Clock Timing

Table 3. Input Clock Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input clock period	t2	10		30	MHz
Input clock rise time	t3			4	ns
Input clock fall time	t4			4	ns
Input clock duty cycle		40	50	60	%

3.3 Clock Configuration

3.3.1 PLL configuration

The SCP220x has five internal PLLs as clock sources; all have the input reference clock as input:

- SYS_PLL: System
- AP_PLL: Array Processor Unit
- DIP_PLL: Display Interface Port
- IF_PLL: Interfaces
- MEM_PLL: Memory

These five configurable internal clock sources have three configurable aspects:

- PLL output frequency
- PLL clock source for divider
- Divider value

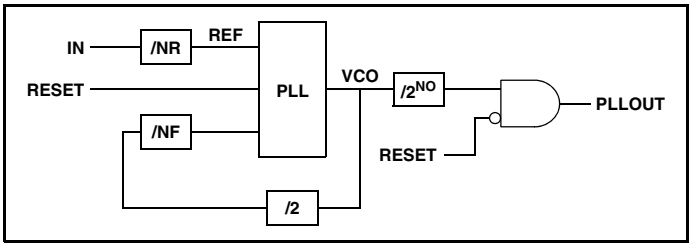


Figure 6. PLL Programming Parameters

Each PLL can be configured to a wide range of output frequencies based on its NF, NR and NO settings, where:

- NF (8-bit) ranges from 0 to 255
- NR (5-bit) ranges from 0 to 31
- NO (3-bit) ranges from 0 to 7; 2NO values: 1, 2, 4, 8, 16, 32, 64, 128

The PLL output frequency is derived from the following equation:

fOUT = (2 * fIN * (NF+1)) / (2NO * (NR+1))

where fIN is the input clock frequency. The following constraints must be met when deriving fPLLOUT.

- fIN = input frequency must meet 10 Mhz < fIN < 30 Mhz
- fREF = comparison frequency = fIN / (NR+1) must meet 10 Mhz < fREF < 30 Mhz
- fVCO = VCO frequency = (2 * fIN * (NF+1)) / (NR+1) must meet 1000 Mhz < fVCO < 2000 Mhz
- fOUT = output frequency must meet 20 Mhz < fOUT < 1000 Mhz

3.3.2 Configuring the clocks

There are two classes of clocks:

- The system clocks (cmem_clk, ac_clk, arm_clk, sys_clk, mem_clk2x, mem_clk) that have extra hardware controls so that changes are managed and they take default value from boot-up configuration; see 3.3.2.1, System Clocks
- The peripherals clocks are unmanaged. See 3.3.2.2, Peripheral Clocks

The PLL power-up default settings are controlled by either the recommended boot-up configuration (see 3.6.2, Boot-up configuration) or software programmable registers (advanced use only).

The SYS_PLL and AP_PLL settings are applied to the PLL when the appropriate configuration bit is set in the clock update register. The PLL has a “lock” time after the settings are applied. During this lock time, the timing_gen block will glitchlessly switch the ARM926EJ-S processor and system clocks. The “lock” time is approximately 520 input clock periods. It should be noted that if only the “NO” setting is changed the lock period is much shorter (9-10 input clock periods).

The IF_PLL, DIP_PLL and MEM_PLL do not have any hardware ensuring clean transition. The appropriate software clock gating must be activated before updating these PLLs.

The configuration of the ARM926EJ-S processor and system clocks is controlled by hardware mechanisms that are initiated by a software “kick”, whereas, the AP and peripheral clock configurations do not have any hardware control mechanisms. Software must manage all aspects of the clock configuration for the AP and peripherals.

The registers are described at 5.2, Clock Configuration Registers.

3.3.2.1 System Clocks

Three aspects of the ARM926EJ-S processor and system clocking can be individually updated:

- The PLL configuration.
- The output divider for the PLL.
- The PLL to be used for the clock generation.

This allows a lot of flexibility during the overall clock configuration such as:

- Using the other PLL while one PLL is locking. This prevents a switch over to the external reference clock while the PLL is locking and may be required for performance reasons in some applications.
- Using a common PLL for both the system and AP so that one of the PLLs can be powered down for current savings.

Care should be taken in the order of the configurations such that an invalid frequency is not generated for the clock domain.

The figure below shows the system clocks diagram:

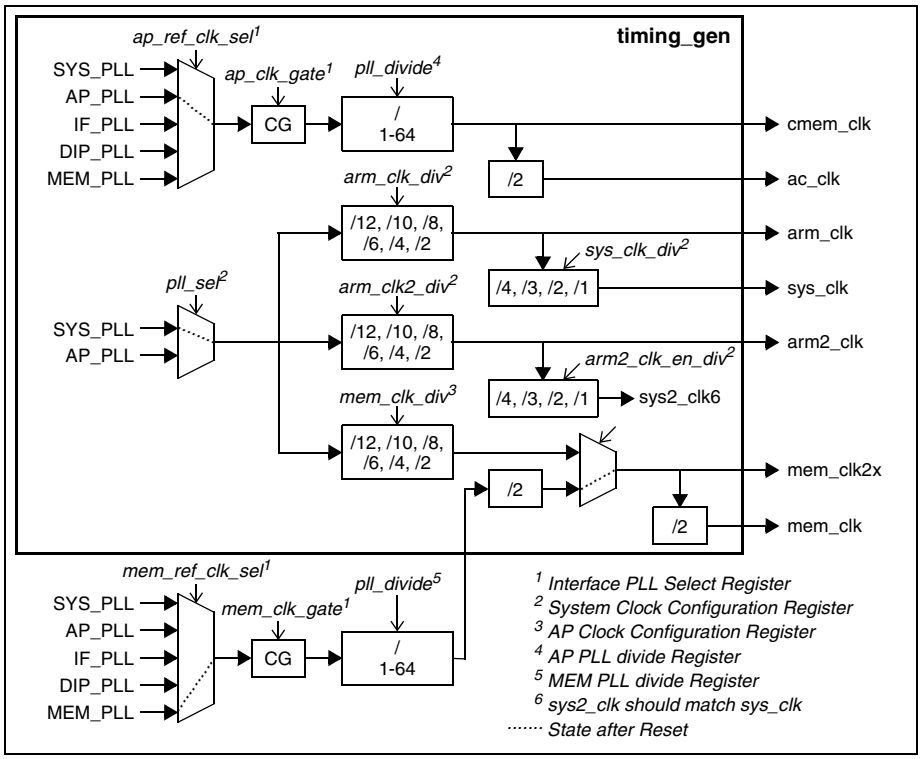


Figure 7. System Clocks

After reset, the system clocks are 96 MHz if the boot-up configuration matches the oscillator or crystal base frequency. DS-10163-00-08 32/234

NOTE

The software bootloader can change the clocks settings but it is important to know that the System initialization (Operating System and hardware subsystems) resets the clock setting as well.

There are three types of clock dividers:

- The integer divider where is clock is divided by a integer value from 1 to 64. Shown as „/1-64. blocks in diagram
- Fixed divider by 2, shown as '/2'. block in diagram
- Multiple choice dividers where clock is divider by one of the choice offered. The diagram shows these blocks by their list of choices such as: '/4, /3, /2, /1'.

To modify the value of SYS_PLL, AP_PLL, corresponding multiplexer and divider, it is required to use the Clock Update register to ensure proper transition, see.

NOTE

For advanced power saving mode, it is possible to gate clocks via the CG blocks. However, because of the reserved multiplexer (memory used with synchronized clock to the ARM926EJ-S processor or from independent clock), the mem_clk_gate should not be used.

3.3.2.2 Peripheral Clocks

There are five peripheral reference clocks:

- if_ref_clock: interfaces reference clock

- xga_ref_clock: digital display reference clock
- sif_ref_clock: sensor interface reference clock
- usb_ref_clock: USB (Universal Serial Bus) reference clock
- tvout_ref_clock: analog display reference clock

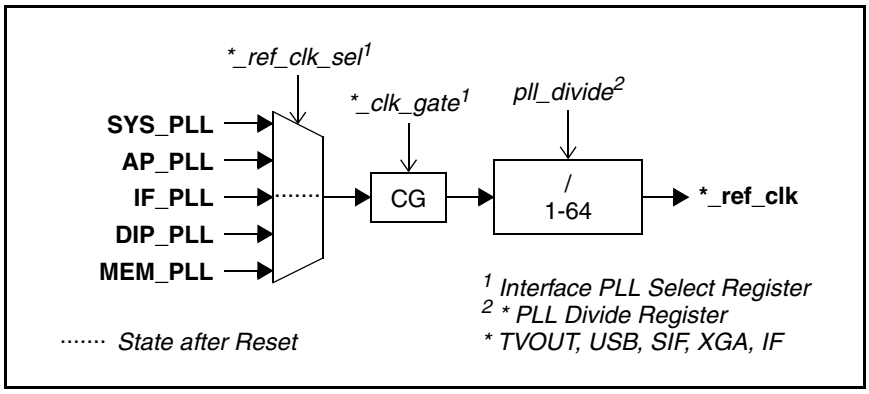


Figure 8. Peripherals Clocks

This if_ref_clk is used as a clock source for the following interfaces so that their external timing is unchanged when the system clock frequency is adjusted.

- mmc. This PLL clock is used to drive the MMC clock generator. The MMC clock generator has its own configurable software divider. The target frequency for the MMC clock has a maximum of 20-25 Mhz so the PLL should be programmed for twice that frequency as a minimum.
- uart. The PLL clock is used in the baud rate generator and front end serializer/de-serializer.
- spi. This PLL is used to drive the SPI clock generator and the SPI frontend interface.
- Audio. This PLL clock is the clock source for the NCO in the audio block. The NCO is used to generate the audio master clock. The audio master clock or a clock source connected to the master clock input is used to derive the audio bit clocks and frame clocks. The NCO operates best at higher frequencies, so a 96 Mhz setting is best for this application.
- OS timer. This PLL clock is the clock source for the timer down counter. The timer has its own configurable divider so the PLL clock frequency for this application is very flexible. RTC. This PLL clock is the clock source for the NCO in the RTC timer. The NCO operates best at higher frequencies, so a 96 Mhz setting is best for this application.

RTC. This PLL clock is the clock source for the NCO in the RTC timer. The NCO operates best at higher frequencies, so a 96 Mhz setting is best for this application.

3.3.2.3 Clock Restrictions

System Clocking Restrictions:

- Maximum ARM926EJ-S processor clock is 347.5 Mhz.
- Maximum system clock is 120 Mhz.

AP Clocking Restrictions:

- Maximum AP clock is 360 Mhz/180 Mhz (cmem_clk/ac_clk)

Other Clocking Restrictions:

- Maximum SYS_PLL frequency is 719 Mhz
- Maximum AP_PLL frequency is 632 Mhz
- Maximum MEM_PLL frequency is 704 Mhz
- Maximum DIP_MEM, IF_PLL frequency is 724 Mhz

- lf_ref_clk, xga_ref_clk, sif_ref_clk, usb_ref_clk and tvout_ref_clk maximum frequency is 133 Mhz

3.4 External Memory Interface

3.4.1 Memory Controller

The following diagram illustrates the system level architecture for the memory controller implementation.

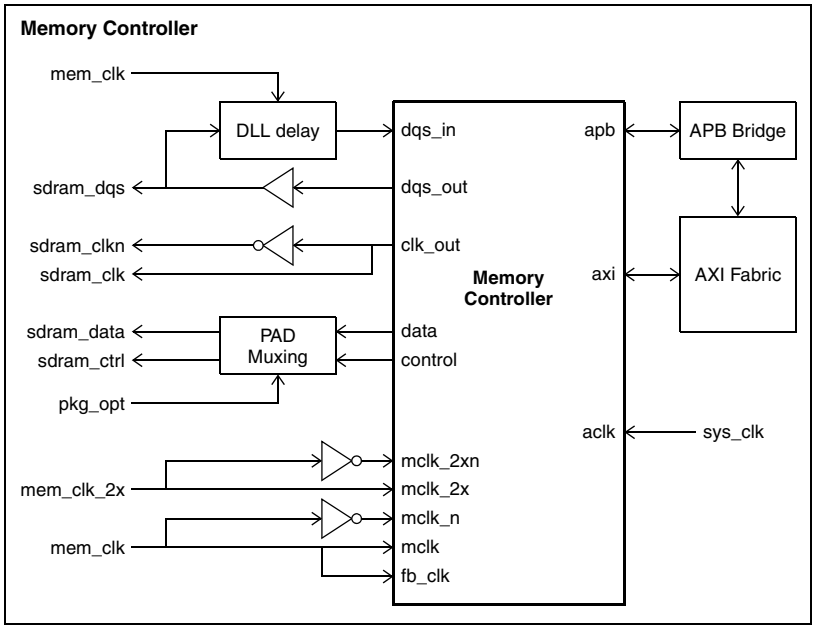


Figure 9. Memory Controller Architecture

The memory controller is configured such that the external memory interface is asynchronous to the system bus. This allows the system to run at a faster speed than the external memory and also allows for dynamic system frequency changes. By default, mem_clk is derived from mem_ref_clk. Dynamically changing this clock frequency likely means that the memory controller cannot be used for the following reasons:

- For DDR applications, the DDL delay line has a lock time whenever the “mclk” frequency is changed. Proper read operation is not guaranteed during this lock time.
- The memory controller timing registers can only be updated when the memory controller is put into a configuration state. During this configuration state, external memory accesses are not allowed.

Registers are described in 5.6, Memory Controller.

3.5 Reset

A SCP220x chip becomes operational after a hardware reset through its reset pin (resetN). This pin, when asserted, keeps the entire chip in a reset state. After the power supply voltages have stabilized, the external reset must remain asserted for at least 1 sec. Then the chip waits for the PLLs lock for 500 input reference clock periods.

Figure 10: Reset Timing below illustrates the time line of events that occur within the chip when the external reset is de-asserted.

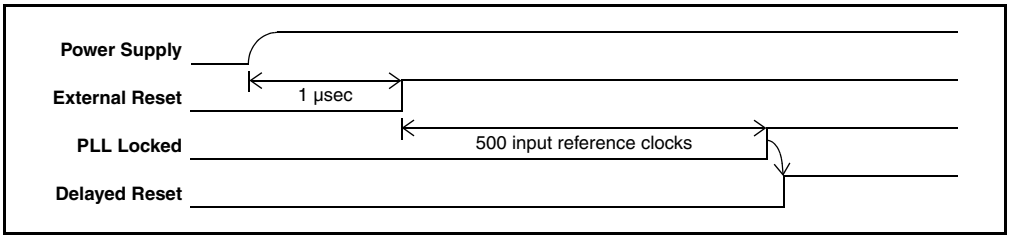


Figure 10. Reset Timing

As explained above, the internal reset architecture is controlled by an external reset pin but also by software initiated reset requests from boot loader, system registers and watchdog [see the Figure below].

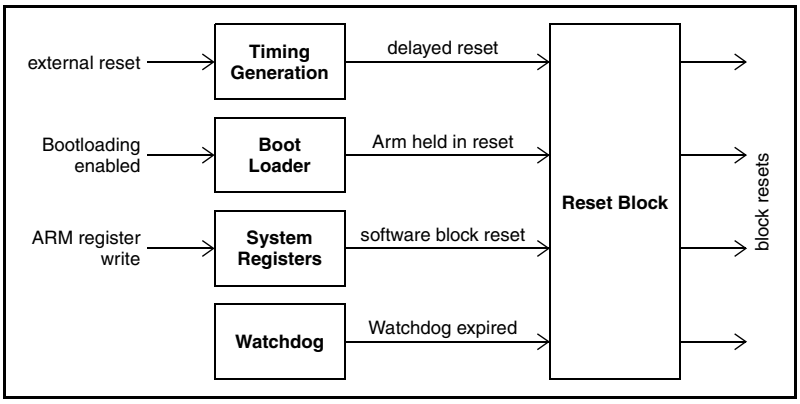


Figure 11. Internal Reset Architecture

NOTE

Most pins are in tri-state during and after reset so no damage could happen.

It is possible to reset the chip or a block through the system registers (See 5.4, Reset and Clock Gating).

3.6 Boot-up

3.6.1 Hardware Boot-up Configuration

The SCP220x chips need some pins to be set appropriately to boot and function properly.

- hw_deep_secure = 0 (low)*
- bootmode = 1 (high)*

*Suggestion: a 100 KOhms (±5%, 1/16 W) can be use as pull-up or pull-down resistor.

3.6.2 Boot-up configuration

The SCP220x chips have a configurable boot mechanism. They offer options depending of the connected hardware and boot configuration.

To configure the boot-up configurable parameters, a subset of the Display Interface Port Data bus pins (dip_data) are sampled when reset is de-asserted. The dip_data pins are tri-stated by default; this allows the pull-up and pull-down values to be sampled. The SCP220x have internal a pull-up and pull-down configuration so a default behavior is available on some pin without requiring external pull-up or pull-down resistors.

The following table describes the configurable options.

Table 4. SCP220x Boot-up Configurable Options

Configurable Feature	dip_data Pins (internal PU/PD)	Operation
Enables full-on power domain usage by default	dip_data[7] PD	0 = DISABLED* 1 = ENABLED
Enable ECC checking for NAND flash booting	dip_data[6] PD	0 = DISABLED* 1 = ENABLED
Boot Loader Mode NEED external resistor! See also section 3.6.4	dip_data[5-3] PD PU PD	000 = DRAM (debug only) 001 = SPI port generic flash 010 = Reserved* 011 = SPI port ATMEL Dataflash 100 = NAND flash
Reserved	dip_data[2]	Reserved
PLL configuration so that the internal clocks are 96 Mhz See Section 3.3	dip_data[1-0] PU PD	00 – input clk = 13 Mhz 01 – input clk = 19.2 Mhz 10 – input clk = 24 Mhz* 11 – input clk = 27 Mhz
* chip default PU / PD : Pull-Up / Pull-Down dip_data: Display Interface Port Data bus pins		

To set a level externally on the dip_data pins, you can use a 4.7 KOhm (±5%, 1/16 W) resistor as pull-up or pull-down.

3.6.3 Boot-up Timeline

The following timeline illustrates events that occur during a successful boot sequence. The internal configuration and software binary image must be resident in the SPI device or NAND flash prior to initiating the boot-up sequence.

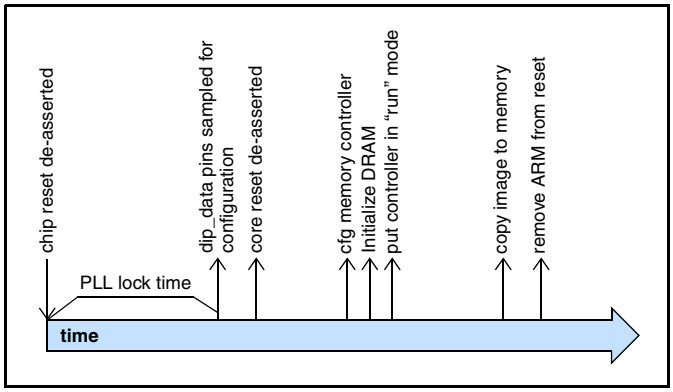


Figure 12. Boot-up Sequence Timeline

It is possible to skip some steps in the boot-up sequence. For instance, if the software image was previously downloaded and the DRAM was put into self refresh mode, then the DRAM initialization and code download steps would not be necessary.

There are several possible ways to load the code into the SCP220x, they are presented in 3.6.4, Hardware Boot Load Modes.

For all cases, the format of the downloaded data contains both the configuration information as well as the software image. The data format is as follows:

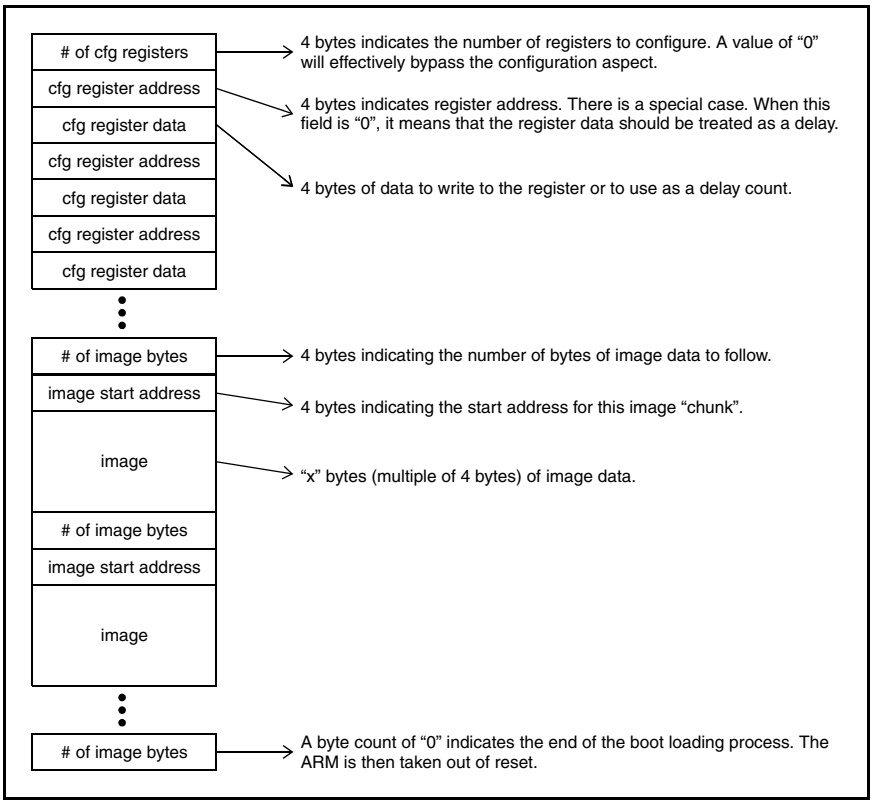


Figure 13. Hardware Boot Loader Load Description

3.6.4 Hardware Boot Load Modes

The hardware boot loader block facilitates code loading from different external interfaces.The external interface gets data from an external device and the boot loader block moves the data from the receive FIFO to DRAM memory

While the Hardware Boot Loader is operating, the ARM926EJ-S processor is held in reset so that it does not start executing code until the complete program store is in place. When the byte counter expires, indicating all code has been copied, the boot loader indicates to the reset block that the ARM926EJ-S processor can be removed from reset.

Possible boot loader configurations, as specified by the downloaded configuration information, are identified in the table below.

Table 5. Configuring Boot Load Using dip_data[5:3] Pins

dip_data[5:3]	Description
b000	This setting will not invoke the boot loader and the ARM926EJ-S processor will be removed from reset immediately. This is a debug mode of operation. Code must be written to memory through some other means (ie. JTAG Port).
b011	Code is resident in a serial NAND flash connected to the SPI port. The serial Flash Memory is an ATMEL DataFlash memory that supports the “continuous array read” command (0xe8).

Table 5. Configuring Boot Load Using dip_data[5:3] Pins

b001	Code is resident in a serial NAND flash connected to the SPI port. The serial Flash Memory is an industry standard memory that supports the “read data bytes” command (0x03).
[b010]	[RESERVED]
b100	Code is resident in NAND flash. The NAND flash block read sequence is: After reset is de-asserted, the bootloader will issue a “reset” command (“ff”) followed by a 25 µsec delay. The boot loader then issues the page read command (“00”) and 5 bytes of address (all “0”). This is followed by a read confirm command (“30”). Before proceeding further, a 50 µsec delay occurs. A 2 Kbyte page is then read. If ECC is enabled four 512 byte page reads are issued.

If booting from NAND Flash, there is an optional ECC checking mode that may be enabled via a software register. If ECC checking is enabled, the boot_loader checks for errors after a block is read from the device. Upon error detection, the boot loader keeps the ARM926EJ-S processor in reset.

For NAND Flash, the data must be organized in the 2K Flash sector as follows.

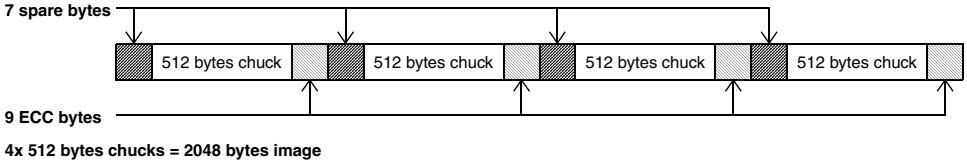


Figure 14. NAND Flash Data Organization

3.7 Low Power Configurations

The SCP220x chips offer three power consumption reduction features described below: voltage islands, clock gating and processor standby. They can be implemented independently for maximum control.

3.7.1 Voltage Islands

The SCP220x provides two voltage islands: Low Power Audio/Video domain and the IC Core domain (see Figure 2., SCP220x Internal Architecture).

The Low Power Audio/Video domain is powered through the VDD_LP pin. This domain allows for processing at reduced power consumption. The ARM926EJ-S processor runs along with some of the blocks offering some processing, audio and display capability (digital out only, see Figure 30., Display Sub-System (DSS) Internal Architecture). Apex is not running and there is no input of images.

The IC Core domain is powered through the VDD_CORE pin. This domain contains the high performance blocks such as the APEX, SIF and USB.

Low power consumption mode is achieved by removing power to the IC Core (VDD_CORE) by an external device (i.e. power MOSFET) optionally controlled via a SCP220x GPIO pin. CogniVue Reference Design Kit (RDK) has this low power option implemented. NOTE: it is recommended that all the other power lines be connected at all times even if the corresponding blocks are not active.

3.7.2 Clock gating

It is possible to idle some blocks by gating their clocks. Clock gating is achieved through registers, see 5.4, Reset and Clock Gating. Note that this functionality is provided by the SDK, direct register setting is recommended only for custom bootloader code as SDK is not available at this stage.

3.7.3 Processor Standby

Another way to save power is to place the ARM926EJ-S processor in standby when no processing is required before an event.

4 Interconnect and Communication

4.1 NAND Flash Interface

The SCP2201 and SCP2207 products have a NAND flash interface for connectivity to an external NAND flash device. The following list details specific NAND flash features:

- 8-bit datapath
- Software configurable external control signal timing
- Incoming and outgoing datapath implemented using FIFOs
- Software controlled command and page address
- Read/Write datapath that bypasses the FIFO and allows direct access
- Configurable page size
- NAND flash read and write algorithms are software driven
- Optional hardware ECC support; a simple ECC (1bit correct, 2 bit detect) as well as a Reed Solomon ECC algorithm (4 bit correct)
- Supports up to 4 external chip selects

4.1.1 NAND Flash connection

The following figure shows the connection between the SCP2201 or SCP2207 and a typical external NAND flash device. It should be noted that the „ry_by. signal is not a dedicated pin on the SCP2201 or SCP2207. Instead this connection, required for command status, is made to a GPIO. Alternatively, a software managed polling routing may be used to determine when various commands are completed.

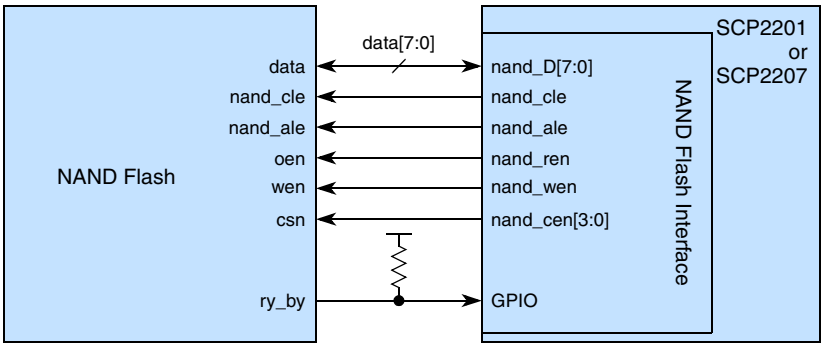


Figure 15. NAND Flash Connectivity

The following table describes the NAND Flash Interface pinout for the SCP220x

Table 6. NAND Flash Interface

Signal	Alternate Function	Pin Direction	Pin Description
nand_D[7:0]	gpio[81:74] or mmcplus_data[7:0]	Bi-dir.	NAND data bus or alternate function
nand_cle	gpio[11]	Bi-dir.	Command latch enable or alternate function
nand_ale	gpio[10]	Bi-dir.	Address latch enable or alternate function
nand_cen[0]	gpio[12] or mmcplus_clk	Bi-dir.	Chip select or alternate function
nand_cen[1]	gpio[70] or mmcplus_cmd	Bi-dir.	Chip select or alternate function
nand_cen[2]	gpio[71] or spi_Rxd1	Bi-dir.	Chip select or alternate function
nand_cen[3]	gpio[72] or spi_Rxd2	Bi-dir.	Chip select or alternate function
nand_ren	gpio[14]	Bi-dir.	Read enable or alternate function
nand_wen	gpio[13]	Bi-dir.	Write enable or alternate function

4.1.2 NAND Flash Hardware Description

The hardware implementation for the NAND Flash block is as shown in the following diagram.

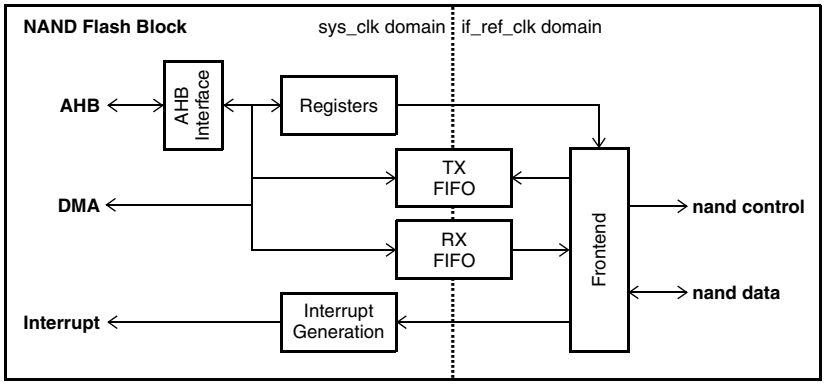


Figure 16. NAND Flash Hardware Architecture

The NAND Flash front-end contains a state machine that drives the external interface based on the configuration settings from the software interface.

The front-end block issues commands, address and data as directed by the particular software configuration. The transmit and receive fifos provide buffer space such that the internal bus-bandwidth required to move data is minimized because AMBA AHB bursts can efficiently move data minimizing overall bus bandwidth usage.

The following diagrams illustrate what the external waveforms look like and also illustrate any software configurable parameters that control the external signals.

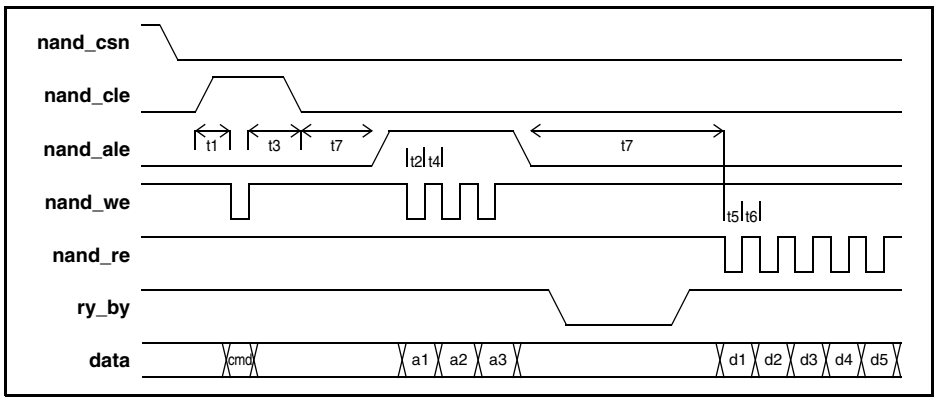


Figure 17. NAND Flash Page Read Cycle

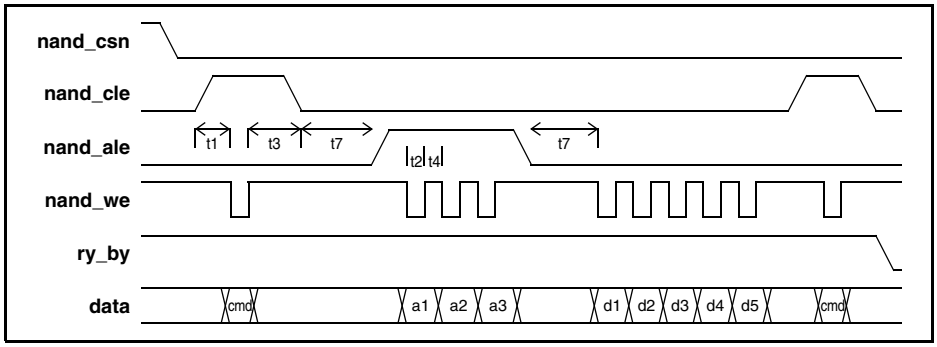


Figure 18. NAND Flash Page Write Cycle

It should be noted that ry_by is not a dedicated pin. Instead it has to be connected to a GPIO or else software must poll the NAND Flash device to determine when various commands are completed.

NAND Flash has a page and block structure as illustrated in the following diagram.

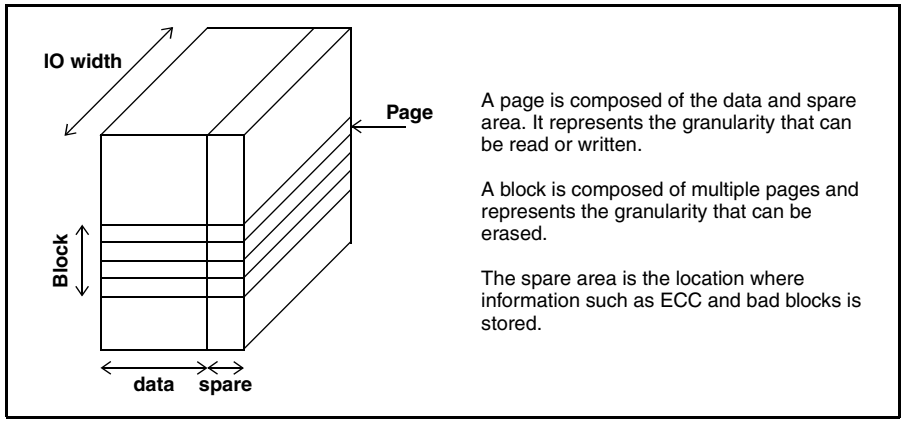


Figure 19. NAND Flash Page and Block Structure

The software configurable parameters in the NAND configuration register allow for a lot of flexibility when programming and reading NAND Flash. There are two fields, page_size and spare size. As an example: lets say that the data space in the NAND is 512 bytes and the spare space is 16 bytes (this is the case Toshiba 128Mx8 device).

Case 1 - ECC parity is enabled and only a single page is going to be written. In that case set page_size=512, ecc_ena = 1, spare_size=0. The interface will write the page of data as it fills into the FIFO, generate parity during this process and append the 6 bytes of ECC to the end of the data stream before interrupting indicating completion. If it was a read, the interface would have read 512 bytes of data and placed them in the FIFO re-generating a new ECC during this process. The interface would have then read the 6 bytes of ECC from the NAND and made the parity check available to software before interrupting indicating completion.

Case 2 - ECC parity is enabled and multiple pages are read. In this case set page_size=512, ecc_ena = 1, spare_size=10. Operation will proceed as described above except after the ECC has been read, 10 dummy bytes are read effectively setting the address pointer to the beginning of the next block of data. The Flash will indicate it is ready for the next block via it.s RY/BY pin at which time another "kick" will initiate another read process. The "command" and "address" aspect of the cycle do not need to be repeated.

Case 3 - ECC parity is not required and a single page is going to be written. In this case set page_size=512, ecc_ena = 0, spare_size=0. The interface will only write the page of data to NAND prior to interrupting.

The description of the control registers for the NAND Flash interface can be found at 5.7, NAND Interface Registers Description.

4.2 UART

The SCP220x has two UARTs, referred to as UART and UART1, used for incoming or outgoing data paths. Note that uart1_Rx and uart1_Tx signals of UART1 are available via shared I/Os and this UART does not support CTS/RTS modem signals.

- The UARTs have the following features:
- Asynchronous interface
- Programmable baud rate
- Parity and framing error detection with indication via interrupts
- Echo, local loopback and remote loopback diagnostic modes
- Single start bit, 8-bit character length, programmable stop bits (1 or 2), programmable parity (even, odd or none)
- Independent receive and transmit FIFOs
- The primary UART supports CTS/RTS modem signals for hardware flow control.

The following table lists the SCP220x pin information for the UART Interface.

Table 7. UART Interface

Signal	Alternate Function	Pin Direction	Pin Description
uart_Rx	gpio[23]	Bi-dir.	UART serial receive data or alternate function
uart_Tx	gpio[22]	Bi-dir.	UART serial transmit data or alternate function
uart_cts	gpio[82] or spi_CS1	Bi-dir.	Clear to send modem signal or alternate function
uart_rts	gpio[83] or spi_CS2	Bi-dir.	Request to send modem signal or alternate function

UART1 signals are accessible only as alternate functions. These signals are listed in 4.12.1, GPIO and Alternate Function List.

4.2.1 UART Hardware Description

The following diagrams illustrate the various loopback modes that are supported.

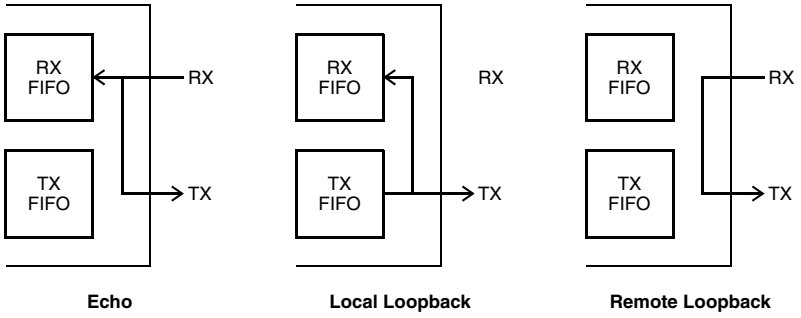


Figure 20. Uart Diagnostic Loopbacks

The following UART protocol is supported with configurability for the stop and parity bits.

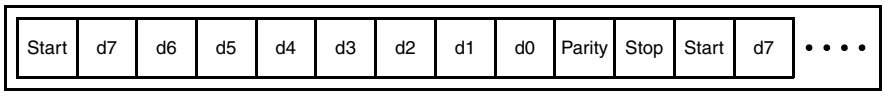


Figure 21. Uart Protocol

The hardware implementation for the UART block is as shown in the following diagram.

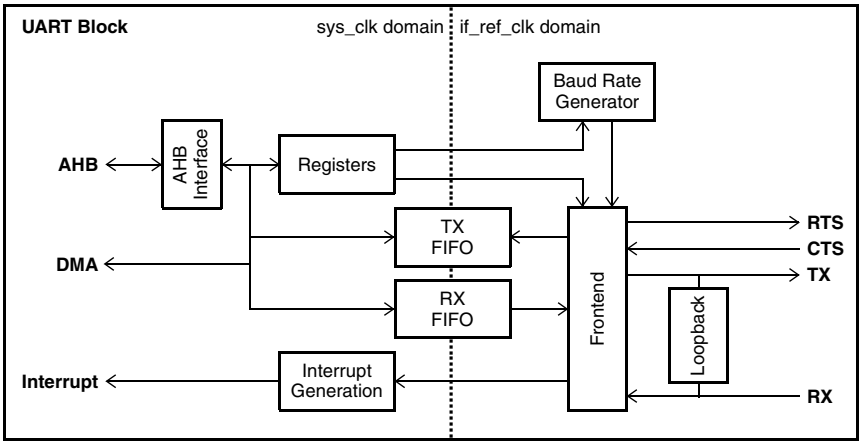


Figure 22. Uart Hardware Architecture

The Baud rate generator uses the software configurable baud rate register as a divider to generate the receive and transmit clock enables.

The FIFOs are identical async fifos. The frontend block reads 8 bit wide data out of the transmit fifo, serializes it, adds start,stop and parity bits and transmits it at the programmed baud rate. Similarily the frontend block receives serial data and forms an 8 bit word. Start,stop and parity bits are stripped off. Parity errors and frame errors are checked and generate interrupts.

The modem signals, when enabled, provide flow control to the hardware. The Clear To Send (CTS) input modem signal indicates to the transmit state machine whether or not to send out a character. The receive state machine generates the ready to send output when there is an appropriate amount of space available in the fifo.

The description of the control registers for the UART interface can be found at 5.8, UART Control Registers.

4.3 SPI

The Serial Peripheral Interface (SPI) provides an alternate data path to and from the SCP220x. The SCP220x device has two SPI blocks on board referred as SPI and SPI1. The SPI blocks in the SCP2201 and SCP2207 devices each have four chip selects (spi_CS, spi_CS1, spi_CS2, spi_CS3), four serial receive data signals (spi_Rx, spi_Rx1, spi_Rx2, spi_Rx3) and a serial transmit data signal (spi_Tx). (Note that three of the chip selects and three receive signals of SPI are accessible via shared I/Os). For more details, refer to 4.12.1, GPIO and Alternate Function List. SPI1 has a dedicated clock, a chip select, and transmit and receive I/Os as shown below in Table 8.

This interface is compatible with the Motorola SPI specification and provides the following features:

- Four wire synchronous full duplex interface using a clock, chip select, serialized receive data and serialized transmit data
- Configurable as master or slave. The master sources the clock and chip select and the slave sinks these pins.
- 128-byte transmit FIFO and 128-byte receive FIFO
- Programmable clock rate (master mode only)
- Programmable frame size
- Supports “continuous” mode of operation
- Programmable clock phase (SPH) and polarity (SPO)

The following table lists the SCP220x pin information for the SPI.

Table 8. SPI Signals

Signal	Alternate Function	Pin Type	Pin Description
spi_Clk	gpio[29]	Bi-dir.	SPI serial clock or alternate function
spi_CS	gpio[28]	Bi-dir.	SPI slave select or alternate function
spi_Tx	gpio[26]	Bi-dir.	SPI serial transmit data or alternate function
spi_Rx	gpio[27]	Bi-dir.	SPI serial receive data or alternate function
spi1_Clk	mp2ts1_clk	Bi-dir.	SPI1 serial clock or alternate function
spi1_CS	mp2ts1_sync	Bi-dir.	SPI1 slave select or alternate function
spi1_Tx	mp2ts1_valid	Bi-dir.	SPI1 serial transmit data or alternate function
spi1_Rx	mp2ts1_data	Bi-dir.	SPI1 serial receive data or alternate function

The clock phase and clock polarity configurability allow for four modes of operation. The clock phase controls which edge of the clock that the transmitter transitions data and the receiver samples data. Transmission and reception of data operate on opposite edges of the clock. The polarity controls whether or not the clock is high or low during the inactive period. The following four diagrams illustrate the operation for these four modes for a byte length transaction.

When SPH=0, data is transmitted on the falling edge and sampled on the rising edge. When SPH=1, data is transmitted on the rising edge and sampled on the falling edge.

When SPO=0, the clock is low during inactivity (ie. chip select de-asserted). When SPO=1, the clock is high during inactivity.

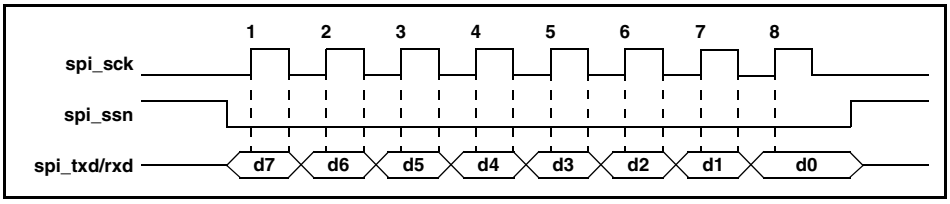


Figure 23. SPI Clock Phase/Polarity - SPH=0, SPO=0

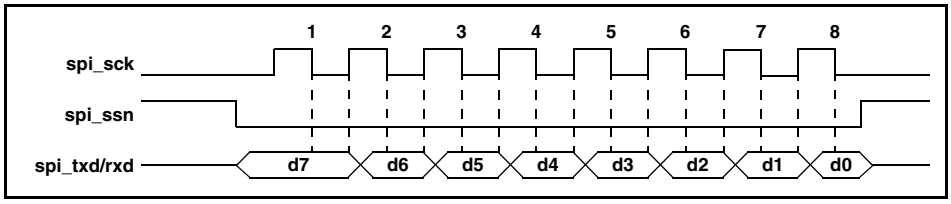


Figure 24. SPI Clock Phase/Polarity - SPH=1, SPO=0

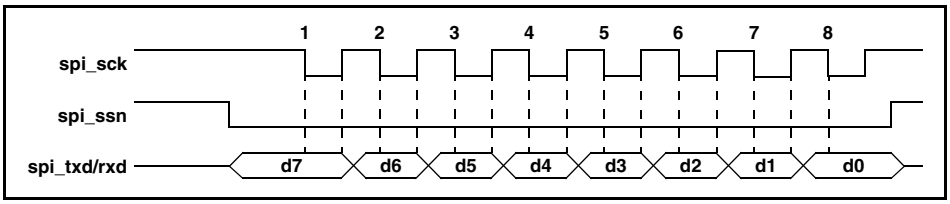


Figure 25. SPI Clock Phase/Polarity - SPH=0, SPO=1

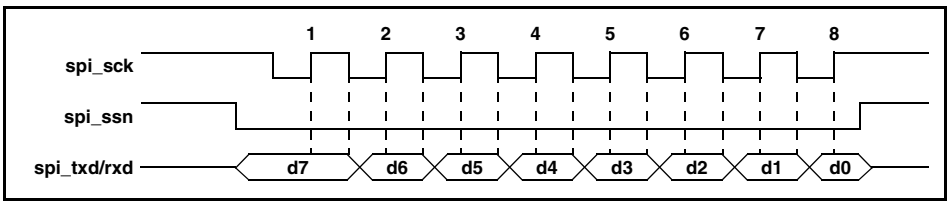


Figure 26. SPI Clock Phase/Polarity - SPH=1, SPO=1

4.3.1 SPI Hardware Description

The hardware implementation for the SPI block is as shown in the following diagram.

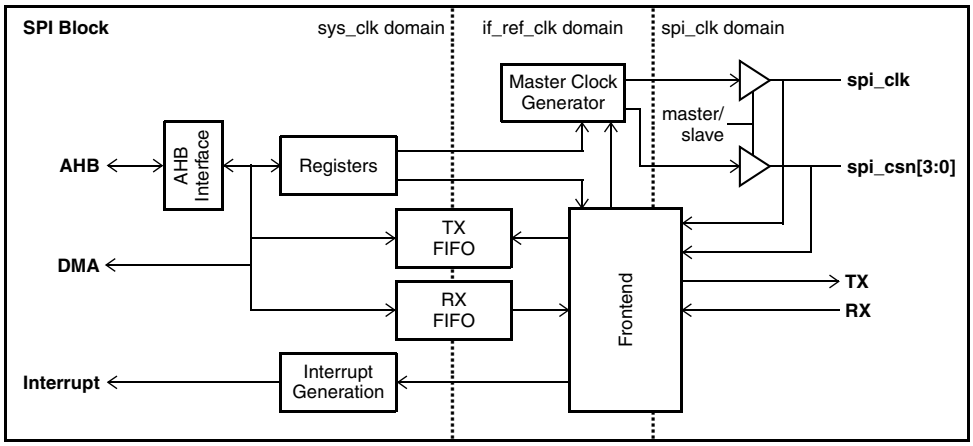


Figure 27. SPI Hardware Architecture

The frontend sub-block is operating in the spi_clk domain whether or not the SPI interface is configured as a master or slave. The clock is tapped off the external pad so that the internal timing

requirements within the frontend are identical for master and slave operation. The frontend block is primarily a serialization and de-serialization engine that pops and pushes data from/to the fifos. When the frontend gets a clock edge when selected, its starts serializing transmit data and de-serializing the receive data. When a “word” has gone through the serialization process, the frontend pushes the word into the receive fifo and pops the transmit fifo. The process then repeats until the chip select is de-asserted.

The master clock generator is only used when the SPI block is configured as a master. The clock generator creates an external SPI clock that is a programmable divider of the interface PLL clock. The clock generator must also create the master chip select since it gets asserted before the external SPI clock starts wiggling. The chip select and external SPI clock start a transaction when a data event has occurred in the FIFOs. This most likely is the event of the transmit FIFO containing data. A FIFO not empty signal traverses clock domains and is used to kick activity within the clock generator. Similarily, when the transmit fifo becomes empty (and the last data has been transmit/received), the clock generator de-activates the external clock and chip select.

The receive and transmit FIFOs are asynchronous with one side in the internal system clock domain and the other side in the interface reference clock domain. The read and write pointers cross clock domains through the technique of converting the pointer to a gray code, double sampling and converting back to a pointer. The limitation that this imposes is that the databus size must not be dynamic. The databus size is software configurable but cannot dynamically change while the fifo is in use.

The description of the control registers for the SPI interface can be found at 5.9, SPI Registers.

4.3.2 SPI Port Timing

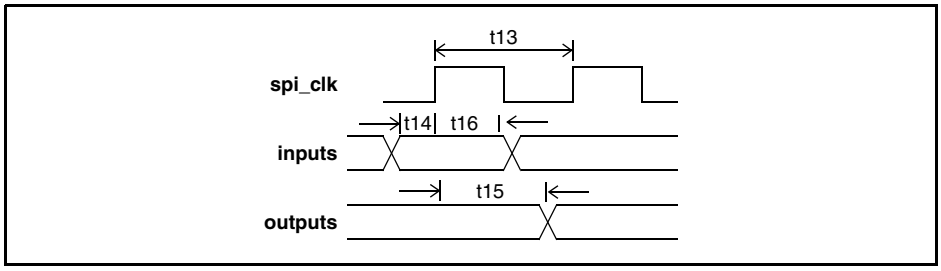


Figure 28. SPI Port Timing

Table 9. SPI Port Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
SPI port clock frequency (master)	t13			50	MHz
SPI port input setup time (master)	t14 (3.0 V)	4.3			ns
SPI port input hold time (master)	t16 (3.0 V)	0.25			ns
SPI port output delay time (master)	t15 (3.0 V)			10.3	ns

4.4 Sensor Interface (SIF)

The Sensor Interface receives data from one of two sources – the external sensor or from memory. Supported format(s) of input data from the sensor:

- YUV422 stream

NOTE

The Sensor Interface block may be programmed to accept YUV422 data in UYVY, YUYV, VYUY and YVYU formats. Input image sizes up to 10M-pixels are supported at clock frequencies up to 160 MHz.

Output image formats supported by the Sensor Interface block are:

- YUV422 Stream
- YUV420 Planar

The Sensor Interface also provides the following functionality:

- scale down:
- average mode scaling by: 1, 1/2, 1/4 and 1/8
- decimation mode scaling by : horizontal and vertical decimation
- adaptive luminance using histogram table build or gamma correction
- image effects: grey scale, sepia, negative, emboss, sketch
- edge enhancement
- image smoothing using a LPF with 9 taps for luminance and 5 taps for chrominance coefficients
- WOI (window of interest) used for cropping the input images

The SIF is controlled via the SDK under the Sensor Device Interface (SDI).

The following table lists the SCP220x external pinout of the Sensor Interface (SIF).

Table 10. Sensor Interface (SIF) External Pinout

Signal	Alternate Function	Pin Direction	Pin Description
sensor_D[9:0]	-	Input	Sensor data
sensor_pclk	-	Input	Sensor pixel clock
sensor_rclk	-	Input	Sensor horizontal sync signal
sensor_fclk	-	Input	Sensor vertical sync signal
sensor_clkout	gpio[1]	Bi-dir.	Sensor source clock or alternate function
sensor_fodd	gpio[52]	Bi-dir.	Field (odd, even) or alternate function

Table 10. Sensor Interface (SIF) External Pinout

sensor_gpio	gpio[53]	Bi-dir.	Sensor GPIO or alternate function

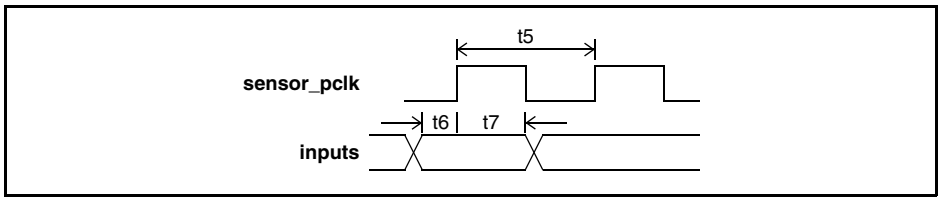


Figure 29. Sensor Interface Timing

Table 11. Sensor Interface Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
Sensor pixel clock frequency	t5			160	MHz
Sensor input setup time	t6(3.0V)	3.2			nsec
	t6(1.8V)	3.2			nsec
Sensor input hold time	t7(3.0V)	0.75			nsec
	t7(1.8V)	0.75			nsec

4.5 Display Sub-System (DSS)

The Display Sub-System (DSS) has two types of outputs through the Display Interface Port:

- Digital, for LCD (1x) or CPU-like (x4) interfaces
- Analog, generates NTSC/PAL composite signal

To minimize the load on the processor and APEX, there are two advanced resize/format blocks Bitblt and Bitblt mini.

The DSS has mixed voltage islands; the figure below gives more precision about the correspondence:

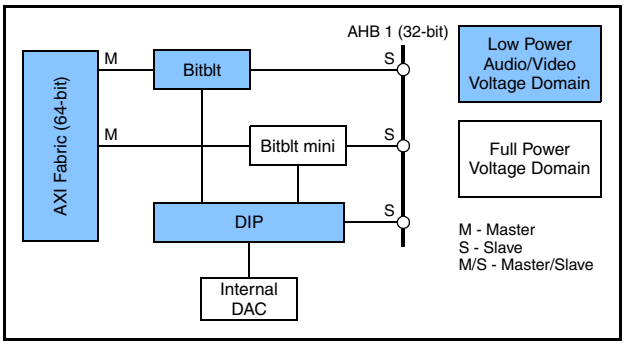


Figure 30. Display Sub-System (DSS) Internal Architecture

4.5.1 Display Interface Port (DIP) and TV Output

The Display Interface Port (DIP) interfaces the SCP2201 or SCP2207 to an external video/display device such as an LCD or a television.

The external display controller in the SCP2201 and SCP2207 devices may be a TFT LCD or four CPU-like interface devices

The DIP has the following features:

- Color format resizing (RGB24 -> RGB666/RGB565; RGB666->RGB24/RGB565; RGB565->RGB24/RGB666, YUV422->YUV444)
- Display Bus Interface (DBI): Drives an LCD, LCD Controller or CPU-type interface. Four chip selects are available to support up to four devices including any combination of LCD Controllers and/or other devices with CPU-type interfaces.
- Timing Interface: Drives an external video device (RBG LCD) with hsync/vsync/blank signals; when this mode is enabled, only TFT LCD devices may be used. This interface supports up to WVGA resolution.
- TV Interface: This analog interface derives timing information from an internal NTSC/PAL video encoder to drive video data at correct intervals. Maximum resolution supported is 640x480 (VGA).

The following figure shows a configuration with single TFT-RGB LCD and a TV connection to the DIP. In this case, the internal SCP220x video encoder and DAC are used to derive the analog TV signal.

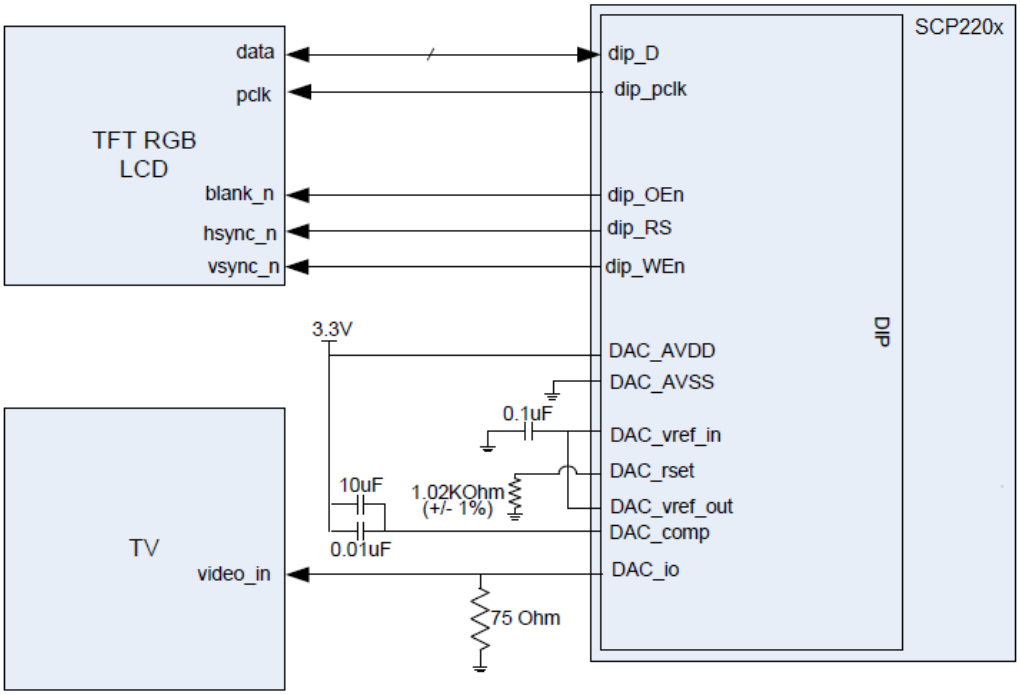


Figure 31. TFT-RGB LCD and TV Connected to DIP (Using Internal DAC)

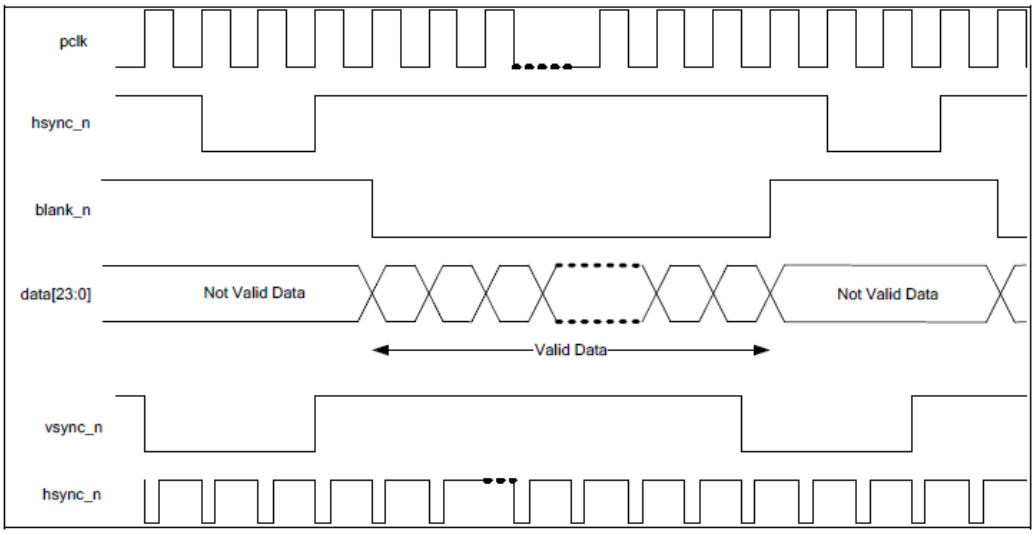


Figure 32. RGB-type Display Waveform Diagram

The following figure shows a configuration with dual LCDs and a TV connection to the DIP. In this case, the internal SCP220x video encoder and DAC are used to derive the analog TV signal.

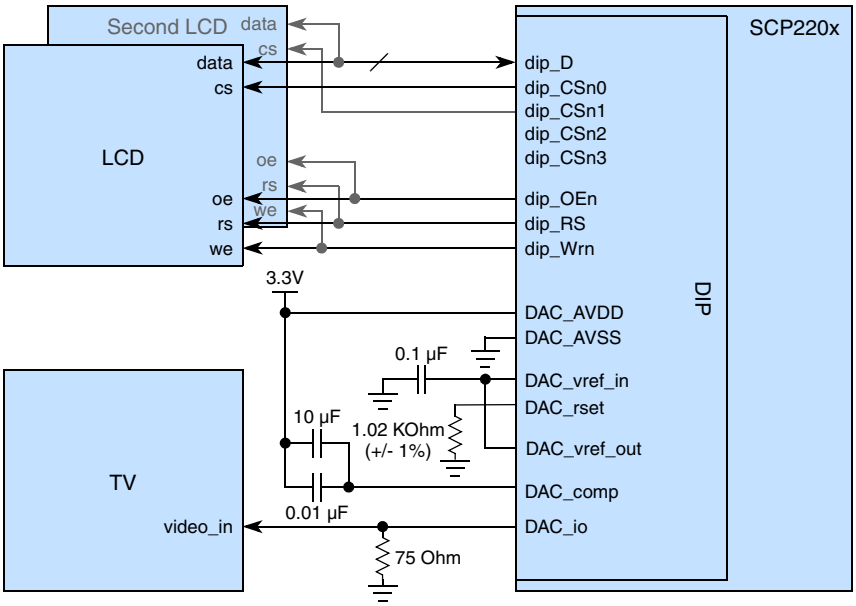
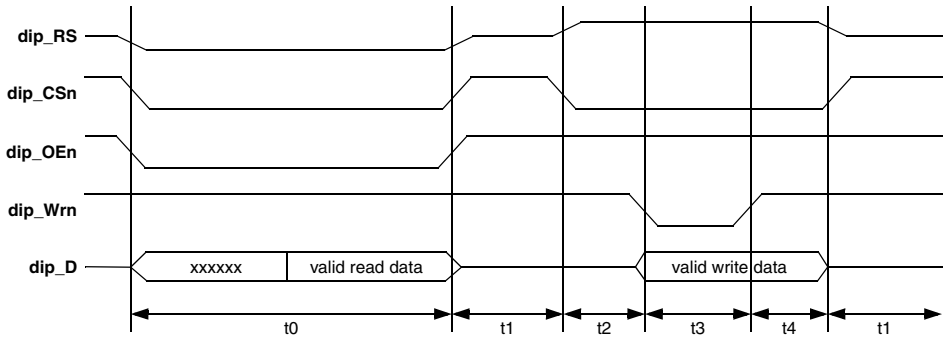


Figure 33. CPU-Type LCD and TV Connected to DIP (Using Internal DAC)

The figure below illustrates timing for DBI connectivity to a CPU-type interface.



Timing parameters depend on CPU clock and modifying the CPU clock frequency will alter the duration of the data transfer.

t0 - Width of the overall read cycle + 1
t1 - Time between transactions + 1
t2 - Time up to start of Wrn pulse and start of write data + 1
t3 - Width of Wrn pulse + 1
t4 - Extend write data after Wrn pulse + 1

Figure 34. DBI to CPU-type Display Waveform Diagram

The following table lists the SCP2201 and SCP2207 pin information for the DIP. The DAC signals correspond to the TV Output feature available for all SCP220x products.

Table 12. Display Interface Pins

SCP2201, SCP2207 Signal	Alternate Function	Pin Direction	Pin Description
dip_data[23]	gpio[51] or uart1_txd	Bi-dir.	Digital video data or alternate function
dip_data[22]	gpio[50] or uart1_rxd	Bi-dir.	Digital video data or alternate function
dip_data[21]	gpio[49]	Bi-dir.	Digital video data or alternate function
dip_data[20]	gpio[48]	Bi-dir.	Digital video data or alternate function
dip_data[19]	gpio[47]	Bi-dir.	Digital video data or alternate function
dip_data[18]	gpio[46]	Bi-dir.	Digital video data or alternate function
dip_data[17]	gpio[3] or scl_sec	Bi-dir.	Digital video data or alternate function
dip_data[16]	gpio[4] or sda_sec	Bi-dir.	Digital video data or alternate function
dip_data[15:0]	-	Bi-dir.	Digital video data bus
dip_pclk	gpio[5]	Bi-dir.	Digital video pixel clock or alternate function
dip_OEn	gpio[15] or dip_blank	Bi-dir.	Digital video output enable or alternate function
dip_RS	dip_hsync	Output	Digital video register select or alternate function
dip_Wrn	dip_vsync	Output	Digital video write enable or alternate function
dip_CSn0	-	Output	Digital video chip select 0

Table 12. Display Interface Pins

dip_CSn1	-	Output	Digital video chip select 1
dip_CSn2	gpio[24]	Bi-dir.	Digital video chip select 2 or alternate function
dip_CSn3	gpio[25]	Bi-dir.	Digital video chip select 3 or alternate function
dip_cpu_vsync	gpio[54]	Bi-dir.	External synchronization frame pulse or alternate function
DAC_comp	-	Analog Output	Analog output of the DAC; signal can drive 1.0 Vpp on 75 ohm load
DAC_vref_out	-	Analog Output	Voltage reference output. This output delivers 1.140 V reference voltage from cell. It is normally connected to the VREFIN pin.
DAC_rset	-	Analog In/Out	An external resistor Rset connecting DAC_rset pin to AVSS adjusts the magnitude of the DAC full-scale output current. Recommended setting is 1.02 KOhm with 1% tolerance.
DAC_vref_in	-	Analog Input	Reference voltage input. It is suggested to place 0.1 µF ceramic capacitor between this pin and AVSS pin externally.
DAC_io	-	Analog Output	Analog output pin (with drive strength) to which a resistor and capacitor is attached to ground to set the output current of the DAC

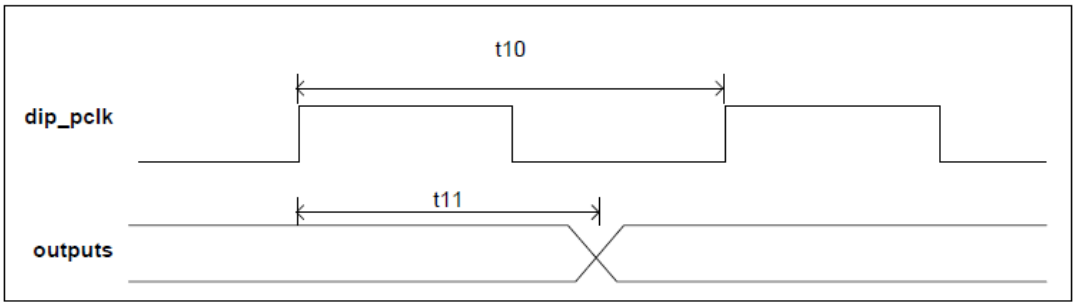


Figure 35. DIP (TFT-RGB-type) Port Timing

Table 13. DIP (TFT-RGB-type) Port Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
DIP port pixel clock frequency	t10			70	MHz
DIP port output delay time	t11 (3.0 V)			5.5	ns

NOTE

DIP's CPU-type protocol does not have a reference clock to determine setup/hold time for dip_data[17:0] (For CPU Read transaction). Timing, as shown in Figure 35, is dependent on individual CPU-type LCD, configurable within DIP.

4.5.2 Bitblt and Bitblt mini

Bitblt and Bitblt mini are supported through the SDK under the Graphic Display Interface (GDI).

4.6 USB 2.0 HIGH SPEED

The USB Interface has the following features:

- USB 2.0 HIGH SPEED compliant
- SCP2201 and SCP2207 devices support USB OTG
- USB 2.0 PHY is integrated on chip
- Supports high-speed (480 MHz), full speed (12 MHz), and low speed (1.5 MHz) operation
- Supports seven physical endpoints - one control and six endpoints configurable as IN or OUT. The IN/OUT endpoints are software configurable as bulk, isochronous, interrupt or control

The following table lists the SCP220x pin information for the USB Interface.

Table 14. USB Interface

Signal	Alternate Function	Pin Direction	Pin Description
usb_phy_id	-	Analog USB pad	Indicates A or B cable
usb_phy_vbus	-	Analog USB pad	Vbus power monitor input. This is a 5 V signal (+/-10%) with a max value of 5.5 V .
usb_phy_Plus	-	Analog USB pad	USB data plus
usb_phy_Minus	-	Analog USB pad	USB data minus
usb_phy_res	-	Analog USB pad	External resistor of 8.2 K ± 1% should be connected from here to ground
utmiotg_drvvbus	gpio[73]	Bi-dir.	Externally controls power source for USB VBUS voltage or alternate function;

The usb_phy_vbus signal monitors the 5.0 V VBus signals for USB 2.0 HIGH SPEED. The following figure illustrates USB interface connectivity with the host.

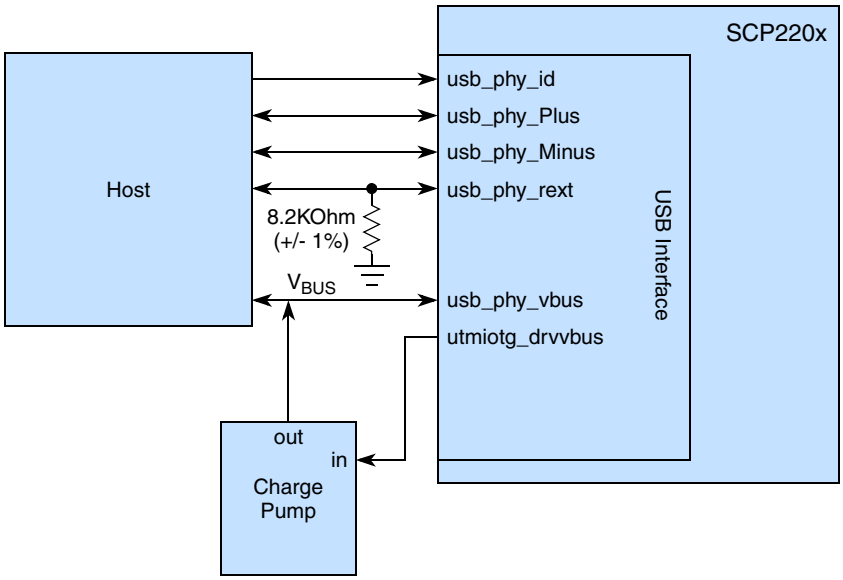


Figure 36. USB/Host Connectivity

4.7 Audio Interface

The Audio Interface provides a direct connection to either voice quality or high-quality audio ADC/DAC. The Audio Interface has the following features:

- Supports I2S or AC97 interface protocol
- Supports full duplex data path
- Separate receive and transmit FIFOs
- Software configurable hardware interface to support a variety of I2S and AC97 applications

Figure 35 shows the SCP2201/SCP2207 audio interface connections to an audio DAC.

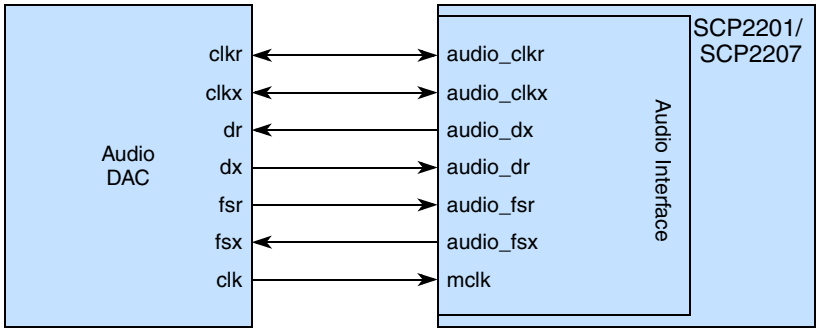


Figure 37. Audio Interface

The following lists the SCP2201 and SCP2207 pin information for the Audio Interface.

Table 15. Audio Interface Pins

Signal	Alternate Function	Pin Direction	Pin Description
audio_clkr	gpio[16]	Bi-dir.	Audio receive bit clock or alternate function
audio_clkx	gpio[19]	Bi-dir.	Audio transmit bit clock or alternate function

Table 15. Audio Interface Pins

audio_dr	gpio[17]	Bi-dir.	Audio receive data or alternate function
audio_dx	gpio[20]	Bi-dir.	Audio transmit data or alternate function
audio_fsr	gpio[18] or pwm2_out	Bi-dir.	Audio receive frame clock or alternate function
audio_fsx	gpio[21]	Bi-dir.	Audio transmit frame clock or alternate function
mclk	-	Bi-dir.	Audio clock source from external audio DAC.

There is a large degree of flexibility within this interface that allows support for the following applications:

- AC97 controller sourcing the bit clock
- AC97 controller sinking the bit clock
- I2S controller with a common clock and sync for both receive and transmit. Clock and sync are configurable as source or sink.
- I2S controller with a separate clock and sync for the receive and transmit. Clocks and syncs are configurable as source or sink.

The following sample waveforms illustrate the configurability available within this interface. The waveforms also identify what timing aspects are software configurable.

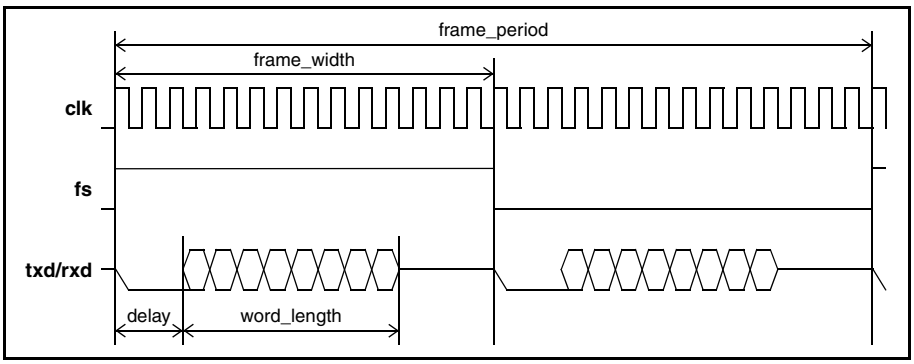


Figure 38. I2S Stereo Transmission

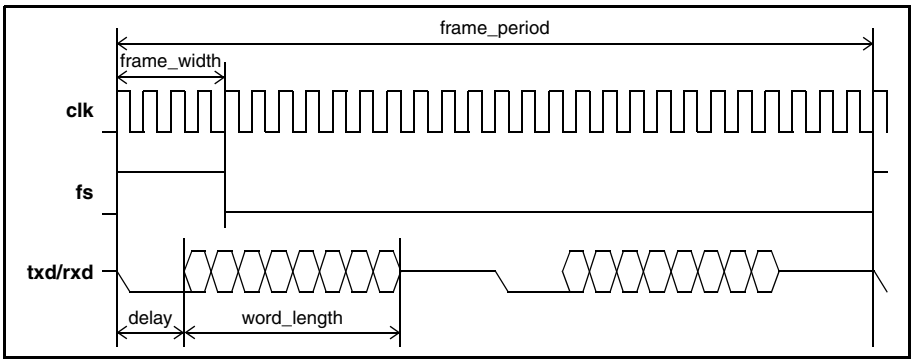


Figure 39. I2S Mono Transmission

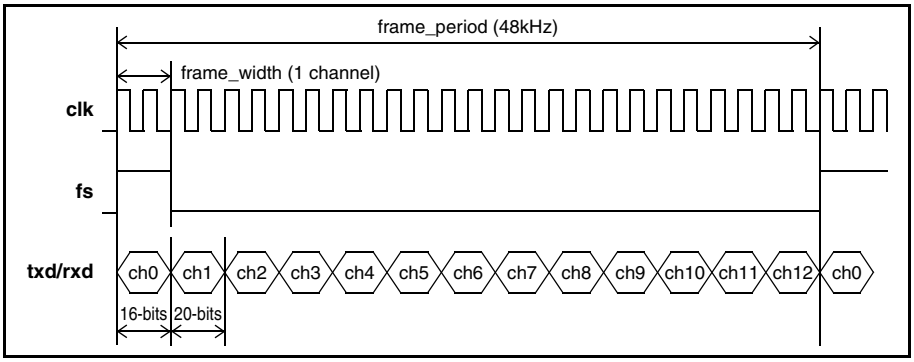


Figure 40. AC97 Mode of Operation

4.7.1 Audio Interface Hardware Description

The hardware implementation for the Audio block is as shown in the following diagram.

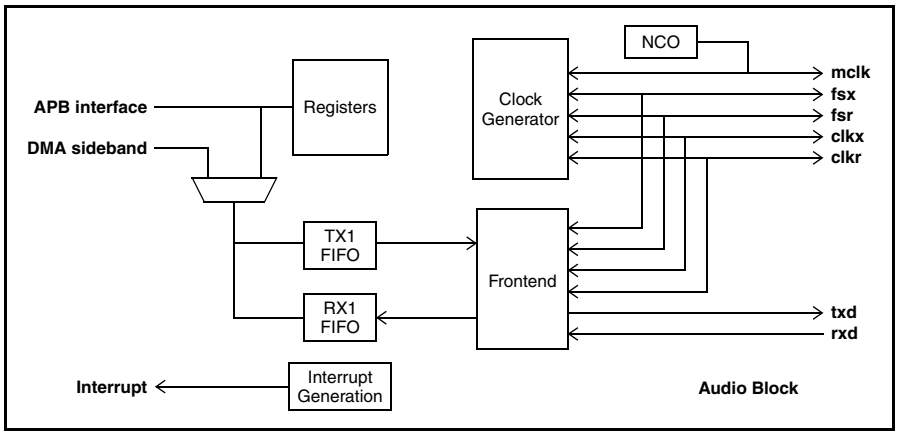


Figure 41. Audio Hardware Architecture

The clock generator block takes the software configuration from the registers block and divides the master clock down appropriately to produce the bit clocks for the transmit and receive. Software configuration also controls the PAD enables at the top level. If the ASIC sources the bit clocks, the PADs are enabled otherwise the bit clocks are inputs and are driven by an external source. Irregardless of the bit clock source, the bit clocks re-enter the audio block and are used as clocks in the frontend block. Software configuration select either the positive edge or negative of the clock and also select whether the receive section has it.s own unique bit clock or uses the same bit clock as the transmit section.

The frontend block primarily serializes and de-serializes the data form the receive and transmit fifos. The frame/sync pulses are also generated through a software configured divide of the bit clocks.

The transmit and receive fifos are asynchronous fifos with the internal side residing in the system clock domain and the external side residing in the bit clock domains. This is the mechanism for crossing between the two clock domains.

The frontend operates quite differently when running in the I2S mode than when running in the AC97 mode. The I2S mode operates in stereo or mono. The difference between the two is that the fifos are accessed twice for the stereo mode of operation and only once for mono mode. The stereo mode sends left/right channel data on one edge of the frame signal and sends right/left channel data on the other edge of the frame signal. The frame sync will typically be configured with a 50% duty cycle. Mono mode only sends data on the assertion of the frame signal. In this case the frame signal is typically a pulse that occurs at the beginning of the frame. Data is always sent MSB first. The fifo can

only be accessed with width increments of 8,16 or 32 bits. If the actual word length is not on these boundaries, a larger data width is used with zeros padded in the extra bits (the data is right justified). The data organization in the fifo is dependant on the word length. Also, for stereo applications, the left and right channel data alternate. The following diagram illustrates some examples of fifo data organization.

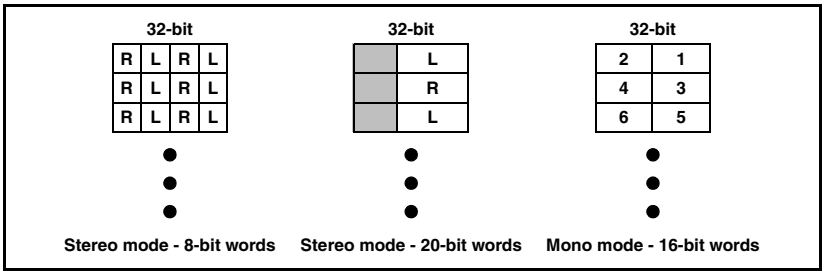


Figure 42. I2S FIFO Data Organization

AC97 is comprised of 13 channels of which the first channel is 16 bits and the rest are 20 bits in length. The frame period is 48 Khz. The following diagram illustrates the channel organization.

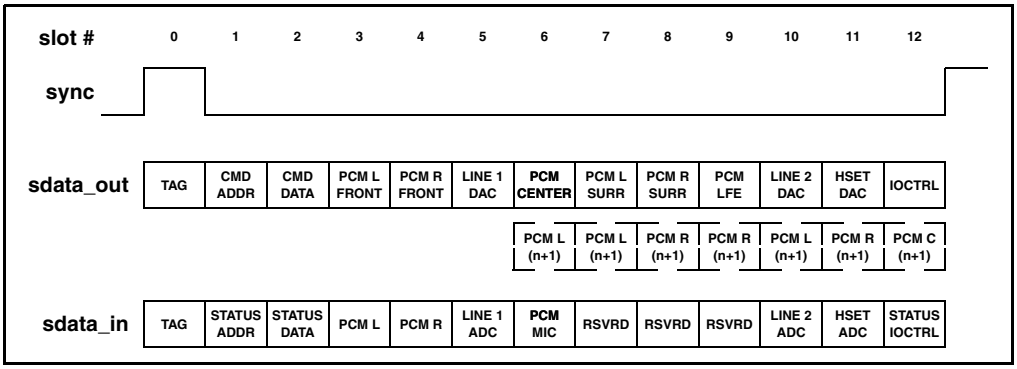


Figure 43. AC97 AC Link Frame Organization

The AC-link output slots are described below. Data for channels 3-12 come from the transmit fifo. If multiple channels are enabled it is assumed that the data is organized in the fifo in the order that the channel gets transmitted. Channels 0, 1, 2 and 12 are driven by software register configurations and accesses.

Slot	Name	Description
0	SDATA_OUT TAG	MSBs indicate which slots contain valid data. LSBs convey codec ID.
1	Control CMD ADDR write port	Read/Write command bit plus 7 bit codec register address.
2	Control DATA write port	16 bit command register write data.
3,4	PCM L&R DAC playback	16,18,20 bit PCM data for left and right channels.
5	Modem line 1 DAC	16 bit modem data for modem line 1 output.
6,7,8,9	PCM center, surround L&R, LFE	16,18,20 bit PCM data for center, surround L&R, LFE channels.
10	Modem Line 2 DAC	16 bit modem data for modem line 2 output.
11	Modem handset DAC	16 bit modem data for modem handset output.
12	Modem IO control	GPIO write port for modem control.
10,11	SPDIF Out	Optional AC-link bandwidth for SPDIF output.

6-12	Double rate audio	Optional AC-link bandwidth for 88.2 or 96khz on L, C, R channels.

The AC-link input slots are described below. Data from slots 3-11 (if valid) is written into the receive fifo in the order that the channel arrives. Valid data from channel 1,2 and 12 is presented in a software register. An interrupt/status indicator informs software that the register contains new data.

Slot	Name	Description
0	SDATA_IN TAG	MSBs indicate which slots contain valid data.
1	STATUS ADDR read port	MSBs echo register address. LSBs indicate which slots request data.
2	STATUS DATA read port	16 bit command register read data.
3,4	PCM L&R ADC record	16,18,20 bit PCM data from left and right channels.
5	Modem line 1 ADC	16 bit modem data for modem line 1 input.
6	Dedicated Microphone ADC	16,18,20 bit PCM data from optional 3rd ADC input.
7,8,9	Vendor reserved	Vendor specific (enhanced input for docking, array mic, etc.
10	Modem Line 2 ADC	16 bit modem data for modem line 2 input.
11	Modem handset ADC	16 bit modem data for modem handset input.
12	Modem IO status	GPIO read port for modem status.

Channel 1&2 are used to read and write registers within the codec. The mechanism to utilize these channels is not through the fifo datapath. Instead software registers exist for the codec address, write data and read data. Configuration of these registers will enable channel 1&2 in the next audio frame. An interrupt/status indicator provides feedback on the completion of register writes or on the availability of register read data. More details of this operation is described in the register definitions. In a similar fashion, the modem IO control and status (channel 12) are also controlled by software registers.

AC97 codecs have a reset input. It is assumed that this is a GPIO and under software control. Whether or not the codec sinks or sources the bit clock is determined by the conditions when the reset is removed. If the codec detects a bit clock present (minimum 5 clocks) while reset is asserted it will be configured to sink the bit clock, otherwise it will source the bit clock. Depending on the application (ASIC sinking or sourcing the bit clock) software must appropriately configure and enable the bit clock generation prior to releasing the codec reset if the application requires the ASIC to source the bit clock. This sequence of events must occur everytime the codec is reset.

There are 3 types of codec resets. The external pin reset (as described above) is a “cold” reset. When a cold reset occurs all codec registers are reset and bit clock sourcing is re-determined. A “warm” reset will re-activate the AC-link without resetting the codec registers. A “warm” reset is indicated by a 1 µsec pulse on the sync line. Software can initiate this process through register configuration. The third reset mechanism is a register bit in the codec. Software has access to this mechanism through the regular codec register configuration process.

The codec can also be placed in “power-down” mode. This is achieved by writing to a particular register in the codec. Since this mechanism requires the hardware to enter a particular state after channel 2 has been sent, in addition to the codec register configuration process, a software indicator bit must be set. To exit from this state a “warm” reset must be issued.

The AC-link provides 12 channels (@ 20 bits) with a frame rate of 48 Khz. The interface also supports a mechanism that allows for sampling rates other than 48 Khz. Data rates of 44.1 Khz, 88.2 Khz and 96 Khz are also supported. The double-rate audio (88.2 Khz or 96 Khz) is supported by combining two slots per DAC channel. This would utilize the optional alternate channel source for channels 6-12.

Up-sampling is not required to support the 44.1 Khz or 88.2 Khz data rates. Channel 0 (in both the incoming and outgoing data stream) contains valid channel flags. This provides the mechanism to send valid data in a sub-set of

the frames being sent. A 44.1 Khz data rate needs valid data in only 441 frames for every 480 frames transferred at 48 Khz. The codec determines when data is sent by setting the channel request bits in the incoming TAG information in channel 0. These are examined by hardware, and the appropriate channels are tagged as valid and filled with data. Since it is assumed that the fifo is filled with the appropriate sequence of data depending on which channels are enabled, the hardware must wait until all channel requests (for channels that are enabled) are requesting since it is not possible to send a channel data in a different sequence than that present in the fifo. Similarily it is assumed that the receive data remains in order (ie. All channels are valid or not valid) and the data is written to the fifo as it is received.

A detailed view of the physical AC-link protocol is illustrated in the following diagrams.

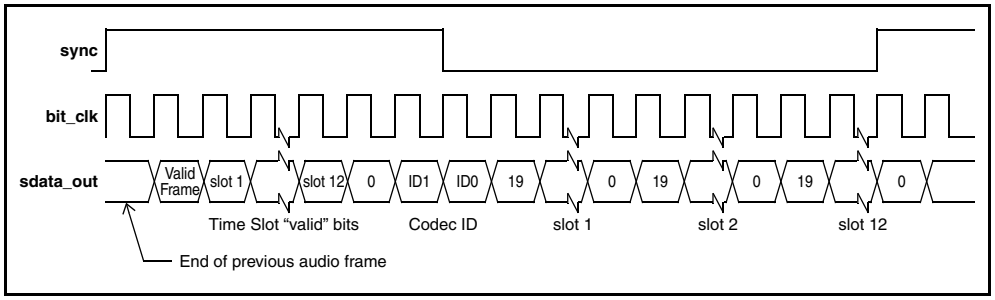


Figure 44. AC97 AC-link Output Frame

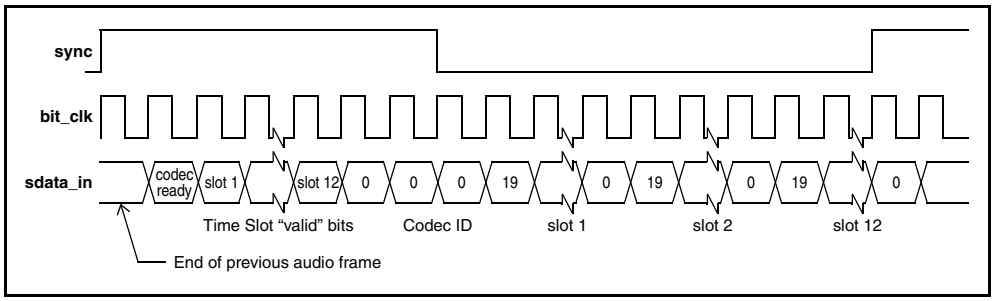


Figure 45. Figure 44 AC97 AC-link Input Frame

Control registers are described at 5.10, Audio Registers.

4.7.2 Audio Port Timing

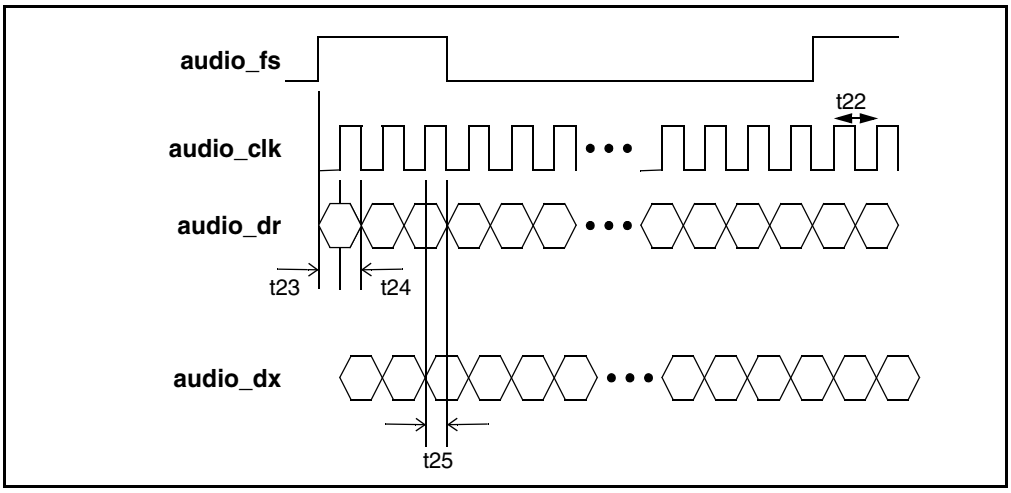


Figure 46. Audio Port Timing

Table 16. Audio Port Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
audio_clk frequency	t22			50	MHz
Input data (audio_fs, audio_dr) setup time to the rising edge of audio_clk	t23(3.0 V)	2.9			ns
Input data (audio_fs, audio_dr) hold time from the rising edge of audio_clk	t24(3.0 V)	0.1			ns
Output data (audio_fs, audio_dx) delay time from the rising edge of audio_clk	t25(3.0 V)			13.25	ns

4.8 Media Storage MMC and MMCPlus blocks (compatible SD/SDHC)

The SCP220x has two MMC blocks both having identical characteristics except that MMCPlus is capable of 8-bit parallel data path:

- MMC: 1 or 4-bit data width
- MMCPlus: 1 or 4 or 8-bit data width

Secure Digital and MMC are supported on both blocks; MMCPlus only on the MMCPlus block.

The Media Storage Interfaces are compatible with the SD and SDHC memory card specifications. SDHC cards are supported up to 32 GB capacity, but only at SD card interface rates (i.e. clock is 25 MHz and not 50 MHz as SDHC allows).

The Media Storage Interfaces have the following features:

- Software programmable external clock
- Support of a 48-bit command through a software accessible command buffer
- Support of both a 48 or 136-bit response through a response buffer
- Support of CRC generation and checking

- Software configurable data width of 1 (MMC mode) or 4 bits (SD/SDHC mode) [or 8 bits (MMCPlus) for the MMCPlus block]
- Incoming and outgoing datapath (implemented using FIFOs) driven by a DMA engine

The interface does not manage the media card power supply. Figure 45 shows the connectivity between the SCP220x and an SD memory card.

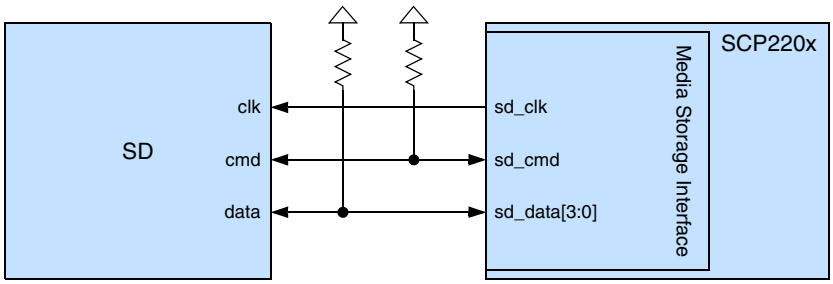


Figure 47. Media Storage Interface to SD Card

The following table lists the SCP220x pin information for the Media Storage Interfaces. Note that the MMCPlus is only accessible from alternate functions.

Table 17. Media Storage Interfaces Pins

Signal	Alternate Function	Pin Type	Pin Description
sd_clk	gpio[35]	Bi-dir.	SD/SDHC clock or alternate function
sd_cmd	gpio[34]	Bi-dir.	SD/SDHC serial command/response or alternate function
sd_D[3:0]	gpio[33:30]	Bi-dir.	SD/SDHC serial data or alternate function
nand_cen_p[0]	gpio[12] / mmcplus_clk	Bi-dir	Clock
nand_cen_p[1]	Gpio[70] / mmcplus_cmd	Bi-dir	Command
nand_data_p[7:0]	gpio[81:74] / mmcplus_data[7:0]	Bi-dir	Data

4.8.1 Media Storage Interfaces Hardware Description

The hardware implementation for the MMC and MMCPlus block are as shown in the following diagram.

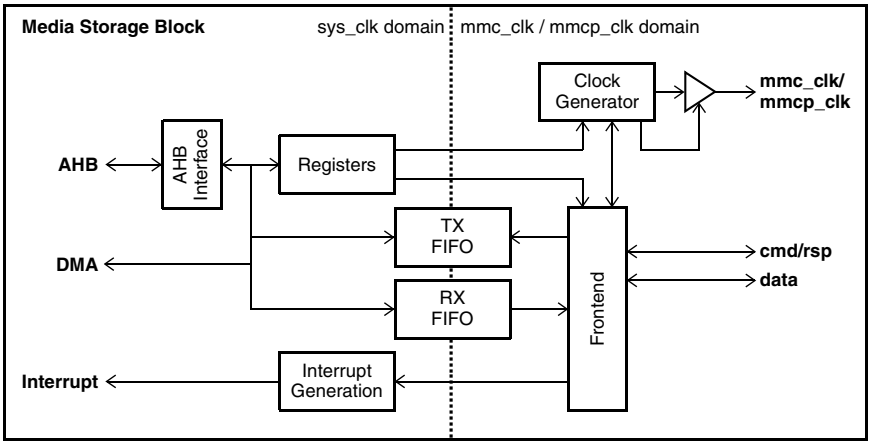


Figure 48. Media Storage MMC/MMCPlus Hardware Architecture

The hardware architecture of the MMC/MMCPlus blocks are similar to the other serial interface blocks. A register section contains the software interface and is read and written from the AMBA bus.

The clock generator block, when enabled, divides the system clock down to create the external serial clock. To simplify timing constraints, the clock is brought back into the block from the outgoing PAD and is used as a clock for the front end interface block as well as the external side of the FIFOs.

The RX/TX FIFOs are asynchronous FIFOs with the internal side driven by sys_clk and the external side driven by mmc_clk/mmcp_clk.

The front-end block is a large state machine that deals with the commands initiated in the register block and generates the appropriate protocol on the external interface.

MMC Control registers are described at 5.11, MMC/SD Control Registers

MMCPlus Control registers are described at 5.12, MMCPlus Control Registers

4.8.2 Programming Model

This section illustrates a number of example software flow diagrams to help illustrate how the interface is used from a software driver perspective. These flows are examples and by no means indicate that this is the only driver flow. Depending on the physical device attached variants of the basic examples shown may very well be required.

The SD interface is based on a command and response architecture. The ASIC will issue the command written by software into the command buffer. The media device will generate a response and this will be capture in the response buffer for software viewing. Data is then read or written on a 4096bit block basis. Single or multiple data blocks can be accessed. Hardware strips off the start stop, transmitter and CRC fields prior to dumping the response or read data in the buffer space. The reverse process happens for the commands and write data blocks.

The following steps illustrate an example media card data read.

1. Initialize the interface pertinent for the media card installed. This would involve configuration of the clock rate and configuration register.
2. Configure the datapath parameters in the data control register.
3. Issue a command by writing the appropriate command and argument to the command and argument registers. The argument register should be written first since the act of writing to the command registers initiates the operation on the interface.
4. If applicable, monitor for a response from the command and verify the response information.
5. Drain the fifo as it fills up. This can be done directly by reading the fifo or a dma channel can be setup to drain the fifo.

6. Continue the draining operation until the “sd_data_complete” interrupt occurs.
7. Check whether or not any errors have occurred.

The following steps illustrate an example media card data write:

- Initialize the interface pertinent for the media card installed. This would involve configuration of the clock rate and configuration register.
- Configure the datapath parameters in the data control register.
- Issue a command by writing the appropriate command and argument to the command and argument registers.
- If applicable, monitor for a response from the command and verify the response information.
- Fill the fifo. This can be done directly by writing to the fifo or a dma channel can be setup to fill the fifo.
- Continue the filling operation until the “sd_data_complete” interrupt occurs.

Check whether or not any errors have occurred.

4.8.3 MMC/MMCPlus Port Timing

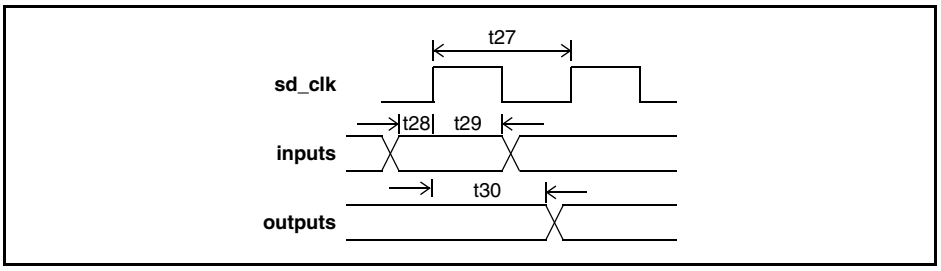


Figure 49. MMC/MMCPlus Port Timing

Table 18. MMC/MMCPlus Port Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
MMC port clock frequency	t27			25	MHz
MMC port input setup time	t28 (3.0 V)	2.5			ns
MMC port input hold time	t29 (3.0 V)	5.0			ns
MMC port output delay time	t30 (3.0 V)			12.5	ns

4.9 I2C Interface

The I2C controller is a peripheral interface intended for configuring external devices such as sensors and audio DACs. The interface consists of the following signals:

- serial clock – This is a clock to sample an incoming serial data stream or to indicate when an outgoing serial stream has valid data. This serial clock pin will “float” high and “drive” low much like an open collector and requires an external pull-up resistor.
- serial data – This is a bi-directional IO that can be driven by either the SCP220x or the peripheral being configured. The serial data pin will “float” high and “drive” low much like an open collector and requires an external pull-up resistor.

The SCP220x has two I2C interfaces on chip. One of these interfaces has primary functionality on dedicated I/Os. The other I2C interface has its serial clock and serial data accessible as alternate functions via shared I/Os as shown in the table below.

Table 19. I2C Interface

Signal	Alternate Function	Pin Direction	Pin Description
scl	gpio[36]	Bi-dir.	Serial configuration clock
sda	gpio[37]	Bi-dir.	Serial configuration data
dip_data[17]	gpio[3] / scl_sec	Bi-dir.	Serial configuration clock
dip_data[16]	gpio[4] / sda_sec	Bi-dir.	Serial configuration data

Transactions on the serial interface follow a particular protocol.

- Transmission Start
- Peripheral Slave Address
- Peripheral Internal Address
- Data transfer
- Transmission Stop

Transmission start and stop are indicated by sequencing the serial clock and data as illustrated in the following diagram. Transmission start occurs when serial data falls while serial clock is high and transmission stop occurs when serial data rises while serial clock is high.

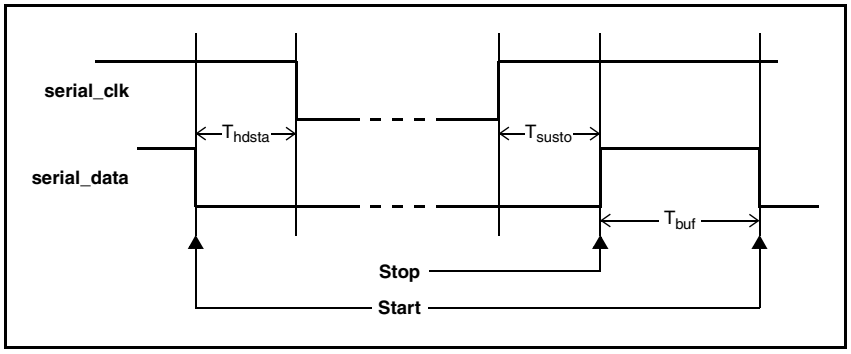


Figure 50. Transmission Stop and Start Conditions

As illustrated there are a few timing parameters that must be satisfied for proper operation. Software configurable counters are used to generate these time intervals since the system clock rate is not fixed.

There is also a software configurable delay parameter as illustrated below:

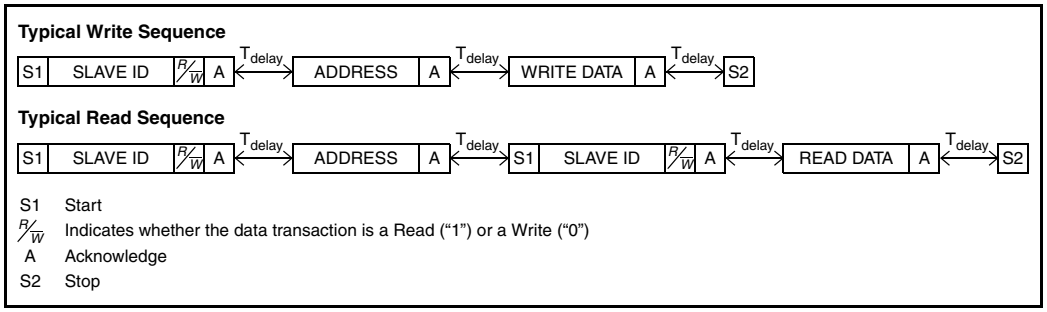


Figure 51. I2C Configurable Delay

The following diagram illustrates the address and data phases of the transaction cycle.

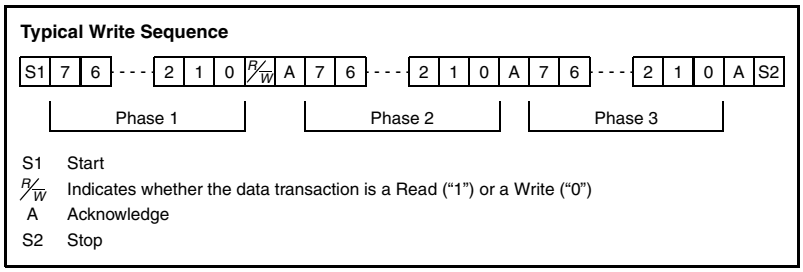


Figure 52. Serial Configuration Transaction Protocol

A basic element of a transaction is called a phase. A transaction can contain either 2-phases or 3- phases depending on the peripheral. Usually a write transaction is a 3-phase transaction specifying the slave address, internal address and data. A read transaction is usually a 2-phase transaction comprised of a peripheral slave address phase and a data phase. The read transaction likely was preceded by a 2-phase write transaction that included a peripheral slave address phase and a peripheral internal address phase.

A phase consists of sequential data transmission of 8-bits that followed by an acknowledge bit. The source of the acknowledge bit is the recipient of the previous 8 bits of data. The external peripheral will source the acknowledge bit for slave and internal address phases as well as write data phases. The controller sources the acknowledge for data read phases.

The peripheral Slave address phase contains a 7 bit slave ID as well as a R/W bit. The R/W bit indicates to the peripheral whether or not the following phases are read or write transactions. R/W=1 indicates a read transaction and R/W=0 indicates a write transaction.

The I2C Controller supports both the master arbitration protocol as well as the Slave stall protocol.

The master arbitration protocol is utilized in systems that have multiple master controllers. The master controller monitors the input SDA line during the SCL high period to see if the data it sent is present on the external SDA line. The SDA line is open-collector, so if another master is driving the SDA line low the input SDA will not match the output SDA for a master that is floating SDA high. This master is deemed to have lost arbitration and will remove itself from the transaction. Master arbitration only occurs during the slave address phase and the write data phases (this is when the master drives the SDA line).

The slave peripheral has a mechanism to stall any part of the I2C transaction. This is done by pulling the SCL line low. The master monitors the SCL input to see if it is held low after the master has driven SCL high. If it is still low, it knows that a slave is stalling the operation and the master pauses until the SCL line floats high (i.e. the slave releases SCL when it is ready for the next part of the transaction.

4.9.1 I2C Hardware Description

The hardware implementation for the serial controller block is fairly simple and only has a few internal blocks as shown in the following diagram.

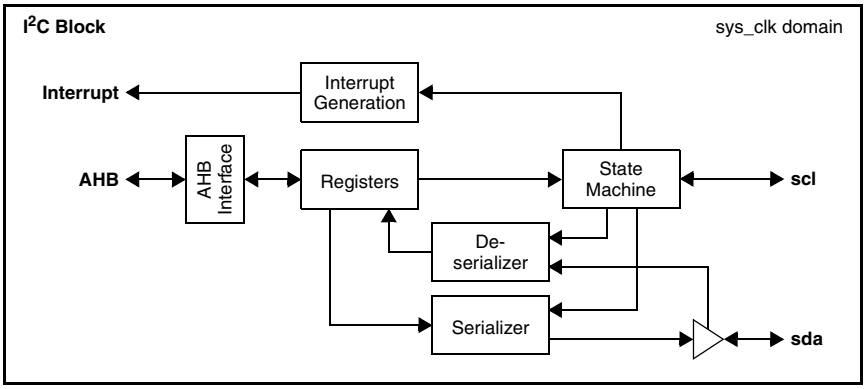


Figure 53. I2C Hardware Architecture

The serial controller is mostly a single state machine that gets initiated by software “kicks”. The various phases of the transaction are initiated by a software register access. The state machine initiates the appropriate hardware protocol at the interface as dictated by the register access.

4.9.2 Programming Model

The serial controller will be used for two very common operations. Either the controller will be used to write to a peripheral to configure the setup of the peripheral or it may be used to read from a register within the peripheral. The following sections provide programming guidelines for these two scenarios.

4.9.2.1 Peripheral Write – Manual Mode

The following steps suggest an algorithm for programming a peripheral configuration.

- Initialize the three configuration registers.
- Write the slave ID to the slave address register along with read_write=0.
- Write the peripheral address to the target address register.
- Write the peripheral data to the target data register.
- Set the start bit in the control register.
- Wait until the acknowledge interrupt occurs.
- Set the stop bit in the control register.
- Wait until the stop interrupt occurs.

4.9.2.2 Peripheral Write – Automatic Mode

The following steps suggest an algorithm for programming a peripheral configuration. This mode of operation is enabled by setting the auto_mode_ena bit in the config1 register

- Initialize the three configuration registers.
- Write the slave ID to the slave address register along with read_write=0.
- Write the peripheral address to the target address register.
- Write the peripheral data to the target data register.
- Set the start bit in the control register.
- Wait until the stop interrupt occurs.

4.9.2.3 Peripheral Read – Manual Mode

The following steps suggest an algorithm for programming a peripheral configuration.

- Initialize the three configuration registers.
- Write the slave ID to the slave address register along with read_write=0.
- Write the peripheral address to the target address register.
- Set the start bit in the control register.
- Wait until the acknowledge interrupt occurs.
- Write the slave ID to the slave address register along with read_write=1.
- Set the start bit in the control register.
- Wait until the acknowledge interrupt occurs.
- Set the stop bit in the control register.
- Wait until the stop interrupt occurs.
- Read the configuration data from the slave data register.

4.9.2.4 Peripheral Read – Auto Mode

The following steps suggest an algorithm for programming a peripheral configuration. This mode of operation is enabled by setting the auto_mode_ena bit in the config1 register

- Initialize the three configuration registers.
- Write the slave ID to the slave address register along with read_write=1.
- Write the peripheral address to the target address register.
- Set the start bit in the control register.
- Wait until the stop interrupt occurs.
- Read the configuration data from the slave data register.

I2C Control registers are described at 5.13, I2C Registers.

4.9.3 I2C Port Timing

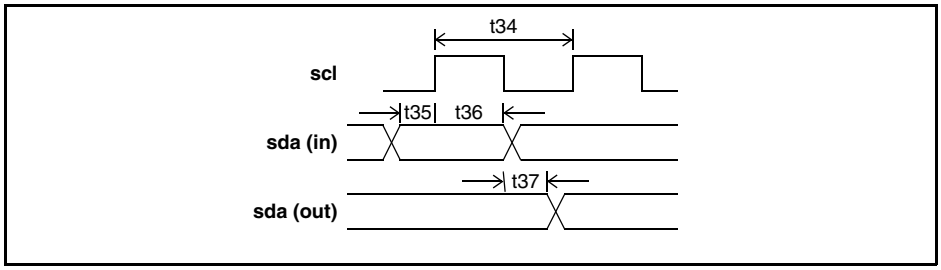


Figure 54. I2C Port Timing

Table 20. I2C Port Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
I^2C port clock frequency (f_{SCL})	t34			400	kHz
I^2C port input setup time ($t_{SU;DAT}$)	t35	100[1]			ns
I^2C port input hold time ($t_{HD;DAT}$)	t36	0			ns

Table 20. I2C Port Timing

I^2C port output delay time ($t_{VD;DAT}$)	t37			900[1]	ns

[1] The I2C Controller design transitions signals at ¼ period intervals of the scl_clk, additionally the I/O pads are designed for operating at >100 Mhz switching frequency, ensuring that the timing specifications of the I2C specification (NXP Semiconductors Document UM10204, I2C-bus specification and user manual, Rev 5 – 9 October 2012) are met.

Example: fSCL = 400 kHZ , scl_clk period = 2500ns;

t35 = ¼ scl_clk period = 625ns ;

t37 = ¼ scl_clk period = 625ns

4.10 Pulse Width Modulated Outputs

Two pulse width modulated (PWM) outputs are available as alternate pin functions. As shown in the following table

Table 21. PWM Function Pinout

Signal	Alternate Function	Pin Direction	Pin Description
audio_fsr	gpio[18] or pwm2	Bi-dir.	Audio receive frame clock, GPIO or PWM signal as alternate function;
sc_fcb	gpio[57] or pwm1	Bi-dir.	Smart card, GPIO or PWM signal as alternate function;

4.10.1 PWM Hardware Description

The Pulse Width Modulation block allows for generating digital signal with variable pulse width with the following features:

- Control of working frequency from system clock (sys_clk) to system clock divided by 4096 (8- bit divider followed by 1, 1/2, 1/4, 1/8 or 1/16)
- Control of period and pulse width through 16-bit registers (from 1 to 65536)
- One shot or free running with posted value updates
- Out signal inverter
- Out signal dead-zone generator through 8-bit register

The following diagram illustrates the architecture of the Pulse Width Modulation (PWM).

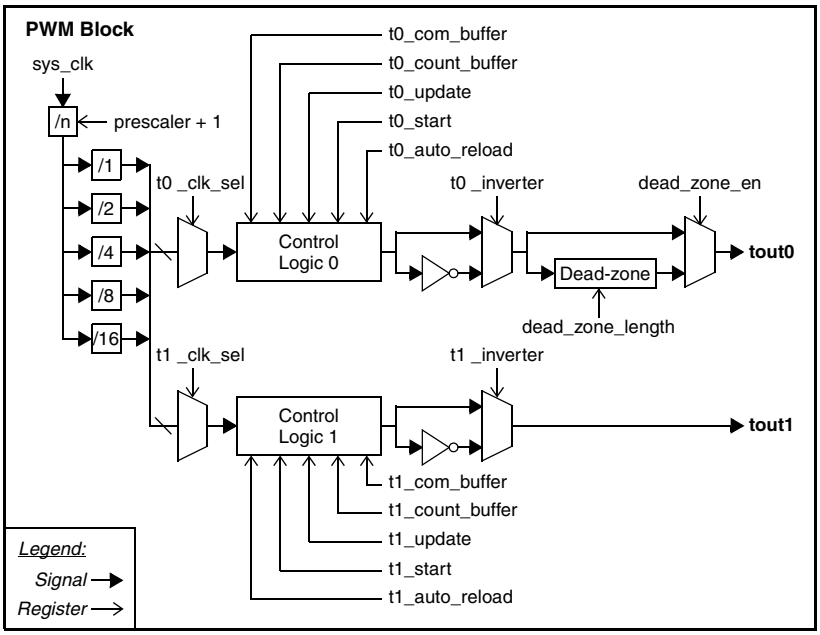


Figure 55. PWM Hardware Architecture

The Tout0 frequency is determined by the following formula:

$$f_{tout0} = \frac{f_{sys_clk}}{(prescaler + 1) * ([1|2|4|8|16]_{t0_clk_sel}) * (t0_count_buffer + 1)}$$

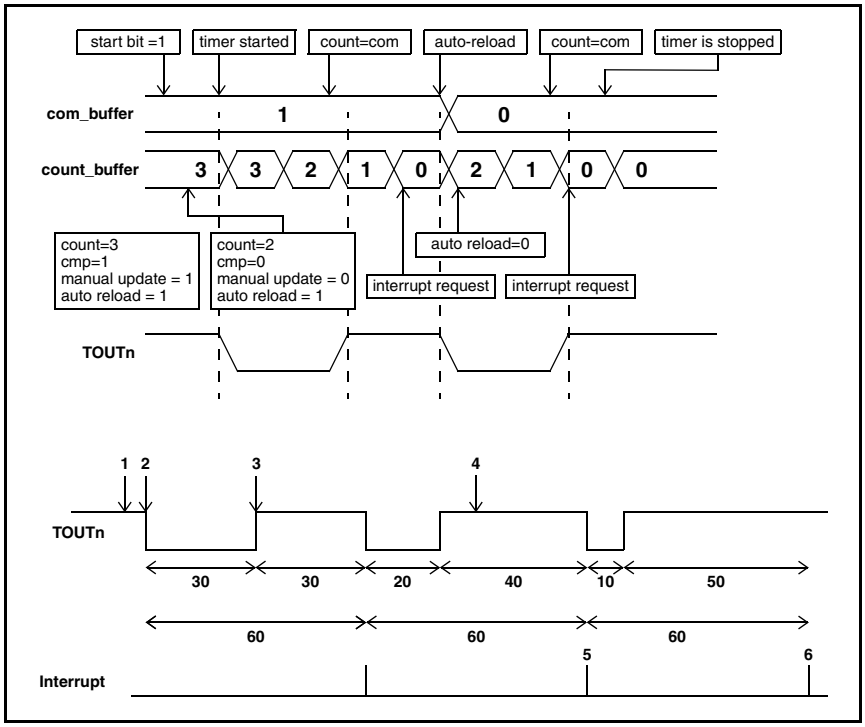
For Tout1, just replace the t0 by t1.

PWM Control registers are described at 5.14, PWM Registers.

4.10.2 PWM Programming Notes

4.10.2.1 Example

The following section provides some details on appropriate programming of the PWM using an example. The following diagram illustrates a timeline for a particular PWM configuration.



The following events refer to the numbering in the above diagram

1. count=60, cmp=30, update=1, auto_reload=1, update=0. The update must be toggled from “1” to “0”. The value of the next count and cmp can be set at step 3. In the case where these values have been set prior to step 1, the update should be disabled. If the next value is set in the enable state, this next value goes into the first value of count and cmp at the “start”.
2. start=1.
3. cmp=20. This can be set as soon as the start was issued in step 2. In the case where auto_reload is enabled, the count is updated when the interrupt occurs so count and cmp must be set prior to the interrupt event. If the cmp value is enough until the next reflection, it can be set after the interrupt.
4. cmp=10.
5. auto_reload=0.
6. start=0.

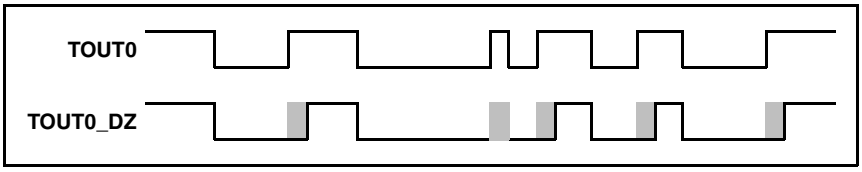
4.10.2.2 PWM as general timer

Another use of the PWM is as a general timer. For this scenario, the cmp value is deducted from the PWM timer. The rest of the settings are identical to the PWM usage. For example to get an interrupt every 10msec (100hz) for a 24 Mhz clock source, the following settings are required.

- 24 Mhz / 100hz = 240,000
- 240,000 = (prescaler + 1) x (1/Tn_clk_sel) x (count + 1)
- Prescaler=99, 1/Tn_clk_sel=16, count=149

4.10.2.3 PWM dead zones

Another PWM usage is to have dead zones. A dead-zone delays the low to high transition point. The following diagram illustrates the concept.



4.11 KeyPad Scan Interface

The SCP220x has an optional keyscan capability. The keyscan processor has four output scan ports and four input scan ports to allow recognition of up to 16 keys.

The Keyscan Interface provides the following features:

- Programmable key scan and sense polarity
- Programmable scan time
- Programmable scan matrix
- Auto clearing of the sense value after it has been read
- Supports typing mode and gaming mode

Figure 54 shows the keyscan system implementation.

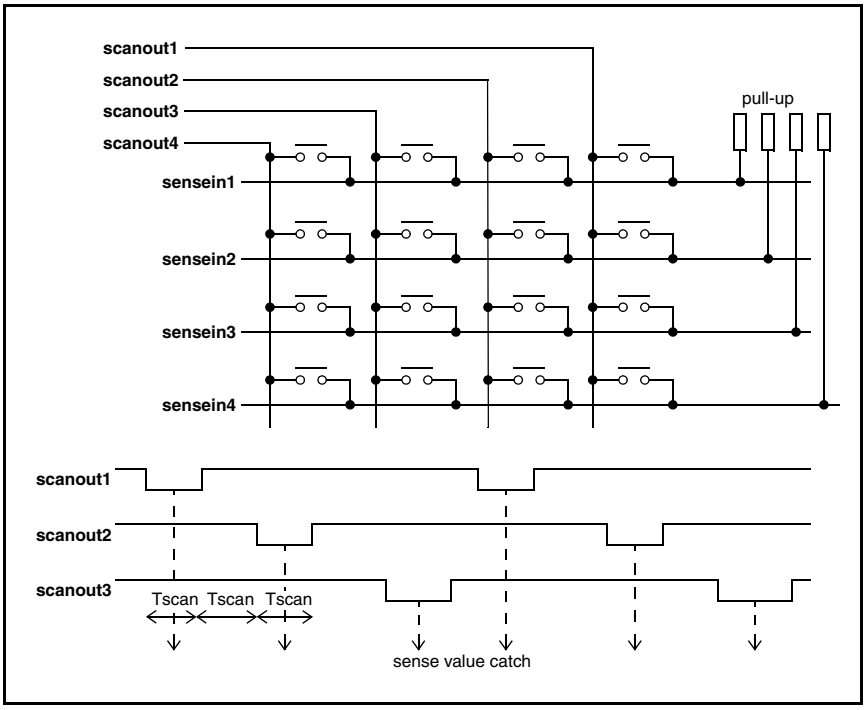


Figure 56. Keyscan Interface

Table 22 lists the SCP220x pin information for the Keyscan Interface. To enable the keyscan interface, the Alternate Function register must be programmed accordingly.

Table 22. Keyscan Interface

Signal	Keyscan Function	Pin Direction	Pin Description
reserved_14	gpio[45] or keyscan_out3	Bi-dir.	GPIO or alternate function

Table 22. Keyscan Interface

reserved_13	gpio[44] or keyscan_out2	Bi-dir.	GPIO or alternate function
reserved_12	gpio[43] or keyscan_out1	Bi-dir.	GPIO or alternate function
reserved_11	gpio[42] or keyscan_out0	Bi-dir.	GPIO or alternate function
reserved_10	gpio[41] or keyscan_in3	Bi-dir.	GPIO or alternate function
reserved_9	gpio[40] or keyscan_in2	Bi-dir.	GPIO or alternate function
reserved_8	gpio[39] or keyscan_in1	Bi-dir.	GPIO or alternate function
reserved_7	gpio[38] or keyscan_in0	Bi-dir.	GPIO or alternate function

Keyscan control registers are described at 5.15, KeyScan Registers.

4.12 GPIOs and Alternate Functions

A number of external pins have additional GPIO functionality and possibly an alternate function as shown in GPIO and Alternate Function List.

The “gpio enable” register controls whether the pin functions as a GPIO. For the pins that have also an alternate function, the “alternate function enable” register selects the alternate function if the GPIO enable bit for that pin is disabled.

NOTE

External pins labeled ‘reserved_#’ should only be used as GPIOs or as the alternate function as the primary functionality is reserved and not intended for use during normal operation. The SDK generates a firmware that handles the pin function already.

The primary function, GPIO, alternate function is listed in 4.12.1, GPIO and Alternate Function List below.
The pinout is listed in Table 88 (SCP220x Pinout)
The GPIO control registers are described at 5.16, GPIO Registers
The PAD and I/O control registers are described at 5.3, PAD and I/O registers
The GPIO enable bits are listed in ,
The Alternate function enable bits are listed in Table 40., Alternate Function Enable Register
The PAD strength enable bits are listed in Table ,
The PAD Types are described below 4.12.2, PAD Type description

4.12.1 GPIO and Alternate Function List

Table 23. GPIOs and Alternate Functions Shared with External Pins

GPIO	PIN - MAIN	ALTERNATE	POWER	PAD TYPE	PAD Resistor/Default
-	dip_rs_p	dip_hsync	LVDD	B	PD/none
-	dip_wen_p	dip_vsync	LVDD	B	PD/none
-	spi1_sck_p	mp2ts1_clk	JVDD	A	PD/none
-	spi1_rxd_p	mp2ts1_d	JVDD	A	PD/none

Table 23. GPIOs and Alternate Functions Shared with External Pins

-	spi1_ssn_p	mp2ts1_sync	JVDD	C	PU/PU
-	spi1_txd_p	mp2ts1_valid	JVDD	A	PD/none
-	sdram_clkn_p	sdram_clk_fb	RVDD	B	-
gpio00	Reserved_1		MVDD	A	PD/none
gpio01	sif_clkout_p		SVDD	B	PD/none
gpio02	Reserved_2		MVDD	A	PD/PD
gpio03	dip_data_p[17]	scl_sec	LVDD	B	PD/none
gpio04	dip_data_p[16]	sda_sec	LVDD	B	PD/none
gpio05	dip_pclk_p		LVDD	B	PD/none
gpio06	Reserved_3	pwi_clk	MVDD	A	PD/none
gpio07	Reserved_4	pwi_data	MVDD	A	PD/none
gpio08	Reserved_5		MVDD	A	PD/none
gpio09	Reserved_6		MVDD	A	PD/none
gpio10	nand_ale_p		NVDD	A	PD/none
gpio11	nand_cle_p		NVDD	A	PD/none
gpio12	nand_cen_p[0]	mmcplus_clk	NVDD	C	PU/PU
gpio13	nand_wen_p		NVDD	A	PD/none
gpio14	nand_ren_p		NVDD	A	PD/none
gpio15	dip_oen_p	dip_blank	LVDD	B	PD/none
gpio16	audio_clkr_p		AUVDD	A	PD/none
gpio17	audio_dr_p		AUVDD	A	PD/none
gpio18	audio_fsr_p	pwm2_out	AUVDD	A	PD/none
gpio19	audio_clkx_p		AUVDD	A	PD/none
gpio20	audio_dx_p		AUVDD	A	PD/none
gpio21	audio_fsx_p		AUVDD	A	PD/none
gpio22	uart_txd_p		JVDD	A	PD/none
gpio23	uart_rxd_p		JVDD	A	PD/none
gpio24	dip_csn2_p		LVDD	D	PU/PU
gpio25	dip_csn3_p		LVDD	D	PU/PU
gpio26	spi_txd_p		JVDD	A	PD/none
gpio27	spi_rxd_p		JVDD	A	PD/none
gpio28	spi_ssn_p		JVDD	C	PU/PU

Table 23. GPIOs and Alternate Functions Shared with External Pins

gpio29	spi_sck_p		JVDD	A	PD/none
gpio30	mmc_data_p[0]		SDVDD	B	PD/none
gpio31	mmc_data_p[1]		SDVDD	B	PD/none
gpio32	mmc_data_p[2]		SDVDD	B	PD/none
gpio33	mmc_data_p[3]		SDVDD	B	PD/none
gpio34	mmc_cmd_p		SDVDD	B	PD/none
gpio35	mmc_clk_p		SDVDD	D	PU/PU
gpio36	scl_p		SVDD	B	PD/none
gpio37	sda_p		SVDD	B	PD/none
gpio38	Reserved_7	keyscan_in0	MVDD	B	PD/none
gpio39	Reserved_8	keyscan_in1	MVDD	B	PD/none
gpio40	Reserved_9	keyscan_in2	MVDD	B	PD/none
gpio41	Reserved_10	keyscan_in3	MVDD	B	PD/none
gpio42	Reserved_11	keyscan_out0	MVDD	B	PD/none
gpio43	Reserved_12	keyscan_out1	MVDD	B	PD/none
gpio44	Reserved_13	keyscan_out2	MVDD	B	PD/none
gpio45	Reserved_14	keyscan_out3	MVDD	B	PD/none
gpio46	dip_data_p[18]		LVDD	B	PD/none
gpio47	dip_data_p[19]		LVDD	B	PD/none
gpio48	dip_data_p[20]		LVDD	B	PD/none
gpio49	dip_data_p[21]		LVDD	B	PD/none
gpio50	dip_data_p[22]	uart1_rxd	LVDD	B	PD/none
gpio51	dip_data_p[23]	uart1_txd	LVDD	B	PD/none
gpio52	fodd_p		SVDD	B	PD/none
gpio53	sif_gpio_p		SVDD	B	PD/none
gpio54	dip_cpu_vsync_p		LVDD	B	PD/none
gpio55	sc_clk_p		SMVDD	D	PU/PU
gpio56	sc_rst_p		SMVDD	B	PD/none
gpio57	sc_fcb_p	pwm1_out	SMVDD	B	PD/none
gpio58	sc_io_p		SMVDD	B	PD/none
gpio59	sc_card_detect_p		SMVDD	B	PD/none
gpio60	sc_power_on_p		SMVDD	B	PD/none

Table 23. GPIOs and Alternate Functions Shared with External Pins

gpio61	sc_card_voltage_p	spi_rxd3	SMVDD	B	PD/none
gpio62	-		-	-	-/-
gpio63	-		-	-	-/-
gpio64	-		-	-	-/-
gpio65	-		-	-	-/-
gpio66	-		-	-	-/-
gpio67	-		-	-	-/-
gpio68	-		-	-	-/-
gpio69	-		-	-	-/-
gpio70	nand_cen_p[1]	mmcplus_cmd	NVDD	C	PU/PU
gpio71	nand_cen_p[2]	spi_rxd1	NVDD	C	PU/PU
gpio72	nand_cen_p[3]	spi_rxd2	NVDD	C	PU/PU
gpio73	utmiotg_drvvbus_p		AUVDD	D	PU/PU
gpio74	nand_data_p[0]	mmcplus_data[0]	NVDD	A	PD/none
gpio75	nand_data_p[1]	mmcplus_data[1]	NVDD	A	PD/none
gpio76	nand_data_p[2]	mmcplus_data[2]	NVDD	A	PD/none
gpio77	nand_data_p[3]	mmcplus_data[3]	NVDD	A	PD/none
gpio78	nand_data_p[4]	mmcplus_data[4]	NVDD	A	PD/none
gpio79	nand_data_p[5]	mmcplus_data[5]	NVDD	A	PD/none
gpio80	nand_data_p[6]	mmcplus_data[6]	NVDD	A	PD/none
gpio81	nand_data_p[7]	mmcplus_data[7]	NVDD	A	PD/none
gpio82	uart_cts_p	spi_ssn1	JVDD	A	PD/none
gpio83	uart_rts_p	spi_ssn2	JVDD	A	PD/none
gpio84	Reserved_15	spi_ssn3	MVDD	A	PD/none
gpio85	sdram_rdy_p		RVDD	B	PD/none
gpio86	dip_csn0_p		LVDD	D	PU/PU
gpio87	dip_csn1_p		LVDD	D	PU/PU
gpio88	mp2ts_d_p		JVDD	A	PD/n.a.
gpio89	mp2ts_clk_p		JVDD	-	PD/n.a.
gpio90	mp2ts_valid_p		JVDD	A	PD/n.a.
gpio91	mp2ts_sync_p		JVDD	A	PD/n.a.
gpio92	-		-	-	-/-

Table 23. GPIOs and Alternate Functions Shared with External Pins

gpio93	-		-	-	-/-
gpio94	-		-	-	-/-
gpio95	-		-	-	-/-

4.12.2 PAD Type description

There are four types of hardware PAD for the GPIOs: A, B, C, D. The correspondence with the pin is listed in the table above.

Following are each GPIO PAD type specifications at 50% transition and the corresponding measurement illustration diagram.

4.12.2.1 GPIO pad type A specifications

Table 24. GPIO PAD type A specifications

Pad Type	Drive Strength	Parameter	Load (pF)	Min	Typ	Max	Unit
A	Low-drive	tpLH	10	2	-	5	ns
			25	3	-	7	ns
			100	10	-	19	ns
			200	18	-	35	ns
		tpHL	10	2	-	5	ns
			25	4	-	8	ns
			100	11	-	23	ns
			200	20	-	42	ns
	High-drive	tpLH	10	1	-	3	ns
			25	2	-	4	ns
			100	5	-	10	ns
			200	8	-	16	ns
		tpHL	10	1	-	3	ns
			25	2	-	5	ns
			100	6	-	12	ns
			200	11	-	22	ns

4.12.2.2 GPIO PAD type B specifications

Table 25. GPIO PAD type B specifications

Pad Type	Drive Strength	Parameter	Load (pF)	Min	Typ	Max	Unit
B	Low-drive	tpLH	10	2	-	4	ns
			25	3	-	6	ns
			100	5	-	11	ns
			200	9	-	18	ns
		tpHL	10	2	-	5	ns
			25	3	-	6	ns
			100	6	-	14	ns
			200	11	-	24	ns
	High-drive	tpLH	10	2	-	4	ns
			25	2	-	5	ns
			100	4	-	8	ns
			200	7	-	13	ns
		tpHL	10	2	-	4	ns
			25	2	-	5	ns
			100	5	-	10	ns
			200	8	-	16	ns

4.12.2.3 GPIO PAD type C specifications

Table 26. GPIO pad type C specificatiopns

Pad Type	Drive Strength	Parameter	Load (pF)	Min	Typ	Max	Unit
C	Low-drive	tpLH	10	2	-	5	ns
			25	3	-	7	ns
			100	10	-	19	ns
			200	18	-	35	ns
		tpHL	10	2	-	5	ns
			25	4	-	8	ns
			100	11	-	23	ns
			200	20	-	42	ns
	High-drive	tpLH	10	1	-	3	ns

Table 26. GPIO pad type C specificatiopns

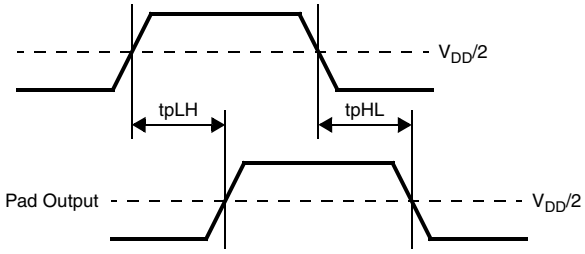
			25	2	-	4	ns
			100	5	-	10	ns
			200	8	-	16	ns
		tpHL	10	1	-	3	ns
			25	2	-	5	ns
			100	6	-	12	ns
			200	11	-	22	ns

4.12.2.4 GPIO PAD type D specifications

Table 27. GPIO pad type D specificatiopns

Pad Type	Drive Strength	Parameter	Load (pF)	Min	Typ	Max	Unit
D	Low-drive	tpLH	10	2	-	4	ns
			25	3	-	6	ns
			100	5	-	11	ns
			200	9	-	18	ns
		tpHL	10	2	-	5	ns
			25	3	-	6	ns
			100	6	-	14	ns
			200	11	-	24	ns
	High-drive	tpLH	10	2	-	4	ns
			25	2	-	5	ns
			100	4	-	8	ns
			200	7	-	13	ns
		tpHL	10	2	-	4	ns
			25	2	-	5	ns
			100	5	-	10	ns
			200	8	-	16	ns

4.12.2.5 GPIO PAD Propagation Schematic



Propagation delays Low to High and High to Low

Figure 57. GPIO PAD Propagation Schematic

4.13 Production Test and System Signals

The following table lists the SCP220x pin information for system and test signals.

Table 28. Production Test and System Singlas

Signal	Pad Resistor	Pin Direction	Pin Description
Clkin	-	Input	Clock input to SCP220x from crystal, oscillator or baseband processor
Clkout	-	Output	Output for crystal connection or ground if Clkin is driven by oscillator
resetN	-	Input	Chip reset
hw_deep_secure	-	Input	set to ‘0’, reserved
bootmode	-	Input	Selects how the part will startup after reset. Must be set to ‘1’.
testmode	PD	Input	Enable testmode (manufacture test only)
tck	PU	Input	JTAG test clock
rtck	-	Output	JTAG return clock
tdi	PU	Input	JTAG test data input
tdo	-	Output	JTAG test data output
Ntrst	PD	Input	JTAG test reset
tms	PU	Input	JTAG test mode

5 Registers

In general, an application should use the SDK to use the blocks available on the chips. Most of the low level interfacing that requires register access is already available ready to use.
In case it is required to act directly on the registers here is the list of the main register groups. It is recommended to use macro functions in the SDK to modify the values of the registers and in most cases the access is done through

the register/bit name rather than directly with the register value.

When writing boot loader code however it is necessary to access the register directly.

5.1 Memory Map

The following table describes the address map.

Table 29. Memory Map

Start Address	End Address	Peripheral	ARM1 HSEL
0x0000_0000	0x0000_003f	Reserved	Reserved
0x0000_0000	0x1fff_ffff	External Memory	21
0x2000_0000	0x2fff_ffff	Section 5.6, Memory Controller	21
0x3000_0000	0x3fff_ffff	Reserved	Reserved
0x4000_0000	0x43ff_ffff	DIP Port	4
0x4400_0000	0x47ff_ffff	BitBlt	23
0x4800_0000	0x4fff_ffff	BitBlt_mini	24
0x5000_0000	0x53ff_ffff	USB OTG	5
0x5400_0000	0x57ff_ffff	Reserved	Reserved
0x5800_0000	0x5Bff_ffff	Reserved	Reserved
0x5c00_0000	0x5fff_ffff	Reserved	Reserved
0x6000_0000	0x67ff_ffff	Section 5.7, NAND Interface Registers Description	7
0x6800_0000	0x6Bff_ffff	Section 5.11, MMC/SD Control Registers	8
0x6c00_0000	0x6fff_ffff	Section 5.12, MMCPlus Control Registers	28
0x7000_0000	0x73ff_ffff	Multi-channel DMA	9
0x7400_0000	0x77ff_ffff	SPI1	30
0x7800_0000	0x7fff_ffff	Reserved	Reserved
0x8000_0000	0x87fff_ffff	Reserved	Reserved
0x8800_0000	0x8fff_ffff	Reserved	Reserved
0x9000_0000	0x9fff_ffff	Reserved	Reserved
0x9400_0000	0x97ff_ffff	Reserved	Reserved
0x9800_0000	0x9fff_ffff	Reserved	Reserved
0xA000_0000	0xA3ff_ffff	Section 5.9, SPI Registers	14
0xA400_0000	0xA7ff_ffff	Reserved	Reserved
0xA800_0000	0xAfff_ffff	Reserved	Reserved

Table 29. Memory Map

0xB000_0000	0xBfff_ffff	Sensor Interface	16
0xC000_0000	0xC3ff_ffff	Reserved	Reserved
0xC400_0000	0xC7ff_ffff	Reserved	Reserved
0xc800_0000	0xcbff_ffff	Reserved	Reserved
0xcc00_0000	0xcfff_ffff	Reserved	Reserved
0xd000_0000	0xdfff_ffff	APB bridge	19
0xe000_0000	0xefff_ffff	Reserved	Reserved
0xf000_0000	0xffff_efff	Reserved	Reserved
0xffff_f000	0xffff_ffff	Reserved	Reserved

The following table decribes the memory mapping through the APB bridge containing most of the peripherals interfaces.

Table 30. Sub-peripherals Memory Map

Start Address	End Address	Sub-peripherals
0xd000_0000	0xd000_ffff	Section 5.10, Audio Registers
0xd001_0000	0xd001_ffff	Section 5.8, UART Control Registers
0xd002_0000	0xd002_ffff	Section 5.15, KeyScan Registers
0xd003_0000	0xd003_ffff	System Registers
0xd004_0000	0xd004_ffff	Section 5.13, I2C Registers
0xd005_0000	0xd005_ffff	OS Timer 1
0xd006_0000	0xd006_ffff	OS Timer 2
0xd007_0000	0xd007_ffff	OS Timer 3
0xd008_0000	0xd008_ffff	RTC Timer
0xd009_0000	0xd009_ffff	Section 5.14, PWM Registers
0xd00a_0000	0xd00a_ffff	Section 5.16, GPIO Registers
0xd00b_0000	0xd00b_ffff	Smart Card
0xd00c_0000	0xd00c_ffff	Reserved
0xd00d_0000	0xd00d_ffff	OS Timer 4
0xd00e_0000	0xd00e_ffff	Uart2
0xd00f_0000	0xd00f_ffff	Reserved

The system register map is summarized below and described in the following sections.

System registers are located from address: 0xd003_0000.

Table 31. System Registers

Register	Address Offset	Mode
Section 5.2.1, System Clock Configuration Register	0x00	RW
Section 5.2.2, AP Clock Configuration Register	0x04	RW
Section 5.2.3, Clock Update Register	0x08	RW
Section 5.4.3, System Reset Register	0x18	RW
Section 5.4.1, System Power Down	0x1c	RW
Section 5.5.1, Chip ID Register	0x28	RO
Section 5.3.1, Alternate Function Enable Register	0x30	RW
Section 5.3.2, Drive Strength Register	0x80	RW
Drive strength2	0x84	RW
Drive strength3	0x88	RW
Section 5.3.3, PAD Resistor Enable	0x8C	RW
PAD resistor enable2	0x90	RW
PAD resistor enable3	0x94	RW
PAD resistor enable4	0x98	RW
Section 5.4.2, System power down1	0xCC	RW
Section 5.4.4, System Reset1 Register	0xd0	RW
Section 5.2.7, Interface PLL Select Register	0xF8	RW
IF PLL divide	0xFC	RW
XGA PLL divide	0x100	RW
SIF PLL divide	0x104	RW
MEM PLL divide	0x108	RW
AP PLL divide	0x10C	RW
USB PLL divide	0x110	RW
TVOUT PLL divide	0x114	RW

5.2 Clock Configuration Registers

If necessary, it is possible to change the clocks configuration, exercise caution doing so especially when modifying the system clocks. It is recommended in any case to change clocks configuration via the SDK.

Here is the list of registers, please refer to 3.3, Clock Configuration for details:

- System Clock Configuration Register
- AP Clock Configuration Register
- Clock Update Register

- Interface Clock Configuration Register
- DIP Clock Configuration Register
- Memory Clock Configuration Register
- Interface PLL Select Register
- PLL Divide Register

5.2.1 System Clock Configuration Register

Table 32. System Clock Configuration Register

System Clock Configuration			
Address: 0x00		Reset = 0xcc0_0000	Type: RW

Name	Bit	Function	Reset
arm2_clk_en_div	31-30	Both ARMs have a clock enable that effectively tells the ARM926EJ-S processor the frequency of the system bus. The primary ARM926EJ-S processor is able to get the appropriate divide value from the sys_clk_div field but the secondary ARM926EJ-S processor needs a separate field to define this value since it can operate at a different frequency than the primary ARM926EJ-S processor . This field is similar to the sys_clk_div field and must be programmed based on the difference between the ARM2 clock and the system clock frequencies. Note: this field does not set the frequency of the system clock. 11 : sys_clk = arm2_clk/4 10 : sys_clk = arm2_clk/3 01 : sys_clk = arm2_clk/2 00 : sys_clk = arm2_clk	0
tcm_clk_sel	29	The primary ARM926EJ-S processor TCM can be used as either TCM memory or system memory (connected to the bus). The clocking source and physical interface change for each application. This bit selects the application. The sys_clk_config makes the switch over occur. 0 = system memory 1 = tcm memory	
arm_clk2_div	28-26	This divide value is applied to the PLL output clock to generate the secondary ARM926EJ-S processor clock. 0,7 : arm_clk = F_{PLLOUT} /12 6 : arm_clk = F_{PLLOUT} /10 5 : arm_clk = F_{PLLOUT} /8 4 : arm_clk = F_{PLLOUT} /6 3 : arm_clk = F_{PLLOUT} /4 2 : arm_clk = F_{PLLOUT} /3 1 : arm_clk = F_{PLLOUT} /2	3
pll_sel	25	This field selects the PLL source for the system clock generation. 0 = sys pll 1 = ap pll	0

Table 32. System Clock Configuration Register

arm_clk_div	24-22	This divide value is applied to the PLL output clock to generate the primary ARM926EJ-S processor clock. 0,7 : arm_clk = F_{PLLOUT} /12 6 : arm_clk = F_{PLLOUT} /10 5 : arm_clk = F_{PLLOUT} /8 4 : arm_clk = F_{PLLOUT} /6 3 : arm_clk = F_{PLLOUT} /4 2 : arm_clk = F_{PLLOUT} /3 1 : arm_clk = F_{PLLOUT} /2	3
sys_clk_div	21-20	This divide value is applied to the ARM926EJ-S processor clock to generate the system clock. The system clock runs all internal logic except the ARM926EJ-S processor and array processor. 11 : sys_clk = arm_clk/4 10 : sys_clk = arm_clk/3 01 : sys_clk = arm_clk/2 00 : sys_clk = arm_clk	0
	19		
Range	18-16	This field must be set according the configured post reference divide frequency. 0=bypass, 1=10-16 Mhz, 2=16-26 Mhz, 3=26-42 Mhz, 4=42-65 Mhz, 5=65-104 Mhz, 6=104-166 Mhz, 7=166 Mhz+	0
NO	15-13	PLL Output Divider value.	0
NR	12-8	PLL Input Divider value. Power-up default of this field is controlled by configuration settings on the EBI address bus.	0
NF	7-0	PLL Feedback divider value.	0

5.2.2 AP Clock Configuration Register

Table 33. AP Clock Configuration Register

AP Clock Configuration			
Address: 0x04		**Reset = 0x2000_0000**	**Type: RW**
Name	**Bit**	**Function**	**Reset**

Table 33. AP Clock Configuration Register

mem_clk_div	31-29	This divide value is applied to the PLL output clock to generate the memory 2x clock. The “sys_clk_config” kicker applies the divide value. This is only pertinent if the power-up config of the memory clock source is the “sync” mode, otherwise the divide value defaults to /2 of the memory PLL. An additional /2 is also applied so that both a clk and clk_2x are generated. The clk frequency (not the clk_2x) must match the system clock frequency. 0,7 : arm_clk = F_{PLLOUT} /12 6 : arm_clk = F_{PLLOUT} /10 5 : arm_clk = F_{PLLOUT} /8 4 : arm_clk = F_{PLLOUT} /6 3 : arm_clk = F_{PLLOUT} /4 2 : arm_clk = F_{PLLOUT} /3 1 : arm_clk = F_{PLLOUT} /2	1
	28-23		
ap_clock_disable	22	1 = The AP PLL is powered down and put in bypass mode 0 = The AP PLL is enabled	0
	21-19		
Range	18-16	This field must be set according the configured post reference divide frequency. 0=bypass, 1=10-16 Mhz, 2=16-26 Mhz, 3=26-42 Mhz, 4=42-65 Mhz, 5=65-104 Mhz, 6=104-166 Mhz, 7=166 Mhz+	0
NO	15-13	PLL Output Divider value.	0
NR	12-8	PLL Input Divider value. Power-up default of this field is controlled by configuration settings on the EBI address bus.	0
NF	7-0	PLL Feedback divider value.	0

5.2.3 Clock Update Register

Table 34. Clock Update Register

Clock Update			
Address: 0x08		**Reset = 0x30**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
	31-7		
sys_pll_select_cfg	6	When a ’1’ is written to this bit, the PLL selection is applied to the system PLL. This bit is self clearing after the pll selection has been applied. When switching to a new PLL source, the new PLL must already be properly configured and locked.	0x0
ap_pll_lock_status	5	This bit is a read only bit and reflects the "locking" status of the AP PLL. 0 = no lock, 1 = lock.	0x1

Table 34. Clock Update Register

sys_pll_lock_status	4	This bit is a read only bit and reflects the "locking" status of the system PLL. 0 = no lock, 1 = lock.	0x1
	3		
sys_clk_config	2	When a '1' is written to this bit, the clock divide settings for the system, ARM926EJ-S processor and mem_clk_div are applied as well as the tcm_clk_sel setting. This bit is self clearing after the clock divide has been applied	0x0
ap_pll_config	1	When a '1' is written to this bit, the AP PLL configuration process is initiated and the AP PLL is re-configured with the divider values programmed into the AP Clock configuration register. This bit is self clearing after the AP PLL has locked.	0x0
sys_pll_config	0	When a '1' is written to this bit, the system PLL configuration process is initiated and the system PLL is re-configured with the divider values programmed into the system Clock configuration register. This bit is self clearing after the system PLL has locked.	0x0

5.2.4 Interface Clock Configuration Register

Table 35. Interface Clock Configuration Register

Interface Clock Configuration			
Address: 0x3C		**Reset = 0x10_0000**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-21	Reserved	
pll_lock_status	20	This bit is a read only bit and reflects the "locking" status of the PLL. 0 = no lock, 1 = lock.	0
clock_disable	19	1 = The PLL is powered down and put in bypass mode 0 = The PLL is enabled	0
Range	18-16	This field must be set according the configured post reference divide frequency. 0=bypass, 1=10-16 Mhz, 2=16-26 Mhz, 3=26-42 Mhz, 4=42-65 Mhz, 5=65-104 Mhz, 6=104-166 Mhz, 7=166 Mhz+	0
NO	15-13	PLL Output Divider value.	0
NR	12-8	PLL Input Divider value.	0
NF	7-0	PLL Feedback divider value.	0

5.2.5 DIP Clock Configuration Register

Table 36. DIP Clock Configuration Register

DIP Clock Configuration			
Address: 0x38		Reset = 0xa8147	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-22	Reserved	
pll_lock_status	20	This bit is a read only bit and reflects the “locking” status of the PLL. 0 = no lock, 1 = lock.	0
clock_disable	19	1 = The PLL is powered down and put in bypass mode 0 = The PLL is enabled	1
Range	18-16	This field must be set according the configured post reference divide frequency. 0=bypass, 1=10-16 Mhz, 2=16-26 Mhz, 3=26-42 Mhz, 4=42-65 Mhz, 5=65-104 Mhz, 6=104-166 Mhz, 7=166 Mhz+	d2
NO	15-13	PLL Output Divider value.	d4
NR	12-8	PLL Input Divider value.	d1
NF	7-0	PLL Feedback divider value.	d71

5.2.6 Memory Clock Configuration Register

Table 37. Memory Clock Configuration Register

Memory Clock Configuration			
Address: 0x4C		Reset = 0x10_0000	Type: RW
Name	**Bit**	**Function**	**Reset**
	31-22		
ddr_dll_disable	21	Some applications may not require the DDR DLLs. This bit will put the DLLs in the powered down state. 1 = The DLLs are powered down 0 = The DLLs are enabled	0
pll_lock_status	20	This bit is a read only bit and reflects the “locking” status of the PLL. 0 = no lock, 1 = lock.	1
clock_disable	19	1 = The PLL is powered down and put in bypass mode 0 = The PLL is enabled	0
Range	18-16	This field must be set according the configured post reference divide frequency. 0=bypass, 1=10-16 Mhz, 2=16-26 Mhz, 3=26-42 Mhz, 4=42-65 Mhz, 5=65-104 Mhz, 6=104-166 Mhz, 7=166 Mhz+	0

Table 37. Memory Clock Configuration Register

NO	15-13	PLL Output Divider value.	0
NR	12-8	PLL Input Divider value.	0
NF	7-0	PLL Feedback divider value.	0

5.2.7 Interface PLL Select Register

Table 38. Interface PLL Select Register

Interface PLL Select			
Address: 0xF8		**Reset = 0x1800**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
	31-28		
tvout_clk_gate	27	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0
usb_clk_gate	26	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0
ap_clk_gate	25	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0
mem_clk_gate	24	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0
sif_clk_gate	23	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0
xga_clk_gate	22	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0

Table 38. Interface PLL Select Register

if_clk_gate	21	This field controls the clock gating cell between the PLL clock source and the clock divide circuitry. If the PLL or ref_clk_sel is being updated, the clock gating cell must be activated first. 0 = normal operation. Clock output is not gated. 1 = Clock output is gated.	0x0
tvout_ref_clk_sel	20-18	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3=DIP PLL, 4=MEM PLL	0x0
usb_ref_clk_sel	17-15	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3=DIP PLL, 4=MEM PLL	0x0
ap_ref_clk_sel	14-12	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3= DIP PLL, 4=MEM PLL	0x1
mem_ref_clk_sel	11-9	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3= DIP PLL, 4=MEM PLL	0x4
sif_ref_clk_sel	8-6	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3= DIP PLL, 4=MEM PLL	0x0
xga_ref_clk_sel	5-3	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3= DIP PLL, 4=MEM PLL	0x0
if_ref_clk_sel	2-0	This field selects the PLL clock source. 0 = IF PLL, 1 = AP PLL, 2 = SYS PLL, 3= DIP PLL, 4=MEM PLL	0x0

5.2.8 PLL Divide Register

Table 39. PLL Divide Register

PLL Divide			
Address: 0xFC-114		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-9	Reserved	
divide_status	8	This is a read only field that provides an indicator as to when the programmed divide value has been applied. The divide value is applied when the “counter” passes through a “0” value so that the resultant output clock remains clean. Depending on the current count value and the previous clock frequency, there may be a delay before the new divide value gets applied to the output clock. This bit gets set when this register is written and slef clears after the new divide value has been applied.	0x0
Reserved	7-6	Reserved	

Table 39. PLL Divide Register

pll_divide	5-0	This field contains an integer divide that is applied to the selected ref_clk to produce the appropriate peripheral clock. 0 = div1 1 = div2 2 = div3 … 63 = div64	0x0

5.3 PAD and I/O registers

5.3.1 Alternate Function Enable Register

Table 40. Alternate Function Enable Register

Alternate Function Enable			
Address: 0xd003_0030		Reset = 0	Type: RW
Name	**Bit**	**Function**	**Reset**
reserved	31-29	reserved	
gps_ena5	28	0 = main function selected (mp2ts_clk) 1 = alternate function selected (gps_clk) mp2ts_clk = gps_clk	0
gps_ena4	27	0 = main function selected (uart_rts) 1 = alternate function selected (gps_m2) uart_rts = gps_m2	0
gps_ena3	26	0 = main function selected (mp2ts_sync) 1 = alternate function selected (gps_m1) mp2ts_sync = gps_m1	0
gps_ena2	25	0 = main function selected (mp2ts_valid) 1 = alternate function selected (gps_m0) mp2ts_valid = gps_m0	0
gps_ena1	24	0 = main function selected (mp2ts_d) 1 = alternate function selected (gps_s) mp2ts_d = gps_s	0
second_uart	23	0 = main function selected (dip_data[23:22]) 1 = alternate function selected (second uart) dip_data22 = uart1_rxd dip_data23 = uart1_txd	0
spi_mp2ts	22	0 = main function selected (2nd spi interface) 1 = alternate function selected (2nd mp2ts interface) spi1_sck = mp2ts_clk spi1_ssn = mp2ts_sync spi1_txd = mp2ts_valid spi1_rxd = mp2ts_d	0

Table 40. Alternate Function Enable Register

pwi interface	21	0 = main function selected (bb_audio_fsx,bb_audio_clkx) 1 = alternate function selected (pwi_clk,pwi_data) reserved_3 = pwi_clk reserved_4 = pwi_data	0
spi_device3	20	0 = main function selected (nand) 1 = alternate function selected (spi_device3) reserved_15 = spi_ssn3 sc_card_voltage = spi_rxd3	0
spi_device2	19	0 = main function selected (nand) 1 = alternate function selected (spi_device2) uart_rts = spi_ssn2 nand_cen3 = spi_rxd2	0
spi_device1	18	0 = main function selected (nand) 1 = alternate function selected (spi_device1) uart_cts = spi_ssn1 nand_cen2 = spi_rxd1	0
nand_or_mmc_plus	17	0 = main function selected (nand) 1 = alternate function selected (mmc_plus) mmc_plus _data[7:0] = nand_data[7:0] mmc_plus _clk = nand_cen0 mmc_plus _cmd = nand_cen1	0
uart_txd	16	0 = main function selected (uart_txd) 1 = alternate function selected (dip_ref_clk)	0
spi_sck	15	0 = main function selected (mmc_data3) 1 = alternate function selected (ac_clk)	0
spi_ssn	14	0 = main function selected (mmc_data2) 1 = alternate function selected (sys_clk)	0
spi_rxd	13	0 = main function selected (mmc_data1) 1 = alternate function selected (mem_ref_clk)	0
spi_txd	12	0 = main function selected (mmc_data0) 1 = alternate function selected (if_ref_clk)	0
reserved_14	11	0 = main function selected (reserved_14) 1 = alternate function selected (keyscan3_out)	0
reserved_13	10	0 = main function selected (reserved_13) 1 = alternate function selected (keyscan2_out)	0
reserved_12	9	0 = main function selected reserved_12) 1 = alternate function selected (keyscan1_out)	0
reserved_11	8	0 = main function selected (reserved_11) 1 = alternate function selected (keyscan0_out)	0
reserved_10	7	0 = main function selected (reserved_10) 1 = alternate function selected (keyscan3_in)	0
reserved_9	6	0 = main function selected (reserved_9) 1 = alternate function selected (keyscan2_in)	0

Table 40. Alternate Function Enable Register

reserved_8	5	0 = main function selected (reserved_8) 1 = alternate function selected (keyscan1_in)	0
reserved_7	4	0 = main function selected (reserved_7) 1 = alternate function selected (keyscan0_in)	0
dip_data17	3	0 = main function selected (dip_data17) 1 = alternate function selected (scl_sec)	0
dip_data16	2	0 = main function selected (dip_data16) 1 = alternate function selected (sda_sec)	0
audio_fsr	1	0 = main function selected (audio_fsr) 1 = alternate function selected (pwm_output2)	0
sc_fcb	0	0 = main function selected (sc_fcb) 1 = alternate function selected (pwm_output1)	0

5.3.2 Drive Strength Register

Table 41. Drive Strength Register

Drive Strength 1			
Address: 0xd003_0080		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
drive_strength[31-0]	31-0	A number of the external PADs have configurable drive strength. This register allows software to configure the drive strength as low or as high. The following table maps the drive_strength bits to the corresponding PAD that they control. 0 = low drive strength 1 = high drive strength	0
Drive Strength 2			
Address: 0xd003_0084		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
drive_strength[63-32]	31-0	A number of the external PADs have configurable drive strength. This register allows software to configure the drive strength as low or as high. The following table maps the drive_strength bits to the corresponding PAD that they control. 0 = low drive strength 1 = high drive strength	0
Drive Strength 3			
Address: 0xd003_0088		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**

Table 41. Drive Strength Register

drive_strength[95-64]	31-0	A number of the external PADs have configurable drive strength. This register allows software to configure the drive strength as low or as high. The following table maps the drive_strength bits to the corresponding PAD that they control. 0 = low drive strength 1 = high drive strength	0

The table below gives the correspondence Drive Strength register bit to pin.

Table 42. Drive Strength register bit to pin correspondence

Bit	Pin		Bit	Pin		Bit	Pin		Bit	Pin
0	scl_p		24	mclk_p		48	spi_sck_p		72	sc_fcb_p
1	sda_p		25	audio_clkr_p		49	spi_ssn_p		73	sc_io_p
2	sif_clkout_p		26	audio_fsr_p		50	spi_txd_p		74	sc_card_detect_p
3			27	audio_clkx_p		51	fodd_p		75	sc_power_on_p
4			28	audio_dr_p		52	sif_gpio_p		76	sc_card_voltage_p
5			29	audio_dx_p		53	nand_ren_p		77	nand_cen_p1
6			30	audio_fsx_p		54	nand_wen_p		78	nand_cen_p2
7	reserved_1		31	reserved_3		55	nand_ale_p		79	nand_cen_p3
8			32	reserved_4		56	nand_cle_p		80	
9			33	reserved_5		57	dip_data_p23		81	
10	reserved_7		34	reserved_6		58	dip_data_p22		82	

Table 42. Drive Strength register bit to pin correspondence

Bit	Pin		Bit	Pin		Bit	Pin		Bit	Pin
11	reserved_8		35	dip_data_p [15:0]		59	dip_data_p 21		83	
12	Reserved_ 9		36	dip_csn0_ p dip_csn1_ p dip_wen_p dip_rs_p		60	dip_data_p 20		84	
13	reserved_1 0		37	mmc_clk_p		61	dip_data_p 19		85	
14	reserved_1 1		38	mmc_cmd _p		62	dip_data_p 18		86	
15	reserved_1 2		39	mmc_data _p3		63	dip_data_p 17		87	
16	reserved_1 3		40	mmc_data _p2		64	dip_data_p 16		88	Utmiotg_dr vvbus_p
17	reserved_1 4		41	mmc_data _p1		65	dip_csn2_ p		89	spi1_rxd_p
18			42	mmc_data _p0		66	dip_csn3_ p		90	spi1_sck_p
19	reserved_2		43	nand_cen_ p0		67	dip_oen_p		91	spi1_ssn_p
20	sdram control		44	nand_data _p[7:0]		68	dip_pclk_p		92	spi1_txd_p
21	sdram_clk _p		45	uart_rxd_p		69	dip_cpu_vs ync_p		93	uart_cts_p
22	ebi_addr_p [12:0]		46	uart_txd_p		70	sc_clk_p		94	uart_cts_p
23	ebi_data_p 31:0		47	spi_rxd_p		71	sc_rst_p		95	reserved_1 5

5.3.3 PAD Resistor Enable

Table 43. PAD Resistor Enable

PAD Resistor Enable 1			
Address: 0xd003_008C		Reset = 0x0801_003c	Type: RW
Name	Bit	Function	Reset

Table 43. PAD Resistor Enable

pad_resistor_ena [31-0]	31-0	A number of the external PADs have configurable pull-up/pull-down resistors in the PAD. This register allows software to enable the resistor. The following table maps these register bits to the corresponding PADs that they control. 0 = PAD resistor disabled 1 = PAD resistor enabled	0x0801_003c
PAD Resistor Enable 2			
Address: 0xd003_0090		**Reset = 0x0008_4f02**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
pad_resistor_ena [63-32]	31-0	A number of the external PADs have configurable pull-up/pull-down resistors in the PAD. This register allows software to enable the resistor. The following table maps these register bits to the corresponding PADs that they control. 0 = PAD resistor disabled 1 = PAD resistor enabled	0x0008_4f02
PAD Resistor Enable 3			
Address: 0xd003_0094		**Reset = 0x01e7_f0d0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
pad_resistor_ena [95-64]	31-0	A number of the external PADs have configurable pull-up/pull-down resistors in the PAD. This register allows software to enable the resistor. The following table maps these register bits to the corresponding PADs that they control. 0 = PAD resistor disabled 1 = PAD resistor enabled	0x01e7_f0d0
PAD Resistor Enable 4			
Address: 0xd003_0098		**Reset = 0x0000_00bf**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
pad_resistor_ena [117-96]	31-0	A number of the external PADs have configurable pull-up/pull-down resistors in the PAD. This register allows software to enable the resistor. The following table maps these register bits to the corresponding PADs that they control. 0 = PAD resistor disabled 1 = PAD resistor enabled	0x0000_00bf

The table indicates whether the PAD has a pull-up or pull-down with the PU/PD nomenclature in the bit location field.

Table 44. PAD has a pull-up or pull-down

Bit	Pin	Bit	Pin	Bit	Pin	Bit	Pin	Bit	Pin
0-PD	sda_p	24	reserved	48-PD	fodd_p	72	reserved	97-PU	dip_data_p4
1-PD	sif_clkout_p	25	reserved	49-PD	mclk_p	73	reserved	98-PD	dip_data_p3
2	reserved	26	reserved	50-PD	sif_gpio_p	74	reserved	99-PD	dip_data_p2
3	reserved	27-PU	mmc_clk_p	51-PD	fclk_p rclk_p pclk_p sensor_data_p	75-PD	scl_p	100-PU	dip_data_p1
4-PU	utmiotg_drvvbus_p	28-PD	mmc_cmd_p	52	reserved	76-PD	ntrst_p	101-PD	dip_data_p0
5	reserved	29-PD	mmc_data_p3	53-PD	nand_data[7:0]	77-PU	tck_p	102-PD	dip_cpu_vsync_p
6-PD	sdram_rdy_p	30-PD	mmc_data_p2	54-PD	dip_data_p23	78-PU	tdi_p	103-PU	sc_clk_p
7	reserved	31-PD	mmc_data_p1	55-PD	dip_data_p22	79-PU	tms_p	104-PD	sc_rst_p
8	reserved	32-PD	mmc_data_p0	56-PD	dip_data_p21	80-PD	dip_data_p6	105-PD	sc_fcb_p
9	reserved	33-PU	nand_cen_p0	57-PD	dip_data_p20	81-PD	dip_data_p7	106-PD	sc_io_p
10	reserved	34-PD	nand_ren_p	58-PD	dip_data_p19	82-PD	dip_data_p8	107-PD	sc_card_detect_p
11	reserved	35-PD	nand_wen_p	59-PD	dip_data_p18	83-PD	dip_data_p9	108-PD	sc_power_on_p
12	reserved	36-PD	nand_ale_p	60-PD	dip_data_p17	84-PD	dip_data_p10	109-PD	sc_card_voltage_p
13	reserved	37-PD	nand_cle_p	61-PD	dip_data_p16	85-PU	nand_cen_p3	110-PD	dip_data_p11
14	reserved	38-PD	uart_rxd_p	62-PD	dip_data_p[15:12]	86-PU	nand_cen_p2	111	reserved
15	reserved	39-PD	uart_txd_p	63-PD	dip_oen_p	87-PU	nand_cen_p1		
16	reserved	40-PU	dip_csn0_p	64-PD	dip_pclk_p	88-PU	spi1_ssn_p		

Table 44. PAD has a pull-up or pull-down

17-PD	audio_clkr_p		41-PU	dip_csn1_p		65-PD	mp2ts_clk_p mp2ts_d_p mp2ts_valid_p mp2ts_sync_p		89-PD	spi1_txd_p		
18-PD	audio_fsr_p		42-PU	dip_csn2_p		66	reserved		90-PD	spi1_rxd_p		
19-PD	audio_clkx_p		43-PU	dip_csn3_p		67	reserved		91-PD	spi1_sck_p		
20-PD	audio_dr_p		44-PD	spi_rxd_p		68	reserved		92-PD	uart_cts_p		
21-PD	audio_dx_p		45-PD	spi_sck_p		69	reserved		93-PD	uart_rts_p		
22-PD	audio_fsx_p		46-PU	spi_ssn_p		70	reserved		94	reserved		
23	reserved		47-PD	spi_txd_p		71	reserved		95	reserved		

5.4 Reset and Clock Gating

The system reset bits are spread out into two registers: System Reset and System Reset1.

The system power down bits are spread out into two registers: System Power Down and System Power Down1.

5.4.1 System Power Down

This timing generation block has clock gating logic for most of the internal blocks. This register provides software with a mechanism to gate the clock of any block that is not required for the application at hand. This will reduce power for certain applications. When a block has its clock gated the block is “disabled” and unusable.

Table 45. System Power Down

System Power Down			
Address: 0x d003_001c		**Reset = 0xffed_d5ff**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31	Reserved	1
mp2ts1_pdown	30	When this bit is written “1” the peripheral has its clock gated.	1
spi1_pdown	29	When this bit is written “1” the peripheral has its clock gated.	1
Pwi_pdown	28	When this bit is written “1” the peripheral has its clock gated.	1
Mmcplus_pdown	27	When this bit is written “1” the peripheral has its clock gated.	1

Table 45. System Power Down

sequencer_pdown	26	When this bit is written “1” the peripheral has its clock gated. This powers down the GOC buffers and decoding” circuitry (VLD).	1
crypto_pdown	25	When this bit is written “1” the peripheral has its clock gated.	1
Smart_card_pdown	24	When this bit is written “1” the peripheral has its clock gated.	1
Rotator_pdown	23	When this bit is written “1” the peripheral has its clock gated.	1
H264_loop_pdown	22	When this bit is written “1” the peripheral has its clock gated.	1
bitblt_mini_pdown	21	When this bit is written “1” the peripheral has its clock gated.	1
ebi_pdown	20	When this bit is written “1” the peripheral has its clock gated.	0
bitblt_pdown	19	When this bit is written “1” the peripheral has its clock gated.	1
vld_pdown	18	When this bit is written “1” the peripheral has its clock gated.	1
cmem_if_pdown	17	When this bit is written “1” the peripheral has its clock gated.	0
ac_pdown	16	When this bit is written “1” the high speed cmem clock is gated.	1
mmc_pdown	15	When this bit is written “1” the peripheral has its clock gated.	1
dip_pdown	14	When this bit is written “1” the peripheral has its clock gated.	1
hpi_pdown	13	When this bit is written “1” the peripheral has its clock gated.	0
usb_pdown	12	When this bit is written “1” the peripheral has its clock gated.	1
nand_pdown	11	When this bit is written “1” the peripheral has its clock gated.	0
sif_pdown	10	When this bit is written “1” the peripheral has its clock gated.	1
spi_pdown	9	When this bit is written “1” the peripheral has its clock gated.	0
cmem_dma_pdown	8	When this bit is written “1” the peripheral has its clock gated.	1
entropy_pdown1	7	When this bit is written “1” the peripheral has its clock gated.	1
bm_pdown	6	When this bit is written “1” the peripheral has its clock gated. This powers down everything but the “decoding” circuitry (VLD).	1
be_pdown	5	When this bit is written “1” the peripheral has its clock gated.	1
multi_dma_pdown	4	When this bit is written “1” the peripheral has its clock gated.	1
Mpeg2_ts_pdown	3	When this bit is written “1” the peripheral has its clock gated.	1
uart_pdown	2	When this bit is written “1” the peripheral has its clock gated.	1
audio_pdown	1	When this bit is written “1” the peripheral has its clock gated.	1
i2c_pdown	0	When this bit is written “1” the peripheral has its clock gated.	1

5.4.2 System power down1

Table 46. System Power Down1

System Power Down1			
Address: 0xd003_00cc		Reset = 0xffff_ff3e	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31:8	Reserved	0xffff_ff
Sys_cmem_if_pdown	7	When this bit is written “1” the low speed cmem clock is gated.	0
system_pdown	6	When this bit is written “1” some of the system related blocks get their clock gated – boot_loader, keypad, pwm, gpio	0
entropy_pdown2	5	When this bit is written “1” the peripheral has its clock gated.	1
tcm_pdown	4	When this bit is written “1” the peripheral has its clock gated.	1
Reserved	3	Reserved	1
sbist_pdown	2	When this bit is written “1” the peripheral has its clock gated.	1
uart1_pdown	1	When this bit is written “1” the peripheral has its clock gated.	1
arm2_pdown	0	When this bit is written “1” the peripheral has its clock gated.	0

5.4.3 System Reset Register

This register provides a mechanism for software to reset any or all of the internal hardware components. Software controls the assertion and de-assertion of the reset for all peripherals except the ARM926EJ-S processor and the EBI block.

Table 47. System Reset Register

System Reset			
Address: 0xd003_0018		Reset = 0	Type: RW
Name	**Bit**	**Function**	**Reset**
suicide	31	Writing a “1” to this bit will pulse the internal global reset. The entire chip will be reset as if the external reset signal was asserted. The bit is self resetting and always returns “0” when read.	0
mp2ts1_rst	30	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
spi1_rst	29	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
pwi_rst	28	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
mmcplus_rst	27	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0

Table 47. System Reset Register

sequencer_rst	26	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
crypto_rst	25	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
Smart_card_rst	24	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
Rotator_rst	23	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
H264_loop_rst	22	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
bitblt_mini_rst	21	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
ebi_rst	20	Writing a “1” to this bit will pulse the EBI block reset. The bit is self resetting and always returns “0” when read.	0
bitblt_rst	19	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
vld_rst	18	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
cmem_if_rst	17	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
ac_rst	16	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
mmc_rst	15	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
dip_rst	14	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
hpi_rst	13	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
usb_rst	12	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
nand_rst	11	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
sif_rst	10	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
spi_rst	9	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
cmem_dma_rst	8	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
entropy_rst	7	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0

Table 47. System Reset Register

bm_rst	6	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
be_rst	5	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
multi_dma_rst	4	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
Mpeg2_ts_rst	3	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
uart_rst	2	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
audio_rst	1	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0
i2c_rst	0	When this bit is written “1” the peripheral in held in reset until this bit is written “0”.	0

5.4.4 System Reset1 Register

Table 48. System Reset1 Register

System Reset1			
Address: 0xd003_00d0		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
axi_fabric_rst	3	When this bit is written “1” the peripheral is held in reset.	0
sbist_rst	2	When this bit is written “1” the peripheral is held in reset.	0
uart1_rst	1	When this bit is written “1” the peripheral is held in reset.	0
arm2_rst	0	When this bit is written “1” the peripheral is held in reset.	0

5.5 Miscellaneous

5.5.1 Chip ID Register

Table 49. Chip ID Register

chip ID		
Address: 0xd003_0028	**Reset = 0**	**Type: RO**

Table 49. Chip ID Register

Name	Bit	Function	Reset
Reserved	31-7	Reserved	
chip_id	6-4	This field reflects the bond-out option for the jtag_sel_p PADs. It can be used by software to differentiate different chips for marketing purposes.	0
chip_rev_num	3-0	This field reflects the silicon revision of the chip.	0

Table 50.

5.6 Memory Controller

5.6.1 Memory Controller Register Description

Register	Address Offset	Mode
memc_status	0x000	RO
memc_cmd	0x004	WO
direct_cmd	0x008	WO
memory_cfg	0x00C	RW
refresh_prd	0x010	RW
cas_latency	0x014	RW
Tdqss	0x018	RW
Tmrd	0x01C	RW
Tras	0x020	RW
Trc	0x024	RW
Trcd	0x028	RW
Trfc	0x02C	RW
Trp	0x030	RW
Trrd	0x034	RW
Twr	0x038	RW
Twtr	0x03C	RW
Txp	0x040	RW
Txsr	0x044	RW
Tesr	0x048	RW
memory_cfg2	0x04C	RW
memory_cfg3	0x050	RW
reserved	0x054-0xFF	RW

id_0_cfg	0x100	RW
id_1_cfg	0x104	RW
id_2_cfg	0x108	RW
id_3_cfg	0x10C	RW
id_4_cfg	0x110	RW
id_5_cfg	0x114	RW
id_6_cfg	0x118	RW
id_7_cfg	0x11C	RW
id_8_cfg	0x120	RW
id_9_cfg	0x124	RW
id_10_cfg	0x128	RW
id_11_cfg	0x12C	RW
id_12_cfg	0x130	RW
id_13_cfg	0x134	RW
id_14_cfg	0x138	RW
id_15_cfg	0x13C	RW
reserved	0x140-0x1FF	RW
chip_0_cfg	0x200	RW
reserved	0x204-0xFDF	RW
Periph_id_0	0xFE0	RO
Periph_id_1	0xFE4	RO
Periph_id_2	0xFE8	RO
Periph_id_3	0xFEC	RO
Pcell_id_0	0xFF0	RO
Pcell_id_1	0xFF4	RO
Pcell_id_2	0xFF8	RO
Pcell_id_3	0xFFC	RO

5.6.2 memc_status register

This register provides information on the configuration of the memory controller and also on the state of the memory controller.

memc_status		
Address: 0x000	**Reset = 0xe34**	**Type: RO**

Name	Bit	Function	Reset
Reserved	31-13	Reserved	
Memory_banks1	12	See bit 9.	d1
Exclusive_monitors	11-10	Returns the number of exclusive access monitor resources implemented in the controller. 00=0 monitor, 01=1 monitor, 10=2 monitors , 11=4 monitors	d3
Memory_banks0	9	This returns the maximum number of banks that the controller supports. 00 = 2 banks 01 = 4 banks 10,11 = reserved	d1
Memory_chips	8-7	This returns the number of chip selects that the controller supports. 00=1 chip, 01=2 chips, 10=3 chips, 11=4 chips	d0
Memory_ddr	6-4	This returns the type of memory controller. 000=SDR sdram, 001=DDR sdram, 011=mobile DDR sdram If mobile DDR sdram or SDR sdram is supported, the cas_half_cycle bit at address offset 0x14 is ignored.	d11
Memory_width	3-2	This returns the width of the external memory. 00=16bit, 01=32bit, 10=64bit	d1
Memc_status	1-0	This returns the state of the memory controller. 00=config, 01=ready, 10=paused, 11=low power	

5.6.3 memc_cmd register

This registers controls the state of the FSM within the controller. By writing to this register the FSM can be traversed. If a new command is received to change state and a previous command has not be completed, the APB3 pready signal is held LOW (bus cycle is waited) until the new command can be carried out.

Memc_cmd			
Address: 0x004		Reset = 0x0	Type: WO

Name	Bit	Function	Reset
Reserved	31-3	Reserved	
Memc_cmd	2-0	Changes the state of the memory controller. 000=go, 001=sleep, 010=wakeup, 011=pause, 100=configure, 111=active pause Active_pause puts the controller into a paused state without draining the arbiter queue. This enables you to enter low-power mode to change configuration settings such as memory frequency or timing register values without requiring co-ordination between masters in a multi-master system. If the controller is put into low-power mode after using the active_pause command, you may not remove power from the controller because this results in data loss and violation of the AXI protocol. The controller does not issue refreshes while in the config state. It is, therefore, recommended that you use the low-power mode to make register updates because this ensures that the memory is put into self-refresh rather than entering the config state when the memory contains valid data.	d0

5.6.4 direct_cmd register

This register passes commands to the external memory. The configuration of this register enables you to write to any type of mode register supported by the external memory device and also to generate NOP, prechargeall and auto refresh commands. This register therefore enables any initialization sequence that an external memory device might require. The only timing information associated with this register are the command delays defined in the timing registers. Therefore, if an initialization sequence requires additional delays between commands, they must be timed by the master driving the initialization sequence. The register can only be written to in the config or low-power state.

Direct_cmd			
Address: 0x08	**Reset = 0c0**	**Type: WO**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-23	Reserved	
ext_memory_cmd	22	See bit 19-18.	d0
Chip_number	21-20	Bits mapped to external memory chip address bits.	d0
Memory_cmd	19-18	Determines the command required. 000=prechargeall, 001=auto-refresh, 010=modereg or extended modereg access, 011=NOP,100=deep power down	d0
Bank_addr	17-16	Bits mapped to external memory bank address bits when command is modereg access.	d0
Reserved	15-14	Reserved	
Addr_13_to_0	13-0	Bits mapped to external memory address bits [13:0] when command is modereg access.	d0

5.6.5 memory_cfg register

This register configures the memory. It can only be read/written in the config or low-power state.

Memory_cfg			
Address: 0x0C	**Reset = 0x10020**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
sr_enable	31	Auto self refersh entry.	d0
fp_time	30-24	Force precharge timeout count.	d0
fp_enable	23	Force precharge enable.	d0
Active_chips	22-21	Enables the refresh command generation for the number of memory chips. It is only possible to generate commands up to and including the number of chips in the configuration that the memc_status register defines. 00=1 chip, 01=2 chips, 10=3 chips, 11=4 chips	d0

Qos_master_bits	20-18	Encodes the 4 bits of the 8 bit AXI ARID that select one of the 16 QOS values. 000=ARID[3:0], 001=ARID[4:1], 010=ARID[5:2], 011=ARID[6:3], 100=ARID[7:4]	d0
Memory_burst	17-15	Encodes the number of data accesses that are performed to the SDRAM for each read or write command. 000=burst1, 001=burst2, 010=burst4, 011=burst8, 100=burst16 The value must also be programmed into the SDRAM mode register using the direct_cmd register at offset 0x8 and must match it.	d3
Stop_mem_clk	14	When enabled, the memory clock is dynamically stopped when not performing an access to the SDRAM.	d0
Auto_power_down	13	When this is set, the memory interface automatically places the SDRAM into the power-down state by de-asserting CKE when the command FIFO has been empty for the PowerDownPrd memory clock cycles.	d0
Power_down_prd	12-7	Number of memory clock cycles for auto power-down of the SDRAM.	d0
Ap_bit	6	Encodes the position of the auto-precharge bit in the memory address. 0=addr10, 1=addr8	d2
Row_bits	5-3	Encodes the number of the AXI address that comprise the row address. 000=11bits, 001=12bits, 010=13bits, 011=14bits, 100=15bits, 101=16bits The combination of row size, column size, BRC/RBC and memory width must ensure that neither the MSB of the row address nor the MSB of the bank address exceed address range [27:0].	d0
Column_bits	2-0	Encodes the number of the AXI address that comprise the column address. 000=8bits, 001=9bits, 010=10bits, 011=11bits, 100=12bits	d0

5.6.6 refresh_prd register

This sets the memory refresh period. It can only be read/written to in the config or low-power state.

Refresh_prd			
Address: 0x10	**Reset = 0xa60**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-15	Reserved	
Refresh_prd	14-0	Memory refresh period in memory clock cycles.	0xa60

5.6.7 cas_latency register

This sets the cas_latency in memory clock cycles. It can only be read/written to in the config or low-power state.

Cas_latency		
Address: 0x14	**Reset = 0x6**	**Type: RW**

Name	Bit	Function	Reset
Reserved	31-4	Reserved	
Cas_latency	3-1	CAS latency in memory clock cycles.	d3
Cas_half_cycle	0	Encodes whether the CAS latency is half a memory clock cycle more than the value given in bite[3:1]. 0=zero cycle offset (is forced in MDDR and SDR mode) 1=half cycle offset to value in [3:1]	d0

5.6.8 Tdqss register

It can only be read/written to in the config or low-power state.

Tdqss			
Address: 0x18		**Reset = 0x1**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-2	Reserved	
Tdqss	1-0	Write to DQS in memory clock cycles.	0x1

5.6.9 Tmrd register

It can only be read/written to in the config or low-power state.

Tmrd			
Address: 0x1C		**Reset = 0x2**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-7	Reserved	
Tmrd	6-0	Sets mode register command time in memory clock cycles.	0x2

5.6.10 Tras Register

It can only be read/written to in the config or low-power state.

Tras			
Address: 0x20		**Reset = 0x7**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
Tras	3-0	Sets RAS to precharge delay in memory clock cycles.	0x7

5.6.11 Trc Register

It can only be read/written to in the config or low-power state.

Tdqss			
Address: 0x24		Reset = 0xB	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
Trc	3-0	Sets active bank x to active bank x delay in memory clock cycles.	0xB

5.6.12 Trcd Register

It can only be read/written to in the config or low-power state.

Trcd			
Address: 0x28		Reset = 0x1D	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-6	Reserved	
Schedule_Trcd	5-3	Sets the RAS to CAS minimum delay in aclk cycles – 3.	0x5
Trcd	2-0	Sets the RAS to CAS minimum delay in memory clock cycles.	0x5

5.6.13 Trfc Register

It can only be read/written to in the config or low-power state.

Trfc			
Address: 0x2C		Reset = 0x212	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-10	Reserved	
Schedule_Trfc	9-5	Sets the auto-refresh command time in aclk cycles - 3.	0x10
Trfc	4-0	Sets the auto-refresh command time in memory clock cycles.	0x12

5.6.14 Trp Register

It can only be read/written to in the config or low-power state.

Trp			
Address: 0x30		Reset = 0x1d	Type: RW
Name	**Bit**	**Function**	**Reset**

Reserved	31-6	Reserved	
Schedule_Trp	5-3	Sets the precharge to RAS delay in aclk cycles - 3.	0x5
Trp	2-0	Sets the precharge to RAS delay in memory clock cycles.	0x5

5.6.15 Trrd Register

It can only be read/written to in the config or low-power state.

Trrd			
Address: 0x34		**Reset = 0x2**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
Trrd	3-0	Sets active bank x to active bank y delay in memory clock cycles.	0x2

5.6.16 Twr Register

It can only be read/written to in the config or low-power state.

Twr			
Address: 0x38		**Reset = 0x3**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-3	Reserved	
Twr	2-0	Sets the write to precharge delay in memory clock cycles.	0x3

5.6.17 Twtr Register

It can only be read/written to in the config or low-power state.

Twtr			
Address: 0x3C		**Reset = 0x2**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-3	Reserved	
Twtr	2-0	Sets the write to read delay in memory clock cycles.	0x2

5.6.18 TxP Register

It can only be read/written to in the config or low-power state.

Txp		

Address: 0x40		Reset = 0x1	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Txp	7-0	Sets the exit power-down command time in memory clock cycles.	0x1

5.6.19 Txsr Register

It can only be read/written to in the config or low-power state.

Txsr			
Address: 0x44		**Reset = 0xa**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Txsr	7-0	Sets the exit self-refresh command time in memory clock cycles.	0xa

5.6.20 Tesr Register

It can only be read/written to in the config or low-power state.

Tesr			
Address: 0x48		**Reset = 0x14**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Tesr	7-0	Sets the self-refresh command time in memory clock cycles.	0x14

5.6.21 Memory_cfg2 Register

It can only be read/written to in the config or low-power state.

Memory_cfg2			
Address: 0x4c		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-11	Reserved	
Read_delay	10-9	Sets the latency in clocks cycles of the PAD interface.	0x0

memory_type	8-6	Sets the memory type. 000 = SDR 001 = DDR 010 = eDRAM 011 = LPDDR	0x0
memory width	5-4	Sets the width of the external memory. 00 = 16 bit 01 = 32 bit 10 = 64 bit 11 = reserved	0x0
cke_init	3	Sets the level of the cke output after reset.	0x0
dqm_init	2	Sets the level of the dqm outputs after reset.	0x0
a_gt_m_sync	1	Required to be set high when aclk and mclk are running synchronous but when aclk running faster than mclk.	0x0
sync	0	Set high when aclk and mclk are synchronous.	0x0

5.6.22 Memory_cfg3 Register

It can only be read/written to in the config or low-power state.

Memory_cfg3			
Address: 0x50		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-12	Reserved	
prescale	11-3	Prescaler counter value.	0x0
max_outs_refs	2-0	Maximum number of outstanding refresh commands.	0x0

5.6.23 Id_x_cfg Registers

It can only be read/written to in the config or low-power state. For reference, the Ids for the masters are as follows:
Id = 0x00 – bridge from primary AHB control bus
Id = 0x20 – bridge from USB master
Id = 0x40 – bridge from ARM926EJ-S processor instruction bus
Id = 0x60 – MC-dma
Id = 0x80 – rotator
Id = 0xa0 – bitblt
Id = 0xc0 – bitblt_mini
Id = 0xe0 – bridge from secondary AHB control bus

Id_x_cfg		
Address: 0x100-0x13c	**Reset = 0x0**	**Type: RW**

Name	Bit	Function	Reset
Reserved	31-10	Reserved	
Qos_max	9-2	Sets a maximum QoS.	0x0
Qos_min	1	Sets a minimum Qos.	0x0
Qos_enable	0	Enables a QoS value to be applied to memory reads from address IS x.	0x0

5.6.24 chip_0_cfg Register

It can only be read/written to in the config or low-power state.

Chip_0_cfg			
Address: 0x200	**Reset = 0xff00**	**Type: RW**	
Name	**Bit**	**Function**	Reset
Reserved	31-17	Reserved	
Brc_n_rbc	16	Selects the memory organization as decoded from the AXI address. 0=row,bank,column organization 1= bank,row,column organization	0x0
Address_match	15-8	Comparison value for AXI address bits [31:24] to determine the chip that is selected.	0xFF
Address_mask	7-0	The mask for the AXI address bite [31:24] to determine the chip that is selected. 1=corresponding address bit is to be used for comparison.	0x0

5.6.25 Peripheral Identification 0-3 registers

Peripheral_id0			
Address: 0xfe0	**Reset = 0x40**	**Type: RO**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Part_number	7-0	Primecell part number.	0x40

Peripheral_id1			
Address: 0xfe4	**Reset = 0x13**	**Type: RO**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
designer	7-4	Primecell designer.	0x1

Part_number	3-0	Primecell part number.	0x3

Peripheral_id2			
Address: 0xfe8	Reset = 0x14	Type: RO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
revision	7-4	Primecell revision number.	0x1
designer	3-0	Primecell designer.	0x4

Peripheral_id3			
Address: 0xfec	Reset = 0x0	Type: RO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
Customer_mod	3-0	Customer Modified number.	0x0

5.6.26 Primecell Identification 0-3 registers

Pcell_id0			
Address: 0xff0	Reset = 0xD	Type: RO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Id_number	7-0	Id_number	0xD

Pcell_id1			
Address: 0xff4	Reset = 0xF0	Type: RO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Id_number	7-0	Id_number	0xF0

Pcell_id2		
Address: 0xff8	Reset = 0x5	Type: RO

Name	Bit	Function	Reset
Reserved	31-8	Reserved	
Id_number	7-0	Id_number	0x5

Pcell_id3		
Address: 0xffc	Reset = 0xB1	Type: RO

Name	Bit	Function	Reset
Reserved	31-8	Reserved	
Id_number	7-0	Id_number	0xB1

5.7 NAND Interface Registers Description

5.7.1 NAND Register Map

The register map is summarized below and described in the following sections.

Table 51. NAND Register Map

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO
Interface Timing	0x0C	RW
NAND Configuration	0x10	RW
NAND Action	0x14	RW
NAND Command	0x18	RW
NAND Address	0x1C	RW
NAND Address (extended)	0x20	RW
NAND Read	0x24	RO
NAND Write	0x28	WO
NAND Status	0x2C	RO
NAND Simple ECC result	0x30-0x3C	RO
NAND Generated Simple ECC	0x40-0x5C	RO
FIFO status	0x60	RO/WO
FIFO flag Configuration	0x64	RW

Table 51. NAND Register Map

RS ECC config	0x68	RW
RS ECC Read Parity1	0x6C	RW
RS ECC Read Parity2	0x70	RW
RS ECC Read Parity3	0x74	RW
RS ECC Write Parity1	0x78	RO
RS ECC Write Parity2	0x7C	RO
RS ECC Write Parity3	0x80	RO
RS ECC Status	0x84	RO
reserved	0x88-0xfff	
transmit FIFO	0x1000-0x1fff	WO
receive FIFO	0x2000-0x2fff	RO

5.7.2 Interrupt Source Register

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Table 52. Interrupt Source Register

Interrupt Source Register			
Address: 0x00		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-15	Reserved	
corrected_data_done	14	This interrupt is asserted (if using the Reed Solomon ECC) when the corrected data for a 512 byte block has been write to the input FIFO.	0x0
read_ecc_complete	13	Indicates that the Reed Solomon ECC engine has finished checking the last read 512 byte block. The RS ECC status register will now contain the ECC results for that read block.	0x0
write_par_complete	12	Indicates that the Reed Solomon ECC engine has generated the write parity and it is available in the write parity registers.	0x0
nand_block_write	11	Indicates that a NAND block write operation is completed	0x0
nand_block_read	10	Indicates that a NAND block read operation is completed	0x0
tx_pop_error	9	asserted when the transmit FIFO experiences an overrun condition or misaligned access. This error is from the perspective of the external interface.	0x0

Table 52. Interrupt Source Register

rx_push_error	8	asserted when the receive FIFO experiences an underrun condition or misaligned access. This error is from the perspective of the external interface.	0x0
dma_pop_error	7	asserted when the receive FIFO experiences an underrun condition or misaligned access. This error is from the perspective of the internal APB bus.	0x0
dma_push_error	6	asserted when the transmit FIFO experiences an overrun condition or misaligned access. A misaligned access can occur if the width of the write has changed from a previous access. For example, if byte writes have previously been used, the number of writes may be non-multiples of 32 bits. If a 32 bit write now occurs, this is a misaligned access because the byte pointers in the FIFO are not pointing to byte '0'. This error is from the perspective of the internal APB bus.	0x0
rx_ff	5	asserted when the receive FIFO has become full	0x0
rx_hf	4	asserted when the receive FIFO level (amount of bytes in the FIFO) is above the software configured "half" empty level.	0x0
rx_fe	3	asserted when the receive FIFO has become NOT empty	0x0
tx_ff	2	asserted when the transmit FIFO has become NOT full	0x0
tx_hf	1	asserted when the transmit FIFO level (amount of space available) is above the software configured "half" full level.	0x0
tx_fe	0	asserted when the transmit FIFO has become empty	0x0

5.7.3 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Table 53. Interrupt Mask Register

Interrupt Mask Register			
Address: 0x04		**Reset = 0xFFFF_FFFF**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-15	Reserved	
corrected_data_done	14	Masks the interrupt. 1=mask, 0=unmask.	0x1
read_ecc_complete	13	Masks the interrupt. 1=mask, 0=unmask.	0x1
write_par_complete	12	Masks the interrupt. 1=mask, 0=unmask.	0x1
nand_block_write	11	Masks the interrupt. 1=mask, 0=unmask.	0x1

Table 53. Interrupt Mask Register

nand_block_read	10	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_pop_error	9	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_push_error	8	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_pop_error	7	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_push_error	6	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_ff	5	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_hf	4	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_fe	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_ff	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_hf	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_fe	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.7.4 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Table 54. Interrupt Clear Register

Interrupt Clear Register			
Address: 0x08		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-15	Reserved	
corrected_data_done	14	Clears the interrupt when written ‘1’.	0x0
read_ecc_complete	13	Clears the interrupt when written ‘1’.	0x0
write_par_complete	12	Clears the interrupt when written ‘1’.	0x0
nand_block_write	11	Clears the interrupt when written ‘1’.	0x0
nand_block_read	10	Clears the interrupt when written ‘1’.	0x0
tx_pop_error	9	Clears the interrupt when written ‘1’.	0x0
rx_push_error	8	Clears the interrupt when written ‘1’.	0x0
dma_pop_error	7	Clears the interrupt when written ‘1’.	0x0
dma_push_error	6	Clears the interrupt when written ‘1’.	0x0
rx_ff	5	Clears the interrupt when written ‘1’.	0x0
rx_hf	4	Clears the interrupt when written ‘1’.	0x0

Table 54. Interrupt Clear Register

rx_fe	3	Clears the interrupt when written ‘1’.	0x0
tx_ff	2	Clears the interrupt when written ‘1’.	0x0
tx_hf	1	Clears the interrupt when written ‘1’.	0x0
tx_fe	0	Clears the interrupt when written ‘1’.	0x0

5.7.5 Interface Timing

All timing parameters refer to increments of the internal reference clock and a value of .0. means 1x refclk, a value of .1. means 2x refclk, etc.

Table 55. Interface Timing

Interface Timing			
Address: 0x0C		**Reset = 0x01001000**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Chip_select_sel	31-28	The nand interface supports 4 external chip selects (of which only one can be active at any one time). This “one-hot” field indicates which external chip select is active. The chip select that is active is turned on and off by the controlling bits in the NAND Action register. 0001 – chip select 0 selected 0010 – chip select 1 selected 0100 – chip select 2 selected 1000 – chip select 3 selected Any other value will disabled all chip selects.	0x0
t7	27-24	Indicates the number of system clocks from NAND_CLE/NAND_ALE inactive to NAND_ALE/ NAND_CLE active. Also indicates the number of system clocks from NAND_ALE inactive to NAND_WE/NAND_RE active. A value of ‘0’ is invalid for this field.	0x1
t6	23-20	Indicates the number of system clocks for the NAND_RE inactive pulse width	0x0
t5	19-16	Indicates the number of system clocks for the NAND_RE active pulse width	0x0
t4	15-12	Indicates the number of system clocks for the NAND_WE inactive pulse width. A value of ‘0’ is invalid for this field.	0x1
t3	11-8	Indicates the number of system clocks from NAND_WE inactive to NAND_CLE inactive	0x0
t2	7-4	Indicates the number of system clocks for the NAND_WE active pulse width	0x0
t1	3-0	Indicates the number of system clocks from NAND_CLE active to NAND_WE active	0x0

5.7.6 NAND configuration

Table 56. NAND Configuration

NAND Configuration			
Address: 0x10		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
nand_read_ena	31	The “NAND Read” register initiates an external access when the register is read. There may be circumstances where this is undesirable (i.e. debug). This bit will disable the operation of that register. When operation is disabled a read of the “NAND Read” register will complete but the data will be invalid. 0 = “NAND Read” register accesses do not initiate external NAND cycles. 1 = Normal operation for “NAND Read” register accesses.	0x0
nand_if_ena	30	Enables the internal state machine as well as the external PADs for the control signals. 0 = interface disabled 1 = interface enabled	0x0
	29		
ecc_ena	28	Enables the writing (for block writes) and checking (for block reads) of ECC. Refer to the hardware description for a more detailed explanation of ECC generation and checking. This enable bit is only pertinent to the “simple” ECC method.	0x0
addr_size	27-24	Indicates how many bytes are associated with an address transaction.	0x0
spare_size	23-16	Indicates the number of bytes in the spare or redundant area of the NAND Flash.	0x0
page_size	15-0	Indicates the number of bytes in the page of data associated with the NAND flash.	0x0

5.7.7 NAND Action

Table 57. NAND Action

NAND Action			
Address: 0x14		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
write_kick	3	Initiates a block write transaction. The NAND hardware interface will do a series of writes that totals the "page_size" + "ecc_size" + "spare_size". Once completed, the NAND_page_write interrupt is asserted. This action will commence after this bit is set and data is present in the transmit FIFO. The bit is self resetting and always returns 0 when read.	0x0

Table 57. NAND Action

read_kick	2	Initiates a block read transaction. The NAND hardware interface will do a series of reads that totals the "page_size" + "ecc_size" + "spare_size". Once completed, the NAND_page_read interrupt is asserted. This action will commence immediately after this bit is set. Software must ensure that the NAND device is available for a page read operation prior to issuing the "kick". The bit is self resetting and always returns 0 when read.	0x0
ce_dis	1	Setting this bit to '1' de-asserts the NAND Flash chip enable. This operation must be done after NAND accesses are completed and the NAND Flash is effectively idle. The bit is self resetting and always returns 0 when read.	0x0
ce_ena	0	Setting this bit to '1' asserts the NAND Flash chip enable. This operation must be done before any NAND Flash reading or writing is initiated. The bit is self resetting and always returns 0 when read.	0x0

5.7.8 NAND command

Table 58. NAND command

NAND Command			
Address: 0x18		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
command	7-0	This register contains and initiates a command cycle to the NAND Flash. Writing a command to the register initiates an external command access. Reading the register will provide the value of the previous command that was issued.	0x0

5.7.9 NAND address

This register contains and initiates a series of address cycles to the NAND Flash. The number of address cycles initiated is controlled by the "addr_size" field in the NAND configuration register. If the number of address bytes is greater than 4, writing to this register does not initiate any external cycles. Instead the write to the extended NAND address register initiates the external address cycles. Reading the register will provide the value of the previous address that was issued.

Table 59. NAND Address

NAND Address			
Address: 0x1C		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
addr3	31-24	4th address byte	0x0
addr2	23-16	3rd address byte	0x0
addr1	15-8	2nd address byte	0x0
addr0	7-0	1st address byte	0x0

5.7.10 NAND address (extended)

Table 60. NAND address (extended)

NAND Address (extended)			
Address: 0x20		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
addr7	31-24	8th address byte	0x0
addr6	23-16	7th address byte	0x0
addr5	15-8	6th address byte	0x0
addr4	7-0	5th address byte	0x0

5.7.11 NAND Read

Table 61. NAND Read

NAND Read			
Address: 0x24		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
	31-16		

Table 61. NAND Read

read_data	15-0	This register, when read will initiate a read cycle to the NAND Flash. It is intended as a status polling mechanism and must be used in conjunction with appropriate command and address sequencing. The MSB is only applicable if 16 bit NAND Flash is enabled.	0x0

5.7.12 NAND Write

Table 62. NAND Write

NAND Write			
Address: 0x28		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
	31-16		
write_data	15-0	This register, when written will initiate a write cycle to the NAND Flash. It is intended as an alternative to a datapath driven by the FIFO. This register is probably only pertinent to debug or NAND maintenance operations. The MSB is only applicable if 16 bit NAND Flash is enabled.	0x0

5.7.13 NAND Status

This register contains status information associated with NAND read activity. ECC generation and checking is done on 256 byte blocks. Error results are stored for each 256 byte block (for instance if the NAND page size is 2048 bytes then there are 8 result fields and error bits that are pertinent to this activity).

Table 63. NAND Status

NAND Status			
Address: 0x2C		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-19	Reserved	
Write_pending	18	When '1', this bit indicates that a single write cycle is still in progress and is not yet completed. If back to back writes need to be issued this bit should be polled and the second write not written until the bit indicates a completion.	0x0
address_pending	17	When '1', this bit indicates that an address cycle is still in progress and is not yet completed. If back to back addresses need to be issued this bit should be polled and the second address not written until the bit indicates a completion.	0x0

Table 63. NAND Status

command_pending	16	When '1', this bit indicates that a command is still in progress and is not yet completed. If back to back commands need to be issued this bit should be polled and the second command not written until the bit indicates a completion.	0x0
ecc_error	15-8	Indicates that there is an un-correctable error and the data is incorrect. There is one bit per 256 byte block read from memory. This is only pertinent to the "simple" ECC method.	0x0
data_error	7-0	Indicates that there is a correctable data error. The ecc_result field registers provide enough information to correct the error. There is one bit per 256 byte block read from memory. This is only pertinent to the "simple" ECC method.	0x0

5.7.14 NAND Simple ECC result

These registers contain the result fields from the NAND ECC checking. These fields are only applicable when ECC is enabled and a block read has taken place. There is a valid result field for every 256 byte block written to the NAND page.

Table 64. NAND Simple ECC result

NAND Simple ECC result			
Address: 0x30-3C		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
	31-27		
ecc_result2 ecc_result4 ecc_result6 ecc_result8	26-16	ecc result field for additional 256 byte blocks	0x0
	15-11		
ecc_result1 ecc_result3 ecc_result5 ecc_result7	10-0	The result field is pertinent when a correctable error has been detected. The NAND status register indicates whether an error has occurred and whether the error is correctable. When the error is correctable (a single bit error in the 256 byte block) the result provides enough information so that it can be corrected. eccx_result[7:0] contains the byte address (within the 256 byte block) that has the bit error. eccx_result[10:8] indicates which bit within the byte is incorrect. To fix the error, software must invert the bit indicated by this result field.	0x0

5.7.15 NAND Generated Simple ECC

These registers contain the generated ECC resulting from a block write or a block read. The registers contain generated ECC whether or not ECC has been enabled. The intent of these registers are to provide flexibility to software. If ECC is enabled, the contents of these registers are automatically written to flash (for a block write) immediately following the block of data. If the application requires the ECC to be in a location other than the first set of bytes in the redundant block, ECC should be disabled so that hardware does not automatically write the ECC.

Software can then read the generated ECC from these registers and write it to a different location in the redundant block. For a block read, the generated ECC and the stored ECC can be applied to the same validation algorithm that the hardware employs to determine how to correct bit errors.

Table 65. NAND Generated Simple ECC

NAND Generated Simple ECC			
Address: 0x40-5C		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
Reserved	31-24	Reserved	
ecc_generated1 ecc_generated2 ecc_generated3 ecc_generated4 ecc_generated5 ecc_generated6 ecc_generated7 ecc_generated8	23-0	The result field contains the generated ECC for the appropriate 256 byte block. This is only pertinent to the “simple” ECC method.	0x0

5.7.16 FIFO Status

Table 66. FIFO Status

FIFO status			
Address: 0x60		Reset = 0x80	Type: RO/WO
Name	**Bit**	**Function**	**Reset**
rx_flush	31	When this bit is written ‘1’, the receive FIFO is flushed. This bit is a write only bit.	0x0
tx_flush	30	When this bit is written ‘1’, the transmit FIFO is flushed. This bit is a write only bit.	0x0
Reserved	29-14	Reserved	
rx_byte_count	15-8	Indicates how many bytes of data is present in the receive FIFO. This is a read only field.	0x0
tx_byte_count	7-0	Indicates how many bytes of free space is available in the transmit FIFO. This is a read only field.	0x80

5.7.17 FIFO Flag Configuration

Table 67. FIFO Flag Configuration

FIFO flag Configuration		

Table 67. FIFO Flag Configuration

Address: 0x64		Reset = 0x28_4040	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-22	Reserved	
rx_FIFO_size	21:20	Although the asynchronous FIFO supports reads of varying sizes (8,16,32 or 64) the size must be configured prior to using the FIFO. This size refers to the side of the FIFO that the internal bus or dma engine reads from. If this field is being updated, the FIFO must be flushed to ensure that the internal pointers are properly aligned. 00 = 8 bit 01 = 16 bit 10 = 32 bit 11 = 64 bit	0x10
tx_FIFO_size	19-18	Although the asynchronous FIFO supports writes of varying sizes (8,16,32 or 64) the size must be configured prior to using the FIFO. This size refers to the side of the FIFO that the internal bus or dma engine writes to. If this field is being updated, the FIFO must be flushed to ensure that the internal pointers are properly aligned. 00 = 8 bit 01 = 16 bit 10 = 32 bit 11 = 64 bit	0x10
Reserved	17-16	Reserved	
rx_half_empty	15-8	Sets the FIFO level (in bytes) that asserts the receive “half” empty flag. The level setting is associated with how much data is in the FIFO. i.e. if the setting is 0x20, then when the FIFO fills above 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40
tx_half_full	7-0	Sets the FIFO level (in bytes) that asserts the transmit “half” full flag. The level setting is associated with how much space is available in the FIFO. i.e. if the setting is 0x20, then when the FIFO drains such that the amount of space available becomes 0x20 bytes, the interrupt will be asserted.	0x40

5.7.18 RS ECC config

Table 68. RS ECC config

RS ECC config			
Address: 0x68		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-3	Reserved	
rs_ecc_writepar_go	2	Some applications require the write parity before the write block is sent to the nand flash. This “kicking” signal allows the write block to pass through the RS ECC block to generate the write ECC bytes without sending the 512 byte block to nand flash. This bit is self clearing, and when it is written the hardware will read 512 bytes from the transmit FIFO and pass it through the RS ECC block, After this process is complete, the 9 bytes of write ECC can be then be read from the “RS ECC write parity” registers and written to nand flash before the actual 512 byte block is written. After the parity is written, a block write can be configured and the 512 bytes of data can be written to the transmit FIFO again. This time the block will be written to the nand flash.	0x0
rs_ecc_correct_go	1	This is a self clearing bit that is used if the RS ECC engine indicates that correctable read errors have been detected. When this bit is written, the input FIFO is loaded up with the corrected data of the 512 byte block.	0x0
rs_ecc_enable	0	This bit is set before any data operation occurs, if RS ECC operation is required. 0 = RS ECC operation disabled. 1 = RS ECC checking and generation is enabled.	0x0

5.7.19 RS ECC read parity

These registers are programmed with the 9 bytes of ECC for the next 512 byte block read. They must be programmed prior to initiating the block read.

RS ECC read parity1			
Address: 0x6C		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Read_parity	31-0	Bits 31:0 of the read parity	0x0

RS ECC read parity2			
Address: 0x70		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**

Read_parity	31-0	Bits [63:32] of the read parity	0x0

Table 69. RS ECC read parity

RS ECC read parity3			
Address: 0x74		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Read_parity	7-0	Bits [71:64] of the read parity	0x0

5.7.20 RS ECC write parity

These registers contain the 9 bytes of generated parity that are generated by the RS ECC engine after a 512 bytes block is written to NAND flash. The 9 bytes can be read and then written to nand flash.

RS ECC write parity1			
Address: 0x78		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
Write_parity	31-0	Bits 31:0 of the write parity	0x0

RS ECC write parity2			
Address: 0x7C		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
Write_parity	31-0	Bits [63:32] of the write parity	0x0

Table 70. RS ECC write parity

RS ECC write parity3			
Address: 0x80		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
Write_parity	7-0	Bits [71:64] of the write parity	0x0

5.7.21 RS ECC Status

Table 71. RS ECC Status

RS ECC Status			
Address: 0x84		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-2	Reserved	
rs_ecc_read_status	1-0	After the 512 byte read operation is completed and the “read_ecc_complete” interrupt is asserted, this field indicates the status of the block read. 00 = no errors 01 = correctable errors 1x = uncorrectable errors	0x0

5.7.22 Transmit FIFO

The Transmit FIFO operates as a FIFO even though it has a range of addresses. The wider range allows bus bursting to fill the FIFO.

Table 72. Transmit FIFO

Transmit FIFO			
Address: 0x1000-0x1fff		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be written to the FIFO. This register supports 8,16 or 32 bit writes and pushes the appropriate amount of data into the FIFO.	0x0

5.7.23 Receive FIFO

The Receive FIFO operates as a FIFO even though it has a range of addresses. The wider range allows bus bursting to drain the FIFO.

Table 73. Receive FIFO

Receive FIFO			
Address: 0x2000-0x2fff		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be read from the FIFO. This register supports 8,16 or 32 bit reads and pops the appropriate amount of data from the FIFO.	0x0

5.8 UART Control Registers

5.8.1 UART Register Description

Table 74. UART Register Map

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO
baud rate control	0x0C	RW
Configuration	0x10	RW
fifo status	0x14	RO/WO
fifo flag Configuration	0x18	RW
reserved	0x1C-0xfff	WO
transmit fifo	0x1000-0x1fff	WO
receive fifo	0x2000-0x2fff	RO

The register map is summarized below and described in the following sections.

5.8.2 Interrupt Source Register

Table 75. Interrupt Source Register

Interrupt Source Register			
Address: 0x00		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-13	Reserved	
RTS_raw	12	This bit reflects the state of the RTS modem signal.	0x0
CTS_raw	11	This bit reflects the state of the CTS modem signal.	0x0
rx_push_error	10	asserted when the receive fifo experiences an overrun condition or mis-aligned access.This error is from the perspective of the external interface.	0x0
parity_error	9	asserted when the receive block detects a parity error	0x0
frame_error	8	asserted when the receive block has detected a frame error (missing stop bit(s))	0x0

Table 75. Interrupt Source Register

bus_pop_error	7	asserted when the transmit fifo experiences an underrun condition or mis-aligned access.This error is from the perspective of the internal APB bus.	0x0
bus_push_error	6	asserted when the receive fifo experiences an overrun condition or mis-aligned access. A mis-aligned access can occur if the width of the write has changed from a previous access. For example, if byte writes have previously been used, the number of writes may be non-multiples of 32 bits. If a 32 bit write now occurs, this is a misaligned access because the byte pointers in the fifo are not pointing to byte '0'. This error is from the perspective of the internal APB bus.	0x0
rx_ff	5	asserted when the receive fifo has become full	0x0
rx_hf	4	asserted when the receive fifo level (amount of bytes in the fifo) has risen above the software configured "half" empty level.	0x0
rx_fe	3	asserted when the receive fifo has become NOT empty	0x0
tx_ff	2	asserted when the transmit fifo has become NOT full	0x0
tx_hf	1	asserted when the transmit fifo level (amount of space available) has risen above the software configured "half" full level.	0x0
tx_fe	0	asserted when the transmit fifo has become empty	0x0

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

5.8.3 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Table 76. Interrupt Mask Register

Interrupt Mask Register			
Address: 0x04		**Reset = 0x1FFF**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-13	Reserved	
RTS_raw	12	Masks the interrupt. 1=mask, 0=unmask.	0x1
CTS_raw	11	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_push_error	10	Masks the interrupt. 1=mask, 0=unmask.	0x1
parity_error	9	Masks the interrupt. 1=mask, 0=unmask.	0x1
frame_error	8	Masks the interrupt. 1=mask, 0=unmask.	0x1
bus_pop_error	7	Masks the interrupt. 1=mask, 0=unmask.	0x1
bus_push_error	6	Masks the interrupt. 1=mask, 0=unmask.	0x1

Table 76. Interrupt Mask Register

rx_ff	5	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_hf	4	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_fe	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_ff	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_hf	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_fe	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.8.4 Interrupt clear Register

Table 77. Interrupt Clear Register

• Interrupt Clear Register		•	•
• Address: 0x08		• Reset = 0x0	• Type: WO
• Name	• Bit	• Function	• Reset
Reserved	31-13	Reserved	
RTS_raw	12	This interrupt can't be cleared. It just reflects the state of the modem signal.	0x0
CTS_raw	11	This interrupt can't be cleared. It just reflects the state of the modem signal.	0x0
rx_push_error	10	Clears the interrupt when written '1'.	0x0
parity_error	9	Clears the interrupt when written '1'.	0x0
frame_error	8	Clears the interrupt when written '1'.	0x0
bus_pop_error	7	Clears the interrupt when written '1'.	0x0
bus_push_error	6	Clears the interrupt when written '1'.	0x0
rx_ff	5	Clears the interrupt when written '1'.	0x0
rx_hf	4	Clears the interrupt when written '1'.	0x0
rx_fe	3	Clears the interrupt when written '1'.	0x0
tx_ff	2	Clears the interrupt when written '1'.	0x0
tx_hf	1	Clears the interrupt when written '1'.	0x0
tx_fe	0	Clears the interrupt when written '1'.	0x0

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

5.8.5 Baud Rate Control

Table 78. Baud Rate Control

Baud Rate Control			
Address: 0x0C		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
rx_baud_rate_div	31-16	The interface reference clock is divided by this value to produce the baud rate for the receive data. Minimum value is d16.	0x0
tx_baud_rate_div	15-0	The interface reference clock is divided by this value to produce the baud rate for the transmit data. Minimum value is d16. For IR mode, the tx_baud_rate_div must be the same as the rx_baud_rate_div.	0x0

5.8.6 Configuration

Table 79. UART Configuration

Configuration			
Address: 0x10		**Reset = 0x14_0000**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
rts_fifo_level	31-24	This field sets the FIFO threshold as to when to de-assert the RTS modem signal. It represents the amount of empty space in the fifo in bytes. If the fifo goes below this amount of empty space, the RTS modem signal is de-asserted.	0x0
Reserved	23	Reserved	
enable_CTS	22	This field enables the use of the CTS signal. When enabled, the transmitter will only send characters when this signal is asserted. 0 = modem signal not enable 1 = modem signal enabled	0x0
enable_RTS	21	This field enables the use of the RTS signal. When enabled, the receiver will assert/de-assert the signal depending on how much space is available in the fifo. If the modem signal is not enabled it will remain de-asserted. 0 = modem signal not enable 1 = modem signal enabled	0x0
rx_fifo_size	20-19	The receive fifo is an async fifo that supports 8,16,32 or 64 bit wide reads. The fifo access size must be set prior to using the fifo. If the fifo size is changed, the fifo must be flushed. 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = 64 bits	0x10

Table 79. UART Configuration

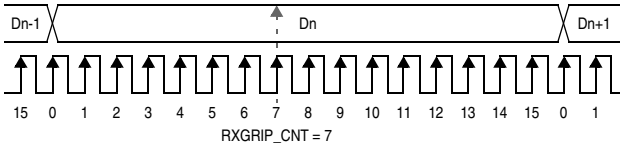
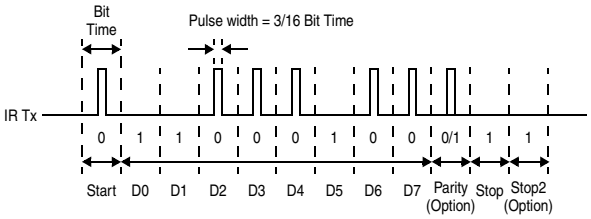
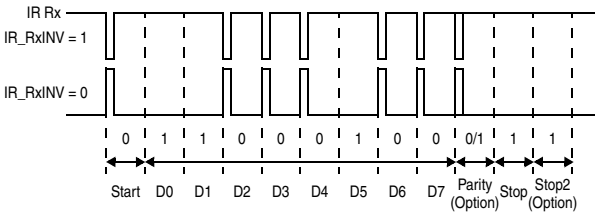
tx_fifo_size	18-17	The transmit fifo is an async fifo that supports 8,16,32 or 64 bit wide writes. The fifo access size must be set prior to using the fifo. If the fifo size is changed, the fifo must be flushed. 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = 64 bits	0x10
external_pad_ena	16	This bit enables the external transmit data pad. It must be enabled before sending any data. 0 = pad disabled 1 = pad enabled	0x0
rx_sampling_pos	15-12	The rx_sampling_pos is internal sampling position in a bit and is usually set to 7 when normal mode and 0 when IR mode.  	0x0
ir_tx_polarity	10	IR TX polarity in IR mode 0 : Active High 1 : Active Low	0x0
ir_rx_polarity	9	IR RX polarity in IR mode 0 : Active High 1 : Active Low	0x0
ir_mode	8	When the ir_mode is enabled, signal format is followed by IR mode timing diagram. 0 = normal mode 1 = IR mode  IR mode TX timing diagram  IR mode RX timing diagram	0x0

Table 79. UART Configuration

echo	7	When the echo is enabled, the incoming receive data is received internally but it is also looped back to the external transmit path. 0 = loopback disabled 1 = loopback enabled	0x0
remote_loop	6	When a remote loopback is enabled, the incoming receive data is loopback to the outgoing transmit data. The internal receive path is disabled. 0 = loopback disabled 1 = loopback enabled	0x0
local_loop	5	When a local loopback is enabled, the transmit data is looped back to the receive data. The transmit data is still transmitted externally. 0 = loopback disabled 1 = loopback enabled	0x0
rx_endian	4	This bit is used to modify the endianess of the data while it passes through the receive fifo. Data is written to the fifo a byte at a time. If data is read from the fifo 32 bits at a time, then an endianess swap can occur. LE data can be changed to BE data on 32 bit boundaries. 0 = maintain endianess 1 = change LE data to BE data	0x0
tx_endian	3	This bit is used to modify the endianess of the data while it passes through the transmit fifo. Data is read from the fifo a byte at a time. If data is written to the fifo 32 bits at a time, then an endianess swap can occur. LE data can be changed to BE data on 32 bit boundaries. 0 = maintain endianess 1 = change LE data to BE data	0x0
parity	2:1	This field indicates the parity type. 00 = even parity 01 = odd parity 10 = no parity 11 = reserved	0x0
stop_bits	0	This field indicates the number of stop bits. 0 = 1 stop bit 1 = 2 stop bits	0x0

5.8.7 FIFO Status

Table 80. FIFO Status

FIFO status			
Address: 0x14		**Reset = 0x40**	**Type: RO/WO**
Name	**Bit**	**Function**	**Reset**
rx_flush	31	When this bit is written ‘1’, the receive fifo is flushed. This bit is a write only bit.	0x0
tx_flush	30	When this bit is written ‘1’, the transmit fifo is flushed. This bit is a write only bit.	0x0

Table 80. FIFO Status

Reserved	29-16	Reserved	
rx_byte_count	15-8	Indicates how many bytes of data is present in the receive fifo. This is a read only field.	0x0
tx_byte_count	7-0	Indicates how many bytes of free space is available in the transmit fifo. This is a read only field.	0x40

5.8.8 FIFO flag configuration

Table 81. FIFO Flag Configuration

FIFO flag Configuration			
Address: 0x18		**Reset = 0x4040**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	
rx_half_empty	15-8	Sets the FIFO level (in bytes) that asserts the receive “half” empty flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes (or more) of data available in the FIFO, the interrupt will be asserted.	0x40
tx_half_full	7-0	Sets the FIFO level (in bytes) that asserts the transmit “half” full flag. The level setting is associated with how much space is available in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes (or more) of space available in the FIFO, the interrupt will be asserted.	0x40

5.8.9 Transmit FIFO

The Transmit FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to fill the fifo.

Table 82. Transmit FIFO

Transmit FIFO			
Address: 0x1000-0x1fff		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be written to the fifo. This register supports 8,16 or 32 bit writes and pushes the appropriate amount of data into the fifo.	0x0

5.8.10 Receive FIFO

The Receive FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to drain the fifo.

Table 83. Receive FIFO

Receive FIFO			
Address: 0x2000-0x2fff		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
data	31-0	**This field contains the data to be read from the fifo. This register supports 8,16 or 32 bit reads and pops the appropriate amount of data from the fifo.**	0x0

5.9 SPI Registers

The register map is summarized below and described in the following sections.

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO
clock rate control	0x0C	RW
Configuration1	0x10	RW
Configuration2	0x14	RW
read_status	0x18	RO
fifo status	0x1C	RO/WO
fifo flag Configuration	0x20	RW
GPS Configuration	0x24	RW
GPS Counter 1	0x28	RW
GPS Counter 2	0x2C	RO
reserved	0x30-0xfff	WO
transmit fifo	0x1000-0x1fff	WO
receive fifo	0x2000-0x2fff	RO

5.9.1 Interrupt Source Register

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Interrupt Source Register			
Address: 0x00		Reset = 0x0	Type: RO

Name	Bit	Function	Reset
Reserved	31-11	Reserved	
rxdata_fall	10	This interrupt source is specific to a SPI application involving MMC cards. When reading or writing blocks of data from an MMC card, there is a period of time after the command has been issued before the card is ready to issue or accept data. During this period the MMC card must be polled to determine when it is ready for the block transaction. It will issue “FF” until its ready and then it issues “FE”. This interrupt eases the software overhead when looking for the “FE”. Software can let the FIFO fill and instead of reading all the data out looking for the “FE” it can continue to flush the fifo until this interrupt occurs.	0x0
tx_pop_error	9	asserted when the receive fifo experiences an overrun condition or mis-aligned access.This error is from the perspective of the external interface.	0x0
rx_push_error	8	asserted when the transmit fifo experiences an underrun condition or mis-aligned access.This error is from the perspective of the external interface.	0x0
dma_pop_error	7	asserted when the transmit fifo experiences an underrun condition or mis-aligned access.This error is from the perspective of the internal APB bus.	0x0
dma_push_error	6	asserted when the receive fifo experiences an overrun condition or mis-aligned access. A mis-aligned access can occur if the width of the write has changed from a previous access. For example, if byte writes have previously been used, the number of writes may be non-multiples of 32 bits. If a 32 bit write now occurs, this is a misaligned access because the byte pointers in the fifo are not pointing to byte ’0’. This error is from the perspective of the internal APB bus.	0x0
rx_ff	5	asserted when the receive fifo has become full	0x0
rx_hf	4	asserted when the receive fifo level is above the software configured “half” empty level.	0x0
rx_fe	3	asserted when the receive fifo has become NOT empty	0x0
tx_ff	2	asserted when the transmit fifo has become NOT full	0x0
tx_hf	1	asserted when the transmit fifo level is below the software configured “half” full level.	0x0
tx_fe	0	asserted when the transmit fifo has become empty	0x0

5.9.2 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Interrupt Mask Register		

Address: 0x04	Reset = 0x7FF	Type: RW	
Name	**Bit**	**Function**	**Reset**
Reserved	31-11	Reserved	
rxdata_fall	10	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_pop_error	9	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_push_error	8	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_pop_error	7	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_push_error	6	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_ff	5	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_hf	4	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_fe	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_ff	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_hf	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_fe	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.9.3 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Interrupt Clear Register			
Address: 0x08	Reset = 0x0	Type: WO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-11	Reserved	
rxdata_fall	10	Clears the interrupt when written ‘1’.	0x0
tx_pop_error	9	Clears the interrupt when written ‘1’.	0x0
rx_push_error	8	Clears the interrupt when written ‘1’.	0x0
dma_pop_error	7	Clears the interrupt when written ‘1’.	0x0
dma_push_error	6	Clears the interrupt when written ‘1’.	0x0
rx_ff	5	Clears the interrupt when written ‘1’.	0x0
rx_hf	4	Clears the interrupt when written ‘1’.	0x0
rx_fe	3	Clears the interrupt when written ‘1’.	0x0
tx_ff	2	Clears the interrupt when written ‘1’.	0x0
tx_hf	1	Clears the interrupt when written ‘1’.	0x0

tx_fe	0	Clears the interrupt when written ‘1’.	0x0

5.9.4 Clock Rate Control

Clock Rate Control			
Address: 0x0C		Reset = 0x2	Type: RW
Name	**Bit**	**Function**	**Reset**
clock_div	31-0	The interface PLL clock is divided by this value to produce the clock rate for the external SPI clock. This register is only applicable for master mode of operation. Values of “0” or “1” are invalid.	0x2

5.9.5 Configuration1

Configuration1			
Address: 0x10		Reset = 0x108100	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31	Reserved	
device_select	30-29	When the SPI interface is configured as a master it can support up to four slave devices. Only one slave device at a time can be accessed, but muxing control allows communication with four different devices (there are 4 chip selects and 4 receive data, the clock and transmit data is common for all 4 devices). This field controls the muxing logic. 0 = device 0 1 = device 1 2 = device 2 3 = device 3	0x0
spi_clkgen_disable	28	This field disables the SPI clock generator. Slave applications do not require the clock generator so it is recommended that this be disabled to minimize power. 0 = SPI clock generator enabled 1 = SPI clock generator disabled	0x0
transaction_cnt	27-20	This field dictates the number of transactions that occur during a chip select assertion for fixed length transactions. The transactions are of a length defined by the "length" field. This field is intended to allow for continuous streams of data. Operation does not begin until the data is in the fifo. Care should be taken in how the fifo is filled. The fifo should be filled with increments (8,16 or 32) that is equal or greater than the “length” field for the SPI transaction. This field is only applicable to master mode of operation. Continuos mode is also supported in slave mode but the chip select duration is controlled externally.	0x1

var_length_start	19	This field is used in conjunction with the var_length_ena field. When “1”, the SPI serial stream will begin once data is in the fifo. The fifo should be filled with increments (8,16 or 32) that is equal or greater than the “length” field for the SPI transaction. The stream will continue as long as data is in the fifo and as long as this bit is asserted. Writing a “0” to this field will de-activate the chip select and stop the serial stream. The fifo should be empty when this mode is stopped.	0x0
var_length_ena	18	When “1”, the variable length continuos stream mode is enabled. This is very similar to the mode enabled by the transaction_cnt field except the start and end are controlled by the toggling the var_length_start field.	0x0
length	17-12	This field dictates the number of bits that are transmitted per transaction on the SPI interface. Valid values range from d3-d32. If the length is not 8,16 or 32 the following characteristics apply. When the de-serialized data is pushed into the fifo it is padded with “0” to align with 8,16 or 32 bits. When data is being serialized and transmitted, data is popped out of the fifo as an 8,16 or 32 bit word and the extra bits up to the 8,16 or 32 bit boundary are dropped.	0x8
Reserved	11-9	Reserved	
tx_dis	8	transmit datapath disable. Since the SPI interface transmits data coincidently with the reception of data, this bit provides some flexibility to software. If the application is only reading and transmit data is non-exisitant, the receive path can be disabled. This will prevent the transmit fifo from underrunning. 0=transmit datapath enable 1=transmit datapath disable.	0x1
rx_endian	7	This bit is used to modify the endianess of the data while it passes through the receive fifo. Data is written to the fifo a byte at a time. If data is read from the fifo 32 bits at a time, then an endianess swap can occur. LE data can be changed to BE data on 32 bit boundaries. 0 = maintain endianess 1 = change LE data to BE data	0x0
tx_endian	6	This bit is used to modify the endianess of the data while it passes through the transmit fifo. Data is read from the fifo a byte at a time. If data is written to the fifo 32 bits at a time, then an endianess swap can occur. LE data can be changed to BE data on 32 bit boundaries. 0 = maintain endianess 1 = change LE data to BE data	0x0
LSB_first	5	Depending on the setting, the LSB or MSB of the transaction will be transmitted or received first. 0=MSB first 1=LSB first	0x0
loopback	4	When loopback is enabled, the transmit data is looped back into the receive data path. 0=no loopback 1=loopback	0x0

rx_dis	3	Receive datapath disable. Since the SPI interface receives data coincidently with the transmission of data, this bit provides some flexibility to software. If the application is only writing and receive data is non-exisitant, the receive path can be disabled. This will prevent the receive fifo from filling up with garbage. Alternatively, the receive path can remain enabled and the receive fifo can be flushed. 0=receive datapath enable 1=receive datapath disable	0x0
ms	2	master/slave select. This bit configures the interface as a slave or a master. 0=slave mode 1=master mode	0x0
spo	1	This field sets the inactive clock polarity. Inactivity is associated with the chip select being de-asserted. 0 = clock low during inactivity 1 = clock high during inactivity	0x0
sph	0	This field sets the SPI clock phase. 0 = transmit on falling edge, receive on rising edge 1 = transmit on rising edge, receive on falling edge	0x0

5.9.6 Configuration2

This register is used to enable a burst of "reads" on the SPI interface. It is only applicable to the master mode of operation.

Configuration2			
Address: 0x14	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
ena	31	This bit acts as a kickoff. When written ’1’ the SPI will perform the number of transactions specified in the read_length field and write the received data to the receive fifo. This bit is self clearing.	0x0
Reserved	30-16	Reserved	
read_length	15-0	These bits dictate the number of transactions that will occur. The transaction width is dictated by the length field in Configuration1.	0x0

5.9.7 Read Status

Read Status			
Address: 0x18	**Reset = 0x0**	**Type: RO**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	

read_length	15-0	This field is pertinent only if burst read transactions have been configured and enabled via the Configuration2 register. When this field is read, it will reflect the current decremented value of the read transaction counter.	0x0

5.9.8 FIFO Status

FIFO status			
Address: 0x1C	Reset = 0x80	Type: RO/WO	
Name	**Bit**	**Function**	**Reset**
rx_flush	31	When this bit is written ‘1’, the receive fifo is flushed. This bit is a write only bit.	0x0
tx_flush	30	When this bit is written ‘1’, the transmit fifo is flushed. This bit is a write only bit.	0x0
Reserved	29-16	Reserved	
rx_byte_count	15-8	Indicates how many bytes of data is present in the receive fifo. This is a read only field.	0x0
tx_byte_count	7-0	Indicates how many bytes of free space is available in the transmit fifo. This is a read only field.	0x80

5.9.9 FIFO Flag Configuration

FIFO flag Configuration			
Address: 0x20	Reset = 0x28_4040	Type: RW	
Name	**Bit**	**Function**	**Reset**
Reserved	31-22	Reserved	
rx_fifo_size	21-20	The receive fifo is an async fifo that supports 8,16,32 or 64 bit wide reads. The fifo access size must be set prior to using the fifo. If the fifo size is changed, the fifo must be flushed. 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = 64 bits	0x10
tx_fifo_size	19-18	The transmit fifo is an async fifo that supports 8,16,32 or 64 bit wide writes. The fifo access size must be set prior to using the fifo. If the fifo size is changed, the fifo must be flushed. 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = 64 bits	0x10
Reserved	17-16	Reserved	

rx_half_empty	15-8	Sets the FIFO level (in bytes) that asserts the receive “half” empty flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40
tx_half_full	7-0	Sets the FIFO level (in bytes) that asserts the transmit “half” full flag. The level setting is associated with how much space is available in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of space available in the FIFO, the interrupt will be asserted.	0x40

5.9.10 GPS Configuration and Control

The SPI block has an embedded alternate GPS function that stores data from a GPS source into the receive fifo. When this mode is enabled the SPI function can no longer use the receive fifo (the transmit fifo is still available for SPI transmit functions). The GPS interface is a very simple serial interface as shown below:

Error! Objects cannot be created from editing field codes.

The Data format stored in the FIFO is software configurable and can be one of the following:

Error! Objects cannot be created from editing field codes.

GPS Configuration			
Address: 0x24	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-6	Reserved	
enter_track_mode	5	During tracking mode, continuous data acquisition is not required. Instead, just the number of samples needs to be stored. 0 – normal acquisition mode (all data is sent to the fifo) 1 – tracking mode (data is no longer stored but a sample counter is incremented).	0x0
invert_clk	4	This field will invert the clock before it is used by the internal hardware. 0 – no inversion 1 – gps_clk is inverted	0x0
mode	3	Indicates how many magnitude bits are associated with the interface. 0 – 1 magnitude bit 1 – 3 magnitude bits	0x0
format	2-1	Sets the serialization format as illustrated above. 00 – unpacked 01 – packed 10 – super packed 11 - reserved	0x0
enable_gps	0	As soon as this mode is enabled, any activity on the GPS interface is de-serialized and pushed into the receive fifo. 0 – disabled 1 - enabled	0x0

GPS Counter 1		

Address: 0x28		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
switch_over_cnt	31-0	This register is used to set the start value of a count down register. The count down register is used when switching from acquisition mode to tracking mode. It's count is used to synchronize when the switch over from storing data to just counting samples (data is thrown away) occurs. Usually it will lineup with a completed DMA transaction. If enter_track_mode is set, and the count down register is "0" data will no longer be stored in the fifo. Instead the sample_cnt will be incremented. When enter_track_mode is cleared, the data is once again stored in the fifo and the down counter starts decrementing from its programmed start value.	0x0

GPS Counter 2			
Address: 0x2C		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
sample_cnt	31-0	This field contains the current sample count when the GPS is in the tracking mode of operation. It is cleared when the following event occurs – the enter_track_mode is set and the count down register is "0". When enter_track_mode becomes "0", the sample_cnt holds its current value.	0x0

The following sequence is a typical use of the GPS registers.

- Initially the acquisition mode is entered. The GPS interface is enabled and the data is moved by the mc-dma to a storage buffer. The switch_over_cnt is programmed with a sample number that matches the chunk size that the mc-dma is programmed with.
- After acquisition, the tracking mode is entered. During this mode, most of the data is not required but the number of samples that have been received needs to be saved. To enter this mode, the "enter_track_mode" field is asserted. When this is asserted, the hardware will wait until the down counter expires (so that a known amount of data is received) and then just counts the number of samples that occur. The incoming data is discarded.
- When additional data samples are required, the "enter_track_mode" field is de-asserted. This will re-enable the data path to the fifo and hold the sample count until the next time the tracking mode is entered.

5.9.11 Transmit FIFO

The Transmit FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to fill the fifo.

Transmit FIFO			
Address: 0x1000-0x1fff		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**

data	31-0	This field contains the data to be written to the fifo. This register supports 8,16 or 32 bit writes and pushes the appropriate amount of data into the fifo. The size of the access must match the size that is programmed into the tx_fifo_size field in the Configuration1 register.	0x0

5.9.12 Receive FIFO

The Receive FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to drain the fifo.

Receive FIFO			
Address: 0x2000-0x2fff	**Reset = 0x0**	**Type: WO**	
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be read from the fifo. This register supports 8,16 or 32 bit reads and pops the appropriate amount of data from the fifo. The size of the access must match the size that is programmed into the rx_fifo_size field in the Configuration1 register.	0x0

5.10 Audio Registers

The Audio register map is summarized below and described in the following sections. All register support 32 bit accesses only. The FIFOs are the exception and they support 8,16 or 32 bit accesses.

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO
Interface Configuration	0x0C	RW
Bit Clock Configuration	0x10	RW
Receive Frame Clock Configuration	0x14	RW
Transmit Frame Clock Configuration	0x18	RW
AC97 Configuration	0x1C	RW
AC97 Command	0x20	RW
AC97 Status	0x24	RO
AC97 Modem Control	0x28	RW
AC97 Modem Status	0x2C	RO
fifo status	0x30	RO/WO
fifo flag Configuration	0x34	RW
NCO Configuration	0x38	RW
reserved	0x3c-0xfff	

tx fifo	0x1000-0x1fff	WO
rx fifo	0x2000-0x2fff	RO

5.10.1 Interrupt Source Register

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Interrupt Source Register			
Address: 0x00		**Reset = 0x2**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-14	Reserved	
modem_status	13	This interrupt is only applicable for the AC97 mode of operation. It is asserted after the modem status field is valid.	0x0
cmd_read_complete	12	This interrupt is only applicable for the AC97 mode of operation. It is asserted after a requested read command (slot 1 & 2) has completed and valid data is now available in the AC97 status register.	0x0
cmd_write_complete	11	This interrupt is only applicable for the AC97 mode of operation. It is asserted after a requested write command (slot 1 & 2) has completed.	0x0
codec_ready	10	This interrupt is only applicable for the AC97 mode of operation. It is asserted when the “codec ready” bit in the incoming TAG channel has transitioned from “0” to “1” indicating that the codec is now ready for operation.	0x0
tx_pop_error	9	asserted when the receive fifo experiences an overrun condition or mis-aligned access.This error is from the perspective of the external interface.	0x0
rx_push_error	8	asserted when the transmit fifo experiences an underrun condition or mis-aligned access.This error is from the perspective of the external interface.	0x0
bus_pop_error	7	asserted when the receive fifo experiences an underrun condition or mis-aligned access.This error is from the perspective of the internal APB bus.	0x0
bus_push_error	6	asserted when the transmit fifo experiences an overrun condition or mis-aligned access. A mis-aligned access can occur if the width of the write has changed from a previous access. For example, if byte writes have previously been used, the number of writes may be non-multiples of 32 bits. If a 32 bit write now occurs, this is a misaligned access because the byte pointers in the fifo are not pointing to byte ’0’. This error is from the perspective of the internal APB bus.	0x0
rx_ff	5	asserted when the receive fifo has become full	0x0
rx_hf	4	asserted when the receive fifo level has risen above the software configured “half” empty level.	0x0
rx_fe	3	asserted when the receive fifo has become NOT empty	0x0

tx_ff	2	asserted when the transmit fifo has become NOT full	0x0
tx_hf	1	asserted when the transmit fifo level has dropped below the software configured “half” full level.	0x0
tx_fe	0	asserted when the transmit fifo has become empty	0x0

5.10.2 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Interrupt Mask Register			
Address: 0x04		**Reset = 0xFFFF_FFFF**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-14	Reserved	
modem_status	13	Masks the interrupt. 1=mask, 0=unmask.	0x1
cmd_read_complete	12	Masks the interrupt. 1=mask, 0=unmask.	0x1
cmd_write_complete	11	Masks the interrupt. 1=mask, 0=unmask.	0x1
codec_ready	10	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_pop_error	9	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_push_error	8	Masks the interrupt. 1=mask, 0=unmask.	0x1
bus_pop_error	7	Masks the interrupt. 1=mask, 0=unmask.	0x1
bus_push_error	6	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_ff	5	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_hf	4	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_fe	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_ff	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_hf	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_fe	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.10.3 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Interrupt Clear Register			
Address: 0x08		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
	31-14		

modem_status	13	Clears the interrupt when written ‘1’.	0x0
cmd_read_complete	12	Clears the interrupt when written ‘1’.	0x0
cmd_write_complete	11	Clears the interrupt when written ‘1’.	0x0
codec_ready	10	Clears the interrupt when written ‘1’.	0x0
tx_pop_error	9	Clears the interrupt when written ‘1’.	0x0
rx_push_error	8	Clears the interrupt when written ‘1’.	0x0
bus_pop_error	7	Clears the interrupt when written ‘1’.	0x0
bus_push_error	6	Clears the interrupt when written ‘1’.	0x0
rx_ff	5	Clears the interrupt when written ‘1’.	0x0
rx_hf	4	Clears the interrupt when written ‘1’.	0x0
rx_fe	3	Clears the interrupt when written ‘1’.	0x0
tx_ff	2	Clears the interrupt when written ‘1’.	0x0
tx_hf	1	Clears the interrupt when written ‘1’.	0x0
tx_fe	0	Clears the interrupt when written ‘1’.	0x0

5.10.4 Interface Configuration

Interface Configuration			
Address: 0x0C	**Reset = 0x8080_0000**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
tx_fifo_size	31-30	The transmit fifo is an async fifo that supports 8,16,32 or 64 bit wide writes. The fifo access size must be set prior to using the fifo. If the fifo size is changed, the fifo must first be flushed. 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = 64 bits	0x2
rxd_word_length	29-24	Number of bits de-serialized in each active channel on the receive interface. Maximum value is 32 for the I2S mode of operation and 20 bits for the AC97 mode of operation. If this field has been changed after the fifos have been used, the fifo must be flushed for proper operation.	0x0
rx_fifo_size	23-22	The receive fifo is an async fifo that supports 8,16,32 or 64 bit wide reads. The fifo access size must be set prior to using the fifo. If the fifo size is changed, the fifo must first be flushed. 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = 64 bits	0x2

txd_word_length	21-16	Number of bits serialized in each active channel on the transmit interface. Maximum value is 32 for the I2S mode of operation and 20 bits for the AC97 mode of operation. If this field has been changed after the fifos have been used, the fifo must be flushed for proper operation.	0x0
rx_stereo	15	This field enables whether or not the stereo mode is enabled for the receive path. When the stereo mode is enabled the frame is organized as a left and right channel where one channel is transmitted during the first half of frame period and the second channel is transmitted during the second half of the frame period. This setting is only applicable when the interface operates in the I2S mode. 0 = mono operation (single channel) 1 = stereo operation (two channels)	0x0
loop_back	14	When “1” the transmit data is looped back to the receive data.	0x0
common_sync	13	The audio interface can utilize a common frame synchronization signal for both the transmit and receive datapath or can use separate synchronization signals. If a common synchronization signal is selected, then the bit clocks must also be configured as common. The transmit frame signal (fsx) is used if common_sync is enabled 1 = receive frame sync is shared with the transmit. 0 = receive frame sync is separate and unique from the transmit.	0x0
rx_bitclk_src	12	The receive interface can operate with its own bit clock or it can share the transmit bit clock. For applications that have a single bit clock, the receive bit clock must be shared with the transmit bit clock. When the receive and transmit share a bit clock, that bit clock is the transmit bit clock. 0 = receive bit clock is shared with the transmit. 1 = receive bit clock is separate and unique from the transmit.	0x0
fsx_polarity	11	This bit sets the polarity of the transmit frame clock. If the external device sources the frame sync, this bit should be set such that the asic receives an active high frame sync. 0 = internally generated frame sync is active high or external frame sync is NOT inverted. 1 = internally generated frame sync is active low or external frame sync is inverted.	0x0
fsr_polarity	10	This bit sets the polarity of the receive frame clock. . If the external device sources the frame sync, this bit should be set such that the asic receives an active high frame sync. 0 = internally generated frame sync is active high or external frame sync is NOT inverted. 1 = internally generated frame sync is active low or external frame sync is inverted.	0x0
clkx_polarity	9	This bit is used to determine which edge of the bit clock is used to transition the transmit data and frame signal. 0 = transmit signaling transitions on the rising edge of the bit clock. 1 = transmit signaling transitions on the falling edge of the bit clock.	0x0
clkr_polarity	8	This bit is used to determine which edge of the bit clock is used to transition the frame signal and sample the receive data. 0 = receive signaling transitions/sampled on the rising edge of the bit clock. 1 = receive signaling transitions/sampled on the falling edge of the bit clock.	0x0

fsx_src	7	This bit reflects whether or not the asic sources the frame synchronization or whether the codec is the source. 0 = pad is disabled and an external device sources the frame sync. 1 = pad is enabled and the asic sources the frame sync. Prior to enabling the frame sync timing must be properly setup.	0x0
fsr_src	6	This bit reflects whether or not the asic sources the frame synchronization or whether the codec is the source. 0 = pad is disabled and an external device sources the frame sync. 1 = pad is enabled and the asic sources the frame sync. Prior to enabling the frame sync timing must be properly setup.	0x0
clkx_src	5	This bit reflects whether or not the asic sources the bit clock or whether the codec is the source. 0 = pad is disabled and an external device sources the bit clock 1 = pad is enabled and the asic sources the bit clock. Prior to enabling the bit clock timing must be properly setup.	0x0
clkr_src	4	This bit reflects whether or not the asic sources the bit clock or whether the codec is the source. 0 = pad is disabled and an external device sources the bit clock 1 = pad is enabled and the asic sources the bit clock. Prior to enabling the bit clock timing must be properly setup.	
ena_transmit	3	This bit enables the datapath for the transmit direction. This allows the sync and bit clocks to be setup and enabled prior to having a datapath enabled in the transmit direction. The transmit data will be tri-stated until this bit is set and then data will be popped from the fifo and serialized. 0 = transmit datapath is disabled 1 = transmit datapath is enabled	0x0
ena_receive	2	This bit enables the datapath for the receive direction. This allows the sync and bit clocks to be setup and enabled prior to having a datapath enabled in the receive direction. The receive data is ignored until this bit is set and then data will be de-serialized and pushed into the fifo. 0 = receive datapath is disabled 1 = receive datapath is enabled	0x0
tx_stereo	1	This field enables whether or not the stereo mode is enabled for the transmit path. When the stereo mode is enabled the frame is organized as a left and right channel where one channel is transmitted during the first half of frame period and the second channel is transmitted during the second half of the frame period. This setting is only applicable when the interface operates in the I2S mode. 0 = mono operation (single channel) 1 = stereo operation (two channels)	0x0
mode	0	This field indicates to the hardware which mode of operation the interface will operate. 0 = I2S mode of operation 1 = AC97 mode of operation	0x0

The interface configuration register allows pretty much any possible configuration of external signals and sources of these signals. The following table provides some typical example settings.

Mode of operation	mclk	fsx	clkx	fsr	clkr

I2S master	18.432 Mhz	ASIC sourced	ASIC sourced	optional	optional
I2S slave	Not used	Codec sourced	Codec sourced	optional	optional
AC97 master	24.576 Mhz	ASIC sourced	ASIC sourced	Not used	Not used
AC97 slave	Not used	ASIC sourced	Codec sourced	Not used	Not used

5.10.5 Bit clock Configuration

Bit Clock Configuration			
Address: 0x10	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
rx_clk_div	31-16	This field contains the divider value applied to the master audio clock (mclk) to produce the receive bit clock (clkr). Values of 2 or greater are valid. This field is only applicable when the receive clock source is the asic as configured in the interface configuration register. The resultant clock will have a 50% duty cycle for even divides and a non-50% duty cycle for odd divides.	0x0
tx_clk_div	15-0	This field contains the divider value applied to the master audio clock (mclk) to produce the transmit bit clock (clkx). Values of 2 or greater are valid. This field is only applicable when the transmit clock source is the asic as configured in the interface configuration register. The resultant clock will have a 50% duty cycle for even divides and a non-50% duty cycle for odd divides.	0x0

5.10.6 Receive Frame Clock Configuration

Receive Frame Clock Configuration			
Address: 0x14	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31	Reserved	0x0
rx_dly	30-24	This field provides the capability to delay the first bit period of valid data from the assertion of the receive frame clock. This field contains the number of receive bit clocks from the active edge of the frame clock to the first valid bit of received data.	0x0
rx_frame_width	23-16	This field controls the active width of the receive frame clock. The field represents the number of receive bit clocks and has a minimum value of 1.	0x0
rx_frame_period	15-0	This field contains the divider value applied to the receive bit clock to produce the receive frame clock. Values of 2 or greater are valid.	0x0

5.10.7 Transmit Frame Clock Configuration

Transmit Frame Clock Configuration		

Address: 0x18		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31	Reserved	0x0
tx_dly	30-24	This field provides the capability to delay the first bit period of valid data from the assertion of the transmit frame clock. This field contains the number of transmit bit clocks from the active edge of the frame clock to the first valid bit of transmitted data.	0x0
tx_frame_width	23-16	This field controls the active width of the transmit frame clock. The field represents the number of transmit bit clocks and has a minimum value of 1.	0x0
tx_frame_period	15-0	This field contains the divider value applied to the transmit bit clock to produce the transmit frame clock. Values of 2 or greater are valid.	0x0

5.10.8 C97 Configuration

AC97 Configuration			
Address: 0x1C		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	0x0
codec_id	15-14	This field contains the codec ID that will be transmitted during slot 0. Multiple codecs are not supported so the ID will probably always be ‘0’ for a primary codec. The field has been made configurable just in case the flexibility is required.	0x0
warm_reset	13	Writing a ‘1’ to this bit will initiate a “warm” reset on the AC-link interface. This should only be initiated if the codec is in the power-down mode. This bit will cause the hardware to assert the sync signal for 1µsec initiating a warm reset within the codec. The bit is self resetting after the reset activity is complete. Prior to writing to this bit initiate a warm reset, the codec_pdown bit must be cleared in the AC97 command register. 0 = no action 1 = hardware initiates a warm reset	0x0
tx_slot_ena	12 -3	Enables the transmit channel for slots 3 to 12. If variable sampling rates are enabled, this bit is ignored and the slot enable information is obtained from the received TAG slot information. 0 = disable slot 1 = enable slot	0x0
Reserved	2-1	Reserved	0x0

variable_rate_ena	0	An AC97 frame is based on a 48 Khz period. If the application is using a 48 Khz sampling rate, this bit should remain disabled and the enabled channels will be transmitted/received in every frame. If the application is utilizing a non-48 Khz period and the codec supports variable sampling rates, then this bit should be set. When this bit is set, the hardware looks at the channel request bits in the received TAG channel. When the channel request bits match the enabled slots as configured in tx_slot_ena, the channels that are requesting data are sent data. The hardware assumes that the data in the fifo matches the channel requests (i.e. the hardware assumes that the requests will maintain their channel order). Data is not sent until all channels are requesting so that the channel order within the frame is maintained.	0x0

5.10.9 AC97 Command

This register provides the mechanism for read and writing the command registers in the codec. This operation occurs in slot 1 and 2 during the AC97 frame. If a command write or status read are initiated, they will occur during the next AC97 frame. Status and interrupt information is provided to indicate the completion of the command write or the availability of the status information.

AC97 Command			
Address: 0x20	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-26	Reserved	0x0
codec_pdown	25	The Codec can be placed in a power down state by writing to its power down register. This operation is accomplished the same as any other register write but the hardware needs to know that this register is being written to power down the codec. If this is the case, all outputs are zeroed after slot 2 has been transmitted. The codec is removed from this state by issuing a warm reset. This bit must be cleared prior to requesting a warm reset. 0 = no action 1 = interface activity ends after valid slot 2 data has been sent.	0x0
control_reg_ena	24	To enable activity in slot 1 and 2, this bit must be set. Once the bit is set, when the next frame occurs address and write data (if applicable) is serialized out in the next frame. This bit is self resetting once this action occurs. The address and write data must be valid when this bit is set. 0 = no action 1 = enable slot 1 or 2 (iff applicable) during the next frame.	0x0
write_data	23-8	This field contains the write data if the register access is a write. This field is serialized out during slot 2 (command data port) if read_write=0.	0x0
read_write	7	This is the read/write bit that is serialized out in the read/write command bit field of slot 1 (command address port). 0 = write 1 = read	0x0
control_reg_index	6-0	When enabled, this field will be serialized out in the address field of slot 1 (command address port). It is the address within the codec of the register being written or read.	0x0

5.10.10 AC97 Status

AC97 Status			
Address: 0x24	**Reset = 0x0**	**Type: RO**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-25	Reserved	0x0
codec_ready	24	This bit reflects the status of the “Codec Ready” bit in the slot 0 TAG information for the most recently received frame. The codec is not ready for operation until this bit gets set. The condition is normal following the de-assertion of power on reset.	0x0
read_data_valid	23	This bit is asserted if the read_data is valid. The bit is self clearing when a new codec register read is requested (as configured in the AC97 command register). It is re-asserted once the codec has returned valid data. 0 = read_data is invalid 1 = read_data is valid	0x0
echoed_index	22-16	This is the register index that has been echoed back by the codec and reflects the register address that was read.	0x0
read_data	15-0	This field contains the last valid read data from a register access. Data is valid as long as the read_data_valid flag is set.	0x0

5.10.11 AC97 Modem Control

This register provides the mechanism for writing to slot 12 when the slot operates as the Modem GPIO control channel.

AC97 Modem Control			
Address: 0x28	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-21	Reserved	0x0
modem_ena	20	To enable activity in slot 12 (for modem control purposes), this bit must be set. Once the bit is set, when the next frame occurs the modem_control field is serialized out in the next frame. This bit is self resetting once this action occurs. . The modem control data must be valid when this bit is set. 0 = no action 1 = enable slot 12 during the next frame.	0x0
modem_control	19-0	When enabled, this field will be serialized out in slot 12.	0x0

5.10.12 AC97 Modem Status

AC97 Modem Status		
Address: 0x2C	**Reset = 0x0**	**Type: RO**

Name	Bit	Function	Reset
Reserved	31-20	Reserved	0x0
modem_control	19-0	This field contains the last valid modem GPIO status from slot 12. An interrupt is raised when the data has been updated.	0x0

5.10.13 FIFO Status

FIFO status1			
Address: 0x30	Reset = 0x80	Type: RO/WO	
Name	**Bit**	**Function**	**Reset**
rx_flush	31	When this bit is written ‘1’, the receive fifo is flushed. This bit is a write only self clearing bit.	0x0
tx_flush	30	When this bit is written ‘1’, the transmit fifo is flushed. This bit is a write only self clearing bit.	0x0
Reserved	29-16	Reserved	0x0
rx_byte_count	15-8	Indicates how many bytes of data is present in the receive fifo. This is a read only field.	0x0
tx_byte_count	7-0	Indicates how many bytes of free space is available in the transmit fifo. This is a read only field.	0x80

5.10.14 FIFO Flag Configuration

FIFO flag Configuration			
Address: 0x34	Reset = 0x4040	Type: RW	
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	
rx_half_empty	15-8	Sets the FIFO level (in bytes) that asserts the receive “half” empty flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40
tx_half_full	7-0	Sets the FIFO level (in bytes) that asserts the transmit “half” full flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40

5.10.15 NCO Configuration

NCO Configuration		

Address: 0x38		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
audio_nco_enable	31	Enables the NCO as the source of MCLK. Also enables the MCLK PAD to drive out the NCO derived clock.	0x0
Reserved	30-20	Reserved	
audio_nco_value	19-0	NCO value. NCO value = round ((2^21 x mclk_freq) / SYS_freq)	0x0

5.11 MMC/SD Control Registers

The register map is summarized below and described in the following sections.

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO
MMC/SD clock rate	0x0C	RW
MMC/SD Configuration	0x10	RW
MMC/SD Data Control	0x14	RW
MMC/SD argument	0x18	RW
MMC/SD command	0x1C	RW
MMC/SD command response	0x20	RO
MMC/SD response	0x24-0x30	RO
fifo status	0x34	RO/WO
fifo flag Configuration	0x38	RW
reserved	0x3C-0xfff	
transmit fifo	0x1000-0x1fff	WO
receive fifo	0x2000-0x2fff	RO

5.11.1 Interrupt Source Register

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Interrupt Source Register			
Address: 0x00		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**

Reserved	31-20	Reserved	
dat3	19	This is not an interrupt but reflects the current state of the mmc_data3 signal.	0x0
sdio_interrupt	18	SDIO cards have a card interrupt mechanism. This bit is the indicator for that interrupt and is only applicable to SDIO cards. This interrupt source can be masked but not cleared by the interrupt clear register. It must be cleared within the SDIO card itself.	0x0
data_complete	17	Indicates that an MMC/SD data operation (read or write) is complete.	0x0
data_crc	16	Indicates that for an MMC/SD data operation, the CRC check failed.	0x0
response	15	Indicates that for MMC/SD operation, an MMC/SD response is available in the response buffer.	0x0
response_crc	14	Indicates that the received MMC/SD response has a CRC check failure.	0x0
dat3_low	13	This interrupt is asserted when the data state machine is idle and the mmc_data3 is low.	0x0
dat3_high	12	This interrupt is asserted when the data state machine is idle and the mmc_data3 is high.	0x0
cmd_complete	11	Indicates that the current MMC/SD command has been sent.	0x0
busy	10	Indicates that an MMC/SD card "busy" condition is no longer present. i.e. the card is no longer busy.	0x0
tx_pop_error	9	asserted when the transmit fifo experiences an overrun condition or misaligned access. This error is from the perspective of the external interface.	0x0
rx_push_error	8	asserted when the receive fifo experiences an underrun condition or misaligned access. This error is from the perspective of the external interface.	0x0
dma_pop_error	7	asserted when the receive fifo experiences an underrun condition or misaligned access. This error is from the perspective of the internal APB bus.	0x0
dma_push_error	6	asserted when the transmit fifo experiences an overrun condition or misaligned access. A misaligned access can occur if the width of the write has changed from a previous access. For example, if byte writes have previously been used, the number of writes may be non-multiples of 32 bits. If a 32 bit write now occurs, this is a misaligned access because the byte pointers in the fifo are not pointing to byte ’0’. This error is from the perspective of the internal APB bus.	0x0
rx_ff	5	asserted when the receive fifo has become full	0x0
rx_hf	4	asserted when the receive fifo level (amount of bytes in the fifo) is above the software configured “half” empty level.	0x0
rx_fe	3	asserted when the receive fifo has become NOT empty	0x0
tx_ff	2	asserted when the transmit fifo has become NOT full	0x0
tx_hf	1	asserted when the transmit fifo level (amount of space available) is above the software configured “half” full level.	0x0
tx_fe	0	asserted when the transmit fifo has become empty	0x0

5.11.2 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Interrupt Mask Register			
Address: 0x04		Reset = 0x7_FFFF	Type: RW

Name	Bit	Function	Reset
	31-19		
sdio_interrupt	18	Masks the interrupt. 1=mask, 0=unmask.	0x1
data_complete	17	Masks the interrupt. 1=mask, 0=unmask.	0x1
data_crc	16	Masks the interrupt. 1=mask, 0=unmask.	0x1
response	15	Masks the interrupt. 1=mask, 0=unmask.	0x1
response_crc	14	Masks the interrupt. 1=mask, 0=unmask.	0x1
dat3_low	13	Masks the interrupt. 1=mask, 0=unmask.	0x1
dat3_high	12	Masks the interrupt. 1=mask, 0=unmask.	0x1
cmd_complete	11	Masks the interrupt. 1=mask, 0=unmask.	0x1
busy	10	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_pop_error	9	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_push_error	8	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_pop_error	7	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_push_error	6	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_ff	5	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_hf	4	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_fe	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_ff	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_hf	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_fe	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.11.3 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Interrupt Clear Register			
Address: 0x08		Reset = 0x0	Type: WO

Name	Bit	Function	Reset

	31-18		
data_complete	17	Clears the interrupt when written ‘1’.	0x0
data_crc	16	Clears the interrupt when written ‘1’.	0x0
response	15	Clears the interrupt when written ‘1’.	0x0
response_crc	14	Clears the interrupt when written ‘1’.	0x0
dat3_low	13	Clears the interrupt when written ‘1’.	0x0
dat3_high	12	Clears the interrupt when written ‘1’.	0x0
cmd_complete	11	Clears the interrupt when written ‘1’.	0x0
busy	10	Clears the interrupt when written ‘1’.	0x0
tx_pop_error	9	Clears the interrupt when written ‘1’.	0x0
rx_push_error	8	Clears the interrupt when written ‘1’.	0x0
dma_pop_error	7	Clears the interrupt when written ‘1’.	0x0
dma_push_error	6	Clears the interrupt when written ‘1’.	0x0
rx_ff	5	Clears the interrupt when written ‘1’.	0x0
rx_hf	4	Clears the interrupt when written ‘1’.	0x0
rx_fe	3	Clears the interrupt when written ‘1’.	0x0
tx_ff	2	Clears the interrupt when written ‘1’.	0x0
tx_hf	1	Clears the interrupt when written ‘1’.	0x0
tx_fe	0	Clears the interrupt when written ‘1’.	0x0

5.11.4 MMC/SD Clock Rate

MMC/SD clock rate			
Address: 0x0C	**Reset = 0x80000000**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
disable	31	This bit provides a lower power standby condition when the SD interface is in use. Setting this bit will halt the clock such that the external serial clock is low. Power-up default is a disabled clock. Software must ensure that the proper divide is programmed prior to enabling the clock. Also, it is recommended to disable the clock whenever a clock_rate_divider change is required so that spurious clock pulses do not occur.	0x1
Reserved	30-16	Reserved	
clock_divider	15-0	The interface PLL clock is divided by the contents of this field to produce the serial clock. A divider of 2 or greater is valid.	0x0

5.11.5 MMC/SD configuration

MMC/SD Configuration		
Address: 0x10	Reset = 0x0	Type: RW

Name	Bit	Function	Reset
Reserved	31-10	Reserved	
data_width	13:12	Configures the width of the external data bus. 00 = 1-bit data 01 = 4-bit data 10,11 = reserved	0x0
Reserved	11-10	Reserved	
tx_fifo_size	9-8	Although the asynchronous fifo supports writes of varying sizes (8,16,32 or 64) the size must be configured prior to using the fifo. This size refers to the side of the fifo that the internal bus or dma engine writes to. If this field is being updated, the fifo must be flushed to ensure that the internal pointers are properly aligned. 00 = 8 bit 01 = 16 bit 10 = 32 bit 11 = 64 bit	0x0
rx_fifo_size	7-6	Although the asynchronous fifo supports reads of varying sizes (8,16,32 or 64) the size must be configured prior to using the fifo. This size refers to the side of the fifo that the internal bus or dma engine reads from. If this field is being updated, the fifo must be flushed to ensure that the internal pointers are properly aligned. 00 = 8 bit 01 = 16 bit 10 = 32 bit 11 = 64 bit	0x0
enable	5	This allows the media interface to function or holds it in an in-operative state. 0=disable 1=enable.	0x0
Reserved	4	Reserved	
block_size	3-0	This field indicates how many bytes are associated with a block for any read or write transaction. The hardware uses this field to break larger data lengths into block size chunks. The block size is defined as 2^{block_size}. 0=reserved, 1=2bytes, 2=4bytes, 3=8bytes, 4=16bytes, 5=32bytes, 6=64bytes, 7=128bytes, 8=256bytes, 9=512bytes, 10=1Kbytes, 11=2Kbytes, 12=4Kbytes, 13=8Kbytes, 14=16Kbytes, 15=32Kbytes	0x0

5.11.6 MMC/SD Data Control

This register is used to indicate to hardware the data path activity associated with a particular command. This register must be configured prior to writing to the command register.

MMC/SD Data Control		

Address: 0x14		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
data_size	31-16	This field indicates how many bytes are transferred before signaling a completion interrupt. This field must be an integer multiple of the configured block size. When this register is read, the data_size field will reflect how many bytes of data remain to be transferred.	0x0
Reserved	15-2	Reserved	
data_ena	1	Data path operation will not commence until this bit is enabled. 0=disable 1=enable.	0x0
data_direction	0	0=read 1=write.	0x0

5.11.7 MMC/SD Argument

MMC/SD Argument			
Address: 0x18		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
cmd_argument	31-0	This contains the argument for the command to be sent to the SD card. A read of the register will provide the previous command argument that was written	0x0

5.11.8 MMC/SD Argument

MMC/SD Command			
Address: 0x1C		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-11	Reserved	
response_type	10-8	This field indicates to the hardware the type of response that is expected for the particular command issued. 000=response type R1 or R6 001=rsponse type R1b 010=response type R2 011=response type R3 or R4 100=no response.	0x0
Reserved	7	Reserved	

data_command	6	This bit is only applicable to SDIO cards that make use of the interrupt feature on mmc_data1. Also it is only applicable when the interface is configured for the 4 bit mode of operation which will use mmc_data1 as a data line. Under this scenario the SDIO interrupt has a window of opportunity that it is valid. For hardware to understand that window of opportunity it must know whether the command that is being issued is associated with data or not. 1 = command is a data related command (read or write operation) 0 = command is not data related.	0x0
command_index	5-0	This is the command index to be sent to the SD card. The action of writing to this register initiates the command transfer process. A read of the register will provide the previous command index that was written.	0x0

5.11.9 MMC/SD Command Response

MMC/SD command response			
Address: 0x20		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-6	Reserved	
command_index	5-0	Contains the command index field of the last received response. If the response is type R2 or R3 the field will be '111111'.	0x0

5.11.10 MMC/SD Response

MMC/SD response			
Address: 0x24		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
card_status CID/CSD[39:8]	31-0	These registers are used to store the current SD response. If the response is type R1, R1b or R6 only the first 32 bits of the response field are valid Contains the card status field of the response or contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

MMC/SD response			
Address: 0x28		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
CID/CSD[71:40]	31-0	Contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

MMC/SD response		

Address: 0x2C		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
CID/CSD[103:72]	31-0	Contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

MMC/SD response			
Address: 0x30		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-24	Reserved	
CID/CSD[127:104]	23-0	Contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

5.11.11 FIFO Status

FIFO status			
Address: 0x34		**Reset = 0x80**	**Type: RO/WO**
Name	**Bit**	**Function**	**Reset**
rx_flush	31	When this bit is written ‘1’, the receive fifo is flushed. This bit is a write only bit.	0x0
tx_flush	30	When this bit is written ‘1’, the transmit fifo is flushed. This bit is a write only bit.	0x0
	29-16		
rx_byte_count	15-8	Indicates how many bytes of data is present in the receive fifo. This is a read only field.	0x0
tx_byte_count	7-0	Indicates how many bytes of free space is available in the transmit fifo. This is a read only field.	0x80

5.11.12 FIFO Flag Configuration

FIFO flag Configuration			
Address: 0x38		**Reset = 0x4040**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	

rx_half_empty	15-8	Sets the FIFO level (in bytes) that asserts the receive “half” empty flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40
tx_half_full	7-0	Sets the FIFO level (in bytes) that asserts the transmit “half” full flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40

5.11.13 Transmit FIFO

The Transmit FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to fill the fifo.

Transmit FIFO			
Address: 0x1000-0x1fff	**Reset = 0x0**	**Type: WO**	
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be written to the fifo. This register supports 8,16 or 32 bit writes and pushes the appropriate amount of data into the fifo. The size of the access must match the size that is programmed into the tx_fifo_size field in the Configuration1 register.	0x0

5.11.14 Receive FIFO

The Receive FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to drain the fifo.

Receive FIFO			
Address: 0x2000-0x2fff	**Reset = 0x0**	**Type: WO**	
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be read from the fifo. This register supports 8,16 or 32 bit reads and pops the appropriate amount of data from the fifo. The size of the access must match the size that is programmed into the rx_fifo_size field in the Configuration1 register.	0x0

5.12 MMCPlus Control Registers

The MMCPLUS register map is summarized below and described in the following sections.

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO

MMCPLUS clock rate	0x0C	RW
MMCPLUS Configuration	0x10	RW
MMCPLUS Data Control	0x14	RW
MMCPLUS argument	0x18	RW
MMCPLUS command	0x1C	RW
MMCPLUS command response	0x20	RO
MMCPLUS response	0x24-0x30	RO
fifo status	0x34	RO/WO
fifo flag Configuration	0x38	RW
reserved	0x3C-0xfff	
transmit fifo	0x1000-0x1fff	WO
receive fifo	0x2000-0x2fff	RO

5.12.1 Interrupt Source Register

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Interrupt Source Register			
Address: 0x00		**Reset = 0x0**	**Type: RO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-20	Reserved	
dat3	19	This is not an interrupt but reflects the current state of the mmc_data3 signal.	0x0
sdio_interrupt	18	SDIO cards have a card interrupt mechanism. This bit is the indicator for that interrupt and is only applicable to SDIO cards. This interrupt source can be masked but not cleared by the interrupt clear register. It must be cleared within the SDIO card itself.	0x0
data_complete	17	Indicates that an MMCPLUS data operation (read or write) is complete.	0x0
data_crc	16	Indicates that for an MMCPLUS data operation, the CRC check failed.	0x0
response	15	Indicates that for MMCPLUS operation, an MMCPLUS response is available in the response buffer.	0x0
response_crc	14	Indicates that the received MMCPLUS response has a CRC check failure.	0x0
dat3_low	13	This interrupt is asserted when the data state machine is idle and the mmc_data3 is low.	0x0
dat3_high	12	This interrupt is asserted when the data state machine is idle and the mmc_data3 is high.	0x0
cmd_complete	11	Indicates that the current MMCPLUS command has been sent.	0x0

busy	10	Indicates that an MMCPLUS card "busy" condition is no longer present. i.e. the card is no longer busy.	0x0
tx_pop_error	9	asserted when the transmit fifo experiences an overrun condition or misaligned access. This error is from the perspective of the external interface.	0x0
rx_push_error	8	asserted when the receive fifo experiences an underrun condition or misaligned access. This error is from the perspective of the external interface.	0x0
dma_pop_error	7	asserted when the receive fifo experiences an underrun condition or misaligned access. This error is from the perspective of the internal APB bus.	0x0
dma_push_error	6	asserted when the transmit fifo experiences an overrun condition or misaligned access. A misaligned access can occur if the width of the write has changed from a previous access. For example, if byte writes have previously been used, the number of writes may be non-multiples of 32 bits. If a 32 bit write now occurs, this is a misaligned access because the byte pointers in the fifo are not pointing to byte ’0’. This error is from the perspective of the internal APB bus.	0x0
rx_ff	5	asserted when the receive fifo has become full	0x0
rx_hf	4	asserted when the receive fifo level (amount of bytes in the fifo) has risen above the software configured “half” empty level.	0x0
rx_fe	3	asserted when the receive fifo has become NOT empty	0x0
tx_ff	2	asserted when the transmit fifo has become NOT full	0x0
tx_hf	1	asserted when the transmit fifo level (amount of space available) has risen above the software configured “half” full level.	0x0
tx_fe	0	asserted when the transmit fifo has become empty	0x0

5.12.2 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Interrupt Mask Register			
Address: 0x04		**Reset = 0x7FFFF**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-19	Reserved	
sdio_interrupt	18	Masks the interrupt. 1=mask, 0=unmask.	0x1
data_complete	17	Masks the interrupt. 1=mask, 0=unmask.	0x1
data_crc	16	Masks the interrupt. 1=mask, 0=unmask.	0x1
response	15	Masks the interrupt. 1=mask, 0=unmask.	0x1
response_crc	14	Masks the interrupt. 1=mask, 0=unmask.	0x1
dat3_low	13	Masks the interrupt. 1=mask, 0=unmask.	0x1

dat3_high	12	Masks the interrupt. 1=mask, 0=unmask.	0x1
cmd_complete	11	Masks the interrupt. 1=mask, 0=unmask.	0x1
busy	10	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_pop_error	9	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_push_error	8	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_pop_error	7	Masks the interrupt. 1=mask, 0=unmask.	0x1
dma_push_error	6	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_ff	5	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_hf	4	Masks the interrupt. 1=mask, 0=unmask.	0x1
rx_fe	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_ff	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_hf	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
tx_fe	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.12.3 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Interrupt Clear Register			
Address: 0x08		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
	31-18		
data_complete	17	Clears the interrupt when written ‘1’.	0x0
data_crc	16	Clears the interrupt when written ‘1’.	0x0
response	15	Clears the interrupt when written ‘1’.	0x0
response_crc	14	Clears the interrupt when written ‘1’.	0x0
dat3_low	13	Clears the interrupt when written ‘1’.	0x0
dat3_high	12	Clears the interrupt when written ‘1’.	0x0
cmd_complete	11	Clears the interrupt when written ‘1’.	0x0
busy	10	Clears the interrupt when written ‘1’.	0x0
tx_pop_error	9	Clears the interrupt when written ‘1’.	0x0
rx_push_error	8	Clears the interrupt when written ‘1’.	0x0
dma_pop_error	7	Clears the interrupt when written ‘1’.	0x0
dma_push_error	6	Clears the interrupt when written ‘1’.	0x0

rx_ff	5	Clears the interrupt when written ‘1’.	0x0
rx_hf	4	Clears the interrupt when written ‘1’.	0x0
rx_fe	3	Clears the interrupt when written ‘1’.	0x0
tx_ff	2	Clears the interrupt when written ‘1’.	0x0
tx_hf	1	Clears the interrupt when written ‘1’.	0x0
tx_fe	0	Clears the interrupt when written ‘1’.	0x0

5.12.4 MMC PLUS Clock Rate

MMCPLUS clock rate			
Address: 0x0C	**Reset = 0x80000000**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
disable	31	This bit provides a lower power standby condition when the SD interface is in use. Setting this bit will halt the clock such that the external serial clock is low. Power-up default is a disabled clock. Software must ensure that the proper divide is programmed prior to enabling the clock. Also, it is recommended to disable the clock whenever a clock_rate_divider change is required so that spurious clock pulses do not occur.	0x1
	30-20	reserved	
ext_clock_delay	19-16	mmcplus external clock delay mmc_clock_out is delayed in proportion to this value.	0x0
clock_divider	15-0	The interface PLL clock is divided by the contents of this field to produce the serial clock. A divider of 2 or greater is valid.	0x0

5.12.5 MMCPLUS Configuration

MMCPLUS Configuration			
Address: 0x10	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-14	Reserved	
data_width	13:12	Configures the width of the external data bus. 00 = 1-bit data 01 = 4-bit data 10 = 8-bit data 11 = reserved	0x0
reserved	11:10	Reserved	

tx_fifo_size	9-8	Although the asynchronous fifo supports writes of varying sizes (8,16,32 or 64) the size must be configured prior to using the fifo. This size refers to the side of the fifo that the internal bus or dma engine writes to. If this field is being updated, the fifo must be flushed to ensure that the internal pointers are properly aligned. 00 = 8 bit 01 = 16 bit 10 = 32 bit 11 = 64 bit	0x0
rx_fifo_size	7-6	Although the asynchronous fifo supports reads of varying sizes (8,16,32 or 64) the size must be configured prior to using the fifo. This size refers to the side of the fifo that the internal bus or dma engine reads from. If this field is being updated, the fifo must be flushed to ensure that the internal pointers are properly aligned. 00 = 8 bit 01 = 16 bit 10 = 32 bit 11 = 64 bit	0x0
enable	5	This allows the media interface to function or holds it in an in-operative state. 0=disable 1=enable.	0x0
reserved	4	reserved	0x0
block_size	3-0	This field indicates how many bytes are associated with a block for any read or write transaction. The hardware uses this field to break larger data lengths into block size chunks. The block size is defined as 2block_size. 0=reserved, 1=2bytes, 2=4bytes, 3=8bytes, 4=16bytes, 5=32bytes, 6=64bytes, 7=128bytes, 8=256bytes, 9=512bytes, 10=1Kbytes, 11=2Kbytes, 12=4Kbytes, 13=8Kbytes, 14=16Kbytes, 15=32Kbytes	0x0

5.12.6 MMCPLUS Data Control

This register is used to indicate to hardware the data path activity associated with a particular command. This register must be configured prior to writing to the command register.

MMCPLUS Data Control			
Address: 0x14	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
data_size	31-16	This field indicates how many bytes are transferred before signaling a completion interrupt. This field must be an integer multiple of the configured block size. When this register is read, the data_size field will reflect how many bytes of data remain to be transferred.	0x0
	15-2		
data_ena	1	Data path operation will not commence until this bit is enabled. 0=disable 1=enable.	0x0
data_direction	0	0=read 1=write.	0x0

5.12.7 MMCPLUS Agreement

MMCPLUS Argument			
Address: 0x18	Reset = 0x0	Type: RW	
Name	**Bit**	**Function**	**Reset**
cmd_argument	31-0	This contains the argument for the command to be sent to the SD card. A read of the register will provide the previous command argument that was written	0x0

5.12.8 MMCPLUS Command

MMCPLUS Command			
Address: 0x1C	Reset = 0x0	Type: RW	
Name	**Bit**	**Function**	**Reset**
Reserved	31-11	Reserved	
response_type	10-8	This field indicates to the hardware the type of response that is expected for the particular command issued. 000=response type R1 or R6 001=rsponse type R1b 010=response type R2 011=response type R3 or R4 100=no response.	0x0
	7		
data_command	6	This bit is only applicable to SDIO cards that make use of the interrupt feature on mmc_data1. Also it is only applicable when the interface is configured for the 4 bit mode of operation which will use mmc_data1 as a data line. Under this scenario the SDIO interrupt has a window of opportunity that it is valid. For hardware to understand that window of opportunity it must know whether the command that is being issued is associated with data or not. 1 = command is a data related command (read or write operation) 0 = command is not data related.	0x0
command_index	5-0	This is the command index to be sent to the SD card. The action of writing to this register initiates the command transfer process. A read of the register will provide the previous command index that was written.	0x0

5.12.9 MMCPLUS Command Response

MMCPLUS command response			
Address: 0x20	Reset = 0x0	Type: RO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-6	Reserved	

command_index	5-0	Contains the command index field of the last received response. If the response is type R2 or R3 the field will be '111111'.	0x0

5.12.10 MMCPLUS Response

MMCPLUS response			
Address: 0x24	Reset = 0x0	Type: RO	
Name	**Bit**	**Function**	**Reset**
card_status CID/CSD[39:8]	31-0	These registers are used to store the current SD response. If the response is type R1, R1b or R6 only the first 32 bits of the response field are valid Contains the card status field of the response or contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

MMCPLUS response			
Address: 0x28	Reset = 0x0	Type: RO	
Name	**Bit**	**Function**	**Reset**
CID/CSD[71:40]	31-0	Contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

MMCPLUS response			
Address: 0x2C	Reset = 0x0	Type: RO	
Name	**Bit**	**Function**	**Reset**
CID/CSD[103:72]	31-0	Contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

MMCPLUS response			
Address: 0x30	Reset = 0x0	Type: RO	
Name	**Bit**	**Function**	**Reset**
Reserved	31-24	Reserved	
CID/CSD[127:104]	23-0	Contains either the CID fields or CSD fields if a command is issued that queries this information.	0x0

5.12.11 FIFO Status

FIFO status		

Address: 0x34		Reset = 0x80	Type: RO/WO
Name	**Bit**	**Function**	**Reset**
rx_flush	31	When this bit is written ‘1’, the receive fifo is flushed. This bit is a write only bit.	0x0
tx_flush	30	When this bit is written ‘1’, the transmit fifo is flushed. This bit is a write only bit.	0x0
Reserved	29-16	Reserved	
rx_byte_count	15-8	Indicates how many bytes of data is present in the receive fifo. This is a read only field.	0x0
tx_byte_count	7-0	Indicates how many bytes of free space is available in the transmit fifo. This is a read only field.	0x80

5.12.12 FIFO Flag Configuration

FIFO flag Configuration			
Address: 0x38		**Reset = 0x4040**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	
rx_half_empty	15-8	Sets the FIFO level (in bytes) that asserts the receive “half” empty flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40
tx_half_full	7-0	Sets the FIFO level (in bytes) that asserts the transmit “half” full flag. The level setting is associated with how much data is in the FIFO. i.e if the setting is 0x20, then when there is 0x20 bytes of data available in the FIFO, the interrupt will be asserted.	0x40

5.12.13 Transmit FIFO

The Transmit FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to fill the fifo.

Transmit FIFO			
Address: 0x1000-0x1fff		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be written to the fifo. This register supports 8,16 or 32 bit writes and pushes the appropriate amount of data into the fifo. The size of the access must match the size that is programmed into the tx_fifo_size field in the Configuration1 register.	0x0

5.12.14 Receive FIFO

The Receive FIFO operates as a fifo even though it has a range of addresses. The wider range allows bus bursting to drain the fifo.

Receive FIFO			
Address: 0x2000-0x2fff		Reset = 0x0	Type: WO
Name	**Bit**	**Function**	**Reset**
data	31-0	This field contains the data to be read from the fifo. This register supports 8,16 or 32 bit reads and pops the appropriate amount of data from the fifo. The size of the access must match the size that is programmed into the rx_fifo_size field in the Configuration1 register.	0x0

5.13 I2C Registers

The register map is summarized below and described in the following sections.

Register	Address Offset	Mode
Interrupt Source	0x00	RO
Interrupt Mask	0x04	RW
Interrupt Clear	0x08	WO
Configuration1	0x0C	RW
Configuration2	0x10	RW
Configuration3	0x14	RW
Slave Address	0x18	RW
Target Data	0x1C	RW
Target Address	0x20	RW
Control	0x24	WO

5.13.1 Interrupt Source Register

The interrupt source register contains the raw unmasked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Interrupt Source Register			
Address: 0x00		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
arb_lost	3	Indicates that arbitration has been lost to another I2C master.	0x0

no_acknowledge	2	Indicates that an acknowledge was not received when the Slave ID was sent or that an acknowledge was not received during the data phase of a write transaction.	0x0
stop	1	Indicates that the “transaction stop” is complete. The serial interface runs at a very slow rate and a stop indication must be completed before software initiates a new “transaction start”.	0x0
acknowledge_complete	0	Indicates that the peripheral has acknowledged the transaction phase.	0x0

5.13.2 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Interrupt Mask Register			
Address: 0x04		**Reset = 0xf**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
arb_lost_mask	3	Masks the interrupt. 1=mask, 0=unmask.	0x1
no_acknowledge_mask	2	Masks the interrupt. 1=mask, 0=unmask.	0x1
stop_mask	1	Masks the interrupt. 1=mask, 0=unmask.	0x1
acknowledge_mask	0	Masks the interrupt. 1=mask, 0=unmask.	0x1

5.13.3 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the raw interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Interrupt Clear Register			
Address: 0x08		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-4	Reserved	
arb_lost_clr	3	Clears the interrupt when written ‘1’.	0x0
no_acknowledge_clr	2	Clears the interrupt when written ‘1’.	0x0
stop_clr	1	Clears the interrupt when written ‘1’.	0x0
acknowledge_clr	0	Clears the interrupt when written ‘1’.	0x0

5.13.4 Configuration1

Configuration1		

Address: 0x0C		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
master_ack	31	This bit provides the flexibility to allow or not allow the master to source an acknowledge for read transactions. When ‘1’, the master will source an acknowledge for read transactions. When ‘0’, the serial data line is not driven during the acknowledge bit period.	0x0
prim_sec	30	The I2C block is physically connected to two sets of IO pins. The primary pins do not have any sharing but the secondary pins are shared with dip_data[17:16]. The duplication of I2C pins is required in some circumstances because of the IO voltage selections. The DIP IOs can be powered with a different IO voltage than the primary I2C IOs. If the I2C is required to configure a DAC, them the secondary I2C port will have to be used if the DIP IOs are not compatible with the I2C IOs. This bit selects which set of IOs that the I2C communicates with. It is possible to have devices connected to both primary and secondary IOs as long as this selector is properly configured for the duration of the I2C communication. When set to the primary I2C, the secondary I2C sees no activity and vice versa. 0 = I2C connected to primary I2C IOs. 1 = I2C connected to secondary I2C IOs.	0x0
auto_mode_ena	29	When this bit is set a more auto-mated mode of the I2C interface is enabled and results in less interrupt over-head. Refer to Programming Model 4.9.2 for a detailed decription of manual and automatic modes. 0 = manual mode 1 = auto-matic mode	
master_arb_ena	28	When this bit is set the arbitration logic for multiple master systems is enabled. 0 = arbitration logic is disabled. 1 = arbitration logic is enabled.	
Reserved	27-16	Reserved	
clock_divider	15-0	This field contains the clock divider that is applied to the system clock to generate the serial data clock. The clock divide ratio applied to the system clock is this field plus one.	0x0

5.13.5 Configuration2

Configuration2			
Address: 0x10		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
TSUSTO	31-16	This field contains the count value that is applied to the system clock to generate the TSUSTO timing parameter. This parameter is the minimum time from serial_clock rising to serial_data rising during a “stop” indication. This is typically 4.0 µsec.	0x0

THDSTA	15-0	This field contains the count value that is applied to the system clock to generate the THDSTA timing parameter. This parameter is the minimum time from serial_data falling to serial_clock falling during a “start” indication. This is typically 4.0 µsec.	0x0

5.13.6 Configuration3

Configuration3			
Address: 0x14	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
TDELAY	31-16	This field contains the count value that is applied to the system clock to generate the TDELAY timing parameter. This parameter is the delay time between a slave address byte transmission and the next byte (read or write), the time between consecutive bytes (read or write), as well as the time between the last byte (read or write) and the stop condition. It should be noted that the hardware also automatically checks for the I2C slave stall indicator (SCL held low) and will stall its activity until the stall condition is removed.	0x0
TBUF	15-0	This field contains the count value that is applied to the system clock to generate the TBUF timing parameter. This parameter is the minimum idle time between a start and stop condition. This is typically 4.7 µsec.	0x0

5.13.7 Slave Address

Slave Address			
Address: 0x18	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
slave_id	7-1	This field contains the slave id of the peripheral being accessed. It should be programmed prior to initiating any transactions.	0x0
read_write	0	This is the R/W field for the Peripheral slave address. 1=read, 0=write.	0x0

5.13.8 Target Data

Target Data			
Address: 0x1C	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	

data	7-0	For I2C write transactions, the target data field is the third byte of data transmitted after the target address and is the data to be written to that target address. If the interface is operating in manual mode, new data can be written after the “acknowledge” interrupt is received. For read transactions, this register will not provide valid information until a transaction is started. Once a transaction is started, when an acknowledge interrupt has been received, valid read data will be present in this register. If the register is read prior to issuing a stop command, another read will occur on the serial control bus. If a stop has been issued, valid data from the read is present and no further action results.	0x0

5.13.9 Target Address

Target Address			
Address: 0x20		**Reset = 0x0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
address	7-0	The first byte of an I2C write transaction (after the slave address byte) is the address of the register that is being accessed. This field is programmed with that device address.	0x0

5.13.10 Control

Control			
Address: 0x24		**Reset = 0x0**	**Type: WO**
Name	**Bit**	**Function**	**Reset**
Reserved	31-2	Reserved	
stop	1	When this bit is written “1”, the serial control interface will create a stop condition.	0x0
start	0	When this bit is written “1”, the serial control interface will create a start condition, serialize out the data in the slave address register and either serialize out the data in the data register for a write or de-serialize the incoming data for a read.	0x0

5.14 PWM Registers

The register map is summarized below and described in the following sections.

Register	Address Offset	Mode
pwm config	0x00	RW
pwm control	0x04	RW
compare buffer 0	0x08	RW

count buffer 0	0x0C	RW
status 0	0x10	RO
compare buffer 1	0x14	RW
count buffer 1	0x18	RW
status 1	0x1C	RO
raw interrupt	0x20	RO

5.14.1 PWM Config

PWM Config		
Address: 0x00	**Reset = 0xc0_0000**	**Type: RW**

Name	Bit	Function	Reset
Reserved	31-26	Reserved	0x0
t1_timeout_clr	25	Clears the timer 1 timeout interrupt. This bit is self-clearing. 1 = clear interrupt.	0x0
t0_timeout_clr	24	Clears the timer 0 timeout interrupt. This bit is self-clearing. 1 = clear interrupt.	0x0
int_mask	23-22	Mask for the two timer interrupts. 1 = interrupt is masked 0 = interrupt is not masked	0x3
t1_clk_sel	21-19	Mux selector for the PWM timer1. 000: ½ 001: ¼ 010: 1/8 011: 1/16 1xx: TCLK0	0x0
t0_clk_sel	18-16	Mux selector for the PWM timer0. 000: ½ 001: ¼ 010: 1/8 011: 1/16 1xx: TCLK0	0x0
dead_zone_length	15-8	Specifies the dead zone length	0x0
prescaler	7-0	Specifies the pre-scaler value	0x0

5.14.2 PWM Control

PWM Control		
Address: 0x04	**Reset = 0x0**	**Type: RW**

Name	Bit	Function	Reset
Reserved	31-9	Reserved	0x0
t1_auto_reload	8	Controls the auto reload function of timer 1. 0 = one shot 1 = interval mode (auto reload)	0x0
t1_inverter	7	Controls the output inverter for timer 1. 0 = inverter off 1 = inverter on	0x0
t1_update	6	Controls the manual update for timer 1. 0 = no operation 1 = the timer is updated with the count buffer value.	0x0
t1_start	5	Controls the operation of timer 1. 0 = timer stopped 1 = timer started	0x0
dead_zone_en	4	Controls the dead zone operation 0 = disabled 1 = enabled	0x3
t0_auto_reload	3	Controls the auto reload function of timer 0. 0 = one shot 1 = interval mode (auto reload)	0x0
t0_inverter	2	Controls the output inverter for timer 0. 0 = inverter off 1 = inverter on	0x0
t0_update	1	Controls the manual update for timer 0. 0 = no operation 1 = the timer is updated with the count buffer value.	0x0
t0_start	0	Controls the operation of timer 0. 0 = timer stopped 1 = timer started	0x0

5.14.3 Compare Buffer 0

Compare buffer 0		
Address: 0x08	**Reset = 0x0**	**Type: RW**

Name	Bit	Function	Reset
Reserved	31-16	Reserved	0x0
t0_com_buffer	15-0	Timer 0 compare buffer.	0x0

5.14.4 Count Buffer 0

Count buffer 0		

Address: 0x0C	Reset = 0x0	Type: RW	
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	0x0
t0_count_buffer	15-0	Timer 0 count buffer.	0x0

5.14.5 Status 0

Status 0			
Address: 0x10	**Reset = 0x0**	**Type: RO**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	0x0
t0_timer_cnt	15-0	Timer 0 count observation register.	0x0

5.14.6 Compare Buffer 1

Compare buffer 1			
Address: 0x14	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	0x0
t1_com_buffer	15-0	Timer 1 compare buffer.	0x0

5.14.7 Count Buffer 1

Count buffer 1			
Address: 0x18	**Reset = 0x0**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	0x0
t1_count_buffer	15-0	Timer 1 count buffer.	0x0

5.14.8 Status 1

Status 1		
Address: 0x1C	**Reset = 0x0**	**Type: RO**

Name	Bit	Function	Reset
Reserved	31-16	Reserved	0x0
t1_timer_cnt	15-0	Timer 1 count observation register.	0x0

5.14.9 Raw Interrupt

Raw Interrupt		
Address: 0x20	**Reset = 0x0**	**Type: RO**

Name	Bit	Function	Reset
Reserved	31-2	Reserved	0x0
t1_timeout	1	Timer 1 timeout interrupt.	0x0
t0_timeout	0	Timer 0 timeout interrupt.	0x0

5.15 KeyScan Registers

The register map is summarized below and described in the following sections.

Register	Address Offset	Mode
Interrupt Mask	0x00	RW
Interrupt Source	0x04	RO
Interrupt Clear	0x08	RW
Keypad control0	0x0C	RW
Keypad control1	0x10	RW
Keypad time	0x14	RW
Keypad value	0x18	RO

5.15.1 Interrupt Mask Register

The interrupt mask register provides a mechanism to individually mask one or more of the interrupt sources.

Interrupt Mask Register			
Address: 0x00		Reset = 0xFFFF	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	
Key sensing	15-0	Masks the interrupt. 1=mask, 0=unmask.	0xFFFF

5.15.2 Interrupt Source Register

The interrupt source register contains the masked interrupts and can be used for polling purposes (instead of the external interrupt pin) or for determining which interrupt(s) have caused the external interrupt pin to assert.

Interrupt Source Register			
Address: 0x01		Reset = 0x0	Type: RO
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	
Key sensing	15-0	Each key sense is detected by hardware.	0x0000

5.15.3 Interrupt Clear Register

The interrupt clear register provides the mechanism for clearing the interrupt sources. Writing a „1. to the interrupt bit location will clear the interrupt.

Reading this register returns unmasked interrupt source that is interrupt source pending register.

Interrupt Clear Register			
Address: 0x08		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**
Reserved	31-16	Reserved	
Key sensing	15-0	Writing a ‘1’ : relative interrupt source will be cleared. Reading : returns unmasked interrupt source	0x0000

5.15.4 Keypad Control0

Keypad control0			
Address: 0x0C		Reset = 0x0	Type: RW
Name	**Bit**	**Function**	**Reset**

Reserved	31-6	Reserved	0x0
keypad_enable	5	Keypad operation enable 1 = enable 0 = disable	0x0
polarity	4	Keyscan output and key sense polarity 1 = active high (external pull-down) 0 = active low (external pull-up)	0x0
mode_sel	3	1 = single input mode. When in this mode, keypad module will sense only one button at a time. It is used in typing mode. 0 = multi input mode. When in this mode, keypad module will sense multi-button at a time. It is useful in gaming mode that requires moving diagonal and shooting with moving.	0x0
Reserved	2	Reserved	0
auto_clr	1	Keypad auto clear enable. When enabled, the keypad value register is cleared after it is read. 1 = enabled 0 = disabled	0x0
value_clr	0	keypad register clear. This bit is self resetting 1 = the keypad value register is cleared. 0 = no action	0x3

5.15.5 Keypad Control1

Keypad Control 1		
Address: 0x10	**Reset = 0x0**	**Type: RW**

Name	**Bit**	**Function**	**Reset**
Reserved	31-8	Reserved	
keysense3_en	7	Keysense3 enable 1 : enable 0 : disable	0x0
keysense2_en	6	Keysense2 enable 1 : enable 0 : disable	0x0
keysense1_en	5	Keysense1 enable 1 : enable 0 : disable	0x0
keysense0_en	4	Keysense0 enable 1 : enable 0 : disable	0x0
keyscan3_en	3	Keyscan3 enable 1 : enable 0 : disable	0x0

keyscan2_en	2	Keyscan2 enable 1 : enable 0 : disable	0x0
keyscan1_en	1	Keyscan1 enable 1 : enable 0 : disable	0x0
keyscan0_en	0	Keyscan0 enable 1 : enable 0 : disable	0x0

5.15.6 Keypad Time

Keypad time			
Address: 0x14	**Reset = 0x1FFF**	**Type: RW**	
Name	**Bit**	**Function**	**Reset**
Reserved	31-13	Reserved	0x0
scan_time	12-0	Key scan driving time. Scan_time_freq = sys_freq / (scan_time+1)	0x1FFF

5.15.7 Keypad Value

Keypad value			
Address: 0x18	Reset = 0x0	Type: RO	
Name	Bit	Function	Reset
Reserved	31-16	Reserved	0x0
sense_value	15-0	Contains the sense value vector. This vector is a dynamic view of the key sense information and will change for every key scan interval. If a latched version of this field is desired, please read the interrupt source register.	0x0

The following table illustrates the sense_value vector.

	key_sense3	key_sense2	key_sense1	key_sense0
key_scan3	sense_value15	sense_value14	sense_value13	sense_value12
key_scan2	sense_value11	sense_value10	sense_value9	sense_value8
key_scan1	sense_value7	sense_value6	sense_value5	sense_value4
key_scan0	sense_value3	sense_value2	sense_value1	sense_value0

5.16 GPIO Registers

The General Purpose Input Output (GPIO) pins are controlled with several registers.

- GPIO Enable Registers activate the pin for GPIO use, otherwise the pin has its primary or alternate function.
- GPIO Direction Registers determine the GPIO as input or output.

- GPIO OutData Registers determine the level of the GPIO when configured as output.
- GPIO InData Registers read the level of GPIO when configured as input.

In case a GPIO pin is used as output, it is preferable to activate the enable bit once the direction and OutData have been set.

See also Section 2.4, Pin Configuration and Section 4.12, GPIOs and Alternate Functions.

5.16.1 GPIO Enable Registers

Table 84. GPIO Enable Registers

GPIO Enable 1			
Address: 0xd00a_0000		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_enable[31-0]	31-0	These bits are used to enable the alternate GPIO functionality for the external pins listed in the preceding table. 0 = GPIO disabled 1 = GPIO enabled	0
GPIO Enable 2			
Address: 0xd00a_0004		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_enable[63-32]]	31-0	These bits are used to enable the alternate GPIO functionality for the external pins listed in the preceding table. 0 = GPIO disabled 1 = GPIO enabled	0
GPIO Enable 3			
Address: 0xd00a_0008		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_enable[95-64]	31-0	These bits are used to enable the alternate GPIO functionality for the external pins listed in the preceding table. 0 = GPIO disabled 1 = GPIO enabled	0

5.16.2 GPIO Direction Registers

Table 85. GPIO direction Registers

GPIO Direction 1			
Address: 0xd00a_000C		**Reset = 0xffff_ffff**	**Type: RW**
Name	**Bit**	**Function**	**Reset**

Table 85. GPIO direction Registers

gpio_dir[31-0]	31-0	If a GPIO has been enabled these bits configure the GPIO as either an input or output. 0 = output 1 = input	0xffff_ffff
GPIO Direction 2			
Address: 0xd00a_0010		**Reset = 0xffff_ffff**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_dir[63-32]	31-0	If a GPIO has been enabled these bits configure the GPIO as either an input or output. 0 = output 1 = input	0xffff_ffff
GPIO Direction 3			
Address: 0xd00a_0014		**Reset = 0xffff_ffff**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_dir[95-64]	31-0	If a GPIO has been enabled these bits configure the GPIO as either an input or output. 0 = output 1 = input	0xffff_ffff

5.16.3 GPIO OutData Registers

Table 86. GPIO OutData Registers

GPIO OutData 1			
Address: 0xd00a_0018		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_out[31-0]	31-0	When the register is written, the GPIO will latch the written value if the GPIO is an output. For an input the write is ignored. When read, it will reflect what was previously written (whether or not the GPIO is configured as an input or output).	0
GPIO OutData 2			
Address: 0xd00a_001c		**Reset = 0**	**Type: RW**
Name	**Bit**	**Function**	**Reset**
gpio_out[63-32]	31-0	When the register is written, the GPIO will latch the written value if the GPIO is an output. For an input the write is ignored. When read, it will reflect what was previously written (whether or not the GPIO is configured as an input or output).	0

Table 86. GPIO OutData Registers

GPIO OutData 3			
Address: 0xd00a_0020		Reset = 0	Type: RW
Name	**Bit**	**Function**	**Reset**
gpio_out[95-64]	31-0	When the register is written, the GPIO will latch the written value if the GPIO is an output. For an input the write is ignored. When read, it will reflect what was previously written (whether or not the GPIO is configured as an input or output).	0

5.16.4 GPIO InData Registers

Table 87. GPIO InData Registers

GPIO InData 1			
Address: 0xd00a_0024		Reset = 0	Type: RO
Name	**Bit**	**Function**	**Reset**
gpio_in[31-0]	31-0	When this register is read, the bits reflect the level of the external signal if the GPIO is an input or reflects the previously written value if the GPIO is an output.	0
GPIO InData 2			
Address: 0xd00a_0028		Reset = 0	Type: RO
Name	**Bit**	**Function**	**Reset**
gpio_in[63-32]	31-0	When this register is read, the bits reflect the level of the external signal if the GPIO is an input or reflects the previously written value if the GPIO is an output.	0
GPIO InData 3			
Address: 0xd00a_002c		Reset = 0	Type: RO
Name	**Bit**	**Function**	**Reset**
gpio_in[95-64]	31-0	When this register is read, the bits reflect the level of the external signal if the GPIO is an input or reflects the previously written value if the GPIO is an output.	0

6 Packaging

6.1 SCP2201

The SCP2201 is available in a 236-ball BGA package of size 9 mm x 9 mm x 1.34 mm. Figure 57 contains SCP220x packaging information. All dimensions are in mm.

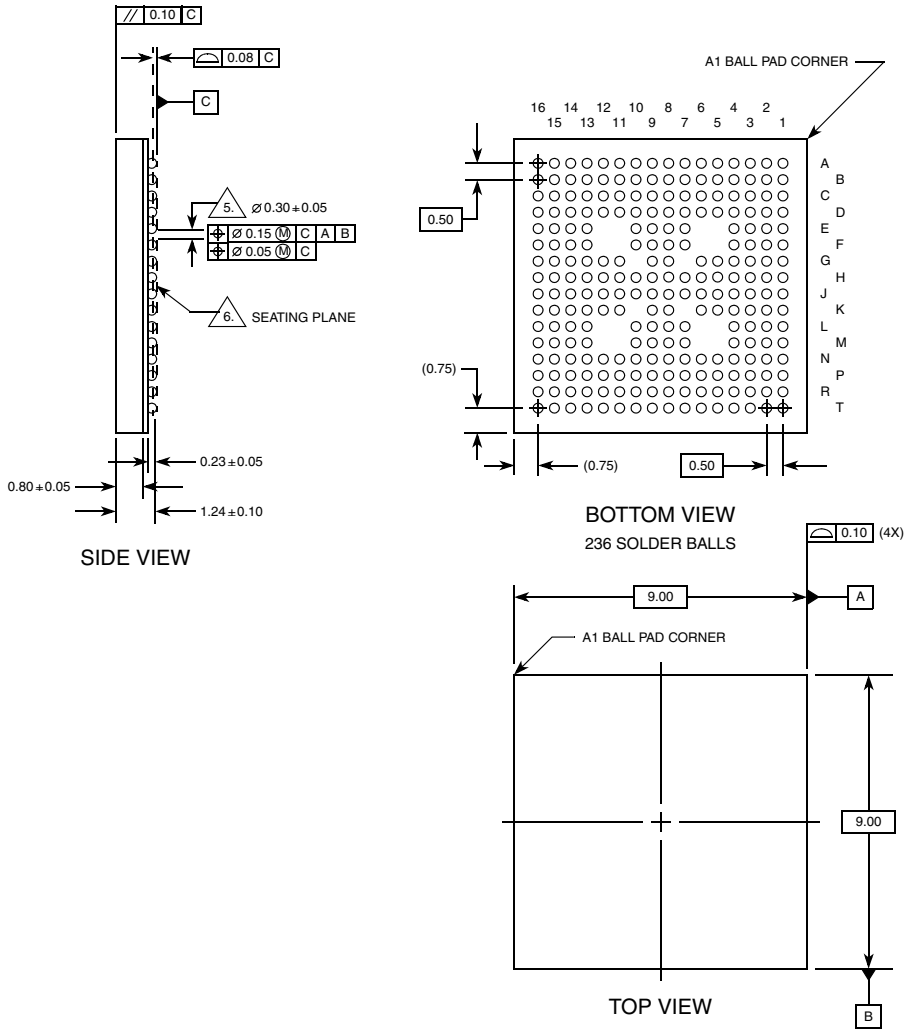


Figure 58. SCP2201 Package

6.2 SCP2207

The SCP2207 is available in a 236-ball BGA package of size 9 mm x 9 mm x 1.34 mm. The following figure contains SCP220x packaging information. All dimensions are in mm.

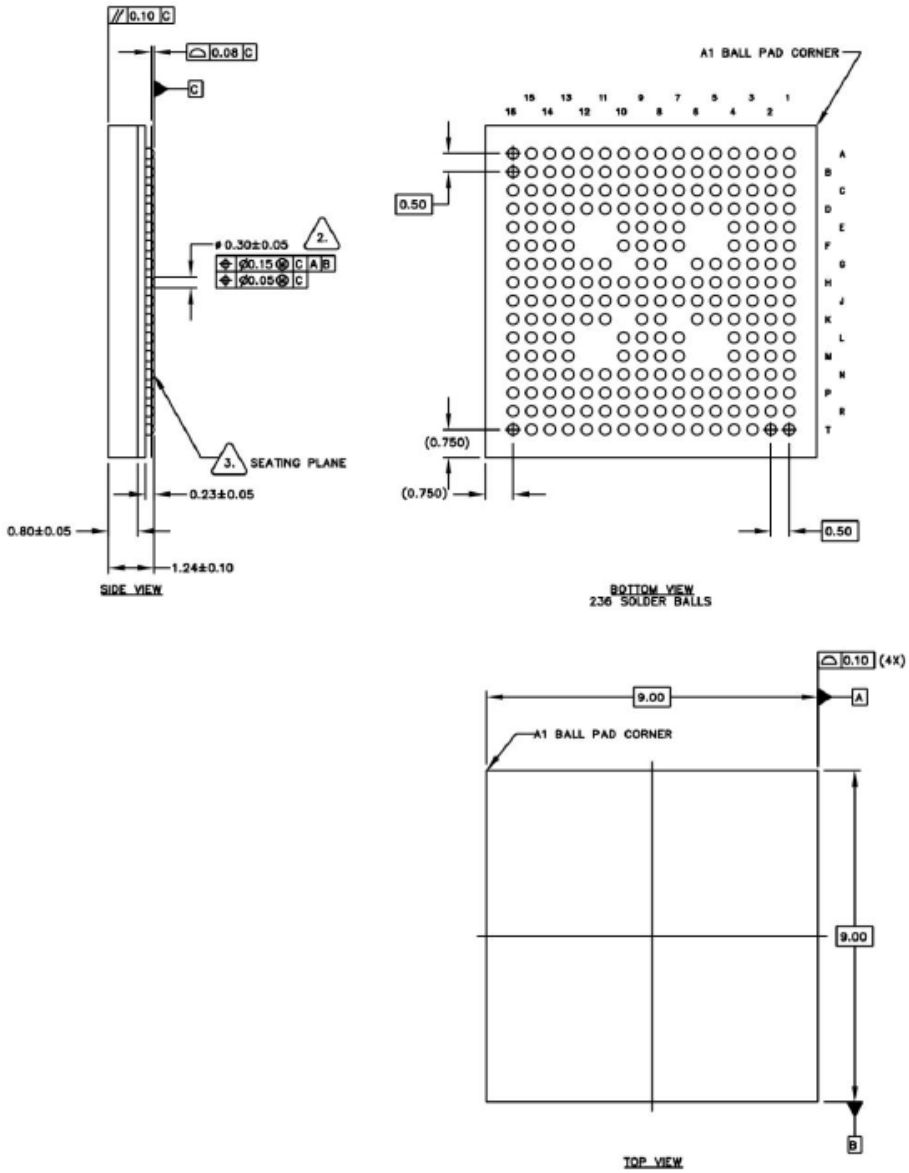


Figure 59. SCP2207 Package

6.3 SCP220x Pinout

The following table describes the physical pins of the devices belonging to the SCP220x series. The pins are organized into functional groups. External interfaces are grouped together on the IO voltage banks that can be powered by either 3.0 V DC ± 10%. Many outputs may be configured as having low or high output drive strength by programming the device. The output drive capability is indicated in the PAD type column.

Note that '-'. indicates the pin does not apply to the device as the signal is not balled out on the package. Pins in the 'No Connect' section of the table are used solely for test purposes and should not be used in normal operating mode. Some pins are designated 'reserved_#'. These pins may only be used as the corresponding GPIOs or alternate functionality as defined in section 4.11. Primary functionality of these pins is reserved and not intended for use.

Table 88. SCP220x Pinout

Pin Name	Power Domain	PAD Type	Default PU/PD	SCP2201 and SCP2207 Ball
Sensor				
sensor_D[0]	SIF	Input	PD	C6
sensor_D[1]	SIF	Input	PD	C7
sensor_D[2]	SIF	Input	PD	D7
sensor_D[3]	SIF	Input	PD	E7
sensor_D[4]	SIF	Input	PD	B8
sensor_D[5]	SIF	Input	PD	C8
sensor_D[6]	SIF	Input	PD	D8
sensor_D[7]	SIF	Input	PD	E8
sensor_D[8]	SIF	Input	PD	B9
sensor_D[9]	SIF	Input	PD	C9
sensor_fclk	SIF	Input	PD	B11
sensor_pclk	SIF	Input	PD	E9
sensor_rclk	SIF	Input	PD	D9
sensor_fodd	SIF	Bi-dir. 4 mA / 8 mA	-	C10
sensor_gpio	SIF	Bi-dir. 4 mA / 8 mA	-	B7
sensor_clkout	SIF	Bi-dir. 4 mA / 8 mA	-	B10
I2C				
scl	SIF	Bi-dir. 4 mA / 8 mA	-	E10
sda	SIF	Bi-dir. 4 mA / 8 mA	-	D10
NAND				
nand_wen	NAND	Bi-dir. 2 mA / 4 mA	-	R1
nand_ren	NAND	Bi-dir. 2 mA / 4 mA	-	T1
nand_cen[3]	NAND	Bi-dir. 2 mA / 4 mA	PU	K3
nand_cen[2]	NAND	Bi-dir. 2 mA / 4 mA	PU	T2
nand_cen[1]	NAND	Bi-dir. 2 mA / 4 mA	PU	R2
nand_cen[0]	NAND	Bi-dir. 2 mA / 4 mA	PU	P2
nand_ale	NAND	Bi-dir. 2 mA / 4 mA	-	L3
nand_cle	NAND	Bi-dir. 2 mA / 4 mA	-	M3

Table 88. SCP220x Pinout

nand_D[0]	NAND	Bi-dir. 2 mA / 4 mA	-	N3
nand_D[1]	NAND	Bi-dir. 2 mA / 4 mA	-	P3
nand_D[2]	NAND	Bi-dir. 2 mA / 4 mA	-	K4
nand_D[3]	NAND	Bi-dir. 2 mA / 4 mA	-	N4
nand_D[4]	NAND	Bi-dir. 2 mA / 4 mA	-	P4
nand_D[5]	NAND	Bi-dir. 2 mA / 4 mA	-	K5
nand_D[6]	NAND	Bi-dir. 2 mA / 4 mA	-	N5
nand_D[7]	NAND	Bi-dir. 2 mA / 4 mA	-	N6
wp_N	NAND	Input	-	-
DAC				
DAC_comp	DAC	Analog I/O	-	L2
DAC_vref_out	DAC	Analog I/O	-	M1
DAC_rset	DAC	Analog I/O	-	M2
DAC_vref_in	DAC	Analog I/O	-	N2
DAC_io	DAC	Analog I/O	-	L1
Display Interface Port				
dip_data[0]	DIP	Bi-dir. 4 mA / 8 mA	PD	K16
dip_data[1]	DIP	Bi-dir. 4 mA / 8 mA	PU	K15
dip_data[2]	DIP	Bi-dir. 4 mA / 8 mA	PD	K14
dip_data[3]	DIP	Bi-dir. 4 mA / 8 mA	PD	K13
dip_data[4]	DIP	Bi-dir. 4 mA / 8 mA	PU	J13
dip_data[5]	DIP	Bi-dir. 4 mA / 8 mA	PD	L16
dip_data[6]	DIP	Bi-dir. 4 mA / 8 mA	PD	L15
dip_data[7]	DIP	Bi-dir. 4 mA / 8 mA	PD	L14
dip_data[8]	DIP	Bi-dir. 4 mA / 8 mA	PD	L13
dip_data[9]	DIP	Bi-dir. 4 mA / 8 mA	-	M16
dip_data10]	DIP	Bi-dir. 4 mA / 8 mA	-	M15
dip_data[11]	DIP	Bi-dir. 4 mA / 8 mA	-	M14
dip_data[12]	DIP	Bi-dir. 4 mA / 8 mA	-	M13
dip_data[13]	DIP	Bi-dir. 4 mA / 8 mA	-	N16
dip_data[14]	DIP	Bi-dir. 4 mA / 8 mA	-	N15
dip_data[15]	DIP	Bi-dir. 4 mA / 8 mA	-	N14

Table 88. SCP220x Pinout

dip_data[16]	DIP	Bi-dir. 4 mA / 8 mA	-	N13
dip_data[17]	DIP	Bi-dir. 4 mA / 8 mA	-	P16
dip_data[18]	DIP	Bi-dir. 4 mA / 8 mA	-	P15
dip_data[19]	DIP	Bi-dir. 4 mA / 8 mA	-	R16
dip_data[20]	DIP	Bi-dir. 4 mA / 8 mA	-	R15
dip_data[21]	DIP	Bi-dir. 4 mA / 8 mA	-	T16
dip_data[22]	DIP	Bi-dir. 4 mA / 8 mA	-	T15
dip_data[23]	DIP	Bi-dir. 4 mA / 8 mA	-	T14
dip_RS	DIP	Output 4 mA / 8 mA	-	R14
dip_CSn0	DIP	Output 4 mA / 8 mA	PU	R13
dip_CSn1	DIP	Output 4 mA / 8 mA	PU	T13
dip_CSn2	DIP	Bi-dir. 4 mA / 8 mA	PU	R12
dip_CSn3	DIP	Bi-dir. 4 mA / 8 mA	PU	P12
dip_Wrn	DIP	Output 4 mA / 8 mA	-	N12
dip_OEn	DIP	Bi-dir. 4 mA / 8 mA	-	R11
dip_pclk	DIP	Bi-dir. 4 mA / 8 mA	-	N11
dip_cpu_vsync	DIP	Bi-dir. 4 mA / 8 mA	-	P11
UART				
uart_Rx	MISCIF	Bi-dir. 2 mA / 4 mA	-	A2
uart_Tx	MISCIF	Bi-dir. 2 mA / 4 mA	-	B1
uart_cts	MISCIF	Bi-dir. 2 mA / 4 mA	-	B2
uart_rts	MISCIF	Bi-dir. 2 mA / 4 mA	-	C1
SPI				
spi_Clk	MISCIF	Bi-dir. 2 mA / 4 mA	-	C2
spi_CS	MISCIF	Bi-dir. 2 mA / 4 mA	PU	C3
spi_Tx	MISCIF	Bi-dir. 2 mA / 4 mA	-	C4
spi_Rx	MISCIF	Bi-dir. 2 mA / 4 mA	-	C5
spi1_Clk	MISCIF	Bi-dir. 2 mA / 4 mA	-	D1
spi1_CS	MISCIF	Bi-dir. 2 mA / 4 mA	PU	D2
spi1_Tx	MISCIF	Bi-dir. 2 mA / 4 mA	-	D3
spi1_Rx	MISCIF	Bi-dir. 2 mA / 4 mA	-	D4
Media Storage				

Table 88. SCP220x Pinout

sd_clk	SDMMC	Bi-dir. 4 mA / 8 mA	-	F1
sd_cmd	SDMMC	Bi-dir. 4 mA / 8 mA	-	F2
sd_D[0]	SDMMC	Bi-dir. 4 mA / 8 mA	-	F3
sd_D[1]	SDMMC	Bi-dir. 4 mA / 8 mA	-	G1
sd_D[2]	SDMMC	Bi-dir. 4 mA / 8 mA	-	G2
sd_D[3]	SDMMC	Bi-dir. 4 mA / 8 mA	-	G3
Audio				
audio_clkr	AUDIO	Bi-dir. 2 mA / 4 mA	-	P10
audio_clkx	AUDIO	Bi-dir. 2 mA / 4 mA	-	N10
audio_dr	AUDIO	Bi-dir. 2 mA / 4 mA	-	M10
audio_dx	AUDIO	Bi-dir. 2 mA / 4 mA	-	N9
audio_fsr	AUDIO	Bi-dir. 2 mA / 4 mA	-	M9
audio_fsx	AUDIO	Bi-dir. 2 mA / 4 mA	-	N8
mclk	AUDIO	Bi-dir. 2 mA / 4 mA	-	M8
MP2TS				
mp2ts_clk	MISCIF	Input	n.a.	E1
mp2ts_valid	MISCIF	Input	n.a.	E2
mp2ts_sync	MISCIF	Input	n.a.	E3
mp2ts_data	MISCIF	Input	n.a.	E4
USB				
usb_phy_id	USB	USB PAD	n.a.	J5
usb_phy_vbus	USB	USB PAD	n.a.	J4
usb_phy_Plus	USB	USB PAD	n.a.	K1
usb_phy_Minus	USB	USB PAD	n.a.	J1
usb_phy_res	USB	USB PAD	n.a.	J3
utmiotg_drvvbus	AUDIO	Bi-dir. 4 mA / 8 mA	PU	R10
Smart Card				
sc_io	SCCARD	Bi-dir. 4 mA / 8 mA	-	H1
sc_card_detect	SCCARD	Bi-dir. 4 mA / 8 mA	-	H2
sc_card_voltage	SCCARD	Bi-dir. 4 mA / 8 mA	-	H3
sc_fcb	SCCARD	Bi-dir. 4 mA / 8 mA	-	H4
sc_clk	SCCARD	Bi-dir. 4 mA / 8 mA	PU	H5

Table 88. SCP220x Pinout

sc_power_on	SCCARD	Bi-dir. 4 mA / 8 mA	-	H6
sc_rst	SCCARD	Bi-dir. 4 mA / 8 mA	-	J6
SDRAM				
DMCLK	SDRAM	Output 4 mA / 8 mA	-	-
DMCLKn	SDRAM	Output 4 mA / 8 mA	-	-
A[0]	SDRAM	Output 4 mA / 8 mA	-	-
A[1]	SDRAM	Output 4 mA / 8 mA	-	-
A[2]	SDRAM	Output 4 mA / 8 mA	-	-
A[3]	SDRAM	Output 4 mA / 8 mA	-	-
A[4]	SDRAM	Output. 4 mA / 8 mA	-	-
A[5]	SDRAM	Output 4 mA / 8 mA	-	-
A[6]	SDRAM	Output 4 mA / 8 mA	-	-
A[7]	SDRAM	Output 4 mA / 8 mA	-	-
A[8]	SDRAM	Output 4 mA / 8 mA	-	-
A[9]	SDRAM	Output 4 mA / 8 mA	-	-
A[10]	SDRAM	Output 4 mA / 8 mA	-	-
A[11]	SDRAM	Output 4 mA / 8 mA	-	-
A[12]	SDRAM	Output 4 mA / 8 mA	-	-
CKE	SDRAM	Output. 4 mA / 8 mA	-	-
WEn	SDRAM	Output 4 mA / 8 mA	-	-
CASn	SDRAM	Output 4 mA / 8 mA	-	-
RASn	SDRAM	Output 4 mA / 8 mA	-	-
CSn	SDRAM	Output 4 mA / 8 mA	-	-
BA[0]	SDRAM	Output 4 mA / 8 mA	-	-
BA[1]	SDRAM	Output 4 mA / 8 mA	-	-
DM[0]	SDRAM	Output 4 mA / 8 mA	-	-
DM[1]	SDRAM	Output 4 mA / 8 mA	-	-
DM[2]	SDRAM	Output 4 mA / 8 mA	-	-
DM[3]	SDRAM	Output 4 mA / 8 mA	-	-
DQS[0]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQS[1]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQS[2]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-

Table 88. SCP220x Pinout

DQS[3]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[0]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[1]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[2]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[3]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[4]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[5]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[6]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[7]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[8]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[9]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[10]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[11]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[12]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[13]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[14]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[15]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[16]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[17]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[18]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[19]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[20]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[21]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[22]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[23]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[24]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[25]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[26]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[27]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[28]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[29]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
DQ[30]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-

Table 88. SCP220x Pinout

DQ[31]	SDRAM	Bi-dir. 4 mA / 8 mA	-	-
System Signals & JTAG				
Clkin	OSC	Oscillator pad	n.a.	H16
Clkout	OSC	Oscillator pad	n.a.	J16
resetN	HPI	Input	-	H11
hw_deep_secure	DIP	Input	-	P13
rtck	MISCIF	Output 4 mA / 8 mA	-	A3
tck	MISCIF	Input	PU	A4
Ntrst	MISCIF	Input	PD	B5
tdi	MISCIF	Input	PU	B6
tdo	MISCIF	Output 4 mA / 8 mA	-	B3
tms	MISCIF	Input	PU	B4
testmode	MISCIF	Input	PD	A1
bootmode (1)	DIP	Input	-	P14
pkg_opt0 (2)	SDRAM	Input	-	-
pkg_opt1 (2)	SDRAM	Input	-	-
pkg_opt2 (2)	SDRAM	Input	-	-
jtag_sel_p0 (2)	MISCIF	Input	-	-
jtag_sel_p1 (2)	MISCIF	Input	-	-
jtag_sel_p2 (2)	MISCIF	Input	-	-
No Connects				

Table 88. SCP220x Pinout

NC	-	-	-	A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, M7, N7, P5, P6, P7, P8, P9, R4, R5, R6, R7, R8, R9, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, A16, B16, B15, B14, B12, C16, B13, C13, D16, D15, D14, D13, E16, E15, E14
Reserved Pins				
reserved_1	HPI	Bi-dir. 2 mA / 4 mA	-	C14
reserved_2	HPI	Bi-dir. 2 mA / 4 mA	PD	C15
reserved_3	HPI	Bi-dir. 2 mA / 4 mA	-	G11
reserved_4	HPI	Bi-dir. 2 mA / 4 mA	-	H13
reserved_5	HPI	Bi-dir. 2 mA / 4 mA	-	G12
reserved_6	HPI	Bi-dir. 2 mA / 4 mA	-	H12
reserved_7	HPI	Bi-dir. 4 mA / 8 mA	-	G14
reserved_8	HPI	Bi-dir. 4 mA / 8 mA	-	G15
reserved_9	HPI	Bi-dir. 4 mA / 8 mA	-	G16
reserved_10	HPI	Bi-dir. 4 mA / 8 mA	-	F13
reserved_11	HPI	Bi-dir. 4 mA / 8 mA	-	F14
reserved_12	HPI	Bi-dir. 4 mA / 8 mA	-	F15
reserved_13	HPI	Bi-dir. 4 mA / 8 mA	-	F16
reserved_14	HPI	Bi-dir. 4 mA / 8 mA	-	E13
reserved_15	HPI	Bi-dir. 2 mA / 4 mA	-	G13
reserved_16	NAND	Input	-	-
reserved_17	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_18	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_19	NAND	Bi-dir. 2 mA / 4 mA	PU	-

Table 88. SCP220x Pinout

reserved_20	NAND	Bi-dir. 2 mA / 4 mA	PU	-
reserved_21	NAND	Bi-dir. 2 mA / 4 mA	PU	-
reserved_22	NAND	Bi-dir. 2 mA / 4 mA	PU	-
reserved_23	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_24	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_25	NAND	Bi-dir. 2 mA / 4 mA	-	-
reserved_26	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_27	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_28	DIP	Bi-dir. 4 mA / 8 mA	-	-
reserved_29	DIP	Bi-dir. 4 mA / 8 mA	-	-

Notes:
(1) Must be pulled-up to VDD_DIP (100 KOhms 1/16 W 5% suggested) for correct operation on SCP2201 and SCP2207.
(2) Software queries the SCP2207 pkg_opt[2:0] and jtag_sel_p[2:0] pins to determine the SDRAM used in the system. Currently CogniVue supports a 128 MB Micron mobile DDR SDRAM, and the pkg_opt[2:0] and jtag_sel_p[2:0] must both be set to binary '101' (decimal value 5). Consult the factory for interfacing to any other memory.

Table 89. SCP220x Power Pin

Power Pin Name	Description	SCP2201 and SCP2207 Ball #
VDD_CORE	Core supply	G8, G9, H8, H9
VDD_LP	Low-power audio/video supply	J7, J10, K11, K12
VDDA_PLL	Analog PLL supply	J14
VSSA_PLL	Return for PLL VDD (**DO NOT CONNECT TO GROUND**)	H14
VDD_SDRAM	SDRAM core and EBI/SDRAM IO supply	F7,L7
VDD_OSC	Analog supply for crystal pad	J15
VDD_USB	Analog supply for USB	K2
VDDL_USB	USB core supply	-
VDDA_DAC	Analog supply for internal DAC	P1
VDD_SENSOR	Sensor Interface (SIF) and I2C IO supply	F10
VDD_GPIO	GPIO and keyscan supply	D12

Table 89. SCP220x Power Pin

VDD_DIP	DIP IO supply	L10
VDD_MISCIF	MP2TS, JTAG, SPI, UART IO supply	D5
VDD_SDMMC	SD/MMC IO supply	F4
VDD_AUDIO	Audio IO supply	K9
VDD_SCCARD	Smart Card power supply	G5
VDD_NAND	NAND Flash IO supply	L4
VSS	Common ground	C11, C12, D6, D11, F8, F9, G4, G6, H7, H10, J8, J9, J11, J12, K6, K8, L8, L9, M4, R3
VSS_DAC	Analog ground for internal DAC	N1
VSS_USB	Analog ground for USB	J2
VSSL_USB	USB core ground	-
VSS_OSC	Analog ground for crystal pad	H15

7 Electrical Specifications

7.1 Absolute Maximum Rating

The following table describes the absolute maximum ratings for the SCP220x.

Table 90. SCP220x Absolute Maximum Rating

Item	Rating	Unit
I/O supply voltage	-0.2 to +3.3	V
Core supply voltage	-0.2 to +1.2	V
Input voltage for a signal pin	-0.3 to +3.3	V
Storage temperature	-40 to +150	°C
Short circuit duration (single output in high state to GND)	1	second

WARNING: Permanent device damage may occur if the absolute maximum ratings are exceeded. Functional operation should be restricted to the conditions as detailed in the operational sections of this data sheet. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

7.2 Recommended Operating Ranges

The following table describes the recommended operating ranges for the SCP220x. Note that IO_VDD refers to the subset of power supplies for the device I/Os as specified in Table 89. These supplies must be powered up before VDD_CORE is powered up.

Table 91. SCP220x Recommended Operating Range

Item	Symbol	Min.	Typ.	Max.	Unit
Core supply voltage Low power voltage	VDD_CORE VDD_LP	0.9	1.0	1.1	V
I/O Supply Voltage	IO_VDD	2.7	3.0	3.3	V
Sensor Interface (SIF)	VDD_SENSOR	2.7	3.0	3.3	V
		1.71	1.8	1.98	V
Analog PLL supply	VDDA_PLL	2.7	3.0	3.3	V
SDRAM memory supply	VDD_SDRAM	1.7	1.8	1.98	V
Analog supply for USB	VDD_USB	3.0	3.3	3.6	V
Analog supply for internal DAC	VDDA_DAC	2.97	3.3	3.63	V
Operating temperature	$T_{operating}$ (Industrial Qualified Parts)	-40		105	°C

Notes:
1. IO_VDD = VDD_GPIO, VDD_DIP, VDD_AUDIO, VDD_NAND, VDD_SENSOR, VDD_SDMMC, VDD_MISCIF, VDD_SCCARD, VDD_OSC
2.Sensor Interface supply (VDD_SENSOR) shall be part of the IO_VDD group when powered at 3.0V.
3.IO_VDD must always be on.
4. VDDA_PLL, VDD_LP must always be on.
5. VDD_DAC and VDD_USB must be turned on/off when VDD_CORE supply is turned on/off.

7.3 Thermal Characteristics

Table 92. SCP220x Thermal Characteristics

Air Velocity (m/s)	θ JA	θ JB	θ JC
0	25.5 °C/W	19.9 °C/W	8.5 °C/W

7.4 DC Characteristics

The following table describes the DC characteristics of the SCP220x.

Table 93. SCP220x DC Characteristics

Item	Symbol	Min.	Typ.	Max.	Unit
Input Voltage, high	V_{IH} 3.0	2		3.3	V
Input Voltage, low	V_{IL} 3.0	-0.3		0.8	V
Input Voltage, high (VDD_SENSOR at 1.8V)	V_{IH} 1.8	1.17		1.98	V
Input Voltage, low (VDD_SENSOR at 1.8V)	V_{IL} 1.8	-0.3		0.63	V
Sensor Voltage, high (VDD_SENSOR at 3.0V)	V_{IH} 3.0	2.7	3.0	3.3	V
Sensor Voltage, low (VDD_SENSOR at 3.0V)	V_{IL} 3.0	1.71	1.8	1.98	V
Output Voltage, high	V_{OH}	IO_VDD* 0.8			V
Output Voltage, low	V_{OL}			0.4	V
Output Current High (VDD=3.0V)	I_{OH_2ma}	2.2	6.1	11.9	mA
	I_{OH_4ma}	5.1	14.4	27.8	mA
	I_{OH_8ma}	7.3	20.5	39.6	mA
Output Current Low (VDD=3.0V)	I_{OL_2ma}	2.8	5	7.8	mA
	I_{OL_4ma}	5.6	9.5	15.5	mA
	I_{OL_8ma}	8.4	15	23.5	mA
Input Capacitance	C_I			4	pF

The following table describes the typical and maximum current consumption of the SCP220x.

Table 94. SCP220x Power Consumption

Description	Supply	Typ. (25°C)	Max. (105°C)	Unit
Core supply voltage + Low power voltage	VDD_CORE + VDD_LP	300	520	mA
I/O Supply	IO_VDD[1]	18	30	mA
Analog PLL supply	VDDA_PLL	9	20	mA
SDRAM memory supply	VDD_SDRAM	22[2]	180[3]	mA
Analog supply for USB	VDD_USB	6 disabled 8 enabled	11 disabled 17 enabled	mA
Analog supply for internal DAC (TVOut)	VDDA_DAC	0.6 disabled 39 enabled	1.0 disabled 55 enabled	mA

Table 94. SCP220x Power Consumption

Notes:
1. IO_VDD is a combined total reading of VDD_GPIO, VDD_DIP, VDD_AUDIO, VDD_NAND, VDD_SENSOR, VDD_SDMMC, VDD_MISCIF, VDD_SCCARD, VDD_OSC measured at 3.0 V, with no IO activity
2. Measured with an application dewarping a VGA image utilizing a 180° field of view lens.
3. Condition where SDRAM is continuously burst written or read.

8 Revision History

Revision	Details of Change	Date
1	Initial Release	04/2014
2	In Section 1.2.7, POWER Supply • “1.0V core...I/O” changed to “1.8V...I/O Power. In Table 2 (SCP220x Power Supply), Added one row for Pin name VDD_SENSOR. In Table 11 (Sensor Interface Timing), Added rows for symbol t6(1.8V) and t7(1.8V). In Table 91 (SCP220x Recommended Operating Range), Added row for item Sensor Interface (1.8V) and also updated footnote 2. In Table 93 (SCP220x DC Characteristics), Added two for item Input Voltage, high (SIF 1.8V) and Input Voltage, low (SIF 1.8V).	03/2015
2.1	• Editorial Changes • Table Caption moved to top for all tables.	06/2015

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address:freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. All other product or service names are the property of their respective owners.

© 2015 Freescale Semiconductor, Inc.

Document Number: SCP220x
Rev.2.1
06/2015