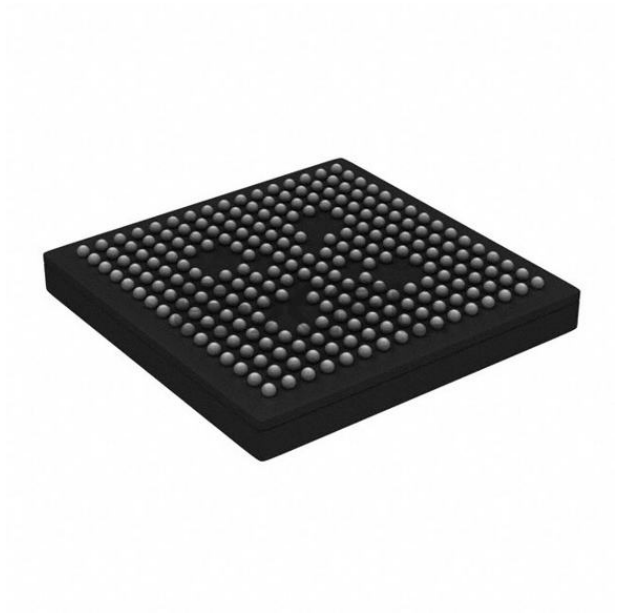

Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	ARM926EJ-S
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	-
Co-Processors/DSP	-
RAM Controllers	DDR, SDRAM
Graphics Acceleration	No
Display & Interface Controllers	LCD
Ethernet	-
SATA	-
USB	USB 2.0 OTG (1)
Voltage - I/O	1.8V, 2.5V, 3.3V
Operating Temperature	-40°C ~ 105°C (TA)
Security Features	Cryptography
Package / Case	236-LFBGA
Supplier Device Package	236-MAPBGA (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/scp2201vmu

1.1 The SCP220x function blocks

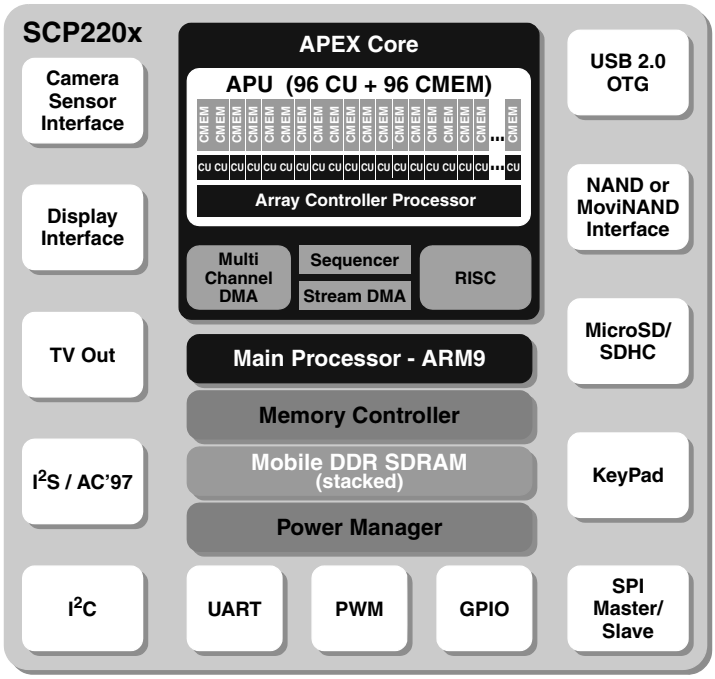

Figure 1. SCP220x Image Cognition Processors

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

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.

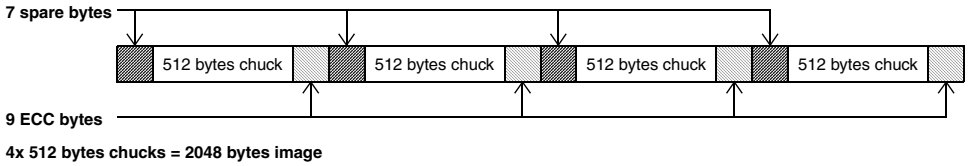

Figure 14. NAND Flash Data Organization

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.

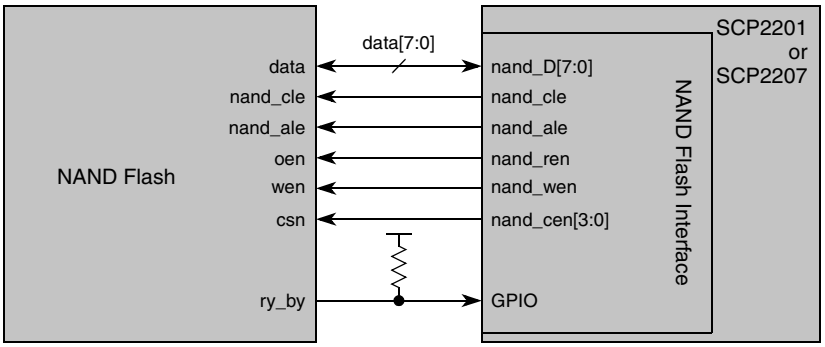

Figure 15. NAND Flash Connectivity

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.

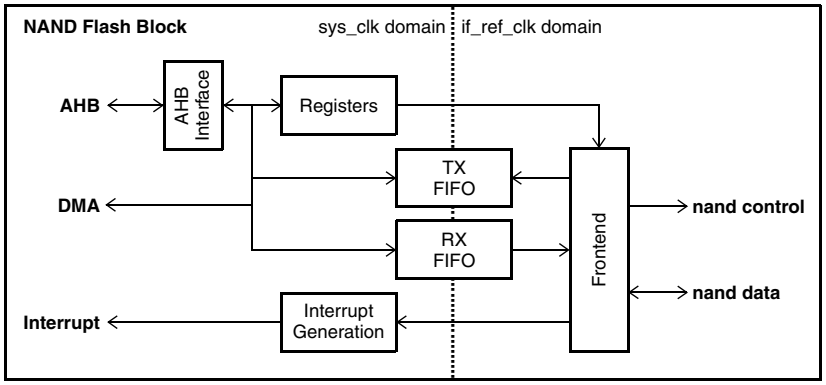

Figure 16. NAND Flash Hardware Architecture

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.

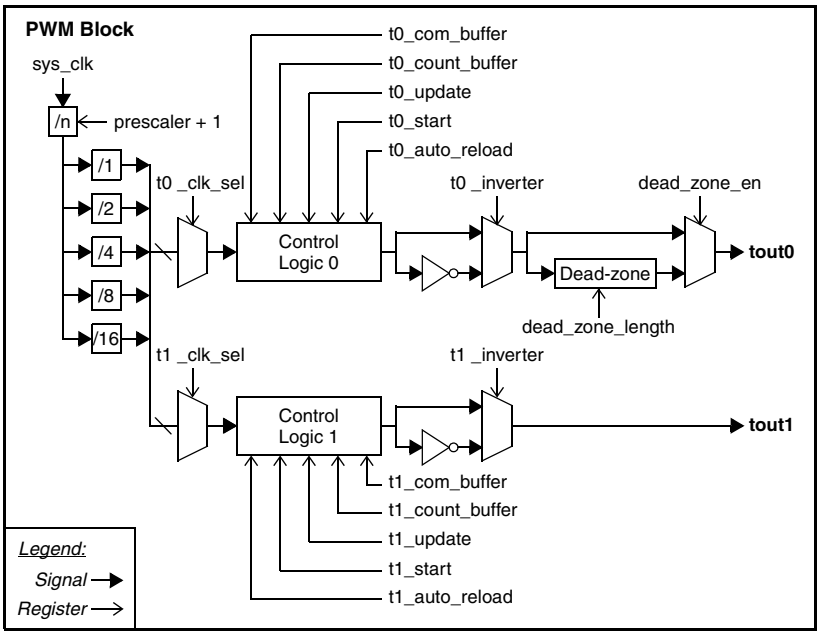

Figure 55. PWM Hardware Architecture

The Tout0 frequency is determined by the following formula:

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.

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 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 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.

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

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**

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

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

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**

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

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

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	

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.