



BCM2835 ARM Peripherals

© 2012 Broadcom Corporation.
All rights reserved

Broadcom Europe Ltd. 406 Science Park Milton Road Cambridge CB4 0WW



Table of Contents

| | | |
|----------|---|------------|
| 1 | Introduction | 4 |
| 1.1 | Overview | 4 |
| 1.2 | Address map | 4 |
| 1.2.1 | Diagrammatic overview | 4 |
| 1.2.2 | ARM virtual addresses (standard Linux kernel only) | 6 |
| 1.2.3 | ARM physical addresses | 6 |
| 1.2.4 | Bus addresses | 6 |
| 1.3 | Peripheral access precautions for correct memory ordering | 7 |
| 2 | Auxiliaries: UART1 & SPI1, SPI2 | 8 |
| 2.1 | Overview | 8 |
| 2.1.1 | AUX registers | 9 |
| 2.2 | Mini UART | 10 |
| 2.2.1 | Mini UART implementation details. | 11 |
| 2.2.2 | Mini UART register details. | 11 |
| 2.3 | Universal SPI Master (2x) | 20 |
| 2.3.1 | SPI implementation details | 20 |
| 2.3.2 | Interrupts | 21 |
| 2.3.3 | Long bit streams | 21 |
| 2.3.4 | SPI register details. | 22 |
| 3 | BSC | 28 |
| 3.1 | Introduction | 28 |
| 3.2 | Register View | 28 |
| 3.3 | 10 Bit Addressing | 36 |
| 4 | DMA Controller | 38 |
| 4.1 | Overview | 38 |
| 4.2 | DMA Controller Registers | 39 |
| 4.2.1 | DMA Channel Register Address Map | 40 |
| 4.3 | AXI Bursts | 63 |
| 4.4 | Error Handling | 63 |
| 4.5 | DMA LITE Engines | 63 |
| 5 | External Mass Media Controller | 65 |
| o | Introduction | 65 |
| o | Registers | 66 |
| 6 | General Purpose I/O (GPIO) | 89 |
| 6.1 | Register View | 90 |
| 6.2 | Alternative Function Assignments | 102 |
| 6.3 | General Purpose GPIO Clocks | 105 |
| 7 | Interrupts | 109 |
| 7.1 | Introduction | 109 |
| 7.2 | Interrupt pending. | 110 |
| 7.3 | Fast Interrupt (FIQ). | 110 |
| 7.4 | Interrupt priority. | 110 |
| 7.5 | Registers | 112 |
| 8 | PCM / I2S Audio | 119 |
| 8.1 | Block Diagram | 120 |
| 8.2 | Typical Timing | 120 |
| 8.3 | Operation | 121 |
| 8.4 | Software Operation | 122 |
| 8.4.1 | Operating in Polled mode | 122 |
| 8.4.2 | Operating in Interrupt mode | 123 |



BCM2835 ARM Peripherals

| | | |
|-----------|---------------------------------|------------|
| 8.4.3 | DMA | 123 |
| 8.5 | Error Handling. | 123 |
| 8.6 | PDM Input Mode Operation | 124 |
| 8.7 | GRAY Code Input Mode Operation | 124 |
| 8.8 | PCM Register Map | 125 |
| 9 | Pulse Width Modulator | 138 |
| 9.1 | Overview | 138 |
| 9.2 | Block Diagram | 138 |
| 9.3 | PWM Implementation | 139 |
| 9.4 | Modes of Operation | 139 |
| 9.5 | Quick Reference | 140 |
| 9.6 | Control and Status Registers | 141 |
| 10 | SPI | 148 |
| 10.1 | Introduction | 148 |
| 10.2 | SPI Master Mode | 148 |
| 10.2.1 | Standard mode | 148 |
| 10.2.2 | Bidirectional mode | 149 |
| 10.3 | LoSSI mode | 150 |
| 10.3.1 | Command write | 150 |
| 10.3.2 | Parameter write | 150 |
| 10.3.3 | Byte read commands | 151 |
| 10.3.4 | 24bit read command | 151 |
| 10.3.5 | 32bit read command | 151 |
| 10.4 | Block Diagram | 152 |
| 10.5 | SPI Register Map | 152 |
| 10.6 | Software Operation | 158 |
| 10.6.1 | Polled | 158 |
| 10.6.2 | Interrupt | 158 |
| 10.6.3 | DMA | 158 |
| 10.6.4 | Notes | 159 |
| 11 | SPI/BSC SLAVE | 160 |
| 11.1 | Introduction | 160 |
| 11.2 | Registers | 160 |
| 12 | System Timer | 172 |
| 12.1 | System Timer Registers | 172 |
| 13 | UART | 175 |
| 13.1 | Variations from the 16C650 UART | 175 |
| 13.2 | Primary UART Inputs and Outputs | 176 |
| 13.3 | UART Interrupts | 176 |
| 13.4 | Register View | 177 |
| 14 | Timer (ARM side) | 196 |
| 14.1 | Introduction | 196 |
| 14.2 | Timer Registers: | 196 |
| 15 | USB | 200 |
| 15.1 | Configuration | 200 |
| 15.2 | Extra / Adapted registers. | 202 |



1 Introduction

1.1 Overview

BCM2835 contains the following peripherals which may safely be accessed by the ARM:

- Timers
- Interrupt controller
- GPIO
- USB
- PCM / I2S
- DMA controller
- I2C master
- I2C / SPI slave
- SPI0, SPI1, SPI2
- PWM
- UART0, UART1

The purpose of this datasheet is to provide documentation for these peripherals in sufficient detail to allow a developer to port an operating system to BCM2835.

There are a number of peripherals which are intended to be controlled by the GPU. These are omitted from this datasheet. Accessing these peripherals from the ARM is not recommended.

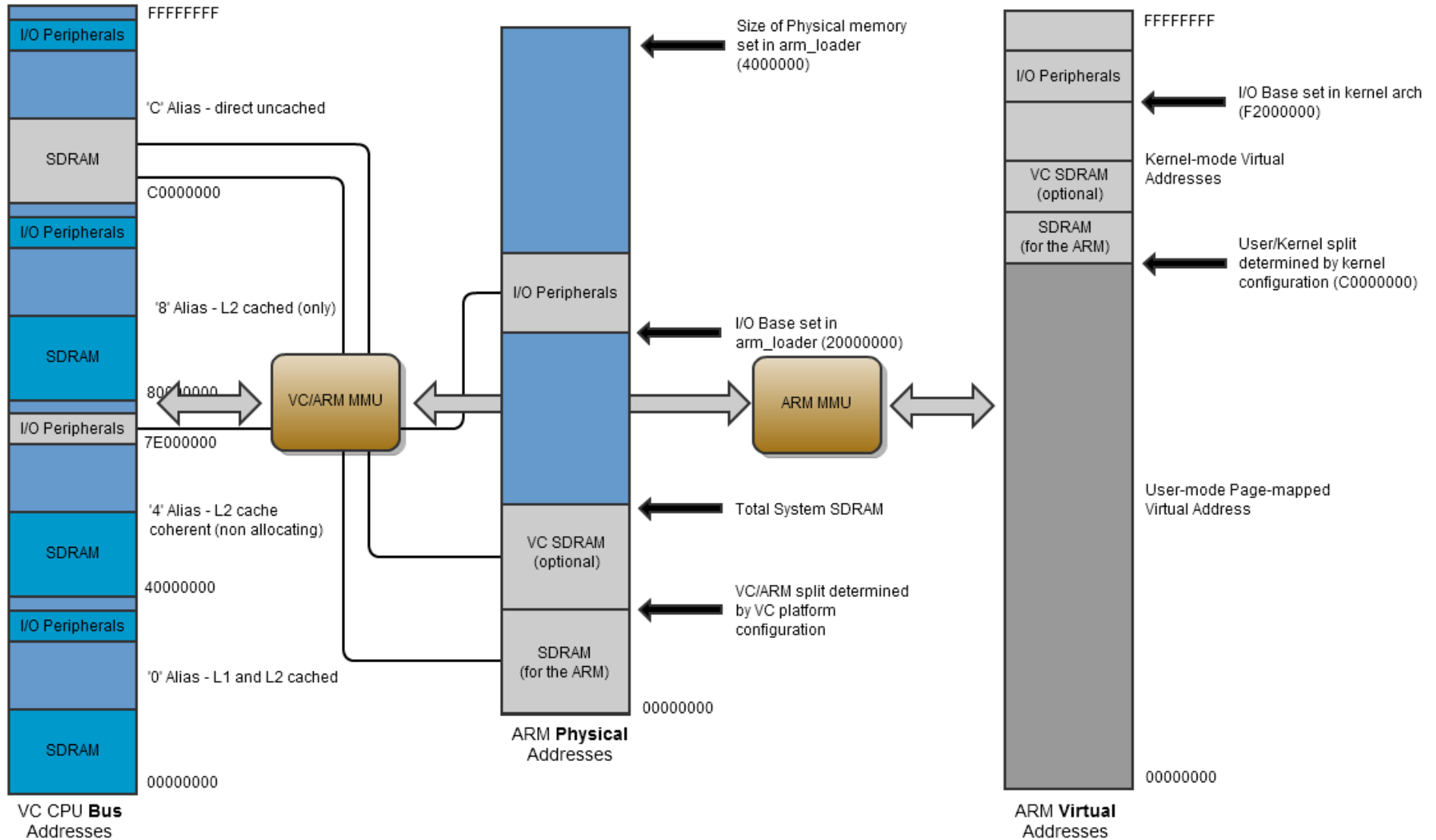
1.2 Address map

1.2.1 Diagrammatic overview

In addition to the ARM's MMU, BCM2835 includes a second coarse-grained MMU for mapping ARM physical addresses onto system bus addresses. This diagram shows the main address spaces of interest:



BCM2835 ARM Peripherals





Addresses in ARM Linux are:

- issued as virtual addresses by the ARM core, then
- mapped into a physical address by the ARM MMU, then
- mapped into a bus address by the ARM mapping MMU, and finally
- used to select the appropriate peripheral or location in RAM.

1.2.2 ARM virtual addresses (standard Linux kernel only)

As is standard practice, the standard BCM2835 Linux kernel provides a contiguous mapping over the whole of available RAM at the top of memory. The kernel is configured for a 1GB/3GB split between kernel and user-space memory.

The split between ARM and GPU memory is selected by installing one of the supplied `start*.elf` files as `start.elf` in the FAT32 boot partition of the SD card. The minimum amount of memory which can be given to the GPU is 32MB, but that will restrict the multimedia performance; for example, 32MB does not provide enough buffering for the GPU to do 1080p30 video decoding.

Virtual addresses in kernel mode will range between `0xC0000000` and `0xEFFFFFFF`.

Virtual addresses in user mode (i.e. seen by processes running in ARM Linux) will range between `0x00000000` and `0xBFFFFFFF`.

Peripherals (at physical address `0x20000000` on) are mapped into the kernel virtual address space starting at address `0xF2000000`. Thus a peripheral advertised here at bus address `0x7Ennnnnn` is available in the ARM kernel at virtual address `0xF2nnnnnn`.

1.2.3 ARM physical addresses

Physical addresses start at `0x00000000` for RAM.

- The ARM section of the RAM starts at `0x00000000`.
- The VideoCore section of the RAM is mapped in only if the system is configured to support a memory mapped display (this is the common case).

The VideoCore MMU maps the ARM physical address space to the bus address space seen by VideoCore (and VideoCore peripherals). The bus addresses for RAM are set up to map onto the uncached¹ bus address range on the VideoCore starting at `0xC0000000`.

Physical addresses range from `0x20000000` to `0x20FFFFFF` for peripherals. The bus addresses for peripherals are set up to map onto the peripheral bus address range starting at `0x7E000000`. Thus a peripheral advertised here at bus address `0x7Ennnnnn` is available at physical address `0x20nnnnnn`.

1.2.4 Bus addresses

The peripheral addresses specified in this document are bus addresses. Software directly accessing peripherals must translate these addresses into physical or virtual addresses, as described above. Software accessing peripherals using the DMA engines must use bus addresses.

¹ BCM2835 provides a 128KB system L2 cache, which is used primarily by the GPU. Accesses to memory are routed either via or around the L2 cache depending on senior two bits of the bus address.



Software accessing RAM directly must use physical addresses (based at 0x00000000). Software accessing RAM using the DMA engines must use bus addresses (based at 0xC0000000).

1.3 Peripheral access precautions for correct memory ordering

The BCM2835 system uses an AMBA AXI-compatible interface structure. In order to keep the system complexity low and data throughput high, the BCM2835 AXI system does not always return read data in-order². The GPU has special logic to cope with data arriving out-of-order; however the ARM core does not contain such logic. Therefore some precautions must be taken when using the ARM to access peripherals.

Accesses to the same peripheral will always arrive and return in-order. It is only when switching from one peripheral to another that data can arrive out-of-order. The simplest way to make sure that data is processed in-order is to place a memory barrier instruction at critical positions in the code. You should place:

- A memory write barrier before the first write to a peripheral.
- A memory read barrier after the last read of a peripheral.

It is **not** required to put a memory barrier instruction after **each** read or write access. Only at those places in the code where it is possible that a peripheral read or write may be followed by a read or write of a **different** peripheral. This is normally at the entry and exit points of the peripheral service code.

As interrupts can appear anywhere in the code so you should safeguard those. If an interrupt routine reads from a peripheral the routine should start with a memory read barrier. If an interrupt routine writes to a peripheral the routine should end with a memory write barrier.

²Normally a processor assumes that if it executes two read operations the data will arrive in order. So a read from location X followed by a read from location Y should return the data of location X first, followed by the data of location Y. Data arriving out of order can have disastrous consequences. For example:

```
a_status = *pointer_to_peripheral_a;  
b_status = *pointer_to_peripheral_b;
```

Without precautions the values ending up in the variables a_status and b_status can be swapped around.

It is theoretical possible for writes to go 'wrong' but that is far more difficult to achieve. The AXI system makes sure the data always arrives in-order at its intended destination. So:

```
*pointer_to_peripheral_a = value_a;  
*pointer_to_peripheral_b = value_b;
```

will always give the expected result. The only time write data can arrive out-of-order is if two different peripherals are connected to the same external equipment.



2 Auxiliaries: UART1 & SPI1, SPI2

2.1 Overview

The Device has three Auxiliary peripherals: One mini UART and two SPI masters. These three peripheral are grouped together as they share the same area in the peripheral register map and they share a common interrupt. Also all three are controlled by the auxiliary enable register.

| Auxiliary peripherals Register Map (offset = 0x7E21 5000) | | | |
|--|----------------------------|------------------------------|------|
| Address | Register Name ³ | Description | Size |
| 0x7E21 5000 | AUX_IRQ | Auxiliary Interrupt status | 3 |
| 0x7E21 5004 | AUX_ENABLES | Auxiliary enables | 3 |
| 0x7E21 5040 | AUX_MU_IO_REG | Mini Uart I/O Data | 8 |
| 0x7E21 5044 | AUX_MU_IER_REG | Mini Uart Interrupt Enable | 8 |
| 0x7E21 5048 | AUX_MU_IIR_REG | Mini Uart Interrupt Identify | 8 |
| 0x7E21 504C | AUX_MU_LCR_REG | Mini Uart Line Control | 8 |
| 0x7E21 5050 | AUX_MU_MCR_REG | Mini Uart Modem Control | 8 |
| 0x7E21 5054 | AUX_MU_LSR_REG | Mini Uart Line Status | 8 |
| 0x7E21 5058 | AUX_MU_MSR_REG | Mini Uart Modem Status | 8 |
| 0x7E21 505C | AUX_MU_SCRATCH | Mini Uart Scratch | 8 |
| 0x7E21 5060 | AUX_MU_CNTL_REG | Mini Uart Extra Control | 8 |
| 0x7E21 5064 | AUX_MU_STAT_REG | Mini Uart Extra Status | 32 |
| 0x7E21 5068 | AUX_MU_BAUD_REG | Mini Uart Baudrate | 16 |
| 0x7E21 5080 | AUX_SPI0_CNTL0_REG | SPI 1 Control register 0 | 32 |
| 0x7E21 5084 | AUX_SPI0_CNTL1_REG | SPI 1 Control register 1 | 8 |
| 0x7E21 5088 | AUX_SPI0_STAT_REG | SPI 1 Status | 32 |
| 0x7E21 5090 | AUX_SPI0_IO_REG | SPI 1 Data | 32 |
| 0x7E21 5094 | AUX_SPI0_PEEK_REG | SPI 1 Peek | 16 |
| 0x7E21 50C0 | AUX_SPI1_CNTL0_REG | SPI 2 Control register 0 | 32 |
| 0x7E21 50C4 | AUX_SPI1_CNTL1_REG | SPI 2 Control register 1 | 8 |

³ These register names are identical to the defines in the AUX_IO header file. For programming purposes these names should be used wherever possible.



BCM2835 ARM Peripherals

| | | | |
|-------------|-------------------|--------------|----|
| 0x7E21 50C8 | AUX_SPI1_STAT_REG | SPI 2 Status | 32 |
| 0x7E21 50D0 | AUX_SPI1_IO_REG | SPI 2 Data | 32 |
| 0x7E21 50D4 | AUX_SPI1_PEEK_REG | SPI 2 Peek | 16 |

2.1.1 AUX registers

There are two Auxiliary registers which control all three devices. One is the interrupt status register, the second is the Auxiliary enable register. The Auxiliary IRQ status register can help to hierarchically determine the source of an interrupt.

AUXIRQ Register (0x7E21 5000)

SYNOPSIS The AUXIRQ register is used to check any pending interrupts which may be asserted by the three Auxiliary sub blocks.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|---------------|---|------|-------|
| 31:3 | | Reserved, write zero, read as don't care | | |
| 2 | SPI 2 IRQ | If set the SPI 2 module has an interrupt pending. | R | 0 |
| 1 | SPI 1 IRQ | If set the SPI1 module has an interrupt pending. | R | 0 |
| 0 | Mini UART IRQ | If set the mini UART has an interrupt pending. | R | 0 |

AUXENB Register (0x7E21 5004)

SYNOPSIS The AUXENB register is used to enable the three modules; UART, SPI1, SPI2.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------------|--|------|-------|
| 31:3 | | Reserved, write zero, read as don't care | | |
| 2 | SPI2 enable | If set the SPI 2 module is enabled. If clear the SPI 2 module is disabled. That also disables any SPI 2 module register access | R/W | 0 |
| 1 | SPI 1 enable | If set the SPI 1 module is enabled. If clear the SPI 1 module is disabled. That also disables any SPI 1 module register access | R/W | 0 |
| 0 | Mini UART enable | If set the mini UART is enabled. The UART will immediately start receiving data, especially if the UART1_RX line is <i>low</i> . If clear the mini UART is disabled. That also disables any mini UART register access | R/W | 0 |



If the enable bits are clear you will have no access to a peripheral. You can not even read or write the registers!

GPIO pins should be set up first the before enabling the UART. The UART core is build to emulate 16550 behaviour. So when it is enabled any data at the inputs will immediately be received . If the UART1_RX line is low (because the GPIO pins have not been set-up yet) that will be seen as a start bit and the UART will start receiving 0x00-characters.

Valid stops bits are not required for the UART. (See also Implementation details). Hence any bit status is acceptable as stop bit and is only used so there is clean timing start for the next bit.

Looking after a reset: the baudrate will be zero and the system clock will be 250 MHz. So only 2.5 µseconds suffice to fill the receive FIFO. The result will be that the FIFO is full and overflowing in no time flat.

2.2 Mini UART

The mini UART is a secondary low throughput⁴ UART intended to be used as a console. It needs to be enabled before it can be used. It is also recommended that the correct GPIO function mode is selected before enabling the mini Uart.

The mini Uart has the following features:

- 7 or 8 bit operation.
- 1 start and 1 stop bit.
- No parities.
- Break generation.
- 8 symbols deep FIFOs for receive and transmit.
- SW controlled RTS, SW readable CTS.
- Auto flow control with programmable FIFO level.
- 16550 *like* registers.
- Baudrate derived from system clock.

This is a mini UART and it does NOT have the following capabilities:

- Break detection
- Framing errors detection.
- Parity bit
- Receive Time-out interrupt
- DCD, DSR, DTR or RI signals.

The implemented UART is not a 16650 compatible UART However as far as possible the first 8 control and status registers are laid out like a 16550 UART. Al 16550 register bits which are not supported can be written but will be ignored and read back as 0. All control bits for simple UART receive/transmit operations are available.

⁴ The UART itself has no throughput limitations in fact it can run up to 32 Mega baud. But doing so requires significant CPU involvement as it has shallow FIFOs and no DMA support.



2.2.1 Mini UART implementation details.

The UART1_CTS and UART1_RX inputs are synchronised and will take 2 system clock cycles before they are processed.

The module does not check for any framing errors. After receiving a start bit and 8 (or 7) data bits the receiver waits for one half bit time and then starts scanning for the next start bit. The mini UART does *not* check if the stop bit is high or wait for the stop bit to appear. As a result of this a UART1_RX input line which is continuously low (a break condition or an error in connection or GPIO setup) causes the receiver to continuously receive 0x00 symbols.

The mini UART uses 8-times oversampling. The Baudrate can be calculated from:

$$baudrate = \frac{system_clock_freq}{8 * (baudrate_reg + 1)}$$

If the system clock is 250 MHz and the baud register is zero the baudrate is 31.25 Mega baud. (25 Mbits/sec or 3.125 Mbytes/sec). The lowest baudrate with a 250 MHz system clock is 476 Baud.

When writing to the data register only the LS 8 bits are taken. All other bits are ignored. When reading from the data register only the LS 8 bits are valid. All other bits are zero.

2.2.2 Mini UART register details.

AUX_MU_IO_REG Register (0x7E21 5040)

SYNOPSIS The AUX_MU_IO_REG register is primary used to write data to and read data from the UART FIFOs.
If the DLAB bit in the line control register is set this register gives access to the LS 8 bits of the baud rate. (Note: there is easier access to the baud rate register)

| Bit(s) | Field Name | Description | Type | Reset |
|--------|---------------------------------------|---|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7:0 | LS 8 bits Baudrate read/write, DLAB=1 | Access to the LS 8 bits of the 16-bit baudrate register. (Only If bit 7 of the line control register (DLAB bit) is set) | R/W | 0 |
| 7:0 | Transmit data write, DLAB=0 | Data written is put in the transmit FIFO (Provided it is not full) (Only If bit 7 of the line control register (DLAB bit) is clear) | W | 0 |
| 7:0 | Receive data read, DLAB=0 | Data read is taken from the receive FIFO (Provided it is not empty) (Only If bit 7 of the line control register (DLAB bit) is clear) | R | 0 |



BCM2835 ARM Peripherals

AUX_MU_IIR_REG Register (0x7E21 5044)

SYNOPSIS The AUX_MU_IER_REG register is primary used to enable interrupts
If the DLAB bit in the line control register is set this register gives access to the MS 8 bits of the baud rate. (Note: there is easier access to the baud rate register)

| Bit(s) | Field Name | Description | Type | Reset |
|--------|--|---|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7:0 | MS 8 bits Baudrate read/write, DLAB=1 | Access to the MS 8 bits of the 16-bit baudrate register. (Only If bit 7 of the line control register (DLAB bit) is set) | R/w | 0 |
| 7:2 | | Reserved, write zero, read as don't care <i>Some of these bits have functions in a 16550 compatible UART but are ignored here</i> | | |
| 1 | Enable receive interrupt (DLAB=0) | If this bit is set the interrupt line is asserted whenever the receive FIFO holds at least 1 byte. If this bit is clear no receive interrupts are generated. | R | 0 |
| 0 | Enable transmit interrupt (DLAB=0) | If this bit is set the interrupt line is asserted whenever the transmit FIFO is empty. If this bit is clear no transmit interrupts are generated. | R | 0 |



BCM2835 ARM Peripherals

AUX_MU_IER_REG Register (0x7E21 5048)

SYNOPSIS The AUX_MU_IIR_REG register shows the interrupt status.
It also has two FIFO enable status bits and (when writing) FIFO clear bits.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|---|---|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7:6 | FIFO enables | Both bits always read as 1 as the FIFOs are always enabled | R | 11 |
| 5:4 | - | Always read as zero | R | 00 |
| 3 | - | Always read as zero as the mini UART has no timeout function | R | 0 |
| 2:1 | READ: Interrupt ID bits WRITE: FIFO clear bits | On read this register shows the interrupt ID bit 00 : No interrupts 01 : Transmit holding register empty 10 : Receiver holds valid byte 11 : <Not possible> On write: Writing with bit 1 set will clear the receive FIFO Writing with bit 2 set will clear the transmit FIFO | R/W | 00 |
| 0 | Interrupt pending | This bit is clear whenever an interrupt is pending | R | 1 |



BCM2835 ARM Peripherals

AUX_MU_LCR_REG Register (0x7E21 504C)

SYNOPSIS The AUX_MU_LCR_REG register controls the line data format and gives access to the baudrate register

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-------------|--|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7 | DLAB access | If set the first to Mini UART register give access the the Baudrate register. During operation this bit must be cleared. | R/W | 0 |
| 6 | Break | If set high the UART1_TX line is pulled low continuously. If held for at least 12 bits times that will indicate a break condition. | R/W | 0 |
| 5:1 | | Reserved, write zero, read as don't care <i>Some of these bits have functions in a 16550 compatible UART but are ignored here</i> | | 0 |
| 0 | data size | If clear the UART works in 7-bit mode If set the UART works in 8-bit mode | R/W | 0 |

AUX_MU_MCR_REG Register (0x7E21 5050)

SYNOPSIS The AUX_MU_MCR_REG register controls the 'modem' signals.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7:2 | | Reserved, write zero, read as don't care <i>Some of these bits have functions in a 16550 compatible UART but are ignored here</i> | | 0 |
| 1 | RTS | If clear the UART1_RTS line is high If set the UART1_RTS line is low This bit is ignored if the RTS is used for auto-flow control. See the Mini Uart Extra Control register description) | R/W | 0 |
| 0 | | Reserved, write zero, read as don't care <i>This bit has a function in a 16550 compatible UART but is ignored here</i> | | 0 |



BCM2835 ARM Peripherals

AUX_MU_LSR_REG Register (0x7E21 5054)

SYNOPSIS The AUX_MU_LSR_REG register shows the data status.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-------------------|---|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7 | | Reserved, write zero, read as don't care <i>This bit has a function in a 16550 compatible UART but is ignored here</i> | | 0 |
| 6 | Transmitter idle | This bit is set if the transmit FIFO is empty and the transmitter is idle. (Finished shifting out the last bit). | R | 1 |
| 5 | Transmitter empty | This bit is set if the transmit FIFO can accept at least one byte. | R | 0 |
| 4:2 | | Reserved, write zero, read as don't care <i>Some of these bits have functions in a 16550 compatible UART but are ignored here</i> | | 0 |
| 1 | Receiver Overrun | This bit is set if there was a receiver overrun. That is: one or more characters arrived whilst the receive FIFO was full. The newly arrived characters have been discarded. This bit is cleared each time this register is read. To do a non-destructive read of this overrun bit use the Mini Uart Extra Status register. | R/C | 0 |
| 0 | Data ready | This bit is set if the receive FIFO holds at least 1 symbol. | R | 0 |

AUX_MU_MSR_REG Register (0x7E21 5058)

SYNOPSIS The AUX_MU_MSR_REG register shows the 'modem' status.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7:6 | | Reserved, write zero, read as don't care <i>Some of these bits have functions in a 16550 compatible UART but are ignored here</i> | | 0 |
| 5 | CTS status | This bit is the inverse of the UART1_CTS input Thus : If set the UART1_CTS pin is low If clear the UART1_CTS pin is high | R | 1 |
| 3:0 | | Reserved, write zero, read as don't care <i>Some of these bits have functions in a 16550 compatible UART but are ignored here</i> | | 0 |



AUX_MU_SCRATCH Register (0x7E21 505C)

SYNOPSIS The AUX_MU_SCRATCH is a single byte storage.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7:0 | Scratch | One whole byte extra on top of the 134217728 provided by the SDC | R/W | 0 |

AUX_MU_CNTL_REG Register (0x7E21 5060)

SYNOPSIS The AUX_MU_CNTL_REG provides access to some extra useful and nice features not found on a normal 16550 UART .

| Bit(s) | Field Name | Description | Type | Reset |
|--------|---|--|------|-------|
| 31:8 | | Reserved, write zero, read as don't care | | |
| 7 | CTS assert level | This bit allows one to invert the CTS auto flow operation polarity. If set the CTS auto flow assert level is low* If clear the CTS auto flow assert level is high* | R/W | 0 |
| 6 | RTS assert level | This bit allows one to invert the RTS auto flow operation polarity. If set the RTS auto flow assert level is low* If clear the RTS auto flow assert level is high* | R/W | 0 |
| 5:4 | RTS AUTO flow level | These two bits specify at what receiver FIFO level the RTS line is de-asserted in auto-flow mode. 00 : De-assert RTS when the receive FIFO has 3 empty spaces left. 01 : De-assert RTS when the receive FIFO has 2 empty spaces left. 10 : De-assert RTS when the receive FIFO has 1 empty space left. 11 : De-assert RTS when the receive FIFO has 4 empty spaces left. | R/W | 0 |
| 3 | Enable transmit Auto flow-control using CTS | If this bit is set the transmitter will stop if the CTS line is de-asserted. If this bit is clear the transmitter will ignore the status of the CTS line | R/W | 0 |



| | | | | |
|---|--|--|-----|---|
| 2 | Enable receive Auto flow-control using RTS | If this bit is set the RTS line will de-assert if the receive FIFO reaches it 'auto flow' level. In fact the RTS line will behave as an RTR (<i>Ready To Receive</i>) line. If this bit is clear the RTS line is controlled by the AUX_MU_MCR_REG register bit 1. | R/W | 0 |
| 1 | Transmitter enable | If this bit is set the mini UART transmitter is enabled. If this bit is clear the mini UART transmitter is disabled | R/W | 1 |
| 0 | Receiver enable | If this bit is set the mini UART receiver is enabled. If this bit is clear the mini UART receiver is disabled | R/W | 1 |

Receiver enable

If this bit is set no new symbols will be accepted by the receiver. Any symbols in progress of reception will be finished.

Transmitter enable

If this bit is set no new symbols will be send the transmitter. Any symbols in progress of transmission will be finished.

Auto flow control

Automatic flow control can be enabled independent for the receiver and the transmitter.

CTS auto flow control impacts the transmitter only. The transmitter will not send out new symbols when the CTS line is de-asserted. Any symbols in progress of transmission when the CTS line becomes de-asserted will be finished.

RTS auto flow control impacts the receiver only. In fact the name RTS for the control line is incorrect and should be RTR (Ready to Receive). The receiver will de-asserted the RTS (RTR) line when its receive FIFO has a number of empty spaces left. Normally 3 empty spaces should be enough.

If looping back a mini UART using full auto flow control the logic is fast enough to allow the RTS auto flow level of '10' (De-assert RTS when the receive FIFO has 1 empty space left).

Auto flow polarity

To offer full flexibility the polarity of the CTS and RTS (RTR) lines can be programmed. This should allow the mini UART to interface with any existing hardware flow control available.



BCM2835 ARM Peripherals

AUX_MU_STAT_REG Register (0x7E21 5064)

SYNOPSIS The AUX_MU_STAT_REG provides a lot of useful information about the internal status of the mini UART not found on a normal 16550 UART.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|--------------------------|---|------|-------|
| 31:28 | | Reserved, write zero, read as don't care | | |
| 27:24 | Transmit FIFO fill level | These bits shows how many symbols are stored in the transmit FIFO The value is in the range 0-8 | R | 0 |
| 23:20 | | Reserved, write zero, read as don't care | | |
| 19:16 | Receive FIFO fill level | These bits shows how many symbols are stored in the receive FIFO The value is in the range 0-8 | R | 0 |
| 15:10 | | Reserved, write zero, read as don't care | | |
| 9 | Transmitter done | This bit is set if the transmitter is idle and the transmit FIFO is empty. It is a logic AND of bits 2 and 8 | R | 1 |
| 8 | Transmit FIFO is empty | If this bit is set the transmitter FIFO is empty. Thus it can accept 8 symbols. | R | 1 |
| 7 | CTS line | This bit shows the status of the UART1_CTS line. | R | 0 |
| 6 | RTS status | This bit shows the status of the UART1_RTS line. | R | 0 |
| 5 | Transmit FIFO is full | This is the inverse of bit 1 | R | 0 |
| 4 | Receiver overrun | This bit is set if there was a receiver overrun. That is: one or more characters arrived whilst the receive FIFO was full. The newly arrived characters have been discarded. This bit is cleared each time the AUX_MU_LSR_REG register is read. | R | 0 |
| 3 | Transmitter is idle | If this bit is set the transmitter is idle. If this bit is clear the transmitter is idle. | R | 1 |
| 2 | Receiver is idle | If this bit is set the receiver is idle. If this bit is clear the receiver is busy. This bit can change unless the receiver is disabled | R | 1 |
| 1 | Space available | If this bit is set the mini UART transmitter FIFO can accept at least one more symbol. If this bit is clear the mini UART transmitter FIFO is full | R | 0 |



BCM2835 ARM Peripherals

| | | | | |
|---|------------------|---|---|---|
| 0 | Symbol available | If this bit is set the mini UART receive FIFO contains at least 1 symbol If this bit is clear the mini UART receiver FIFO is empty | R | 0 |
|---|------------------|---|---|---|

Receiver is idle

This bit is only useful if the receiver is disabled. The normal use is to disable the receiver. Then check (or wait) until the bit is set. Now you can be sure that no new symbols will arrive. (e.g. now you can change the baudrate...)

Transmitter is idle

This bit tells if the transmitter is idle. Note that the bit will set only for a short time if the transmit FIFO contains data. Normally you want to use bit 9: Transmitter done.

RTS status

This bit is useful only in receive Auto flow-control mode as it shows the status of the RTS line.

AUX_MU_BAUD Register (0x7E21 5068)

SYNOPSIS The AUX_MU_BAUD register allows direct access to the 16-bit wide baudrate counter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | Reserved, write zero, read as don't care | | |
| 15:0 | Baudrate | mini UART baudrate counter | R/W | 0 |

This is the same register as is accessed using the LABD bit and the first two register, but much easier to access.

2.3 Universal SPI Master (2x)

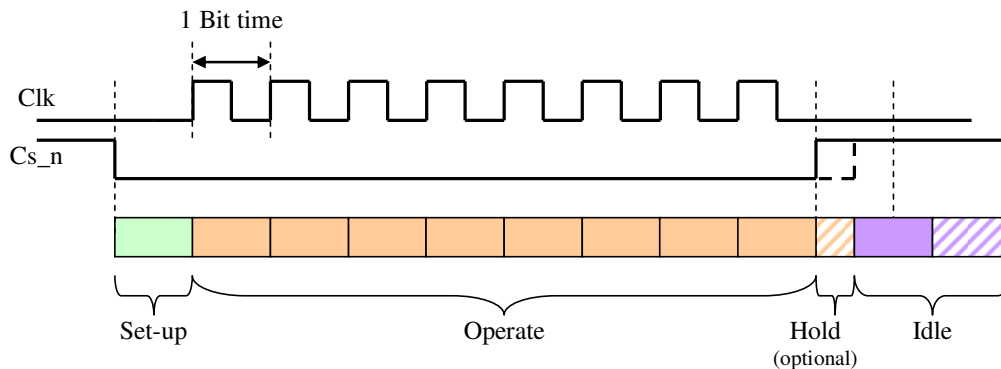
The two universal SPI masters are secondary low throughput⁵ SPI interfaces. Like the UART the devices need to be enabled before they can be used. Each SPI master has the following features:

- Single beat bit length between 1 and 32 bits.
- Single beat variable bit length between 1 and 24 bits
- Multi beat infinite bit length.
- 3 independent chip selects per master.
- 4 entries 32-bit wide transmit and receive FIFOs.
- Data out on rising or falling clock edge.
- Data in on rising or falling clock edge.
- Clock inversion (Idle high or idle low).
- Wide clocking range.
- Programmable data out hold time.
- Shift in/out MS or LS bit first

A major issue with an SPI interface is that there is no SPI standard in any form. Because the SPI interface has been around for a long time some pseudo-standard rules have appeared mostly when interfacing with memory devices. The universal SPI master has been developed to work even with the most 'non-standard' SPI devices.

2.3.1 SPI implementation details

The following diagram shows a typical SPI access cycle. In this case we have 8 SPI clocks.



One bit time before any clock edge changes the CS_n will go low. This makes sure that the MOSI signal has a full bit-time of set-up against any changing clock edges.

The operation normally ends after the last clock cycle. Note that at the end there is one half-bit time where the clock does not change but which still is part of the operation cycle.

There is an option to add a half bit cycle hold time. This makes sure that any MISO data has at least a full SPI bit time to arrive. (Without this hold time, data clocked out of the SPI device on the last clock edge would have only half a bit time to arrive).

⁵ Again the SPIs themselves have no throughput limitations in fact they can run with an SPI clock of 125 MHz. But doing so requires significant CPU involvement as they have shallow FIFOs and no DMA support.



Last there is a guarantee of at least a full bit time where the spi chip select is high. A longer CS_n high period can be programmed for another 1-7 cycles.

The SPI clock frequency is:

$$SPIx_CLK = \frac{system_clock_freq}{2 * (speed_field + 1)}$$

If the system clock is 250 MHz and the speed field is zero the SPI clock frequency is 125 MHz. The practical SPI clock will be lower as the I/O pads can not transmit or receive signals at such high speed. The lowest SPI clock frequency with a 250 MHz system clock is 30.5 KHz.

The hardware has an option to add hold time to the MOSI signal against the SPI clk. This is again done using the system clock. So a 250 MHz system clock will add hold times in units of 4 ns. Hold times of 0, 1, 4 and 7 system clock cycles can be used. (So at 250MHz an additional hold time of 0, 4, 16 and 28 ns can be achieved). The hold time is *additional* to the normal output timing as specified in the data sheet.

2.3.2 Interrupts

The SPI block has two interrupts: TX FIFO is empty, SPI is Idle.

TX FIFO is empty:

This interrupt will be asserted as soon as the last entry has been read from the transmit FIFO. At that time the interface will still be busy shifting out that data. This also implies that the receive FIFO will not yet contain the last received data. It is possible at that time to fill the TX FIFO again and read the receive FIFO entries which have been received. Note that there is no "receive FIFO full" interrupt as the number of entries received is always equal to the number of entries transmitted.

SPI is IDLE:

This interrupt will be asserted when the transmit FIFO is empty and the SPI block has finished all actions (including the CS-high time) By this time the receive FIFO will have all received data as well.

2.3.3 Long bit streams

The SPI module works in bursts of maximum 32 bits. Some SPI devices require data which is longer the 32 bits. To do this the user must make use of the two different data TX addresses: Tx data written to one address cause the CS to remain asserted. Tx data written to the other address cause the CS to be de-asserted at the end of the transmit cycle. So in order to exchange 96 bits you do the following:

Write the first two data words to one address, then write the third word to the other address.



2.3.4 SPI register details.

AUXSPI0/1_CNTL0 Register (0x7E21 5080,0x7E21 50C0)

SYNOPSIS The AUXSPIx_CNTL0 register control many features of the SPI interfaces.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------|---|------|-------|
| 31:20 | Speed | Sets the SPI clock speed. spi clk freq = system_clock_freq/2*(speed+1) | R/W | 0 |
| 19:17 | chip selects | The pattern output on the CS pins when active. | R/W | 111 |
| 16 | post-input mode | If set the SPI input works in post input mode. For details see text further down | R/W | 0 |
| 15 | Variable CS | If 1 the SPI takes the CS pattern and the data from the TX fifo If 0 the SPI takes the CS pattern from bits 17-19 of this register Set this bit only if also bit 14 (variable width) is set | R/W | 0 |
| 14 | Variable width | If 1 the SPI takes the shift length and the data from the TX fifo If 0 the SPI takes the shift length from bits 0-5 of this register | R/W | 0 |
| 13:12 | DOUT Hold time | Controls the extra DOUT hold time in system clock cycles. 00 : No extra hold time 01 : 1 system clock extra hold time 10 : 4 system clocks extra hold time 11 : 7 system clocks extra hold time | R/W | 0 |
| 11 | Enable | Enables the SPI interface. Whilst disabled the FIFOs can still be written to or read from This bit should be 1 during normal operation. | R/W | 0 |
| 10 | In rising | If 1 data is clocked in on the rising edge of the SPI clock If 0 data is clocked in on the falling edge of the SPI clock | R/W | 0 |
| 9 | Clear FIFOs | If 1 the receive and transmit FIFOs are held in reset (and thus flushed.) This bit should be 0 during normal operation. | R/W | 0 |

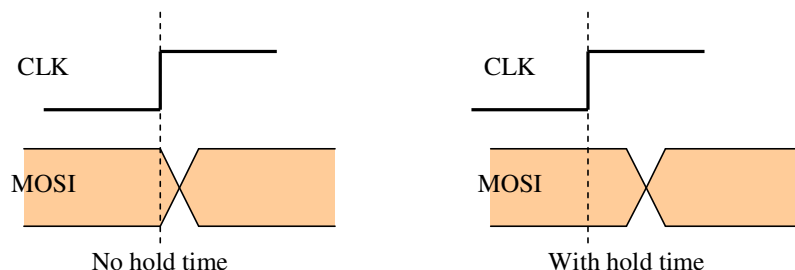
| | | | | |
|-----|------------------------|---|-----|---|
| 8 | Out rising | If 1 data is clocked out on the rising edge of the SPI clock If 0 data is clocked out on the falling edge of the SPI clock | R/W | 0 |
| 7 | Invert SPI CLK | If 1 the 'idle' clock line state is high. If 0 the 'idle' clock line state is low. | R/W | 0 |
| 6 | Shift out MS bit first | If 1 the data is shifted out starting with the MS bit. (bit 15 or bit 11) If 0 the data is shifted out starting with the LS bit. (bit 0) | R/W | 0 |
| 5:0 | Shift length | Specifies the number of bits to shift This field is ignored when using 'variable shift' mode | R/W | 0 |

Invert SPI CLK

Changing this bit will immediately change the polarity of the SPI clock output. It is recommended not to do this when also the CS is active as the connected devices will see this as a clock change.

DOUT hold time

Because the interface runs on fast silicon the MOSI hold time against the clock will be very short. This can cause considerable problems on SPI slaves. To make it easier for the slave to see the data the hold time of the MOSI out against the SPI clock out is programmable.



Variable width

In this mode the shift length is taken from the transmit FIFO. The transmit data bits 28:24 are used as shift length and the data bits 23:0 are the actual transmit data. If the option 'shift MS out first' is selected the first bit shifted out will be bit 23. The receive data will arrive as normal.

Variable CS

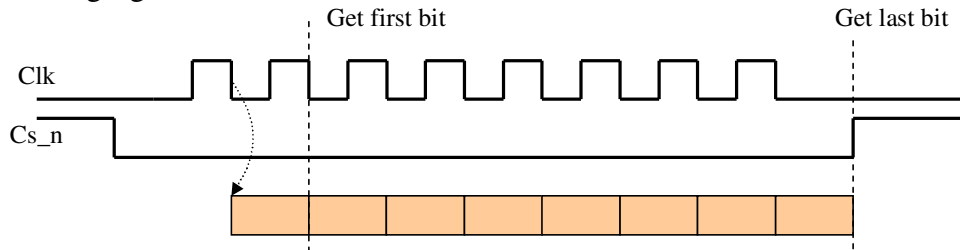
This mode is used together with the variable width mode. In this mode the CS pattern is taken from the transmit FIFO. The transmit data bits 31:29 are used as CS and the data bits 23:0 are the actual transmit data. This allows the CPU to write to different SPI devices without having to change the CS bits. However the data length is limited to 24 bits.

Post-input mode

Some rare SPI devices output data on the falling clock edge which then has to be picked up on the next falling clock edge. There are two problems with this:

1. The very first falling clock edge there is no valid data arriving.
2. After the last clock edge there is one more 'dangling' bit to pick up.

The post-input mode is specifically to deal with this sort of data. If the post-input mode bit is set, the data arriving at the first falling clock edge is ignored. Then after the last falling clock edge the CS remain asserted and after a full bit time the last data bit is picked up. The following figure shows this behaviour:



In this mode the CS will go high 1 full SPI clock cycle after the last clock edge. This guarantees a full SPI clock cycle time for the data to settle and arrive at the MISO input.

AUXSPI0/1_CNTL1 Register (0x7E21 5084,0x7E21 50C4)

SYNOPSIS The AUXSPIx_CNTL1 registers control more features of the SPI interfaces.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------------|---|------|-------|
| 31:18 | - | Reserved, write zero, read as don't care | | |
| 10:8 | CS high time | Additional SPI clock cycles where the CS is high. | R/W | 0 |
| 7 | TX empty IRQ | If 1 the interrupt line is high when the transmit FIFO is empty | R/W | 0 |
| 6 | Done IRQ | If 1 the interrupt line is high when the interface is idle | R/W | 0 |
| 5:2 | - | Reserved, write zero, read as don't care | | |
| 1 | Shift in MS bit first | If 1 the data is shifted in starting with the MS bit. (bit 15) If 0 the data is shifted in starting with the LS bit. (bit 0) | R/W | 0 |
| 0 | Keep input | If 1 the receiver shift register is NOT cleared. Thus new data is concatenated to old data. If 0 the receiver shift register is cleared before each transaction. | R/W | 0 |

Keep input

Setting the 'Keep input' bit will make that the input shift register is not cleared between transactions. However the contents of the shift register is still written to the receive FIFO at the end of each transaction. E.g. if you receive two 8 bit values 0x81 followed by 0x46 the receive FIFO will contain: 0x0081 in the first entry and 0x8146 in the second entry. This mode may save CPU time concatenating bits (4 bits followed by 12 bits).



BCM2835 ARM Peripherals

CS high time

The SPI CS will always be high for at least 1 SPI clock cycle. Some SPI devices need more time to process the data. This field will set a longer CS-high time. So the actual CS high time is (CS_high_time + 1) (In SPI clock cycles).

Interrupts

The SPI block has two interrupts: TX FIFO is empty, SPI is Idle.

TX FIFO is empty:

This interrupt will be asserted as soon as the last entry has been read from the transmit FIFO. At that time the interface will still be busy shifting out that data. This also implies that the receive FIFO will not yet contain the last received data.

It is possible at that time to fill the TX FIFO again and read the receive FIFO entries which have been received. There is a RX FIFO level field which tells exactly how many words are in the receive FIFO. In general at that time the receive FIFO should contain the number of Tx items minus one (the last one still being received). Note that there is no "receive FIFO full" interrupt or "receive FIFO overflow" flag as the number of entries received can never be more than the number of entries transmitted.

AUX is IDLE:

This interrupt will be asserted when the module has finished all activities, including waiting the minimum CS high time. This guarantees that any receive data will be available and 'transparent' changes can be made to the configuration register (e.g. inverting the SPI clock polarity).

AUXSPI0/1_STAT Register (0x7E21 5088,0x7E21 50C8)

SYNOPSIS The AUXSPIx_STAT registers show the status of the SPI interfaces.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|---------------|---|------|-------|
| 31:24 | TX FIFO level | The number of data units in the transmit data FIFO | R/W | 0 |
| 23:12 | RX FIFO level | The number of data units in the receive data FIFO. | R/W | 0 |
| 11:5 | - | Reserved, write zero, read as don't care | | |
| 4 | TX Full | If 1 the transmit FIFO is full If 0 the transmit FIFO can accept at least 1 data unit. | R/W | 0 |
| 3 | TX Empty | If 1 the transmit FIFO is empty If 0 the transmit FIFO holds at least 1 data unit. | R/W | 0 |
| 2 | RX Empty | If 1 the receiver FIFO is empty If 0 the receiver FIFO holds at least 1 data unit. | R/W | 0 |
| 6 | Busy | Indicates the module is busy transferring data. | R/W | 0 |
| 5:0 | Bit count | The number of bits still to be processed. Starts with 'shift-length' and counts down. | R/W | 0 |



BCM2835 ARM Peripherals

Busy

This status bit indicates if the module is busy. It will be clear when the TX FIFO is empty and the module has finished all activities, including waiting the minimum CS high time.

AUXSPI0/1_PEEK Register (0x7E21 508C,0x7E21 50CC)

SYNOPSIS The AUXSPIx_PEEK registers show received data of the SPI interfaces.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:16 | - | Reserved, write zero, read as don't care | | |
| 15:0 | Data | Reads from this address will show the top entry from the receive FIFO, but the data is not taken from the FIFO. This provides a means of inspecting the data but not removing it from the FIFO. | RO | 0 |

AUXSPI0/1_IO Register (0x7E21 50A0-0x7E21 50AC 0x7E21 50E0-0x7E21 50EC)

SYNOPSIS The AUXSPIx_IO registers are the primary data port of the SPI interfaces
These four addresses all write to the same FIFO.

Writing to any of these addresses causes the SPI CS_n pins to be de-asserted at the end of the access

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | - | Reserved, write zero, read as don't care | | |
| 15:0 | Data | Writes to this address range end up in the transmit FIFO. Data is lost when writing whilst the transmit FIFO is full. Reads from this address will take the top entry from the receive FIFO. Reading whilst the receive FIFO is will return the last data received. | R/W | 0 |



BCM2835 ARM Peripherals

AUXSPI0/1_TXHOLD Register

(0x7E21 50B0-0x7E21 50BC

0x7E21 50F0-0x7E21 50FC)

SYNOPSIS The AUXSPIx_TXHOLD registers are the extended CS port of the SPI interfaces. These four addresses all write to the same FIFO.

Writing to these addresses causes the SPI CS_n pins to remain asserted at the end of the access

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | - | Reserved, write zero, read as don't care | | |
| 15:0 | Data | Writes to this address range end up in the transmit FIFO. Data is lost when writing whilst the transmit FIFO is full. Reads from this address will take the top entry from the receive FIFO. Reading whilst the receive FIFO is will return the last data received. | R/W | 0 |



3 BSC

3.1 Introduction

The Broadcom Serial Controller (BSC) controller is a master, fast-mode (400Kb/s) BSC controller. The Broadcom Serial Control bus is a proprietary bus compliant with the Philips® I2C bus/interface version 2.1 January 2000.

- I²C single master only operation (supports clock stretching wait states)
- Both 7-bit and 10-bit addressing is supported.
 - Timing completely software controllable via registers

3.2 Register View

The BSC controller has eight memory-mapped registers. All accesses are assumed to be 32-bit. Note that the BSC2 master is used dedicated with the HDMI interface and should not be accessed by user programs.

There are three BSC masters inside BCM. The register addresses starts from

- BSC0: 0x7E20_5000
- BSC1: 0x7E80_4000
- BSC2 : 0x7E80_5000

The table below shows the address of I²C interface where the address is an offset from one of the three base addresses listed above.

| I2C Address Map | | | |
|-----------------|----------------------|---------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | C | Control | 32 |
| 0x4 | S | Status | 32 |
| 0x8 | DLEN | Data Length | 32 |
| 0xc | A | Slave Address | 32 |
| 0x10 | FIFO | Data FIFO | 32 |
| 0x14 | DIV | Clock Divider | 32 |
| 0x18 | DEL | Data Delay | 32 |



BCM2835 ARM Peripherals

| | | | |
|------|----------------------|-----------------------|----|
| 0x1c | CLKT | Clock Stretch Timeout | 32 |
|------|----------------------|-----------------------|----|

C Register

Synopsis The control register is used to enable interrupts, clear the FIFO, define a read or write operation and start a transfer.

The READ field specifies the type of transfer.

The CLEAR field is used to clear the FIFO. Writing to this field is a one-shot operation which will always read back as zero. The CLEAR bit can set at the same time as the start transfer bit, and will result in the FIFO being cleared just prior to the start of transfer. Note that clearing the FIFO during a transfer will result in the transfer being aborted.

The ST field starts a new BSC transfer. This has a one shot action, and so the bit will always read back as 0 .

The INTD field enables interrupts at the end of a transfer the DONE condition. The interrupt remains active until the DONE condition is cleared by writing a 1 to the I2CS.DONE field. Writing a 0 to the INTD field disables interrupts on DONE.

The INTT field enables interrupts whenever the FIFO is or more empty and needs writing (i.e. during a write transfer) - the TXW condition. The interrupt remains active until the TXW condition is cleared by writing sufficient data to the FIFO to complete the transfer. Writing a 0 to the INTT field disables interrupts on TXW.

The INTR field enables interrupts whenever the FIFO is or more full and needs reading (i.e. during a read transfer) - the RXR condition. The interrupt remains active until the RXW condition is cleared by reading sufficient data from the RX FIFO. Writing a 0 to the INTR field disables interrupts on RXR.

The I2CEN field enables BSC operations. If this bit is 0 then transfers will not be performed. All register accesses are still permitted however.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:16 | | Reserved - Write as 0, read as don't care | | |
| 15 | I2CEN | <u>I2C Enable</u> 0 = BSC controller is disabled 1 = BSC controller is enabled | RW | 0x0 |
| 14:11 | | Reserved - Write as 0, read as don't care | | |
| 10 | INTR | <u>INTR Interrupt on RX</u> 0 = Don t generate interrupts on RXR condition. 1 = Generate interrupt while RXR = 1. | RW | 0x0 |
| 9 | INTT | <u>INTT Interrupt on TX</u> 0 = Don t generate interrupts on TXW condition. 1 = Generate interrupt while TXW = 1. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|-------|---|----|-----|
| 8 | INTD | <u>INTD Interrupt on DONE</u> 0 = Don t generate interrupts on DONE condition. 1 = Generate interrupt while DONE = 1. | RW | 0x0 |
| 7 | ST | <u>ST Start Transfer</u> 0 = No action. 1 = Start a new transfer. One shot operation. Read back as 0. | RW | 0x0 |
| 6 | | Reserved - Write as 0, read as don't care | | |
| 5:4 | CLEAR | <u>CLEAR FIFO Clear</u> 00 = No action. x1 = Clear FIFO. One shot operation. 1x = Clear FIFO. One shot operation. If CLEAR and ST are both set in the same operation, the FIFO is cleared before the new frame is started. Read back as 0. Note: 2 bits are used to maintain compatibility to previous version. | RW | 0x0 |
| 3:1 | | Reserved - Write as 0, read as don't care | | |
| 0 | READ | <u>READ Read Transfer</u> 0 = Write Packet Transfer. 1 = Read Packet Transfer. | RW | 0x0 |

S Register



BCM2835 ARM Peripherals

Synopsis The status register is used to record activity status, errors and interrupt requests. The TA field indicates the activity status of the BSC controller. This read-only field returns a 1 when the controller is in the middle of a transfer and a 0 when idle. The DONE field is set when the transfer completes. The DONE condition can be used with I2CC.INTD to generate an interrupt on transfer completion. The DONE field is reset by writing a 1, writing a 0 to the field has no effect. The read-only TXW bit is set during a write transfer and the FIFO is less than full and needs writing. Writing sufficient data (i.e. enough data to either fill the FIFO more than full or complete the transfer) to the FIFO will clear the field. When the I2CC.INTT control bit is set, the TXW condition can be used to generate an interrupt to write more data to the FIFO to complete the current transfer. If the I2C controller runs out of data to send, it will wait for more data to be written into the FIFO. The read-only RXR field is set during a read transfer and the FIFO is or more full and needs reading. Reading sufficient data to bring the depth below will clear the field. When I2CC.INTR control bit is set, the RXR condition can be used to generate an interrupt to read data from the FIFO before it becomes full. In the event that the FIFO does become full, all I2C operations will stall until data is removed from the FIFO. The read-only TXD field is set when the FIFO has space for at least one byte of data. TXD is clear when the FIFO is full. The TXD field can be used to check that the FIFO can accept data before any is written. Any writes to a full TX FIFO will be ignored. The read-only RXD field is set when the FIFO contains at least one byte of data. RXD is cleared when the FIFO becomes empty. The RXD field can be used to check that the FIFO contains data before reading. Reading from an empty FIFO will return invalid data. The read-only TXE field is set when the FIFO is empty. No further data will be transmitted until more data is written to the FIFO. The read-only RXF field is set when the FIFO is full. No more clocks will be generated until space is available in the FIFO to receive more data. The ERR field is set when the slave fails to acknowledge either its address or a data byte written to it. The ERR field is reset by writing a 1, writing a 0 to the field has no effect. The CLKT field is set when the slave holds the SCL signal high for too long (clock stretching). The CLKT field is reset by writing a 1, writing a 0 to the field has no effect.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:10 | | Reserved - Write as 0, read as don't care | | |
| 9 | CLKT | <u>CLKT Clock Stretch Timeout</u> 0 = No errors detected. 1 = Slave has held the SCL signal low (clock stretching) for longer and that specified in the I2CCLKT register Cleared by writing 1 to the field. | RW | 0x0 |
| 8 | ERR | <u>ERR ACK Error</u> 0 = No errors detected. 1 = Slave has not acknowledged its address. Cleared by writing 1 to the field. | RW | 0x0 |
| 7 | RXF | <u>RXF - FIFO Full</u> 0 = FIFO is not full. 1 = FIFO is full. If a read is underway, no further serial data will be received until data is read from FIFO. | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|------|--|----|-----|
| 6 | TXE | <u>TXE - FIFO Empty</u> 0 = FIFO is not empty. 1 = FIFO is empty. If a write is underway, no further serial data can be transmitted until data is written to the FIFO. | RO | 0x1 |
| 5 | RXD | <u>RXD - FIFO contains Data</u> 0 = FIFO is empty. 1 = FIFO contains at least 1 byte. Cleared by reading sufficient data from FIFO. | RO | 0x0 |
| 4 | TXD | <u>TXD - FIFO can accept Data</u> 0 = FIFO is full. The FIFO cannot accept more data. 1 = FIFO has space for at least 1 byte. | RO | 0x1 |
| 3 | RXR | <u>RXR - FIFO needs Reading (full)</u> 0 = FIFO is less than full and a read is underway. 1 = FIFO is or more full and a read is underway. Cleared by reading sufficient data from the FIFO. | RO | 0x0 |
| 2 | TXW | <u>TXW - FIFO needs Writing (full)</u> 0 = FIFO is at least full and a write is underway (or sufficient data to send). 1 = FIFO is less than full and a write is underway. Cleared by writing sufficient data to the FIFO. | RO | 0x0 |
| 1 | DONE | <u>DONE Transfer Done</u> 0 = Transfer not completed. 1 = Transfer complete. Cleared by writing 1 to the field. | RW | 0x0 |
| 0 | TA | <u>TA Transfer Active</u> 0 = Transfer not active. 1 = Transfer active. | RO | 0x0 |

DLEN Register

Synopsis The data length register defines the number of bytes of data to transmit or receive in the I2C transfer. Reading the register gives the number of bytes remaining in the current transfer.

The DLEN field specifies the number of bytes to be transmitted/received. Reading the DLEN field when a transfer is in progress (TA = 1) returns the number of bytes still to be transmitted or received. Reading the DLEN field when the transfer has just completed (DONE = 1) returns zero as there are no more bytes to transmit or receive. Finally, reading the DLEN field when TA = 0 and DONE = 0 returns the last value written. The DLEN field can be left over multiple transfers.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|------|------|---|----|-----|
| 15:0 | DLEN | <u>Data Length.</u> Writing to DLEN specifies the number of bytes to be transmitted/received. Reading from DLEN when TA = 1 or DONE = 1, returns the number of bytes still to be transmitted or received. Reading from DLEN when TA = 0 and DONE = 0, returns the last DLEN value written. DLEN can be left over multiple packets. | RW | 0x0 |
|------|------|---|----|-----|

A Register

Synopsis The slave address register specifies the slave address and cycle type. The address register can be left across multiple transfers
 The ADDR field specifies the slave address of the I2C device.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:7 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 6:0 | ADDR | <u>Slave Address.</u> | RW | 0x0 |

FIFO Register

Synopsis The Data FIFO register is used to access the FIFO. Write cycles to this address place data in the 16-byte FIFO, ready to transmit on the BSC bus. Read cycles access data received from the bus.
 Data writes to a full FIFO will be ignored and data reads from an empty FIFO will result in invalid data. The FIFO can be cleared using the I2CC.CLEAR field.
 The DATA field specifies the data to be transmitted or received.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:8 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 7:0 | DATA | <u>Writes to the register write transmit data to the FIFO. Reads from register reads received data from the FIFO.</u> | RW | 0x0 |

DIV Register



BCM2835 ARM Peripherals

Synopsis The clock divider register is used to define the clock speed of the BSC peripheral. The CDIV field specifies the core clock divider used by the BSC.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 15:0 | CDIV | <u>Clock Divider</u> SCL = core clock / CDIV Where core_clk is nominally 150 MHz. If CDIV is set to 0, the divisor is 32768. CDIV is always rounded down to an even number. The default value should result in a 100 kHz I2C clock frequency. | RW | 0x5dc |

DEL Register

Synopsis The data delay register provides fine control over the sampling/launch point of the data.
The REDL field specifies the number core clocks to wait after the rising edge before sampling the incoming data.
The FEDL field specifies the number core clocks to wait after the falling edge before outputting the next data bit.
Note: Care must be taken in choosing values for FEDL and REDL as it is possible to cause the BSC master to malfunction by setting values of CDIV/2 or greater. Therefore the delay values should always be set to less than CDIV/2.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:16 | FEDL | <u>FEDL Falling Edge Delay</u> Number of core clock cycles to wait after the falling edge of SCL before outputting next bit of data. | RW | 0x30 |
| 15:0 | REDL | <u>REDL Rising Edge Delay</u> Number of core clock cycles to wait after the rising edge of SCL before reading the next bit of data. | RW | 0x30 |

CLKT Register



BCM2835 ARM Peripherals

Synopsis The clock stretch timeout register provides a timeout on how long the master waits for the slave to stretch the clock before deciding that the slave has hung. The TOUT field specifies the number I2C SCL clocks to wait after releasing SCL high and finding that the SCL is still low before deciding that the slave is not responding and moving the I2C machine forward. When a timeout occurs, the I2CS.CLKT bit is set. Writing 0x0 to TOUT will result in the Clock Stretch Timeout being disabled.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 15:0 | TOUT | <u>TOUT Clock Stretch Timeout Value</u> Number of SCL clock cycles to wait after the rising edge of SCL before deciding that the slave is not responding. | RW | 0x40 |

3.3 10 Bit Addressing

10 Bit addressing is an extension to the standard 7-bit addressing mode. This section describes in detail how to read/write using 10-bit addressing with this I2C controller.

10-bit addressing is compatible with, and can be combined with, 7 bit addressing. Using 10 bits for addressing exploits the reserved combination 1111 0xx for the first byte following a START (S) or REPEATED START (Sr) condition.

The 10 bit slave address is formed from the first two bytes following a S or Sr condition.

The first seven bits of the first byte are the combination 11110XX of which the last two bits (XX) are the two *most significant* bits of the 10-bit address. The eighth bit of the first byte is the R/W bit. If the R/W bit is '0' (write) then the following byte contains the remaining 8 bits of the 10-bit address. If the R/W bit is '1' then the next byte contains data transmitted from the slave to the master.

Writing

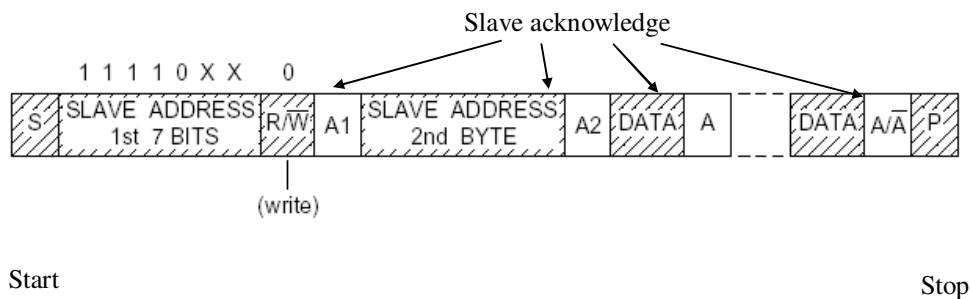


Figure 3-1 Write to a slave with 10 bit address

Figure 3-1 shows a write to a slave with a 10-bit address, to perform this using the controller one must do the following:

Assuming we are in the 'stop' state: (and the FIFO is empty)

1. Write the number of data bytes to written (plus one) to the I2CDLEN register.
2. Write 'XXXXXXXX' to the FIFO where 'XXXXXXXX' are the least 8 significant bits of the 10-bit slave address.
3. Write other data to be transmitted to the FIFO.
4. Write '11110XX' to Slave Address Register where 'XX' are the two most significant bits of the 10-bit address. Set I2CC.READ = 0 and I2CC.ST = 1, this will start a write transfer.

Reading

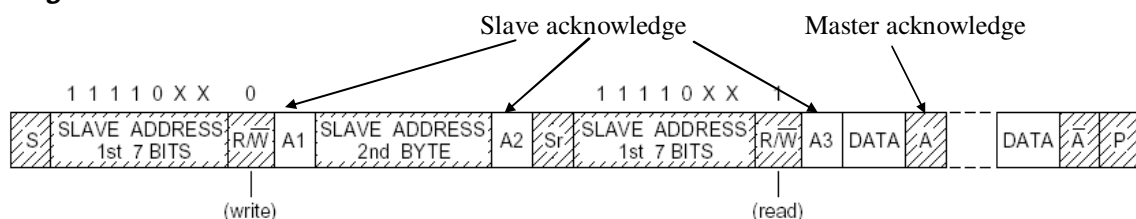


Figure 3-2 Read from slave with 10 bit address

Figure 3-2 shows how a read from a slave with a 10-bit address is performed. Following is the procedure for performing a read using the controller:

1. Write 1 to the I2CDLEN register.
2. Write 'XXXXXXXX' to the FIFO where 'XXXXXXXX' are the least 8 significant bits of the 10-bit slave address.
3. Write '11110XX' to the Slave Address Register where 'XX' are the two most significant bits of the 10-bit address. Set I2CC.READ = 0 and I2CC.ST = 1, this will start a write transfer.
4. Poll the I2CS.TA bit, waiting for the transfer has started.
5. Write the number of data bytes to read to the I2CDLEN register.
6. Set I2CC.READ = 1 and I2CC.ST = 1, this will send the repeat start bit, new slave address and R/W bit (which is '1') initiating the read.

4 DMA Controller

4.1 Overview

The majority of hardware pipelines and peripherals within the BCM2835 are bus masters, enabling them to efficiently satisfy their own data requirements. This reduces the requirements of the DMA controller to block-to-block memory transfers and supporting some of the simpler peripherals. In addition, the DMA controller provides a *read only* prefetch mode to allow data to be brought into the L2 cache in anticipation of its later use.

Beware that the DMA controller is directly connected to the peripherals. Thus the DMA controller must be set-up to use the Physical (hardware) addresses of the peripherals.

The BCM2835 DMA Controller provides a total of 16 DMA channels. Each channel operates independently from the others and is internally arbitrated onto one of the 3 system busses. This means that the amount of bandwidth that a DMA channel may consume can be controlled by the arbiter settings.

Each DMA channel operates by loading a *Control Block* (CB) data structure from memory into internal registers. The *Control Block* defines the required DMA operation. Each *Control Block* can point to a further *Control Block* to be loaded and executed once the operation described in the current *Control Block* has completed. In this way a linked list of *Control Blocks* can be constructed in order to execute a sequence of DMA operations without software intervention.

The DMA supports AXI read bursts to ensure efficient external SDRAM use. The DMA control block contains a burst parameter which indicates the required burst size of certain memory transfers. In general the DMA doesn't do write bursts, although wide writes will be done in 2 beat bursts if possible.

Memory-to-Peripheral transfers can be paced by a *Data Request* (DREQ) signal which is generated by the peripheral. The DREQ signal is level sensitive and controls the DMA by gating its AXI bus requests.

A peripheral can also provide a *Panic* signal alongside the DREQ to indicate that there is an imminent danger of FIFO underflow or overflow or similar critical situation. The *Panic* is used to select the AXI apriority level which is then passed out onto the AXI bus so that it can be used to influence arbitration in the rest of the system.

The allocation of peripherals to DMA channels is programmable.

The DMA can deal with byte aligned transfers and will minimise bus traffic by buffering and packing misaligned accesses.

Each DMA channel can be fully disabled via a top level power register to save power.



BCM2835 ARM Peripherals

4.2 DMA Controller Registers

The DMA Controller is comprised of several identical DMA Channels depending upon the required configuration. Each individual DMA channel has an identical register map (although LITE channels have less functionality and hence less registers).

DMA Channel 0 is located at the address of 0x7E007000, Channel 1 at 0x7E007100, Channel 2 at 0x7E007200 and so on. Thus adjacent DMA Channels are offset by 0x100.

DMA Channel 15 however, is physically removed from the other DMA Channels and so has a different address base of 0x7EE05000.

| DMA Channel Offsets | |
|---|-----------------------------|
| DMA Channels 0 – 14 Register Set Offsets from DMA0_BASE | |
| 0x000 | DMA Channel 0 Register Set |
| 0x100 | DMA Channel 1 Register Set |
| 0x200 | DMA Channel 2 Register Set |
| 0x300 | DMA Channel 3 Register Set |
| 0x400 | DMA Channel 4 Register Set |
| 0x500 | DMA Channel 5 Register Set |
| 0x600 | DMA Channel 6 Register Set |
| 0x700 | DMA Channel 7 Register Set |
| 0x800 | DMA Channel 8 Register Set |
| 0x900 | DMA Channel 9 Register Set |
| 0xa00 | DMA Channel 10 Register Set |
| 0xb00 | DMA Channel 11 Register Set |
| 0xc00 | DMA Channel 12 Register Set |
| 0xd00 | DMA Channel 13 Register Set |
| 0xe00 | DMA Channel 14 Register Set |
| DMA Channel 15 Register Set Offset from DMA15_BASE | |
| 0x000 | DMA Channel 15 Register Set |

Table 4-1 – DMA Controller Register Address Map

4.2.1 DMA Channel Register Address Map

Each DMA channel has an identical register map, only the base address of each channel is different.

There is a global enable register at the top of the Address map that can disable each DMA for powersaving.

Only three registers in each channels register set are directly writeable (CS, CONBLK_AD and DEBUG). The other registers (TI, SOURCE_AD, DEST_AD, TXFR_LEN, STRIDE & NEXTCONBK), are automatically loaded from a *Control Block* data structure held in external memory.

4.2.1.1 Control Block Data Structure

Control Blocks (CB) are 8 words (256 bits) in length and must start at a 256-bit aligned address. The format of the CB data structure in memory, is shown below.

Each 32 bit word of the control block is automatically loaded into the corresponding 32 bit DMA control block register at the start of a DMA transfer. The descriptions of these registers also defines the corresponding bit locations in the CB data structure in memory.

| 32-bit Word Offset | Description | Associated Read-Only Register |
|--------------------|--------------------------------|-------------------------------|
| 0 | Transfer Information | TI |
| 1 | Source Address | SOURCE_AD |
| 2 | Destination Address | DEST_AD |
| 3 | Transfer Length | TXFR_LEN |
| 4 | 2D Mode Stride | STRIDE |
| 5 | Next Control Block Address | NEXTCONBK |
| 6-7 | <i>Reserved – set to zero.</i> | N/A |

Table 4-2 – DMA Control Block Definition

The DMA is started by writing the address of a CB structure into the CONBLK_AD register and then setting the ACTIVE bit. The DMA will fetch the CB from the address set in the SCB_ADDR field of this reg and it will load it into the *read-only* registers described below. It will then begin a DMA transfer according to the information in the CB.

When it has completed the current DMA transfer (length => 0) the DMA will update the CONBLK_AD register with the contents of the NEXTCONBK register, fetch a new CB from that address, and start the whole procedure once again.

The DMA will stop (and clear the ACTIVE bit) when it has completed a DMA transfer and the NEXTCONBK register is set to 0x0000_0000. It will load this value into the CONBLK_AD reg and then stop.



BCM2835 ARM Peripherals

Most of the control block registers cannot be written to directly as they loaded automatically from memory. They can be read to provide status information, and to indicate the progress of the current DMA transfer. The value loaded into the NEXTCONBK register can be overwritten so that the linked list of Control Block data structures can be dynamically altered. However it is only safe to do this when the DMA is paused.

4.2.1.2 Register Map

| DMA Address Map | | | |
|-----------------|-----------------------------|--|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | 0_CS | DMA Channel 0 Control and Status | 32 |
| 0x4 | 0_CONBLK_AD | DMA Channel 0 Control Block Address | 32 |
| 0x8 | 0_TI | DMA Channel 0 CB Word 0 (Transfer Information) | 32 |
| 0xc | 0_SOURCE_AD | DMA Channel 0 CB Word 1 (Source Address) | 32 |
| 0x10 | 0_DEST_AD | DMA Channel 0 CB Word 2 (Destination Address) | 32 |
| 0x14 | 0_TXFR_LEN | DMA Channel 0 CB Word 3 (Transfer Length) | 32 |
| 0x18 | 0_STRIDE | DMA Channel 0 CB Word 4 (2D Stride) | 32 |
| 0x1c | 0_NEXTCONBK | DMA Channel 0 CB Word 5 (Next CB Address) | 32 |
| 0x20 | 0_DEBUG | DMA Channel 0 Debug | 32 |
| 0x100 | 1_CS | DMA Channel 1 Control and Status | 32 |
| 0x104 | 1_CONBLK_AD | DMA Channel 1 Control Block Address | 32 |
| 0x108 | 1_TI | DMA Channel 1 CB Word 0 (Transfer Information) | 32 |
| 0x10c | 1_SOURCE_AD | DMA Channel 1 CB Word 1 (Source Address) | 32 |
| 0x110 | 1_DEST_AD | DMA Channel 1 CB Word 2 (Destination Address) | 32 |
| 0x114 | 1_TXFR_LEN | DMA Channel 1 CB Word 3 (Transfer Length) | 32 |



BCM2835 ARM Peripherals

| | | | |
|-------|-----------------------------|--|----|
| 0x118 | 1 STRIDE | DMA Channel 1 CB Word 4 (2D Stride) | 32 |
| 0x11c | 1 NEXTCONBK | DMA Channel 1 CB Word 5 (Next CB Address) | 32 |
| 0x120 | 1 DEBUG | DMA Channel 1 Debug | 32 |
| 0x200 | 2 CS | DMA Channel 2 Control and Status | 32 |
| 0x204 | 2 CONBLK_AD | DMA Channel 2 Control Block Address | 32 |
| 0x208 | 2 TI | DMA Channel 2 CB Word 0 (Transfer Information) | 32 |
| 0x20c | 2 SOURCE_AD | DMA Channel 2 CB Word 1 (Source Address) | 32 |
| 0x210 | 2 DEST_AD | DMA Channel 2 CB Word 2 (Destination Address) | 32 |
| 0x214 | 2 TXFR_LEN | DMA Channel 2 CB Word 3 (Transfer Length) | 32 |
| 0x218 | 2 STRIDE | DMA Channel 2 CB Word 4 (2D Stride) | 32 |
| 0x21c | 2 NEXTCONBK | DMA Channel 2 CB Word 5 (Next CB Address) | 32 |
| 0x220 | 2 DEBUG | DMA Channel 2 Debug | 32 |
| 0x300 | 3 CS | DMA Channel 3 Control and Status | 32 |
| 0x304 | 3 CONBLK_AD | DMA Channel 3 Control Block Address | 32 |
| 0x308 | 3 TI | DMA Channel 3 CB Word 0 (Transfer Information) | 32 |
| 0x30c | 3 SOURCE_AD | DMA Channel 3 CB Word 1 (Source Address) | 32 |
| 0x310 | 3 DEST_AD | DMA Channel 3 CB Word 2 (Destination Address) | 32 |
| 0x314 | 3 TXFR_LEN | DMA Channel 3 CB Word 3 (Transfer Length) | 32 |
| 0x318 | 3 STRIDE | DMA Channel 3 CB Word 4 (2D Stride) | 32 |
| 0x31c | 3 NEXTCONBK | DMA Channel 3 CB Word 5 (Next CB Address) | 32 |
| 0x320 | 3 DEBUG | DMA Channel 0 Debug | 32 |



BCM2835 ARM Peripherals

| | | | |
|-------|-----------------------------|--|----|
| 0x400 | 4_CS | DMA Channel 4 Control and Status | 32 |
| 0x404 | 4_CONBLK_AD | DMA Channel 4 Control Block Address | 32 |
| 0x408 | 4_TI | DMA Channel 4 CB Word 0 (Transfer Information) | 32 |
| 0x40c | 4_SOURCE_AD | DMA Channel 4 CB Word 1 (Source Address) | 32 |
| 0x410 | 4_DEST_AD | DMA Channel 4 CB Word 2 (Destination Address) | 32 |
| 0x414 | 4_TXFR_LEN | DMA Channel 4 CB Word 3 (Transfer Length) | 32 |
| 0x418 | 4_STRIDE | DMA Channel 4 CB Word 4 (2D Stride) | 32 |
| 0x41c | 4_NEXTCONBK | DMA Channel 4 CB Word 5 (Next CB Address) | 32 |
| 0x420 | 4_DEBUG | DMA Channel 0 Debug | 32 |
| 0x500 | 5_CS | DMA Channel 5 Control and Status | 32 |
| 0x504 | 5_CONBLK_AD | DMA Channel 5 Control Block Address | 32 |
| 0x508 | 5_TI | DMA Channel 5 CB Word 0 (Transfer Information) | 32 |
| 0x50c | 5_SOURCE_AD | DMA Channel 5 CB Word 1 (Source Address) | 32 |
| 0x510 | 5_DEST_AD | DMA Channel 5 CB Word 2 (Destination Address) | 32 |
| 0x514 | 5_TXFR_LEN | DMA Channel 5 CB Word 3 (Transfer Length) | 32 |
| 0x518 | 5_STRIDE | DMA Channel 5 CB Word 4 (2D Stride) | 32 |
| 0x51c | 5_NEXTCONBK | DMA Channel 5 CB Word 5 (Next CB Address) | 32 |
| 0x520 | 5_DEBUG | DMA Channel 5 Debug | 32 |
| 0x600 | 6_CS | DMA Channel 6 Control and Status | 32 |
| 0x604 | 6_CONBLK_AD | DMA Channel 6 Control Block Address | 32 |
| 0x608 | 6_TI | DMA Channel 6 CB Word 0 (Transfer Information) | 32 |



BCM2835 ARM Peripherals

| | | | |
|-------|-----------------------------|--|----|
| 0x60c | 6_SOURCE_AD | DMA Channel 6 CB Word 1 (Source Address) | 32 |
| 0x610 | 6_DEST_AD | DMA Channel 6 CB Word 2 (Destination Address) | 32 |
| 0x614 | 6_TXFR_LEN | DMA Channel 6 CB Word 3 (Transfer Length) | 32 |
| 0x618 | 6_STRIDE | DMA Channel 6 CB Word 4 (2D Stride) | 32 |
| 0x61c | 6_NEXTCONBK | DMA Channel 6 CB Word 5 (Next CB Address) | 32 |
| 0x620 | 6_DEBUG | DMA Channel 6 Debug | 32 |
| 0x700 | 7_CS | DMA Channel 7 Control and Status | 32 |
| 0x704 | 7_CONBLK_AD | DMA Channel 7 Control Block Address | 32 |
| 0x708 | 7_TI | DMA Channel 7 CB Word 0 (Transfer Information) | 32 |
| 0x70c | 7_SOURCE_AD | DMA Channel 7 CB Word 1 (Source Address) | 32 |
| 0x710 | 7_DEST_AD | DMA Channel 7 CB Word 2 (Destination Address) | 32 |
| 0x714 | 7_TXFR_LEN | DMA Channel 7 CB Word 3 (Transfer Length) | 32 |
| 0x71c | 7_NEXTCONBK | DMA Channel 7 CB Word 5 (Next CB Address) | 32 |
| 0x720 | 7_DEBUG | DMA Channel 7 Debug | 32 |
| 0x800 | 8_CS | DMA Channel 8 Control and Status | 32 |
| 0x804 | 8_CONBLK_AD | DMA Channel 8 Control Block Address | 32 |
| 0x808 | 8_TI | DMA Channel 8 CB Word 0 (Transfer Information) | 32 |
| 0x80c | 8_SOURCE_AD | DMA Channel 8 CB Word 1 (Source Address) | 32 |
| 0x810 | 8_DEST_AD | DMA Channel 8 CB Word 2 (Destination Address) | 32 |
| 0x814 | 8_TXFR_LEN | DMA Channel 8 CB Word 3 (Transfer Length) | 32 |
| 0x81c | 8_NEXTCONBK | DMA Channel 8 CB Word 5 (Next CB Address) | 32 |



BCM2835 ARM Peripherals

| | | | |
|-------|------------------------------|---|----|
| 0x820 | 8_DEBUG | DMA Channel 8 Debug | 32 |
| 0x900 | 9_CS | DMA Channel 9 Control and Status | 32 |
| 0x904 | 9_CONBLK_AD | DMA Channel 9 Control Block Address | 32 |
| 0x908 | 9_TI | DMA Channel 9 CB Word 0 (Transfer Information) | 32 |
| 0x90c | 9_SOURCE_AD | DMA Channel 9 CB Word 1 (Source Address) | 32 |
| 0x910 | 9_DEST_AD | DMA Channel 9 CB Word 2 (Destination Address) | 32 |
| 0x914 | 9_TXFR_LEN | DMA Channel 9 CB Word 3 (Transfer Length) | 32 |
| 0x91c | 9_NEXTCONBK | DMA Channel 9 CB Word 5 (Next CB Address) | 32 |
| 0x920 | 9_DEBUG | DMA Channel 9 Debug | 32 |
| 0xa00 | 10_CS | DMA Channel 10 Control and Status | 32 |
| 0xa04 | 10_CONBLK_AD | DMA Channel 10 Control Block Address | 32 |
| 0xa08 | 10_TI | DMA Channel 10 CB Word 0 (Transfer Information) | 32 |
| 0xa0c | 10_SOURCE_AD | DMA Channel 10 CB Word 1 (Source Address) | 32 |
| 0xa10 | 10_DEST_AD | DMA Channel 10 CB Word 2 (Destination Address) | 32 |
| 0xa14 | 10_TXFR_LEN | DMA Channel 10 CB Word 3 (Transfer Length) | 32 |
| 0xa1c | 10_NEXTCONBK | DMA Channel 10 CB Word 5 (Next CB Address) | 32 |
| 0xa20 | 10_DEBUG | DMA Channel 10 Debug | 32 |
| 0xb00 | 11_CS | DMA Channel 11 Control and Status | 32 |
| 0xb04 | 11_CONBLK_AD | DMA Channel 11 Control Block Address | 32 |
| 0xb08 | 11_TI | DMA Channel 11 CB Word 0 (Transfer Information) | 32 |
| 0xb0c | 11_SOURCE_AD | DMA Channel 11 CB Word 1 (Source Address) | 32 |



BCM2835 ARM Peripherals

| | | | |
|-------|------------------------------|---|----|
| 0xb10 | 11_DEST_AD | DMA Channel 11 CB Word 2 (Destination Address) | 32 |
| 0xb14 | 11_TXFR_LEN | DMA Channel 11 CB Word 3 (Transfer Length) | 32 |
| 0xb1c | 11_NEXTCONBK | DMA Channel 11 CB Word 5 (Next CB Address) | 32 |
| 0xb20 | 11_DEBUG | DMA Channel 11 Debug | 32 |
| 0xc00 | 12_CS | DMA Channel 12 Control and Status | 32 |
| 0xc04 | 12_CONBLK_AD | DMA Channel 12 Control Block Address | 32 |
| 0xc08 | 12_TI | DMA Channel 12 CB Word 0 (Transfer Information) | 32 |
| 0xc0c | 12_SOURCE_AD | DMA Channel 12 CB Word 1 (Source Address) | 32 |
| 0xc10 | 12_DEST_AD | DMA Channel 12 CB Word 2 (Destination Address) | 32 |
| 0xc14 | 12_TXFR_LEN | DMA Channel 12 CB Word 3 (Transfer Length) | 32 |
| 0xc1c | 12_NEXTCONBK | DMA Channel 12 CB Word 5 (Next CB Address) | 32 |
| 0xc20 | 12_DEBUG | DMA Channel 12 Debug | 32 |
| 0xd00 | 13_CS | DMA Channel 13 Control and Status | 32 |
| 0xd04 | 13_CONBLK_AD | DMA Channel 13 Control Block Address | 32 |
| 0xd08 | 13_TI | DMA Channel 13 CB Word 0 (Transfer Information) | 32 |
| 0xd0c | 13_SOURCE_AD | DMA Channel 13 CB Word 1 (Source Address) | 32 |
| 0xd10 | 13_DEST_AD | DMA Channel 13 CB Word 2 (Destination Address) | 32 |
| 0xd14 | 13_TXFR_LEN | DMA Channel 13 CB Word 3 (Transfer Length) | 32 |
| 0xd1c | 13_NEXTCONBK | DMA Channel 13 CB Word 5 (Next CB Address) | 32 |
| 0xd20 | 13_DEBUG | DMA Channel 13 Debug | 32 |
| 0xe00 | 14_CS | DMA Channel 14 Control and Status | 32 |



BCM2835 ARM Peripherals

| | | | |
|-------|------------------------------|---|----|
| 0xe04 | 14_CONBLK_AD | DMA Channel 14 Control Block Address | 32 |
| 0xe08 | 14_TI | DMA Channel 14 CB Word 0 (Transfer Information) | 32 |
| 0xe0c | 14_SOURCE_AD | DMA Channel 14 CB Word 1 (Source Address) | 32 |
| 0xe10 | 14_DEST_AD | DMA Channel 14 CB Word 2 (Destination Address) | 32 |
| 0xe14 | 14_TXFR_LEN | DMA Channel 14 CB Word 3 (Transfer Length) | 32 |
| 0xe1c | 14_NEXTCONBK | DMA Channel 14 CB Word 5 (Next CB Address) | 32 |
| 0xe20 | 14_DEBUG | DMA Channel 14 Debug | 32 |
| 0xfe0 | INT_STATUS | Interrupt status of each DMA channel | 32 |
| 0xff0 | ENABLE | Global enable bits for each DMA channel | 32 |

0_CS 1_CS 2_CS 3_CS 4_CS 5_CS 6_CS 7_CS 8_CS 9_CS 10_CS 11_CS 12_CS 13_CS 14_CS Register

Synopsis DMA Control And Status register contains the main control and status bits for this DMA channel.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31 | RESET | <u>DMA Channel Reset</u> Writing a 1 to this bit will reset the DMA. The bit cannot be read, and will self clear. | W1SC | 0x0 |
| 30 | ABORT | <u>Abort DMA</u> Writing a 1 to this bit will abort the current DMA CB. The DMA will load the next CB and attempt to continue. The bit cannot be read, and will self clear. | W1SC | 0x0 |
| 29 | DISDEBUG | <u>Disable debug pause signal</u> When set to 1, the DMA will not stop when the debug pause signal is asserted. | RW | 0x0 |

| | | | | |
|-------|-----------------------------|---|----|-----|
| 28 | WAIT_FOR_OUTSTANDING_WRITES | <p><u>Wait for outstanding writes</u> When set to 1, the DMA will keep a tally of the AXI writes going out and the write responses coming in. At the very end of the current DMA transfer it will wait until the last outstanding write response has been received before indicating the transfer is complete. Whilst waiting it will load the next CB address (but will not fetch the CB), clear the active flag (if the next CB address = zero), and it will defer setting the END flag or the INT flag until the last outstanding write response has been received.</p> <p>In this mode, the DMA will pause if it has more than 13 outstanding writes at any one time.</p> | RW | 0x0 |
| 27:24 | | Reserved - Write as 0, read as don't care | | |
| 23:20 | PANIC_PRIORITY | <p><u>AXI Panic Priority Level</u> Sets the priority of panicking AXI bus transactions. This value is used when the panic bit of the selected peripheral channel is 1. Zero is the lowest priority.</p> | RW | 0x0 |
| 19:16 | PRIORITY | <p><u>AXI Priority Level</u> Sets the priority of normal AXI bus transactions. This value is used when the panic bit of the selected peripheral channel is zero. Zero is the lowest priority.</p> | RW | 0x0 |
| 15:9 | | Reserved - Write as 0, read as don't care | | |
| 8 | ERROR | <p><u>DMA Error</u> Indicates if the DMA has detected an error. The error flags are available in the debug register, and have to be cleared by writing to that register. 1 = DMA channel has an error flag set. 0 = DMA channel is ok.</p> | RO | 0x0 |
| 7 | | Reserved - Write as 0, read as don't care | | |

| | | | | |
|---|--------------------------------|--|-----|-----|
| 6 | WAITING_FOR_OUTSTANDING_WRITES | <p><u>DMA is Waiting for the Last Write to be Received</u> Indicates if the DMA is currently waiting for any outstanding writes to be received, and is not transferring data. 1 = DMA channel is waiting.</p> | RO | 0x0 |
| 5 | DREQ_STOPS_DMA | <p><u>DMA Paused by DREQ State</u> Indicates if the DMA is currently paused and not transferring data due to the DREQ being inactive.. 1 = DMA channel is paused. 0 = DMA channel is running.</p> | RO | 0x0 |
| 4 | PAUSED | <p><u>DMA Paused State</u> Indicates if the DMA is currently paused and not transferring data. This will occur if: the active bit has been cleared, if the DMA is currently executing wait cycles or if the debug_pause signal has been set by the debug block, or the number of outstanding writes has exceeded the max count. 1 = DMA channel is paused. 0 = DMA channel is running.</p> | RO | 0x0 |
| 3 | DREQ | <p><u>DREQ State</u> Indicates the state of the selected DREQ (Data Request) signal, ie. the DREQ selected by the PERMAP field of the transfer info. 1 = Requesting data. This will only be valid once the DMA has started and the PERMAP field has been loaded from the CB. It will remain valid, indicating the selected DREQ signal, until a new CB is loaded. If PERMAP is set to zero (un-paced transfer) then this bit will read back as 1. 0 = No data request.</p> | RO | 0x0 |
| 2 | INT | <p><u>Interrupt Status</u> This is set when the transfer for the CB ends and INTEN is set to 1. Once set it must be manually cleared down, even if the next CB has INTEN = 0. Write 1 to clear.</p> | W1C | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|--------|---|-----|-----|
| 1 | END | <u>DMA End Flag</u> Set when the transfer described by the current control block is complete. Write 1 to clear. | W1C | 0x0 |
| 0 | ACTIVE | <u>Activate the DMA</u> This bit enables the DMA. The DMA will start if this bit is set and the CB_ADDR is non zero. The DMA transfer can be paused and resumed by clearing, then setting it again. This bit is automatically cleared at the end of the complete DMA transfer, ie. after a NEXTCONBK = 0x0000_0000 has been loaded. | RW | 0x0 |

0_CONBLK_AD 1_CONBLK_AD 2_CONBLK_AD 3_CONBLK_AD 4_CONBLK_AD 5_CONBLK_AD 6_CONBLK_AD 7_CONBLK_AD 8_CONBLK_AD 9_CONBLK_AD 10_CONBLK_AD 11_CONBLK_AD 12_CONBLK_AD 13_CONBLK_AD 14_CONBLK_AD Register

Synopsis DMA Control Block Address register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | SCB_ADDR | <u>Control Block Address</u> This tells the DMA where to find a Control Block stored in memory. When the ACTIVE bit is set and this address is non zero, the DMA will begin its transfer by loading the contents of the addressed CB into the relevant DMA channel registers. At the end of the transfer this register will be updated with the ADDR field of the NEXTCONBK control block register. If this field is zero, the DMA will stop. Reading this register will return the address of the currently active CB (in the linked list of CB s). The address must be 256 bit aligned, so the bottom 5 bits of the address must be zero. | RW | 0x0 |

0_TI 1_TI 2_TI 3_TI 4_TI 5_TI 6_TI Register

Synopsis DMA Transfer Information.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
|--------|------------|-------------|------|-------|



BCM2835 ARM Peripherals

| | | | | |
|-------|----------------|--|----|-----|
| 31:27 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 26 | NO_WIDE_BURSTS | <p><u>Don't Do wide writes as a 2 beat burst</u> This prevents the DMA from issuing wide writes as 2 beat AXI bursts. This is an inefficient access mode, so the default is to use the bursts.</p> | RW | 0x0 |
| 25:21 | WAITS | <p><u>Add Wait Cycles</u> This slows down the DMA throughput by setting the number of dummy cycles burnt after each DMA read or write operation is completed. A value of 0 means that no wait cycles are to be added.</p> | RW | 0x0 |
| 20:16 | PERMAP | <p><u>Peripheral Mapping</u> Indicates the peripheral number (1-31) whose ready signal shall be used to control the rate of the transfers, and whose panic signals will be output on the DMA AXI bus. Set to 0 for a continuous un-paced transfer.</p> | RW | 0x0 |
| 15:12 | BURST_LENGTH | <p><u>Burst Transfer Length</u> Indicates the burst length of the DMA transfers. The DMA will attempt to transfer data as bursts of this number of words. A value of zero will produce a single transfer. Bursts are only produced for specific conditions, see main text.</p> | RW | 0x0 |
| 11 | SRC_IGNORE | <p><u>Ignore Reads</u> 1 = Do not perform source reads. In addition, destination writes will zero all the write strobes. This is used for fast cache fill operations. 0 = Perform source reads..</p> | RW | 0x0 |
| 10 | SRC_DREQ | <p><u>Control Source Reads with DREQ</u> 1 = The DREQ selected by PER_MAP will gate the source reads. 0 = DREQ has no effect.</p> | RW | 0x0 |
| 9 | SRC_WIDTH | <p><u>Source Transfer Width</u> 1 = Use 128-bit source read width. 0 = Use 32-bit source read width.</p> | RW | 0x0 |
| 8 | SRC_INC | <p><u>Source Address Increment</u> 1 = Source address increments after each read. The address will increment by 4, if S_WIDTH=0 else by 32. 0 = Source address does not change.</p> | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|-------------|--|----|-----|
| 7 | DEST_IGNORE | <u>Ignore Writes</u> 1 = Do not perform destination writes. 0 = Write data to destination. | RW | 0x0 |
| 6 | DEST_DREQ | <u>Control Destination Writes with DREQ</u> 1 = The DREQ selected by PERMAP will gate the destination writes. 0 = DREQ has no effect. | RW | 0x0 |
| 5 | DEST_WIDTH | <u>Destination Transfer Width</u> 1 = Use 128-bit destination write width. 0 = Use 32-bit destination write width. | RW | 0x0 |
| 4 | DEST_INC | <u>Destination Address Increment</u> 1 = Destination address increments after each write The address will increment by 4, if DEST_WIDTH=0 else by 32. 0 = Destination address does not change. | RW | 0x0 |
| 3 | WAIT_RESP | <u>Wait for a Write Response</u> When set this makes the DMA wait until it receives the AXI write response for each write. This ensures that multiple writes cannot get stacked in the AXI bus pipeline. 1= Wait for the write response to be received before proceeding. 0 = Don t wait; continue as soon as the write data is sent. | RW | 0x0 |
| 2 | | Reserved - Write as 0, read as don't care | | |
| 1 | TDMODE | <u>2D Mode</u> 1 = 2D mode interpret the TXFR_LEN register as YLENGTH number of transfers each of XLENGTH, and add the strides to the address after each transfer. 0 = Linear mode interpret the TXFR register as a single transfer of total length {YLENGTH ,XLENGTH}. | RW | 0x0 |
| 0 | INTEN | <u>Interrupt Enable</u> 1 = Generate an interrupt when the transfer described by the current Control Block completes. 0 = Do not generate an interrupt. | RW | 0x0 |

0_SOURCE_AD 1_SOURCE_AD 2_SOURCE_AD 3_SOURCE_AD 4_SOURCE_AD 5_SOURCE_AD
6_SOURCE_AD 7_SOURCE_AD 8_SOURCE_AD 9_SOURCE_AD 10_SOURCE_AD 11_SOURCE_AD
12_SOURCE_AD 13_SOURCE_AD 14_SOURCE_AD Register



BCM2835 ARM Peripherals

Synopsis DMA Source Address

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | S_ADDR | <u>DMA Source Address</u> Source address for the DMA operation. Updated by the DMA engine as the transfer progresses. | RW | 0x0 |

0_DEST_AD 1_DEST_AD 2_DEST_AD 3_DEST_AD 4_DEST_AD 5_DEST_AD 6_DEST_AD 7_DEST_AD 8_DEST_AD 9_DEST_AD 10_DEST_AD 11_DEST_AD 12_DEST_AD 13_DEST_AD 14_DEST_AD Register

Synopsis DMA Destination Address

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | D_ADDR | <u>DMA Destination Address</u> Destination address for the DMA operation. Updated by the DMA engine as the transfer progresses. | RW | 0x0 |

0_TXFR_LEN 1_TXFR_LEN 2_TXFR_LEN 3_TXFR_LEN 4_TXFR_LEN 5_TXFR_LEN 6_TXFR_LEN Register

Synopsis DMA Transfer Length. This specifies the amount of data to be transferred in bytes. In normal (non 2D) mode this specifies the amount of bytes to be transferred. In 2D mode it is interpreted as an X and a Y length, and the DMA will perform Y transfers, each of length X bytes and add the strides onto the addresses after each X leg of the transfer. The length register is updated by the DMA engine as the transfer progresses, so it will indicate the data left to transfer.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:30 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 29:16 | YLENGTH | When in 2D mode, This is the Y transfer length, indicating how many xlength transfers are performed. When in normal linear mode this becomes the top bits of the XLENGTH | RW | 0x0 |
| 15:0 | XLENGTH | <u>Transfer Length in bytes.</u> | RW | 0x0 |



BCM2835 ARM Peripherals

0_STRIDE 1_STRIDE 2_STRIDE 3_STRIDE 4_STRIDE 5_STRIDE 6_STRIDE Register

Synopsis DMA 2D Stride

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | D_STRIDE | <u>Destination Stride (2D Mode)</u> Signed (2 s complement) byte increment to apply to the destination address at the end of each row in 2D mode. | RW | 0x0 |
| 15:0 | S_STRIDE | <u>Source Stride (2D Mode)</u> Signed (2 s complement) byte increment to apply to the source address at the end of each row in 2D mode. | RW | 0x0 |

0_NEXTCONBK 1_NEXTCONBK 2_NEXTCONBK 3_NEXTCONBK 4_NEXTCONBK 5_NEXTCONBK 6_NEXTCONBK 7_NEXTCONBK 8_NEXTCONBK 9_NEXTCONBK 10_NEXTCONBK 11_NEXTCONBK 12_NEXTCONBK 13_NEXTCONBK 14_NEXTCONBK Register

Synopsis DMA Next Control Block Address

The value loaded into this register can be overwritten so that the linked list of Control Block data structures can be altered. However it is only safe to do this when the DMA is paused. The address must be 256 bit aligned and so the bottom 5 bits cannot be set and will read back as zero.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | ADDR | <u>Address of next CB for chained DMA operations.</u> | RW | 0x0 |

0_DEBUG 1_DEBUG 2_DEBUG 3_DEBUG 4_DEBUG 5_DEBUG 6_DEBUG Register

Synopsis DMA Debug register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:29 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|-------|-------------------------|---|----|-----|
| 28 | LITE | <u>DMA Lite</u> Set if the DMA is a reduced performance LITE engine. | RO | 0x0 |
| 27:25 | VERSION | <u>DMA Version</u> DMA version number, indicating control bit filed changes. | RO | 0x2 |
| 24:16 | DMA_STATE | <u>DMA State Machine State</u> Returns the value of the DMA engines state machine for this channel. | RO | 0x0 |
| 15:8 | DMA_ID | <u>DMA ID</u> Returns the DMA AXI ID of this DMA channel. | RO | 0x0 |
| 7:4 | OUTSTANDING_WRITES | <u>DMA Outstanding Writes Counter</u> Returns the number of write responses that have not yet been received. This count is reset at the start of each new DMA transfer or with a DMA reset. | RO | 0x0 |
| 3 | | Reserved - Write as 0, read as don't care | | |
| 2 | READ_ERROR | <u>Slave Read Response Error</u> Set if the read operation returned an error value on the read response bus. It can be cleared by writing a 1, | RW | 0x0 |
| 1 | FIFO_ERROR | <u>Fifo Error</u> Set if the optional read Fifo records an error condition. It can be cleared by writing a 1, | RW | 0x0 |
| 0 | READ_LAST_NOT_SET_ERROR | <u>Read Last Not Set Error</u> If the AXI read last signal was not set when expected, then this error bit will be set. It can be cleared by writing a 1. | RW | 0x0 |

7_TI 8_TI 9_TI 10_TI 11_TI 12_TI 13_TI 14_TI Register

Synopsis DMA Transfer Information.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:26 | | Reserved - Write as 0, read as don't care | | |

| | | | | |
|-------|--------------|--|----|-----|
| 25:21 | WAITS | <u>Add Wait Cycles</u> This slows down the DMA throughput by setting the number of dummy cycles burnt after each DMA read or write operation is completed. A value of 0 means that no wait cycles are to be added. | RW | 0x0 |
| 20:16 | PERMAP | <u>Peripheral Mapping</u> Indicates the peripheral number (1-31) whose ready signal shall be used to control the rate of the transfers, and whose panic signals will be output on the DMA AXI bus. Set to 0 for a continuous un-paced transfer. | RW | 0x0 |
| 15:12 | BURST_LENGTH | <u>Burst Transfer Length</u> Indicates the burst length of the DMA transfers. The DMA will attempt to transfer data as bursts of this number of words. A value of zero will produce a single transfer. Bursts are only produced for specific conditions, see main text. | RW | 0x0 |
| 11 | SRC_IGNORE | | RW | 0x0 |
| 10 | SRC_DREQ | <u>Control Source Reads with DREQ</u> 1 = The DREQ selected by PER_MAP will gate the source reads. 0 = DREQ has no effect. | RW | 0x0 |
| 9 | SRC_WIDTH | <u>Source Transfer Width</u> 1 = Use 128-bit source read width. 0 = Use 32-bit source read width. | RW | 0x0 |
| 8 | SRC_INC | <u>Source Address Increment</u> 1 = Source address increments after each read. The address will increment by 4, if S_WIDTH=0 else by 32. 0 = Source address does not change. | RW | 0x0 |
| 7 | DEST_IGNORE | | RW | 0x0 |
| 6 | DEST_DREQ | <u>Control Destination Writes with DREQ</u> 1 = The DREQ selected by PERMAP will gate the destination writes. 0 = DREQ has no effect. | RW | 0x0 |
| 5 | DEST_WIDTH | <u>Destination Transfer Width</u> 1 = Use 128-bit destination write width. 0 = Use 32-bit destination write width. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|-----------|---|----|-----|
| 4 | DEST_INC | <u>Destination Address Increment</u> 1 = Destination address increments after each write The address will increment by 4, if DEST_WIDTH=0 else by 32. 0 = Destination address does not change. | RW | 0x0 |
| 3 | WAIT_RESP | <u>Wait for a Write Response</u> When set this makes the DMA wait until it receives the AXI write response for each write. This ensures that multiple writes cannot get stacked in the AXI bus pipeline. 1= Wait for the write response to be received before proceeding. 0 = Don t wait; continue as soon as the write data is sent. | RW | 0x0 |
| 2:1 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 0 | INTEN | <u>Interrupt Enable</u> 1 = Generate an interrupt when the transfer described by the current Control Block completes. 0 = Do not generate an interrupt. | RW | 0x0 |

7_TXFR_LEN 8_TXFR_LEN 9_TXFR_LEN 10_TXFR_LEN 11_TXFR_LEN 12_TXFR_LEN 13_TXFR_LEN 14_TXFR_LEN Register

Synopsis DMA Transfer Length

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 15:0 | XLENGTH | <u>Transfer Length</u> Length of transfer, in bytes. Updated by the DMA engine as the transfer progresses. | RW | 0x0 |

7_DEBUG 8_DEBUG 9_DEBUG 10_DEBUG 11_DEBUG 12_DEBUG 13_DEBUG 14_DEBUG Register

Synopsis DMA Lite Debug register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
|--------|------------|-------------|------|-------|



BCM2835 ARM Peripherals

| | | | | |
|-------|-------------------------|---|----|-----|
| 31:29 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 28 | LITE | <u>DMA Lite</u> Set if the DMA is a reduced performance LITE engine. | RO | 0x1 |
| 27:25 | VERSION | <u>DMA Version</u> DMA version number, indicating control bit filed changes. | RO | 0x2 |
| 24:16 | DMA_STATE | <u>DMA State Machine State</u> Returns the value of the DMA engines state machine for this channel. | RO | 0x0 |
| 15:8 | DMA_ID | <u>DMA ID</u> Returns the DMA AXI ID of this DMA channel. | RO | 0x0 |
| 7:4 | OUTSTANDING_WRITES | <u>DMA Outstanding Writes Counter</u> Returns the number of write responses that have not yet been received. This count is reset at the start of each new DMA transfer or with a DMA reset. | RO | 0x0 |
| 3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2 | READ_ERROR | <u>Slave Read Response Error</u> Set if the read operation returned an error value on the read response bus. It can be cleared by writing a 1, | RW | 0x0 |
| 1 | FIFO_ERROR | <u>Fifo Error</u> Set if the optional read Fifo records an error condition. It can be cleared by writing a 1, | RW | 0x0 |
| 0 | READ_LAST_NOT_SET_ERROR | <u>Read Last Not Set Error</u> If the AXI read last signal was not set when expected, then this error bit will be set. It can be cleared by writing a 1. | RW | 0x0 |

INT_STATUS Register

Synopsis Interrupt status of each DMA engine

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
|--------|------------|-------------|------|-------|



BCM2835 ARM Peripherals

| | | | | |
|-------|-------|--|----|-----|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 15 | INT15 | <u>Interrupt status of DMA engine 15</u> | RW | 0x0 |
| 14 | INT14 | <u>Interrupt status of DMA engine 14</u> | RW | 0x0 |
| 13 | INT13 | <u>Interrupt status of DMA engine 13</u> | RW | 0x0 |
| 12 | INT12 | <u>Interrupt status of DMA engine 12</u> | RW | 0x0 |
| 11 | INT11 | <u>Interrupt status of DMA engine 11</u> | RW | 0x0 |
| 10 | INT10 | <u>Interrupt status of DMA engine 10</u> | RW | 0x0 |
| 9 | INT9 | <u>Interrupt status of DMA engine 9</u> | RW | 0x0 |
| 8 | INT8 | <u>Interrupt status of DMA engine 8</u> | RW | 0x0 |
| 7 | INT7 | <u>Interrupt status of DMA engine 7</u> | RW | 0x0 |
| 6 | INT6 | <u>Interrupt status of DMA engine 6</u> | RW | 0x0 |
| 5 | INT5 | <u>Interrupt status of DMA engine 5</u> | RW | 0x0 |
| 4 | INT4 | <u>Interrupt status of DMA engine 4</u> | RW | 0x0 |
| 3 | INT3 | <u>Interrupt status of DMA engine 3</u> | RW | 0x0 |
| 2 | INT2 | <u>Interrupt status of DMA engine 2</u> | RW | 0x0 |
| 1 | INT1 | <u>Interrupt status of DMA engine 1</u> | RW | 0x0 |
| 0 | INT0 | <u>Interrupt status of DMA engine 0</u> | RW | 0x0 |

ENABLE Register

Synopsis Global enable bits for each channel

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:15 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|----|------|-----------------------------|----|-----|
| 14 | EN14 | <u>enable dma engine 14</u> | RW | 0x1 |
| 13 | EN13 | <u>enable dma engine 13</u> | RW | 0x1 |
| 12 | EN12 | <u>enable dma engine 12</u> | RW | 0x1 |
| 11 | EN11 | <u>enable dma engine 11</u> | RW | 0x1 |
| 10 | EN10 | <u>enable dma engine 10</u> | RW | 0x1 |
| 9 | EN9 | <u>enable dma engine 9</u> | RW | 0x1 |
| 8 | EN8 | <u>enable dma engine 8</u> | RW | 0x1 |
| 7 | EN7 | <u>enable dma engine 7</u> | RW | 0x1 |
| 6 | EN6 | <u>enable dma engine 6</u> | RW | 0x1 |
| 5 | EN5 | <u>enable dma engine 5</u> | RW | 0x1 |
| 4 | EN4 | <u>enable dma engine 4</u> | RW | 0x1 |
| 3 | EN3 | <u>enable dma engine 3</u> | RW | 0x1 |
| 2 | EN2 | <u>enable dma engine 2</u> | RW | 0x1 |
| 1 | EN1 | <u>enable dma engine 1</u> | RW | 0x1 |
| 0 | EN0 | <u>enable dma engine 0</u> | RW | 0x1 |

4.2.1.3 Peripheral DREQ Signals

A DREQ (Data Request) mechanism is used to pace the data flow between the DMA and a peripheral.

Each peripheral is allocated a permanent DREQ signal. Each DMA channel can select which of the DREQ signals should be used to pace the transfer by controlling the DMA reads, DMA writes or both. Note that DREQ 0 is permanently enabled and can be used if no DREQ is required.

When a DREQ signal is being used to pace the DMA reads, the DMA will wait until it has sampled DREQ high before launching a single or burst read operation. It will then wait for all the read data to be returned before re-checking the DREQ and starting the next read. Thus once a peripheral receives the read request it should remove its DREQ as soon as possible to prevent the DMA from re-sampling the same DREQ assertion.

DREQ's are not required when reading from AXI peripherals. In this case, the DMA will request data from the peripheral and the peripheral will only send the data when it is available. The DMA will not request data that it does not have room for, so no pacing of the data flow is required.

DREQ's are required when reading from APB peripherals as the AXI-to-APB bridge will not wait for an APB peripheral to be ready and will just perform the APB read regardless. Thus an APB peripheral needs to make sure that it has all of its read data ready before it drives its DREQ high.

When writing to peripherals, a DREQ is always required to pace the data. However, due to the pipelined nature of the AXI bus system, several writes may be in flight before the peripheral receives any data and withdraws its DREQ signal. Thus the peripheral must ensure that it has sufficient room in its input FIFO to accommodate the maximum amount of data that it might receive. If the peripheral is unable to do this, the DMA WAIT_RESP mechanism can be used to ensure that only one write is in flight at any one time, however this is less efficient transfer mechanism.

The mapping of peripherals to DREQ's is as follows:

| DREQ | Peripheral |
|------|---|
| 0 | DREQ = 1 This is always on so use this channel if no DREQ is required. |
| 1 | DSI |
| 2 | PCM TX |
| 3 | PCM RX |
| 4 | SMI |
| 5 | PWM |
| 6 | SPI TX |
| 7 | SPI RX |



BCM2835 ARM Peripherals

| | |
|----|-----------------------|
| 8 | BSC/SPI Slave TX |
| 9 | BSC/SPI Slave RX |
| 10 | unused |
| 11 | <i>e.MMC</i> |
| 12 | <i>UART TX</i> |
| 13 | SD HOST |
| 14 | UART RX. |
| 15 | DSI |
| 16 | SLIMBUS MCTX. |
| 17 | HDMI |
| 18 | SLIMBUS MCRX |
| 19 | SLIMBUS DC0 |
| 20 | SLIMBUS DC1 |
| 21 | SLIMBUS DC2 |
| 22 | SLIMBUS DC3 |
| 23 | SLIMBUS DC4 |
| 24 | Scaler FIFO 0 & SMI * |
| 25 | Scaler FIFO 1 & SMI * |
| 26 | Scaler FIFO 2 & SMI * |
| 27 | SLIMBUS DC5 |
| 28 | SLIMBUS DC6 |
| 29 | SLIMBUS DC7 |
| 30 | SLIMBUS DC8 |
| 31 | SLIMBUS DC9 |

* The SMI element of the Scaler FIFO 0 & SMI DREQs can be disabled by setting the SMI_DISABLE bit in the DMA_DREQ_CONTROL register in the system arbiter control block.

4.3 AXI Bursts

The DMA supports bursts under specific conditions. Up to 16 beat bursts can be accommodated.

Peripheral (32 bit wide) read bursts are supported. The DMA will generate the burst if there is sufficient room in its read buffer to accommodate all the data from the burst. This limits the burst size to a maximum of 8 beats.

Read bursts in destination ignore mode (DEST_IGNORE) are supported as there is no need for the DMA to deal with the data. This allows wide bursts of up to 16 beats to be used for efficient L2 cache fills.

DMA channel 0 and 15 are fitted with an external 128 bit 8 word read FIFO. This enables efficient memory to memory transfers to be performed. This FIFO allows the DMA to accommodate a wide read burst up to the size of the FIFO. In practice this will allow a 128 bit wide read burst of 9 as the first word back will be immediately read into the DMA engine (or a 32 bit peripheral read burst of 16 – 8 in the input buffer and 8 in the fifo). On any DMA channel, if a read burst is selected that is too large, the AXI read bus will be stalled until the DMA has written out the data. This may lead to inefficient system operation, and possibly AXI lock up if it causes a circular dependency.

In general write bursts are not supported. However to increase the efficiency of L2 cache fills, src_ignore (SRC_IGNORE) transfers can be specified with a write burst. In this case the DMA will issue a write burst address sequence followed by the appropriate number of zero data, zero strobe write bus cycles, which will cause the cache to pre-fetch the data. To improve the efficiency of the 128 bit wide bus architecture, and to make use of the DMAs internal 256 bit registers, the DMA will generate 128 bit wide writes as 2 beat bursts wherever possible, although this behaviour can be disabled.

4.4 Error Handling

If the DMA detects a Read Response error it will record the fact in the READ_ERROR flag in the debug register. This will remain set until it is cleared by writing a 1 to it. The DMA will clear its active flag and generate an interrupt. Any outstanding read data transactions (remainder of a burst) will be honoured. This allows the operator to either restart the DMA by clearing the error bit and setting the active bit, or to abort the DMA transfer by clearing the NEXTCONBK register and restarting the DMA with the ABORT bit set.

The DMA will also record any errors from an external read FIFO. These will be latched in the FIFO_ERROR bit in the debug register until they are cleared by writing a '1' to the bit. (note that only DMA0 and 15 have an external read fifo)

If the DMA detects that a read occurred without the AXI rlast set as expected then it will set the READ_LAST_NOT_SET_ERROR bit in the debug register. This can be cleared by writing a '1' to it.

The error bits are logically OR'd together and presented as a general ERROR bit in the CS register.

4.5 DMA LITE Engines

Several of the DMA engines are of the LITE design. This is a reduced specification engine designed to save space. The engine behaves in the same way as a normal DMA engine except for the following differences.

1. The internal data structure is 128 bits instead of 256 bits. This means that if you do a 128 bit wide read burst of more than 1 beat, the DMA input register will be full and the read bus will be stalled. The normal DMA engine can accept a read burst of 2 without stalling. If you do a narrow 32 bit read burst from the peripherals then the lite engine can cope with a burst of 4 as opposed to a burst of 8 for the normal engine. Note that stalling the read bus will potentially reduce the overall system performance, and may possibly cause a system lockup if you end up with a conflict where the DMA cannot free the read bus as the read stall has prevented it writing out its data due to some circular system relationship.
2. The Lite engine does not support 2D transfers. The TDMODE, S_STRIDE, D_STRIDE and YLENGTH registers will all be removed. Setting these registers will have no effect.
3. The DMA length register is now 16 bits, limiting the maximum transferrable length to 65536 bytes.
4. Source ignore (SRC_IGNORE) and destination ignore (DEST_IGNORE) modes are removed.

The Lite engine will have about half the bandwidth of a normal DMA engine, and are intended for low bandwidth peripheral servicing.

5 External Mass Media Controller

○ Introduction

The External Mass Media Controller (EMMC) is an embedded MultiMedia™ and SD™ card interface provided by Arasan™. It is compliant to the following standards:

- SD™ Host Controller Standard Specification Version 3.0 Draft 1.0
- SDIO™ card specification version 3.0
- SD™ Memory Card Specification Draft version 3.0
- SD™ Memory Card Security Specification version 1.01
- MMC™ Specification version 3.31,4.2 and 4.4

For convenience in the following text card is used as a placeholder for SD™, embedded MultiMedia and SDIO™ cards.

For detailed information about the EMMC internals please refer to the Arasan™ document [SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf](#) but make sure to read the following chapter which lists the changes made to Arasan™'s IP.

Because the EMMC module shares pins with other functionality it must be selected in the GPIO interface. Please refer to the GPIO section for further details.

The interface to the card uses its own clock `clk_emmc` which is provided by the clock manager module. The frequency of this clock should be selected between 50 MHz and 100 MHz. Having a separate clock allows high performance access to the card even if the VideoCore runs at a reduced clock frequency. The EMMC module contains its own internal clock divider to generate the card's clock from `clk_emmc`.

Additionally can the sampling clock for the response and data from the card be delayed in up to 40 steps with a configurable delay between 200ps to 1100ps per step typically. The delay is intended to cancel the internal delay inside the card (up to 14ns) when reading. The delay per step will vary with temperature and supply voltage. Therefore it is better to use a bigger delay than necessary as there is no restriction for the maximum delay.

The EMMC module handles the handshaking process on the command and data lines and all CRC processing automatically.

Command execution is commenced by writing the command plus the appropriate flags to the `CMDTM` register after loading any required argument into the `ARG1` register. The EMMC module calculates the CRC checksum, transfers the command to the card, receives the response and checks its CRC. Once the command has executed or timed-out bit 0 of register `INTERRUPT` will be set. Please note that the `INTERRUPT` register is not self clearing, so the software has first to reset it by writing 1 before using it to detect if a command has finished.



BCM2835 ARM Peripherals

The software is responsible for checking the status bits of the card's response in order to verify successful processing by the card.

In order to transfer data from/to the card register DATA is accessed after configuring the host and sending the according commands to the card using CMDTM. Because the EMMC module doesn't interpret the commands sent to the card it is important to configure it identical to the card setup using the CONTROL0 register. Especial care should be taken to make sure that the width of the data bus is configured identical for host and card. The card is synchronized to the data flow by switching off its clock appropriately. A handshake signal dma_req is available for paced data transfers. Bit 1 of the INTERRUPT register can be used to determine whether a data transfer has finished. Please note that the INTERRUPT register is not self-clearing, so the software has first to reset it by writing 1 before using it to detect if a data transfer has finished.

The EMMC module restricts the maximum block size to the size of the internal data FIFO which is 1k bytes. In order to get maximum performance for data transfers it is necessary to use multiple block data transfers. In this case the EMMC module uses two FIFOs in ping-pong mode, i.e. one is used to transfer data to/from the card while the other is simultaneously accessed by DMA via the AXI bus. If the EMMC module is configured for single block transfers only one FIFO is used, so no DMA access is possible while data is transferred to/from the card and vice versa resulting in long dead times.

○ Registers

Contrary to Arasan™'s documentation the EMMC module registers can only be accessed as 32 bit registers, i.e. the two LSBs of the address are always zero.

The EMMC register base address is 0x7E300000

| EMMC Address Map | | | |
|------------------|----------------------------|---------------------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | ARG2 | ACMD23 Argument | 32 |
| 0x4 | BLKSIZECNT | Block Size and Count | 32 |
| 0x8 | ARG1 | Argument | 32 |
| 0xc | CMDTM | Command and Transfer Mode | 32 |
| 0x10 | RESP0 | Response bits 31 : 0 | 32 |
| 0x14 | RESP1 | Response bits 63 : 32 | 32 |



BCM2835 ARM Peripherals

| | | | |
|------|--------------------------------|-----------------------------------|----|
| 0x18 | RESP2 | Response bits 95 : 64 | 32 |
| 0x1c | RESP3 | Response bits 127 : 96 | 32 |
| 0x20 | DATA | Data | 32 |
| 0x24 | STATUS | Status | 32 |
| 0x28 | CONTROL0 | Host Configuration bits | 32 |
| 0x2c | CONTROL1 | Host Configuration bits | 32 |
| 0x30 | INTERRUPT | Interrupt Flags | 32 |
| 0x34 | IRPT_MASK | Interrupt Flag Enable | 32 |
| 0x38 | IRPT_EN | Interrupt Generation Enable | 32 |
| 0x3c | CONTROL2 | Host Configuration bits | 32 |
| 0x50 | FORCE_IRPT | Force Interrupt Event | 32 |
| 0x70 | BOOT_TIMEOUT | Timeout in boot mode | 32 |
| 0x74 | DBG_SEL | Debug Bus Configuration | 32 |
| 0x80 | EXRDFIFO_CFG | Extension FIFO Configuration | 32 |
| 0x84 | EXRDFIFO_EN | Extension FIFO Enable | 32 |
| 0x88 | TUNE_STEP | Delay per card clock tuning step | 32 |
| 0x8c | TUNE_STEPS_STD | Card clock tuning steps for SDR | 32 |
| 0x90 | TUNE_STEPS_DDR | Card clock tuning steps for DDR | 32 |
| 0xf0 | SPI_INT_SPT | SPI Interrupt Support | 32 |
| 0xfc | SLOTISR_VER | Slot Interrupt Status and Version | 32 |

ARG2 Register



BCM2835 ARM Peripherals

Synopsis This register contains the argument for the SD card specific command ACMD23 (SET_WR_BLK_ERASE_COUNT). ARG2 must be set before the ACMD23 command is issued using the CMDTM register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | ARGUMENT | <u>Argument to be issued with ACMD23</u> | RW | 0x0 |

BLKSIZECNT Register

Synopsis This register must not be accessed or modified while any data transfer between card and host is ongoing. It contains the number and size in bytes for data blocks to be transferred. Please note that the EMMC module restricts the maximum block size to the size of the internal data FIFO which is 1k bytes. BLKCNT is used to tell the host how many blocks of data are to be transferred. Once the data transfer has started and the TM_BLKCNT_EN bit in the CMDTM register is set the EMMC module automatically decreases the BNTCNT value as the data blocks are transferred and stops the transfer once BLKCNT reaches 0.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | BLKCNT | <u>Number of blocks to be transferred</u> | RW | 0x0 |
| 15:10 | | Reserved - Write as 0, read as don't care | | |
| 9:0 | BLKSIZE | <u>Block size in bytes</u> | RW | 0x0 |

ARG1 Register

Synopsis This register contains the arguments for all commands except for the SD card specific command ACMD23 which uses ARG2. ARG1 must be set before the command is issued using the CMDTM register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | ARGUMENT | <u>Argument to be issued with command</u> | RW | 0x0 |

CMDTM Register

Synopsis This register is used to issue commands to the card. Besides the command it also contains flags informing the EMMC module what card response and type of data transfer to expect. Incorrect flags will result in strange behaviour.

For data transfers two modes are supported: either transferring a single block of data or several blocks of the same size. The SD card uses two different sets of commands to differentiate between them but the host needs to be additionally configured using TM_MULTI_BLOCK. It is important that this bit is set correct for the command sent to the card, i.e. 1 for CMD18 and CMD25 and 0 for CMD17 and CMD24. Multiple block transfer gives a better performance.

The BLKSIZECNT register is used to configure the size and number of blocks to be transferred. If bit TM_BLKCNT_EN of this register is set the transfer stops automatically after the number of data blocks configured in the BLKSIZECNT register has been transferred.

The TM_AUTO_CMD_EN bits can be used to make the host to send automatically a command to the card telling it that the data transfer has finished once the BLKCNT bits in the BLKSIZECNT register are 0.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|---|------|-------|
| 31:30 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 29:24 | CMD_INDEX | <u>Index of the command to be issued to the card</u> | RW | 0x0 |
| 23:22 | CMD_TYPE | <u>Type of command to be issued to the card:</u> 00 = normal 01 = suspend (the current data transfer) 10 = resume (the last data transfer) 11 = abort (the current data transfer) | RW | 0x0 |
| 21 | CMD_ISDATA | <u>Command involves data transfer:</u> 0 = no data transfer command 1 = data transfer command | RW | 0x0 |
| 20 | CMD_IXCHK_EN | <u>Check that response has same index as command:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 19 | CMD_CRCCHK_EN | <u>Check the responses CRC:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 18 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 17:16 | CMD_RSPNS_TYPE | <u>Type of expected response from card:</u> 00 = no response 01 = 136 bits response 10 = 48 bits response 11 = 48 bits response using busy | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|----------------|---|----|-----|
| 15:6 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 5 | TM_MULTI_BLOCK | <u>Type of data transfer</u> 0 = single block 1 = multiple block | RW | 0x0 |
| 4 | TM_DAT_DIR | <u>Direction of data transfer:</u> 0 = from host to card 1 = from card to host | RW | 0x0 |
| 3:2 | TM_AUTO_CMD_EN | <u>Select the command to be send after completion of a data transfer:</u> 00 = no command 01 = command CMD12 10 = command CMD23 11 = reserved | RW | 0x0 |
| 1 | TM_BLKCNT_EN | <u>Enable the block counter for multiple block transfers:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 0 | | <i>Reserved - Write as 0, read as don't care</i> | | |

RESP0 Register

Synopsis This register contains the status bits of the SD card s response. In case of commands CMD2 and CMD10 it contains CID[31:0] and in case of command CMD9 it contains CSD[31:0].
 Note: this register is only valid once the last command has completed and no new command was issued.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | RESPONSE | <u>Bits 31:0 of the card s response</u> | RW | 0x0 |

RESP1 Register

Synopsis In case of commands CMD2 and CMD10 this register contains CID[63:32] and in case of command CMD9 it contains CSD[63:32].
 Note: this register is only valid once the last command has completed and no new command was issued.



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | RESPONSE | <u>Bits 63:32 of the card s response</u> | RW | 0x0 |

RESP2 Register

Synopsis In case of commands CMD2 and CMD10 this register contains CID[95:64] and in case of command CMD9 it contains CSD[95:64].
Note: this register is only valid once the last command has completed and no new command was issued.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | RESPONSE | <u>Bits 95:64 of the card s response</u> | RW | 0x0 |

RESP3 Register

Synopsis In case of commands CMD2 and CMD10 this register contains CID[127:96] and in case of command CMD9 it contains CSD[127:96].
Note: this register is only valid once the last command has completed and no new command was issued.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | RESPONSE | <u>Bits 127:96 of the card s response</u> | RW | 0x0 |

DATA Register

Synopsis This register is used to transfer data to/from the card.
Bit 1 of the INTERRUPT register can be used to check if data is available. For paced DMA transfers the high active signal dma_req can be used.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|------------------------------|------|-------|
| 31:0 | DATA | <u>Data to/from the card</u> | RW | 0x0 |

STATUS Register

Synopsis This register contains information intended for debugging. Its values change automatically according to the hardware. As it involves resynchronisation between different clock domains it changes only after some latency and it is easy sample the values too early. Therefore it is not recommended to use this register for polling. Instead use the INTERRUPT register which implements a handshake mechanism which makes it impossible to miss a change when polling.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31:29 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 28:25 | DAT_LEVEL1 | <u>Value of data lines DAT7 to DAT4</u> | RW | 0xf |
| 24 | CMD_LEVEL | <u>Value of command line CMD</u> | RW | 0x1 |
| 23:20 | DAT_LEVEL0 | <u>Value of data lines DAT3 to DAT0</u> | RW | 0xf |
| 19:10 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 9 | READ_TRANSFER | <u>New data can be read from EMMC:</u> 0 = no 1 = yes | RW | 0x0 |
| 8 | WRITE_TRANSFER | <u>New data can be written to EMMC:</u> 0 = no 1 = yes | RW | 0x0 |
| 7:3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2 | DAT_ACTIVE | <u>At least one data line is active:</u> 0 = no 1 = yes | RW | 0x0 |
| 1 | DAT_INHIBIT | <u>Data lines still used by previous data transfer:</u> 0 = no 1 = yes | RW | 0x0 |
| 0 | CMD_INHIBIT | <u>Command line still used by previous command:</u> 0 = no 1 = yes | RW | 0x0 |



BCM2835 ARM Peripherals

CONTROL0 Register

Synopsis This register is used to configure the EMMC module.
 For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-------------|--|------|-------|
| 31:23 | | Reserved - Write as 0, read as don't care | | |
| 22 | ALT_BOOT_EN | <u>Enable alternate boot mode access:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 21 | BOOT_EN | <u>Boot mode access:</u> 0 = stop boot mode access 1 = start boot mode access | RW | 0x0 |
| 20 | SPI_MODE | <u>SPI mode enable:</u> 0 = normal mode 1 = SPI mode | RW | 0x0 |
| 19 | GAP_IEN | <u>Enable SDIO interrupt at block gap (only valid if the HCTL_DWIDTH bit is set):</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 18 | READWAIT_EN | <u>Use DAT2 read-wait protocol for SDIO cards supporting this:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 17 | GAP_RESTART | <u>Restart a transaction which was stopped using the GAP_STOP bit:</u> 0 = ignore 1 = restart | RW | 0x0 |
| 16 | GAP_STOP | <u>Stop the current transaction at the next block gap:</u> 0 = ignore 1 = stop | RW | 0x0 |
| 15:6 | | Reserved - Write as 0, read as don't care | | |
| 5 | HCTL_8BIT | <u>Use 8 data lines:</u> 0 = disabled 1 = enabled | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|-------------|--|----|-----|
| 4:3 | | Reserved - Write as 0, read as don't care | | |
| 2 | HCTL_HS_EN | <u>Select high speed mode (i.e. DAT and CMD lines change on the rising CLK edge):</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 1 | HCTL_DWIDTH | <u>Use 4 data lines:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 0 | | Reserved - Write as 0, read as don't care | | |

CONTROL1 Register

Synopsis This register is used to configure the EMMC module.
 For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.
 CLK_STABLE seems contrary to its name only to indicate that there was a rising edge on the clk_emmc input but not that the frequency of this clock is actually stable.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-------------|---|------|-------|
| 31:27 | | Reserved - Write as 0, read as don't care | | |
| 26 | SRST_DATA | <u>Reset the data handling circuit:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 25 | SRST_CMD | <u>Reset the command handling circuit:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 24 | SRST_HC | <u>Reset the complete host circuit:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 23:20 | | Reserved - Write as 0, read as don't care | | |
| 19:16 | DATA_TOUNIT | <u>Data timeout unit exponent:</u> 1111 = disabled $x = TMCLK * 2^{(x+13)}$ | RW | 0x0 |
| 15:8 | CLK_FREQ8 | <u>SD clock base divider LSBs</u> | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|--------------|---|----|-----|
| 7:6 | CLK_FREQ_MS2 | <u>SD clock base divider MSBs</u> | RW | 0x0 |
| 5 | CLK_GENSEL | <u>Mode of clock generation:</u> 0 = divided 1 = programmable | RW | 0x0 |
| 4:3 | | Reserved - Write as 0, read as don't care | | |
| 2 | CLK_EN | <u>SD clock enable:</u> 0 = disabled 1 = enabled | RW | 0x0 |
| 1 | CLK_STABLE | <u>SD clock stable:</u> 0 = no 1 = yes | RO | 0x0 |
| 0 | CLK_INTLEN | <u>Clock enable for internal EMMC clocks for power saving:</u> 0 = disabled 1 = enabled | RW | 0x0 |

INTERRUPT Register

Synopsis This register holds the interrupt flags. Each flag can be disabled using the according bit in the IRPT_MASK register.
 For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.
 ERR is a generic flag and is set if any of the enabled error flags is set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:25 | | Reserved - Write as 0, read as don't care | | |
| 24 | ACMD_ERR | <u>Auto command error:</u> 0 = no error 1 = error | RW | 0x0 |
| 23 | | Reserved - Write as 0, read as don't care | | |
| 22 | DEND_ERR | <u>End bit on data line not 1:</u> 0 = no error 1 = error | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|----------|--|----|-----|
| 21 | DCRC_ERR | <u>Data CRC error:</u> 0 = no error 1 = error | RW | 0x0 |
| 20 | DTO_ERR | <u>Timeout on data line:</u> 0 = no error 1 = error | RW | 0x0 |
| 19 | CBAD_ERR | <u>Incorrect command index in response:</u> 0 = no error 1 = error | RW | 0x0 |
| 18 | CEND_ERR | <u>End bit on command line not 1:</u> 0 = no error 1 = error | RW | 0x0 |
| 17 | CCRC_ERR | <u>Command CRC error:</u> 0 = no error 1 = error | RW | 0x0 |
| 16 | CTO_ERR | <u>Timeout on command line:</u> 0 = no error 1 = error | RW | 0x0 |
| 15 | ERR | <u>An error has occurred:</u> 0 = no error 1 = error | RO | 0x0 |
| 14 | ENDBOOT | <u>Boot operation has terminated:</u> 0 = no 1 = yes | RW | 0x0 |
| 13 | BOOTACK | <u>Boot acknowledge has been received:</u> 0 = no 1 = yes | RW | 0x0 |
| 12 | RETUNE | <u>Clock retune request was made:</u> 0 = no 1 = yes | RO | 0x0 |
| 11:9 | | Reserved - Write as 0, read as don't care | | |
| 8 | CARD | <u>Card made interrupt request:</u> 0 = no 1 = yes | RO | 0x0 |
| 7:6 | | Reserved - Write as 0, read as don't care | | |



BCM2835 ARM Peripherals

| | | | | |
|---|-----------|---|----|-----|
| 5 | READ_RDY | <u>DATA register contains data to be read:</u> 0 = no 1 = yes | RW | 0x0 |
| 4 | WRITE_RDY | <u>Data can be written to DATA register:</u> 0 = no 1 = yes | RW | 0x0 |
| 3 | | Reserved - Write as 0, read as don't care | | |
| 2 | BLOCK_GAP | <u>Data transfer has stopped at block gap:</u> 0 = no 1 = yes | RW | 0x0 |
| 1 | DATA_DONE | <u>Data transfer has finished:</u> 0 = no 1 = yes | RW | 0x0 |
| 0 | CMD_DONE | <u>Command has finished:</u> 0 = no 1 = yes | RW | 0x0 |

IRPT_MASK Register

Synopsis This register is used to mask the interrupt flags in the INTERRUPT register. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:25 | | Reserved - Write as 0, read as don't care | | |
| 24 | ACMD_ERR | <u>Set flag if auto command error:</u> 0 = no 1 = yes | RW | 0x0 |
| 23 | | Reserved - Write as 0, read as don't care | | |
| 22 | DEND_ERR | <u>Set flag if end bit on data line not 1:</u> 0 = no 1 = yes | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|----------|---|----|-----|
| 21 | DCRC_ERR | <u>Set flag if data CRC error:</u> 0 = no 1 = yes | RW | 0x0 |
| 20 | DTO_ERR | <u>Set flag if timeout on data line:</u> 0 = no 1 = yes | RW | 0x0 |
| 19 | CBAD_ERR | <u>Set flag if incorrect command index in response:</u> 0 = no 1 = yes | RW | 0x0 |
| 18 | CEND_ERR | <u>Set flag if end bit on command line not 1:</u> 0 = no 1 = yes | RW | 0x0 |
| 17 | CCRC_ERR | <u>Set flag if command CRC error:</u> 0 = no 1 = yes | RW | 0x0 |
| 16 | CTO_ERR | <u>Set flag if timeout on command line:</u> 0 = no 1 = yes | RW | 0x0 |
| 15 | | Reserved - Write as 0, read as don't care | | |
| 14 | ENDBOOT | <u>Set flag if boot operation has terminated:</u> 0 = no 1 = yes | RW | 0x0 |
| 13 | BOOTACK | <u>Set flag if boot acknowledge has been received:</u> 0 = no 1 = yes | RW | 0x0 |
| 12 | RETUNE | <u>Set flag if clock retune request was made:</u> 0 = no 1 = yes | RW | 0x0 |
| 11:9 | | Reserved - Write as 0, read as don't care | | |
| 8 | CARD | <u>Set flag if card made interrupt request:</u> 0 = no 1 = yes | RW | 0x0 |
| 7:6 | | Reserved - Write as 0, read as don't care | | |
| 5 | READ_RDY | <u>Set flag if DATA register contains data to be read:</u> 0 = no 1 = yes | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|-----------|---|----|-----|
| 4 | WRITE_RDY | <u>Set flag if data can be written to DATA register:</u> 0 = no 1 = yes | RW | 0x0 |
| 3 | | Reserved - Write as 0, read as don't care | | |
| 2 | BLOCK_GAP | <u>Set flag if data transfer has stopped at block gap:</u> 0 = no 1 = yes | RW | 0x0 |
| 1 | DATA_DONE | <u>Set flag if data transfer has finished:</u> 0 = no 1 = yes | RW | 0x0 |
| 0 | CMD_DONE | <u>Set flag if command has finished:</u> 0 = no 1 = yes | RW | 0x0 |

IRPT_EN Register

Synopsis This register is used to enable the different interrupts in the INTERRUPT register to generate an interrupt on the int_to_arm output. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:25 | | Reserved - Write as 0, read as don't care | | |
| 24 | ACMD_ERR | <u>Create interrupt if auto command error:</u> 0 = no 1 = yes | RW | 0x0 |
| 23 | | Reserved - Write as 0, read as don't care | | |
| 22 | DEND_ERR | <u>Create interrupt if end bit on data line not 1:</u> 0 = no 1 = yes | RW | 0x0 |
| 21 | DCRC_ERR | <u>Create interrupt if data CRC error:</u> 0 = no 1 = yes | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|----------|---|----|-----|
| 20 | DTO_ERR | <u>Create interrupt if timeout on data line:</u> 0 = no 1 = yes | RW | 0x0 |
| 19 | CBAD_ERR | <u>Create interrupt if incorrect command index in response:</u> 0 = no 1 = yes | RW | 0x0 |
| 18 | CEND_ERR | <u>Create interrupt if end bit on command line not 1:</u> 0 = no 1 = yes | RW | 0x0 |
| 17 | CCRC_ERR | <u>Create interrupt if command CRC error:</u> 0 = no 1 = yes | RW | 0x0 |
| 16 | CTO_ERR | <u>Create interrupt if timeout on command line:</u> 0 = no 1 = yes | RW | 0x0 |
| 15 | | Reserved - Write as 0, read as don't care | | |
| 14 | ENDBOOT | <u>Create interrupt if boot operation has terminated:</u> 0 = no 1 = yes | RW | 0x0 |
| 13 | BOOTACK | <u>Create interrupt if boot acknowledge has been received:</u> 0 = no 1 = yes | RW | 0x0 |
| 12 | RETUNE | <u>Create interrupt if clock retune request was made:</u> 0 = no 1 = yes | RW | 0x0 |
| 11:9 | | Reserved - Write as 0, read as don't care | | |
| 8 | CARD | <u>Create interrupt if card made interrupt request:</u> 0 = no 1 = yes | RW | 0x0 |
| 7:6 | | Reserved - Write as 0, read as don't care | | |
| 5 | READ_RDY | <u>Create interrupt if DATA register contains data to be read:</u> 0 = no 1 = yes | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|-----------|---|----|-----|
| 4 | WRITE_RDY | <u>Create interrupt if data can be written to DATA register:</u> 0 = no 1 = yes | RW | 0x0 |
| 3 | | Reserved - Write as 0, read as don't care | | |
| 2 | BLOCK_GAP | <u>Create interrupt if data transfer has stopped at block gap:</u> 0 = no 1 = yes | RW | 0x0 |
| 1 | DATA_DONE | <u>Create interrupt if data transfer has finished:</u> 0 = no 1 = yes | RW | 0x0 |
| 0 | CMD_DONE | <u>Create interrupt if command has finished:</u> 0 = no 1 = yes | RW | 0x0 |

CONTROL2 Register

Synopsis This register is used to enable the different interrupts in the INTERRUPT register to generate an interrupt on the int_to_arm output. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:24 | | Reserved - Write as 0, read as don't care | | |
| 23 | TUNED | <u>Tuned clock is used for sampling data:</u> 0 = no 1 = yes | RW | 0x0 |
| 22 | TUNEON | <u>Start tuning the SD clock:</u> 0 = not tuned or tuning complete 1 = tuning | RW | 0x0 |
| 21:19 | | Reserved - Write as 0, read as don't care | | |



BCM2835 ARM Peripherals

| | | | | |
|-------|------------|---|----|-----|
| 18:16 | UHSMODE | Select the speed mode of the SD card: 000 = SDR12 001 = SDR25 010 = SDR50 011 = SDR104 100 = DDR50 other = reserved | RW | 0x0 |
| 15:8 | | Reserved - Write as 0, read as don't care | | |
| 7 | NOTC12_ERR | <u>Error occurred during auto command CMD12 execution:</u> 0 = no error 1 = error | RO | 0x0 |
| 6:5 | | Reserved - Write as 0, read as don't care | | |
| 4 | ACBAD_ERR | <u>Command index error occurred during auto command execution:</u> 0 = no error 1 = error | RO | 0x0 |
| 3 | ACEND_ERR | <u>End bit is not 1 during auto command execution:</u> 0 = no error 1 = error | RO | 0x0 |
| 2 | ACCRC_ERR | <u>Command CRC error occurred during auto command execution:</u> 0 = no error 1 = error | RO | 0x0 |
| 1 | ACTO_ERR | <u>Timeout occurred during auto command execution:</u> 0 = no error 1 = error | RO | 0x0 |
| 0 | ACNOX_ERR | <u>Auto command not executed due to an error:</u> 0 = no 1 = yes | RO | 0x0 |

FORCE_IRPT Register

Synopsis This register is used to fake the different interrupt events for debugging. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:25 | | Reserved - Write as 0, read as don't care | | |
| 24 | ACMD_ERR | <u>Create auto command error:</u> 0 = no 1 = yes | RW | 0x0 |
| 23 | | Reserved - Write as 0, read as don't care | | |
| 22 | DEND_ERR | <u>Create end bit on data line not 1:</u> 0 = no 1 = yes | RW | 0x0 |
| 21 | DCRC_ERR | <u>Create data CRC error:</u> 0 = no 1 = yes | RW | 0x0 |
| 20 | DTO_ERR | <u>Create timeout on data line:</u> 0 = no 1 = yes | RW | 0x0 |
| 19 | CBAD_ERR | <u>Create incorrect command index in response:</u> 0 = no 1 = yes | RW | 0x0 |
| 18 | CEND_ERR | <u>Create end bit on command line not 1:</u> 0 = no 1 = yes | RW | 0x0 |
| 17 | CCRC_ERR | <u>Create command CRC error:</u> 0 = no 1 = yes | RW | 0x0 |
| 16 | CTO_ERR | <u>Create timeout on command line:</u> 0 = no 1 = yes | RW | 0x0 |
| 15 | | Reserved - Write as 0, read as don't care | | |
| 14 | ENDBOOT | <u>Create boot operation has terminated:</u> 0 = no 1 = yes | RW | 0x0 |
| 13 | BOOTACK | <u>Create boot acknowledge has been received:</u> 0 = no 1 = yes | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|-----------|---|----|-----|
| 12 | RETUNE | <u>Create clock retune request was made:</u> 0 = no 1 = yes | RW | 0x0 |
| 11:9 | | Reserved - Write as 0, read as don't care | | |
| 8 | CARD | <u>Create card made interrupt request:</u> 0 = no 1 = yes | RW | 0x0 |
| 7:6 | | Reserved - Write as 0, read as don't care | | |
| 5 | READ_RDY | <u>Create DATA register contains data to be read:</u> 0 = no 1 = yes | RW | 0x0 |
| 4 | WRITE_RDY | <u>Create data can be written to DATA register:</u> 0 = no 1 = yes | RW | 0x0 |
| 3 | | Reserved - Write as 0, read as don't care | | |
| 2 | BLOCK_GAP | <u>Create interrupt if data transfer has stopped at block gap:</u> 0 = no 1 = yes | RW | 0x0 |
| 1 | DATA_DONE | <u>Create data transfer has finished:</u> 0 = no 1 = yes | RW | 0x0 |
| 0 | CMD_DONE | <u>Create command has finished:</u> 0 = no 1 = yes | RW | 0x0 |

BOOT_TIMEOUT Register

Synopsis This register configures after how many card clock cycles a timeout for e.MMC cards in boot mode is flagged

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | TIMEOUT | <u>Number of card clock cycles after which a timeout during boot mode is flagged</u> | RW | 0x0 |



BCM2835 ARM Peripherals

DBG_SEL Register

Synopsis This register selects which submodules are accessed by the debug bus. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:1 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 0 | SELECT | Submodules accessed by debug bus: 0 = receiver and fifo_ctrl 1 = others | RW | 0x0 |

EXRDFIFO_CFG Register

Synopsis This register allows fine tuning the dma_req generation for paced DMA transfers when reading from the card. If the extension data FIFO contains less than RD_THRSH 32 bits words dma_req becomes inactive until the card has filled the extension data FIFO above threshold. This compensates the DMA latency. When writing data to the card the extension data FIFO feeds into the EMMC module s FIFO and no fine tuning is required Therefore the RD_THRSH value is in this case ignored.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2:0 | RD_THRSH | <u>Read threshold in 32 bits words</u> | RW | 0x0 |

EXRDFIFO_EN Register

Synopsis This register enables the extension data register. It should be enabled for paced DMA transfers and be bypassed for burst DMA transfers.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:1 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|---|--------|---|----|-----|
| 0 | ENABLE | Enable the extension FIFO: 0 = bypass 1 = enabled | RW | 0x0 |
|---|--------|---|----|-----|

TUNE_STEP Register

Synopsis This register is used to delay the card clock when sampling the returning data and command response from the card.
DELAY determines by how much the sampling clock is delayed per step.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2:0 | DELAY | Sampling clock delay per step: 000 = 200ps typically 001 = 400ps typically 010 = 400ps typically 011 = 600ps typically 100 = 700ps typically 101 = 900ps typically 110 = 900ps typically 111 = 1100ps typically | RW | 0x0 |

TUNE_STEPS_STD Register

Synopsis This register is used to delay the card clock when sampling the returning data and command response from the card. It determines by how many steps the sampling clock is delayed in SDR mode.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:6 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 5:0 | STEPS | Number of steps (0 to 40) | RW | 0x0 |

TUNE_STEPS_DDR Register



BCM2835 ARM Peripherals

Synopsis This register is used to delay the card clock when sampling the returning data and command response from the card. It determines by how many steps the sampling clock is delayed in DDR mode.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:6 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 5:0 | STEPS | <u>Number of steps (0 to 40)</u> | RW | 0x0 |

SPI_INT_SPT Register

Synopsis This register controls whether assertion of interrupts in SPI mode is possible independent of the card select line.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bit marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:8 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 7:0 | SELECT | <u>Interrupt independent of card select line:</u> 0 = no 1 = yes | RW | 0x0 |

SLOTISR_VER Register

Synopsis This register contains the version information and slot interrupt status.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bit marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:24 | VENDOR | <u>Vendor Version Number</u> | RW | 0x0 |
| 23:16 | SDVERSION | <u>Host Controller specification version</u> | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|-------------|--|----|-----|
| 15:8 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 7:0 | SLOT_STATUS | <u>Logical OR of interrupt and wakeup signal for each slot</u> | RW | 0x0 |

6 General Purpose I/O (GPIO)

There are 54 general-purpose I/O (GPIO) lines split into two banks. All GPIO pins have at least two alternative functions within BCM. The alternate functions are usually peripheral IO and a single peripheral may appear in each bank to allow flexibility on the choice of IO voltage. Details of alternative functions are given in section 6.2. Alternative Function Assignments.

The block diagram for an individual GPIO pin is given below :

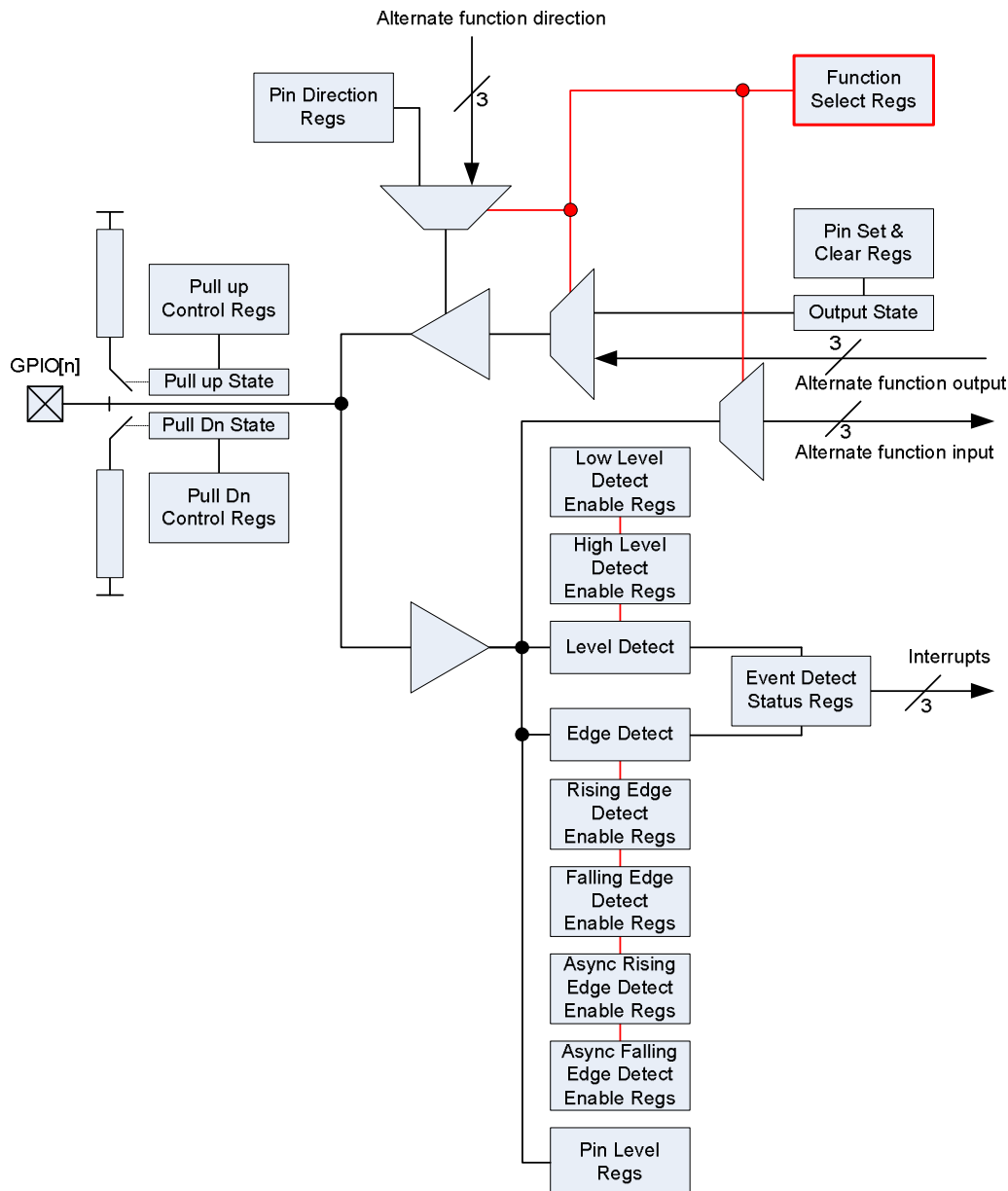


Figure 6-1 GPIO Block Diagram



BCM2835 ARM Peripherals

The GPIO peripheral has three dedicated interrupt lines. These lines are triggered by the setting of bits in the event detect status register. Each bank has its' own interrupt line with the third line shared between all bits.

The Alternate function table also has the pull state (pull-up/pull-down) which is applied after a power down.

6.1 Register View

The GPIO has 41 registers. All accesses are assumed to be 32-bit.

| Address | Field Name | Description | Size | Read/Write |
|--------------|------------|---------------------------------------|------|------------|
| 0x 7E20 0000 | GPFSEL0 | GPIO Function Select 0 | 32 | R/W |
| 0x 7E20 0000 | GPFSEL0 | GPIO Function Select 0 | 32 | R/W |
| 0x 7E20 0004 | GPFSEL1 | GPIO Function Select 1 | 32 | R/W |
| 0x 7E20 0008 | GPFSEL2 | GPIO Function Select 2 | 32 | R/W |
| 0x 7E20 000C | GPFSEL3 | GPIO Function Select 3 | 32 | R/W |
| 0x 7E20 0010 | GPFSEL4 | GPIO Function Select 4 | 32 | R/W |
| 0x 7E20 0014 | GPFSEL5 | GPIO Function Select 5 | 32 | R/W |
| 0x 7E20 0018 | - | Reserved | - | - |
| 0x 7E20 001C | GPSET0 | GPIO Pin Output Set 0 | 32 | W |
| 0x 7E20 0020 | GPSET1 | GPIO Pin Output Set 1 | 32 | W |
| 0x 7E20 0024 | - | Reserved | - | - |
| 0x 7E20 0028 | GPCLR0 | GPIO Pin Output Clear 0 | 32 | W |
| 0x 7E20 002C | GPCLR1 | GPIO Pin Output Clear 1 | 32 | W |
| 0x 7E20 0030 | - | Reserved | - | - |
| 0x 7E20 0034 | GPLEV0 | GPIO Pin Level 0 | 32 | R |
| 0x 7E20 0038 | GPLEV1 | GPIO Pin Level 1 | 32 | R |
| 0x 7E20 003C | - | Reserved | - | - |
| 0x 7E20 0040 | GPEDS0 | GPIO Pin Event Detect Status 0 | 32 | R/W |
| 0x 7E20 0044 | GPEDS1 | GPIO Pin Event Detect Status 1 | 32 | R/W |
| 0x 7E20 0048 | - | Reserved | - | - |
| 0x 7E20 004C | GPREN0 | GPIO Pin Rising Edge Detect Enable 0 | 32 | R/W |
| 0x 7E20 0050 | GPREN1 | GPIO Pin Rising Edge Detect Enable 1 | 32 | R/W |
| 0x 7E20 0054 | - | Reserved | - | - |
| 0x 7E20 0058 | GPFEN0 | GPIO Pin Falling Edge Detect Enable 0 | 32 | R/W |
| 0x 7E20 005C | GPFEN1 | GPIO Pin Falling Edge Detect Enable 1 | 32 | R/W |



BCM2835 ARM Peripherals

| Address | Field Name | Description | Size | Read/Write |
|--------------|------------|---------------------------------------|------|------------|
| 0x 7E20 0060 | - | Reserved | - | - |
| 0x 7E20 0064 | GPHEN0 | GPIO Pin High Detect Enable 0 | 32 | R/W |
| 0x 7E20 0068 | GPHEN1 | GPIO Pin High Detect Enable 1 | 32 | R/W |
| 0x 7E20 006C | - | Reserved | - | - |
| 0x 7E20 0070 | GPLEN0 | GPIO Pin Low Detect Enable 0 | 32 | R/W |
| 0x 7E20 0074 | GPLEN1 | GPIO Pin Low Detect Enable 1 | 32 | R/W |
| 0x 7E20 0078 | - | Reserved | - | - |
| 0x 7E20 007C | GPAREN0 | GPIO Pin Async. Rising Edge Detect 0 | 32 | R/W |
| 0x 7E20 0080 | GPAREN1 | GPIO Pin Async. Rising Edge Detect 1 | 32 | R/W |
| 0x 7E20 0084 | - | Reserved | - | - |
| 0x 7E20 0088 | GPAFEN0 | GPIO Pin Async. Falling Edge Detect 0 | 32 | R/W |
| 0x 7E20 008C | GPAFEN1 | GPIO Pin Async. Falling Edge Detect 1 | 32 | R/W |
| 0x 7E20 0090 | - | Reserved | - | - |
| 0x 7E20 0094 | GPPUD | GPIO Pin Pull-up/down Enable | 32 | R/W |
| 0x 7E20 0098 | GPPUDCLK0 | GPIO Pin Pull-up/down Enable Clock 0 | 32 | R/W |
| 0x 7E20 009C | GPPUDCLK1 | GPIO Pin Pull-up/down Enable Clock 1 | 32 | R/W |
| 0x 7E20 00A0 | - | Reserved | - | - |
| 0x 7E20 00B0 | - | Test | 4 | R/W |

Table 6-1 GPIO Register Assignment

GPIO Function Select Registers (GPFSELn)

SYNOPSIS The function select registers are used to define the operation of the general-purpose I/O pins. Each of the 54 GPIO pins has at least two alternative functions as defined in section 16.2. The FSEL{n} field determines the functionality of the nth GPIO pin. All unused alternative function lines are tied to ground and will output a "0" if selected. All pins reset to normal GPIO input operation.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
| 31-30 | --- | Reserved | R | 0 |



BCM2835 ARM Peripherals

| | | | | |
|-------|-------|--|-----|---|
| 29-27 | FSEL9 | FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO Pin 9 takes alternate function 3 011 = GPIO Pin 9 takes alternate function 4 010 = GPIO Pin 9 takes alternate function 5 | R/W | 0 |
| 26-24 | FSEL8 | FSEL8 - Function Select 8 | R/W | 0 |
| 23-21 | FSEL7 | FSEL7 - Function Select 7 | R/W | 0 |
| 20-18 | FSEL6 | FSEL6 - Function Select 6 | R/W | 0 |
| 17-15 | FSEL5 | FSEL5 - Function Select 5 | R/W | 0 |
| 14-12 | FSEL4 | FSEL4 - Function Select 4 | R/W | 0 |
| 11-9 | FSEL3 | FSEL3 - Function Select 3 | R/W | 0 |
| 8-6 | FSEL2 | FSEL2 - Function Select 2 | R/W | 0 |
| 5-3 | FSEL1 | FSEL1 - Function Select 1 | R/W | 0 |
| 2-0 | FSEL0 | FSEL0 - Function Select 0 | R/W | 0 |

Table 6-2 – GPIO Alternate function select register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31-30 | --- | Reserved | R | 0 |
| 29-27 | FSEL19 | FSEL19 - Function Select 19 000 = GPIO Pin 19 is an input 001 = GPIO Pin 19 is an output 100 = GPIO Pin 19 takes alternate function 0 101 = GPIO Pin 19 takes alternate function 1 110 = GPIO Pin 19 takes alternate function 2 111 = GPIO Pin 19 takes alternate function 3 011 = GPIO Pin 19 takes alternate function 4 010 = GPIO Pin 19 takes alternate function 5 | R/W | 0 |
| 26-24 | FSEL18 | FSEL18 - Function Select 18 | R/W | 0 |
| 23-21 | FSEL17 | FSEL17 - Function Select 17 | R/W | 0 |
| 20-18 | FSEL16 | FSEL16 - Function Select 16 | R/W | 0 |
| 17-15 | FSEL15 | FSEL15 - Function Select 15 | R/W | 0 |
| 14-12 | FSEL14 | FSEL14 - Function Select 14 | R/W | 0 |
| 11-9 | FSEL13 | FSEL13 - Function Select 13 | R/W | 0 |
| 8-6 | FSEL12 | FSEL12 - Function Select 12 | R/W | 0 |
| 5-3 | FSEL11 | FSEL11 - Function Select 11 | R/W | 0 |
| 2-0 | FSEL10 | FSEL10 - Function Select 10 | R/W | 0 |

Table 6-3 – GPIO Alternate function select register 1



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31-30 | --- | Reserved | R | 0 |
| 29-27 | FSEL29 | <u>FSEL29 - Function Select 29</u> 000 = GPIO Pin 29 is an input 001 = GPIO Pin 29 is an output 100 = GPIO Pin 29 takes alternate function 0 101 = GPIO Pin 29 takes alternate function 1 110 = GPIO Pin 29 takes alternate function 2 111 = GPIO Pin 29 takes alternate function 3 011 = GPIO Pin 29 takes alternate function 4 010 = GPIO Pin 29 takes alternate function 5 | R/W | 0 |
| 26-24 | FSEL28 | FSEL28 - Function Select 28 | R/W | 0 |
| 23-21 | FSEL27 | FSEL27 - Function Select 27 | R/W | 0 |
| 20-18 | FSEL26 | FSEL26 - Function Select 26 | R/W | 0 |
| 17-15 | FSEL25 | FSEL25 - Function Select 25 | R/W | 0 |
| 14-12 | FSEL24 | FSEL24 - Function Select 24 | R/W | 0 |
| 11-9 | FSEL23 | FSEL23 - Function Select 23 | R/W | 0 |
| 8-6 | FSEL22 | FSEL22 - Function Select 22 | R/W | 0 |
| 5-3 | FSEL21 | FSEL21 - Function Select 21 | R/W | 0 |
| 2-0 | FSEL20 | FSEL20 - Function Select 20 | R/W | 0 |

Table 6-4 – GPIO Alternate function select register 2

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31-30 | --- | Reserved | R | 0 |
| 29-27 | FSEL39 | <u>FSEL39 - Function Select 39</u> 000 = GPIO Pin 39 is an input 001 = GPIO Pin 39 is an output 100 = GPIO Pin 39 takes alternate function 0 101 = GPIO Pin 39 takes alternate function 1 110 = GPIO Pin 39 takes alternate function 2 111 = GPIO Pin 39 takes alternate function 3 011 = GPIO Pin 39 takes alternate function 4 010 = GPIO Pin 39 takes alternate function 5 | R/W | 0 |
| 26-24 | FSEL38 | FSEL38 - Function Select 38 | R/W | 0 |
| 23-21 | FSEL37 | FSEL37 - Function Select 37 | R/W | 0 |
| 20-18 | FSEL36 | FSEL36 - Function Select 36 | R/W | 0 |
| 17-15 | FSEL35 | FSEL35 - Function Select 35 | R/W | 0 |
| 14-12 | FSEL34 | FSEL34 - Function Select 34 | R/W | 0 |
| 11-9 | FSEL33 | FSEL33 - Function Select 33 | R/W | 0 |
| 8-6 | FSEL32 | FSEL32 - Function Select 32 | R/W | 0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|--------|-----------------------------|-----|---|
| 5-3 | FSEL31 | FSEL31 - Function Select 31 | R/W | 0 |
| 2-0 | FSEL30 | FSEL30 - Function Select 30 | R/W | 0 |

Table 6-5 – GPIO Alternate function select register 3

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31-30 | --- | Reserved | R | 0 |
| 29-27 | FSEL49 | <u>FSEL49 - Function Select 49</u> 000 = GPIO Pin 49 is an input 001 = GPIO Pin 49 is an output 100 = GPIO Pin 49 takes alternate function 0 101 = GPIO Pin 49 takes alternate function 1 110 = GPIO Pin 49 takes alternate function 2 111 = GPIO Pin 49 takes alternate function 3 011 = GPIO Pin 49 takes alternate function 4 010 = GPIO Pin 49 takes alternate function 5 | R/W | 0 |
| 26-24 | FSEL48 | FSEL48 - Function Select 48 | R/W | 0 |
| 23-21 | FSEL47 | FSEL47 - Function Select 47 | R/W | 0 |
| 20-18 | FSEL46 | FSEL46 - Function Select 46 | R/W | 0 |
| 17-15 | FSEL45 | FSEL45 - Function Select 45 | R/W | 0 |
| 14-12 | FSEL44 | FSEL44 - Function Select 44 | R/W | 0 |
| 11-9 | FSEL43 | FSEL43 - Function Select 43 | R/W | 0 |
| 8-6 | FSEL42 | FSEL42 - Function Select 42 | R/W | 0 |
| 5-3 | FSEL41 | FSEL41 - Function Select 41 | R/W | 0 |
| 2-0 | FSEL40 | FSEL40 - Function Select 40 | R/W | 0 |

Table 6-6 – GPIO Alternate function select register 4

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31-12 | --- | Reserved | R | 0 |
| 11-9 | FSEL53 | <u>FSEL53 - Function Select 53</u> 000 = GPIO Pin 53 is an input 001 = GPIO Pin 53 is an output 100 = GPIO Pin 53 takes alternate function 0 101 = GPIO Pin 53 takes alternate function 1 110 = GPIO Pin 53 takes alternate function 2 111 = GPIO Pin 53 takes alternate function 3 011 = GPIO Pin 53 takes alternate function 4 010 = GPIO Pin 53 takes alternate function 5 | R/W | 0 |
| 8-6 | FSEL52 | FSEL52 - Function Select 52 | R/W | 0 |
| 5-3 | FSEL51 | FSEL51 - Function Select 51 | R/W | 0 |
| 2-0 | FSEL50 | FSEL50 - Function Select 50 | R/W | 0 |



Table 6-7 – GPIO Alternate function select register 5

GPIO Pin Output Set Registers (GPSETn)

SYNOPSIS The output set registers are used to set a GPIO pin. The SET{n} field defines the respective GPIO pin to set, writing a “0” to the field has no effect. If the GPIO pin is being used as in input (by default) then the value in the SET{n} field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31-0 | SETn (n=0..31) | 0 = No effect 1 = Set GPIO pin <i>n</i> | R/W | 0 |

Table 6-8 – GPIO Output Set Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|--------------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | SETn (n=32..53) | 0 = No effect 1 = Set GPIO pin <i>n</i> . | R/W | 0 |

Table 6-9 – GPIO Output Set Register 1

GPIO Pin Output Clear Registers (GPCLRn)

SYNOPSIS The output clear registers) are used to clear a GPIO pin. The CLR{n} field defines the respective GPIO pin to clear, writing a “0” to the field has no effect. If the GPIO pin is being used as in input (by default) then the value in the CLR{n} field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31-0 | CLRn (n=0..31) | 0 = No effect 1 = Clear GPIO pin <i>n</i> | R/W | 0 |

Table 6-10 – GPIO Output Clear Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
| 31-22 | - | Reserved | R | 0 |



| | | | | |
|------|--------------------------------|--|-----|---|
| 21-0 | CLR _n (n=32..53) | 0 = No effect 1 = Set GPIO pin <i>n</i> | R/W | 0 |
|------|--------------------------------|--|-----|---|

Table 6-11 – GPIO Output Clear Register 1

GPIO Pin Level Registers (GPLEV_n)

SYNOPSIS The pin level registers return the actual value of the pin. The LEV{*n*} field gives the value of the respective GPIO pin.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------------------|---|------|-------|
| 31-0 | LEV _n (n=0..31) | 0 = GPIO pin <i>n</i> is low 1 = GPIO pin <i>n</i> is high | R/W | 0 |

Table 6-12 – GPIO Level Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------------------|---|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | LEV _n (n=32..53) | 0 = GPIO pin <i>n</i> is low 1 = GPIO pin <i>n</i> is high | R/W | 0 |

Table 6-13 – GPIO Level Register 1

GPIO Event Detect Status Registers (GPEDS_n)

SYNOPSIS The event detect status registers are used to record level and edge events on the GPIO pins. The relevant bit in the event detect status registers is set whenever: 1) an edge is detected that matches the type of edge programmed in the rising/falling edge detect enable registers, or 2) a level is detected that matches the type of level programmed in the high/low level detect enable registers. The bit is cleared by writing a “1” to the relevant bit.

The interrupt controller can be programmed to interrupt the processor when any of the status bits are set. The GPIO peripheral has three dedicated interrupt lines. Each GPIO bank can generate an independent interrupt. The third line generates a single interrupt whenever any bit is set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------------------|--|------|-------|
| 31-0 | EDS _n (n=0..31) | 0 = Event not detected on GPIO pin <i>n</i> 1 = Event detected on GPIO pin <i>n</i> | R/W | 0 |

Table 6-14 – GPIO Event Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|--------------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | EDSn (n=32..53) | 0 = Event not detected on GPIO pin <i>n</i> 1 = Event detected on GPIO pin <i>n</i> | R/W | 0 |

Table 6-15 – GPIO Event Detect Status Register 1

GPIO Rising Edge Detect Enable Registers (GPRENn)

SYNOPSIS The rising edge detect enable registers define the pins for which a rising edge transition sets a bit in the event detect status registers (GPEDSn). When the relevant bits are set in both the GPRENn and GPFENn registers, any transition (1 to 0 and 0 to 1) will set a bit in the GPEDSn registers. The GPRENn registers use synchronous edge detection. This means the input signal is sampled using the system clock and then it is looking for a “011” pattern on the sampled signal. This has the effect of suppressing glitches.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31-0 | RENn (n=0..31) | 0 = Rising edge detect disabled on GPIO pin <i>n</i> . 1 = Rising edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-16 – GPIO Rising Edge Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|--------------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | RENn (n=32..53) | 0 = Rising edge detect disabled on GPIO pin <i>n</i> . 1 = Rising edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-17 – GPIO Rising Edge Detect Status Register 1

GPIO Falling Edge Detect Enable Registers (GPRENn)

SYNOPSIS The falling edge detect enable registers define the pins for which a falling edge transition sets a bit in the event detect status registers (GPEDSn). When the relevant bits are set in both the GPRENn and GPFENn registers, any transition (1 to 0 and 0 to 1) will set a bit in the GPEDSn registers. The GPFENn registers use synchronous edge detection. This means the input signal is sampled using the system clock and then it is looking for a “100” pattern on the sampled signal. This has the effect of suppressing glitches.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31-0 | FENn (n=0..31) | 0 = Falling edge detect disabled on GPIO pin <i>n</i> . 1 = Falling edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-18 – GPIO Falling Edge Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | FENn (n=32..53) | 0 = Falling edge detect disabled on GPIO pin <i>n</i> . 1 = Falling edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-19 – GPIO Falling Edge Detect Status Register 1

GPIO High Detect Enable Registers (GPHENn)

SYNOPSIS The high level detect enable registers define the pins for which a high level sets a bit in the event detect status register (GPEDSn). If the pin is still high when an attempt is made to clear the status bit in GPEDSn then the status bit will remain set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31-0 | HENn (n=0..31) | 0 = High detect disabled on GPIO pin <i>n</i> 1 = High on GPIO pin <i>n</i> sets corresponding bit in GPEDS | R/W | 0 |

Table 6-20 – GPIO High Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | HENn (n=32..53) | 0 = High detect disabled on GPIO pin <i>n</i> 1 = High on GPIO pin <i>n</i> sets corresponding bit in GPEDS | R/W | 0 |

Table 6-21 – GPIO High Detect Status Register 1



GPIO Low Detect Enable Registers (GPLENn)

SYNOPSIS The low level detect enable registers define the pins for which a low level sets a bit in the event detect status register (GPEDSn). If the pin is still low when an attempt is made to clear the status bit in GPEDSn then the status bit will remain set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------|--|------|-------|
| 31-0 | LENn (n=0..31) | 0 = Low detect disabled on GPIO pin <i>n</i> 1 = Low on GPIO pin <i>n</i> sets corresponding bit in GPEDS | R/W | 0 |

Table 6-22 – GPIO Low Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | LENn (n=32..53) | 0 = Low detect disabled on GPIO pin <i>n</i> 1 = Low on GPIO pin <i>n</i> sets corresponding bit in GPEDS | R/W | 0 |

Table 6-23 – GPIO Low Detect Status Register 1

GPIO Asynchronous rising Edge Detect Enable Registers (GPARENn)

SYNOPSIS The asynchronous rising edge detect enable registers define the pins for which a asynchronous rising edge transition sets a bit in the event detect status registers (GPEDSn).

Asynchronous means the incoming signal is not sampled by the system clock. As such rising edges of very short duration can be detected.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------|--|------|-------|
| 31-0 | ARENn (n=0..31) | 0 = Asynchronous rising edge detect disabled on GPIO pin <i>n</i> . 1 = Asynchronous rising edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-24 – GPIO Asynchronous rising Edge Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | ARENn (n=32..53) | 0 = Asynchronous rising edge detect disabled on GPIO pin <i>n</i> . 1 = Asynchronous rising edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |



Table 6-25 – GPIO Asynchronous rising Edge Detect Status Register 1

GPIO Asynchronous Falling Edge Detect Enable Registers (GPAFENn)

SYNOPSIS The asynchronous falling edge detect enable registers define the pins for which a asynchronous falling edge transition sets a bit in the event detect status registers (GPEDSn). Asynchronous means the incoming signal is not sampled by the system clock. As such falling edges of very short duration can be detected.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-----------------|--|------|-------|
| 31-0 | AFENn (n=0..31) | 0 = Asynchronous falling edge detect disabled on GPIO pin <i>n</i> . 1 = Asynchronous falling edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-26 – GPIO Asynchronous Falling Edge Detect Status Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------------|--|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | AFENn (n=32..53) | 0 = Asynchronous falling edge detect disabled on GPIO pin <i>n</i> . 1 = Asynchronous falling edge on GPIO pin <i>n</i> sets corresponding bit in EDSn. | R/W | 0 |

Table 6-27 – GPIO Asynchronous Falling Edge Detect Status Register 1

GPIO Pull-up/down Register (GPPUD)

SYNOPSIS The GPIO Pull-up/down Register controls the actuation of the internal pull-up/down control line to ALL the GPIO pins. This register must be used in conjunction with the 2 GPPUDCLKn registers.

Note that it is not possible to read back the current Pull-up/down settings and so it is the users' responsibility to 'remember' which pull-up/downs are active. The reason for this is that GPIO pull-ups are maintained even in power-down mode when the core is off, when all register contents is lost.

The Alternate function table also has the pull state which is applied after a power down.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
|--------|------------|-------------|------|-------|



BCM2835 ARM Peripherals

| | | | | |
|------|-----|--|-----|---|
| 31-2 | --- | Unused | R | 0 |
| 1-0 | PUD | <u>PUD - GPIO Pin Pull-up/down</u> 00 = Off – disable pull-up/down 01 = Enable Pull Down control 10 = Enable Pull Up control 11 = Reserved *Use in conjunction with GPPUDCLK0/1/2 | R/W | 0 |

Table 6-28 – GPIO Pull-up/down Register (GPPUD)

GPIO Pull-up/down Clock Registers (GPPUDCLKn)

SYNOPSIS

The GPIO Pull-up/down Clock Registers control the actuation of internal pull-downs on the respective GPIO pins. These registers must be used in conjunction with the GPPUD register to effect GPIO Pull-up/down changes. The following sequence of events is required:

1. Write to GPPUD to set the required control signal (i.e. Pull-up or Pull-Down or neither to remove the current Pull-up/down)
2. Wait 150 cycles – this provides the required set-up time for the control signal
3. Write to GPPUDCLK0/1 to clock the control signal into the GPIO pads you wish to modify – NOTE only the pads which receive a clock will be modified, all others will retain their previous state.
4. Wait 150 cycles – this provides the required hold time for the control signal
5. Write to GPPUD to remove the control signal
6. Write to GPPUDCLK0/1 to remove the clock

| Bit(s) | Field Name | Description | Type | Reset |
|--------|----------------------|---|------|-------|
| (31-0) | PUDCLKn (n=0..31) | 0 = No Effect 1 = Assert Clock on line (<i>n</i>) *Must be used in conjunction with GPPUD | R/W | 0 |

Table 6-29 – GPIO Pull-up/down Clock Register 0

| Bit(s) | Field Name | Description | Type | Reset |
|--------|--------------------|---|------|-------|
| 31-22 | - | Reserved | R | 0 |
| 21-0 | PUDCLKn (n=32..53) | 0 = No Effect 1 = Assert Clock on line (<i>n</i>) *Must be used in conjunction with GPPUD | R/W | 0 |

Table 6-30 – GPIO Pull-up/down Clock Register 1



BCM2835 ARM Peripherals

6.2 Alternative Function Assignments

Every GPIO pin can carry an alternate function. Up to 6 alternate function are available but not every pin has that many alternate functions. The table below gives a quick over view.

| | Pull | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 |
|--------|------|------------|---------------|------------|------------------|------------|----------|
| GPIO0 | High | SDA0 | SA5 | <reserved> | | | |
| GPIO1 | High | SCL0 | SA4 | <reserved> | | | |
| GPIO2 | High | SDA1 | SA3 | <reserved> | | | |
| GPIO3 | High | SCL1 | SA2 | <reserved> | | | |
| GPIO4 | High | GPCLK0 | SA1 | <reserved> | | | ARM_TDI |
| GPIO5 | High | GPCLK1 | SA0 | <reserved> | | | ARM_TDO |
| GPIO6 | High | GPCLK2 | SOE_N / SE | <reserved> | | | ARM_RTCK |
| GPIO7 | High | SPI0_CE1_N | SWE_N / SBW_N | <reserved> | | | |
| GPIO8 | High | SPI0_CE0_N | SD0 | <reserved> | | | |
| GPIO9 | Low | SPI0_MISO | SD1 | <reserved> | | | |
| GPIO10 | Low | SPI0_MOSI | SD2 | <reserved> | | | |
| GPIO11 | Low | SPI0_SCLK | SD3 | <reserved> | | | |
| GPIO12 | Low | PWM0 | SD4 | <reserved> | | | ARM_TMS |
| GPIO13 | Low | PWM1 | SD5 | <reserved> | | | ARM_TCK |
| GPIO14 | Low | TXD0 | SD6 | <reserved> | | | TXD1 |
| GPIO15 | Low | RXD0 | SD7 | <reserved> | | | RXD1 |
| GPIO16 | Low | <reserved> | SD8 | <reserved> | CTS0 | SPI1_CE2_N | CTS1 |
| GPIO17 | Low | <reserved> | SD9 | <reserved> | RTS0 | SPI1_CE1_N | RTS1 |
| GPIO18 | Low | PCM_CLK | SD10 | <reserved> | BSCSL SDA / MOSI | SPI1_CE0_N | PWM0 |
| GPIO19 | Low | PCM_FS | SD11 | <reserved> | BSCSL SCL / SCLK | SPI1_MISO | PWM1 |
| GPIO20 | Low | PCM_DIN | SD12 | <reserved> | BSCSL / MISO | SPI1_MOSI | GPCLK0 |
| GPIO21 | Low | PCM_DOUT | SD13 | <reserved> | BSCSL / CE_N | SPI1_SCLK | GPCLK1 |
| GPIO22 | Low | <reserved> | SD14 | <reserved> | SD1_CLK | ARM_TRST | |
| GPIO23 | Low | <reserved> | SD15 | <reserved> | SD1_CMD | ARM_RTCK | |
| GPIO24 | Low | <reserved> | SD16 | <reserved> | SD1_DAT0 | ARM_TDO | |
| GPIO25 | Low | <reserved> | SD17 | <reserved> | SD1_DAT1 | ARM_TCK | |
| GPIO26 | Low | <reserved> | <reserved> | <reserved> | SD1_DAT2 | ARM_TDI | |
| GPIO27 | Low | <reserved> | <reserved> | <reserved> | SD1_DAT3 | ARM_TMS | |
| GPIO28 | - | SDA0 | SA5 | PCM_CLK | <reserved> | | |
| GPIO29 | - | SCL0 | SA4 | PCM_FS | <reserved> | | |
| GPIO30 | Low | <reserved> | SA3 | PCM_DIN | CTS0 | | CTS1 |
| GPIO31 | Low | <reserved> | SA2 | PCM_DOUT | RTS0 | | RTS1 |
| GPIO32 | Low | GPCLK0 | SA1 | <reserved> | TXD0 | | TXD1 |
| GPIO33 | Low | <reserved> | SA0 | <reserved> | RXD0 | | RXD1 |
| GPIO34 | High | GPCLK0 | SOE_N / SE | <reserved> | <reserved> | | |
| GPIO35 | High | SPI0_CE1_N | SWE_N / SBW_N | | <reserved> | | |
| GPIO36 | High | SPI0_CE0_N | SD0 | TXD0 | <reserved> | | |
| GPIO37 | Low | SPI0_MISO | SD1 | RXD0 | <reserved> | | |
| GPIO38 | Low | SPI0_MOSI | SD2 | RTS0 | <reserved> | | |
| GPIO39 | Low | SPI0_SCLK | SD3 | CTS0 | <reserved> | | |
| GPIO40 | Low | PWM0 | SD4 | | <reserved> | SPI2_MISO | TXD1 |
| | Pull | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 |



BCM2835 ARM Peripherals

| | | | | | | | |
|--------|------|------------|------|------------|------------|------------|------|
| GPIO41 | Low | PWM1 | SD5 | <reserved> | <reserved> | SPI2_MOSI | RXD1 |
| GPIO42 | Low | GPCLK1 | SD6 | <reserved> | <reserved> | SPI2_SCLK | RTS1 |
| GPIO43 | Low | GPCLK2 | SD7 | <reserved> | <reserved> | SPI2_CE0_N | CTS1 |
| GPIO44 | - | GPCLK1 | SDA0 | SDA1 | <reserved> | SPI2_CE1_N | |
| GPIO45 | - | PWM1 | SCL0 | SCL1 | <reserved> | SPI2_CE2_N | |
| GPIO46 | High | <Internal> | | | | | |
| GPIO47 | High | <Internal> | | | | | |
| GPIO48 | High | <Internal> | | | | | |
| GPIO49 | High | <Internal> | | | | | |
| GPIO50 | High | <Internal> | | | | | |
| GPIO51 | High | <Internal> | | | | | |
| GPIO52 | High | <Internal> | | | | | |
| GPIO53 | High | <Internal> | | | | | |

Table 6-31 GPIO Pins Alternative Function Assignment

Entries which are white should **not** be used. These may have unexpected results as some of these have special functions used in test mode. e.g. they may drive the output with high frequency signals.

Special function legend:

| Name | Function | See section |
|------------------|-------------------------------------|----------------------------|
| SDA0 | BSC ⁶ master 0 data line | BSC |
| SCL0 | BSC master 0 clock line | BSC |
| SDA1 | BSC master 1 data line | BSC |
| SCL1 | BSC master 1 clock line | BSC |
| GPCLK0 | General purpose Clock 0 | <TBD> |
| GPCLK1 | General purpose Clock 1 | <TBD> |
| GPCLK2 | General purpose Clock 2 | <TBD> |
| SPI0_CE1_N | SPI0 Chip select 1 | SPI |
| SPI0_CE0_N | SPI0 Chip select 0 | SPI |
| SPI0_MISO | SPI0 MISO | SPI |
| SPI0_MOSI | SPI0 MOSI | SPI |
| SPI0_SCLK | SPI0 Serial clock | SPI |
| PWMx | Pulse Width Modulator 0..1 | Pulse Width Modulator |
| TXD0 | UART 0 Transmit Data | UART |
| RXD0 | UART 0 Receive Data | UART |
| CTS0 | UART 0 Clear To Send | UART |
| RTS0 | UART 0 Request To Send | UART |
| PCM_CLK | PCM clock | PCM Audio |
| PCM_FS | PCM Frame Sync | PCM Audio |
| PCM_DIN | PCM Data in | PCM Audio |
| PCM_DOUT | PCM data out | PCM Audio |
| SAx | Secondary mem Address bus | Secondary Memory Interface |
| SOE_N / SE | Secondary mem. Controls | Secondary Memory Interface |
| SWE_N / SRW_N | Secondary mem. Controls | Secondary Memory Interface |
| SDx | Secondary mem. data bus | Secondary Memory Interface |
| BSCSL SDA / MOSI | BSC slave Data, SPI slave MOSI | BSC ISP slave |
| BSCSL SCL / SCLK | BSC slave Clock, SPI slave clock | BSC ISP slave |
| BSCSL - / MISO | BSC <not used>, SPI MISO | BSC ISP slave |
| BSCSL - / CE_N | BSC <not used>, SPI CSn | BSC ISP slave |

⁶ The Broadcom Serial Control bus is a proprietary bus compliant with the Philips[®] I2C bus/interface



BCM2835 ARM Peripherals

| Name | Function | See section |
|------------|------------------------|---------------|
| SPI1_CEx_N | SPI1 Chip select 0-2 | Auxiliary I/O |
| SPI1_MISO | SPI1 MISO | Auxiliary I/O |
| SPI1_MOSI | SPI1 MOSI | Auxiliary I/O |
| SPI1_SCLK | SPI1 Serial clock | Auxiliary I/O |
| TXD0 | UART 1 Transmit Data | Auxiliary I/O |
| RXD0 | UART 1 Receive Data | Auxiliary I/O |
| CTS0 | UART 1 Clear To Send | Auxiliary I/O |
| RTS0 | UART 1 Request To Send | Auxiliary I/O |
| SPI2_CEx_N | SPI2 Chip select 0-2 | Auxiliary I/O |
| SPI2_MISO | SPI2 MISO | Auxiliary I/O |
| SPI2_MOSI | SPI2 MOSI | Auxiliary I/O |
| SPI2_SCLK | SPI2 Serial clock | Auxiliary I/O |
| ARM_TRST | ARM JTAG reset | <TBD> |
| ARM_RTCK | ARM JTAG return clock | <TBD> |
| ARM_TDO | ARM JTAG Data out | <TBD> |
| ARM_TCK | ARM JTAG Clock | <TBD> |
| ARM_TDI | ARM JTAG Data in | <TBD> |
| ARM_TMS | ARM JTAG Mode select | <TBD> |

6.3 General Purpose GPIO Clocks

The General Purpose clocks can be output to GPIO pins. They run from the peripherals clock sources and use clock generators with noise-shaping MASH dividers. These allow the GPIO clocks to be used to drive audio devices.

The fractional divider operates by periodically dropping source clock pulses, therefore the output frequency will periodically switch between:

$$\frac{\text{source_frequency}}{\text{DIVI}} \quad \& \quad \frac{\text{source_frequency}}{\text{DIVI} + 1}$$

Jitter is therefore reduced by increasing the source clock frequency. In applications where jitter is a concern, the fastest available clock source should be used.

The General Purpose clocks have MASH noise-shaping dividers which push this fractional divider jitter out of the audio band.

MASH noise-shaping is incorporated to push the fractional divider jitter out of the audio band if required. The MASH can be programmed for 1, 2 or 3-stage filtering. MASH filter, the frequency is spread around the requested frequency and the user must ensure that the module is not exposed to frequencies higher than 25MHz. Also, the MASH filter imposes a low limit on the range of DIVI.

| MASH | min DIVI | min output freq | average output freq | max output freq |
|----------------|----------|-----------------------|---------------------------------|-----------------------|
| 0 (int divide) | 1 | source / (DIVI) | source / (DIVI) | source / (DIVI) |
| 1 | 2 | source / (DIVI) | source / (DIVI + DIVF / 1024) | source / (DIVI + 1) |
| 2 | 3 | source / (DIVI - 1) | source / (DIVI + DIVF / 1024) | source / (DIVI + 2) |
| 3 | 5 | source / (DIVI - 3) | source / (DIVI + DIVF / 1024) | source / (DIVI + 4) |

Table 6-32 Effect of MASH Filter on Frequency

The following example illustrates the spreading of output clock frequency resulting from the use of the MASH filter. Note that the spread is greater for lower divisors.

| PLL freq (MHz) | target freq (MHz) | MASH | divisor | DIVI | DIVF | min freq (MHz) | ave freq (MHz) | max freq (MHz) | error |
|----------------|-------------------|------|---------|------|------|----------------|----------------|----------------|-------|
| 650 | 18.32 | 0 | 35.480 | 35 | 492 | 18.57 | 18.57 | 18.57 | ok |
| 650 | 18.32 | 1 | 35.480 | 35 | 492 | 18.06 | 18.32 | 18.57 | ok |
| 650 | 18.32 | 2 | 35.480 | 35 | 492 | 17.57 | 18.32 | 19.12 | ok |
| 650 | 18.32 | 3 | 35.480 | 35 | 492 | 16.67 | 18.32 | 20.31 | ok |
| 400 | 18.32 | 0 | 21.834 | 21 | 854 | 19.05 | 19.05 | 19.05 | ok |
| 400 | 18.32 | 1 | 21.834 | 21 | 854 | 18.18 | 18.32 | 19.05 | ok |
| 400 | 18.32 | 2 | 21.834 | 21 | 854 | 17.39 | 18.32 | 20.00 | ok |
| 400 | 18.32 | 3 | 21.834 | 21 | 854 | 16.00 | 18.32 | 22.22 | ok |
| 200 | 18.32 | 0 | 10.917 | 10 | 939 | 20.00 | 20.00 | 20.00 | ok |
| 200 | 18.32 | 1 | 10.917 | 10 | 939 | 18.18 | 18.32 | 20.00 | ok |
| 200 | 18.32 | 2 | 10.917 | 10 | 939 | 16.67 | 18.32 | 22.22 | ok |
| 200 | 18.32 | 3 | 10.917 | 10 | 939 | 14.29 | 18.32 | 28.57 | error |

Table 6-33 Example of Frequency Spread when using MASH Filtering

It is beyond the scope of this specification to describe the operation of a MASH filter or to determine under what conditions the available levels of filtering are beneficial.



BCM2835 ARM Peripherals

Operating Frequency

The maximum operating frequency of the General Purpose clocks is ~125MHz at 1.2V but this will be reduced if the GPIO pins are heavily loaded or have a capacitive load.



BCM2835 ARM Peripherals

Register Definitions

Clock Manager General Purpose Clocks Control (CM_GP0CTL, GP1CTL & GP2CTL)

Address 0x 7e10 1070 CM_GP0CTL
 0x 7e10 1078 CM_GP1CTL
 0x 7e10 1080 CM_GP2CTL

| Bit Number | Field Name | Description | Read/Write | Reset |
|------------|------------|---|------------|-------|
| 31-24 | PASSWD | Clock Manager password "5a" | W | 0 |
| 23-11 | - | Unused | R | 0 |
| 10-9 | MASH | <u>MASH control</u> 0 = integer division 1 = 1-stage MASH (equivalent to non-MASH dividers) 2 = 2-stage MASH 3 = 3-stage MASH To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB. | R/W | 0 |
| 8 | FLIP | <u>Invert the clock generator output</u> This is intended for use in test/debug only. Switching this control will generate an edge on the clock generator output. To avoid output glitches do not switch this control while BUSY=1. | R/W | 0 |
| 7 | BUSY | <u>Clock generator is running</u> Indicates the clock generator is running. To avoid glitches and lock-ups, clock sources and setups must not be changed while this flag is set. | R | 0 |
| 6 | - | Unused | R | 0 |
| 5 | KILL | <u>Kill the clock generator</u> 0 = no action 1 = stop and reset the clock generator This is intended for test/debug only. Using this control may cause a glitch on the clock generator output. | R/W | 0 |
| 4 | ENAB | <u>Enable the clock generator</u> This requests the clock to start or stop without glitches. The output clock will not stop immediately because the cycle must be allowed to complete to avoid glitches. The BUSY flag will go low when the final cycle is completed. | R/W | 0 |
| 3-0 | SRC | <u>Clock source</u> 0 = GND 1 = oscillator 2 = testdebug0 3 = testdebug1 4 = PLLA per 5 = PLLC per 6 = PLLD per 7 = HDMI auxiliary 8-15 = GND To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB. | R/W | 0 |



BCM2835 ARM Peripherals

Table 6-34 General Purpose Clocks Control

Clock Manager General Purpose Clock Divisors (CM_GP0DIV, CM_GP1DIV & CM_GP2DIV)

Address 0x 7e10 1074 CM_GP0DIV
 0x 7e10 107c CM_GP1DIV
 0x 7e10 1084 CM_GP2DIV

| Bit Number | Field Name | Description | Read/Write | Reset |
|------------|------------|--|------------|-------|
| 31-24 | PASSWD | Clock Manager password "5a" | W | 0 |
| 23-12 | DIVI | <u>Integer part of divisor</u> This value has a minimum limit determined by the MASH setting. See text for details. To avoid lock-ups and glitches do not change this control while BUSY=1. | R/W | 0 |
| 11-0 | DIVF | <u>Fractional part of divisor</u> To avoid lock-ups and glitches do not change this control while BUSY=1. | R/W | 0 |

Table 6-35 General Purpose Clock Divisors

7 Interrupts

7.1 Introduction

The ARM has two types of interrupt sources:

1. Interrupts coming from the GPU peripherals.
2. Interrupts coming from local ARM control peripherals.

The ARM processor gets three types of interrupts:

1. Interrupts from ARM specific peripherals.
2. Interrupts from GPU peripherals.
3. Special events interrupts.

The ARM specific interrupts are:

- One timer.
- One Mailbox.
- Two Doorbells.
- Two GPU halted interrupts.
- Two Address/access error interrupt

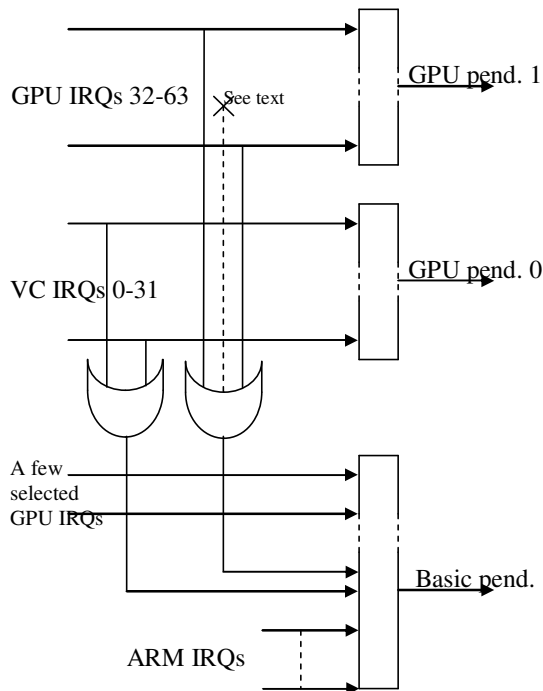
The Mailbox and Doorbell registers are not for general usage.

For each interrupt source (ARM or GPU) there is an interrupt enable bit (read/write) and an interrupt pending bit (Read Only). All interrupts generated by the arm control block are level sensitive interrupts. Thus all interrupts remain asserted until disabled or the interrupt source is cleared.

Default the interrupts from doorbell 0,1 and mailbox 0 go to the ARM this means that these resources should be written by the GPU and read by the ARM. The opposite holds for doorbells 2, 3 and mailbox 1.

7.2 Interrupt pending.

An interrupt vector module has NOT been implemented. To still have adequate interrupt processing the interrupt pending bits are organized as follows:



There are three interrupt pending registers.

One basic pending register and two GPU pending registers.

Basic pending register.

The basic pending register has interrupt pending bits for the ARM specific interrupts .

To speed up the interrupt processing it also has a number of selected GPU interrupts which are deemed most likely to be required in ARM drivers.

Further there are two special GPU pending bits which tell if any of the two other pending registers has bits set, one bit if a GPU interrupt 0-31 is pending, a second bit if a GPU interrupt 32-63 is pending. The 'selected GPU interrupts' on the basic pending registers are NOT taken into account for these two status bits. So the two pending 0,1 status bits tell you that 'there are more interrupt which you have not seen yet'.

GPU pending registers.

There are two GPU pending registers with one bit per GPU interrupt source.

7.3 Fast Interrupt (FIQ).

The ARM also supports a Fast Interrupt (FIQ). One interrupt sources can be selected to be connected to the ARM FIQ input. There is also one FIQ enable. An interrupt which is selected as FIQ should have its normal interrupt enable bit cleared. Otherwise an normal and a FIQ interrupt will be fired at the same time. Not a good idea!

7.4 Interrupt priority.

There is no priority for any interrupt. If one interrupt is much more important then all others it can be routed to the FIQ. Any remaining interrupts have to be processed by polling the pending



BCM2835 ARM Peripherals

registers. It is up to the ARM software to devise a strategy. e.g. First start looking for specific pending bits or process them all shifting one bit at a time.

As interrupt may arrive whilst this process is ongoing the usual care for any 'race-condition critical' code must be taken. The following ARM assembly code has been proven to work:

```
.macro get_irqnr_preamble, base, tmp
ldr \base, =IO_ADDRESS(ARMCTRL_IC_BASE)
.endm
```

```
.macro get_irqnr_and_base, irqnr, irqstat, base, tmp
ldr \irqstat, [\base, #(ARM_IRQ_PEND0 - ARMCTRL_IC_BASE)] @ get masked status
mov \irqnr, #(ARM_IRQ0_BASE + 31)
and \tmp, \irqstat, #0x300 @ save bits 8 and 9
bics \irqstat, \irqstat, #0x300 @ clear bits 8 and 9, and test
bne 1010f
```

```
tst \tmp, #0x100
ldrne \irqstat, [\base, #(ARM_IRQ_PEND1 - ARMCTRL_IC_BASE)]
movne \irqnr, #(ARM_IRQ1_BASE + 31)
@ Mask out the interrupts also present in PEND0 - see SW-5809
bicne \irqstat, #((1<<7) | (1<<9) | (1<<10))
bicne \irqstat, #((1<<18) | (1<<19))
bne 1010f
```

```
tst \tmp, #0x200
ldrne \irqstat, [\base, #(ARM_IRQ_PEND2 - ARMCTRL_IC_BASE)]
movne \irqnr, #(ARM_IRQ2_BASE + 31)
@ Mask out the interrupts also present in PEND0 - see SW-5809
bicne \irqstat, #((1<<21) | (1<<22) | (1<<23) | (1<<24) | (1<<25))
bicne \irqstat, #((1<<30))
beq 1020f
```

1010:

```
@ For non-zero x, LSB(x) = 31 - CLZ(x^(x-1))
@ N.B. CLZ is an ARM5 instruction.
sub \tmp, \irqstat, #1
eor \irqstat, \irqstat, \tmp
clz \tmp, \irqstat
sub \irqnr, \tmp
```

1020: @ EQ will be set if no irqs pending

7.5 Registers

The base address for the ARM interrupt register is 0x7E00B000.

Registers overview:

| Address offset ⁷ | Name | Notes |
|--------------------------------|--------------------|-------|
| 0x200 | IRQ basic pending | |
| 0x204 | IRQ pending 1 | |
| 0x208 | IRQ pending 2 | |
| 0x20C | FIQ control | |
| 0x210 | Enable IRQs 1 | |
| 0x214 | Enable IRQs 2 | |
| 0x218 | Enable Basic IRQs | |
| 0x21C | Disable IRQs 1 | |
| 0x220 | Disable IRQs 2 | |
| 0x224 | Disable Basic IRQs | |

The following is a table which lists all interrupts which can come from the peripherals which can be handled by the ARM.

⁷ This is the offset which needs to be added to the base address to get the full hardware address.



BCM2835 ARM Peripherals

ARM peripherals interrupts table.

| # | IRQ 0-15 | # | IRQ 16-31 | # | IRQ 32-47 | # | IRQ 48-63 |
|----|----------|----|-----------|----|-----------------|----|-------------|
| 0 | | 16 | | 32 | | 48 | smi |
| 1 | | 17 | | 33 | | 49 | gpio_int[0] |
| 2 | | 18 | | 34 | | 50 | gpio_int[1] |
| 3 | | 19 | | 35 | | 51 | gpio_int[2] |
| 4 | | 20 | | 36 | | 52 | gpio_int[3] |
| 5 | | 21 | | 37 | | 53 | i2c_int |
| 6 | | 22 | | 38 | | 54 | spi_int |
| 7 | | 23 | | 39 | | 55 | pcm_int |
| 8 | | 24 | | 40 | | 56 | |
| 9 | | 25 | | 41 | | 57 | uart_int |
| 10 | | 26 | | 42 | | 58 | |
| 11 | | 27 | | 43 | i2c_spi_slv_int | 59 | |
| 12 | | 28 | | 44 | | 60 | |
| 13 | | 29 | Aux int | 45 | pwa0 | 61 | |
| 14 | | 30 | | 46 | pwa1 | 62 | |
| 15 | | 31 | | 47 | | 63 | |

The table above has many empty entries. These should not be enabled as they will interfere with the GPU operation.

ARM peripherals interrupts table.

| | |
|---|---|
| 0 | ARM Timer |
| 1 | ARM Mailbox |
| 2 | ARM Doorbell 0 |
| 3 | ARM Doorbell 1 |
| 4 | GPU0 halted (Or GPU1 halted if bit 10 of control register 1 is set) |
| 5 | GPU1 halted |
| 6 | Illegal access type 1 |
| 7 | Illegal access type 0 |

Basic pending register.

The basic pending register shows which interrupt are pending. To speed up interrupts processing, a number of 'normal' interrupt status bits have been added to this register. This makes the 'IRQ pending base' register different from the other 'base' interrupt registers

| Name: IRQ pend base | | Address: 0x200 | Reset: 0x000 |
|---------------------|-----|----------------|--------------|
| Bit(s) | R/W | Function | |
| 31:21 | - | <unused> | |
| 20 | R | GPU IRQ 62 | |



BCM2835 ARM Peripherals

| Name: IRQ pend base | | Address: 0x200 | Reset: 0x000 |
|---------------------|---|--|--------------|
| 19 | R | GPU IRQ 57 | |
| 18 | R | GPU IRQ 56 | |
| 17 | R | GPU IRQ 55 | |
| 16 | R | GPU IRQ 54 | |
| 15 | R | GPU IRQ 53 | |
| 14 | R | GPU IRQ 19 | |
| 13 | R | GPU IRQ 18 | |
| 12 | R | GPU IRQ 10 | |
| 11 | R | GPU IRQ 9 | |
| 10 | R | GPU IRQ 7 | |
| 9 | R | One or more bits set in pending register 2 | |
| 8 | R | One or more bits set in pending register 1 | |
| 7 | R | Illegal access type 0 IRQ pending | |
| 6 | R | Illegal access type 1 IRQ pending | |
| 5 | R | GPU1 halted IRQ pending | |
| 4 | R | GPU0 halted IRQ pending (Or GPU1 halted if bit 10 of control register 1 is set) | |
| 3 | R | ARM Doorbell 1 IRQ pending | |
| 2 | R | ARM Doorbell 0 IRQ pending | |
| 1 | R | ARM Mailbox IRQ pending | |
| 0 | R | ARM Timer IRQ pending | |

GPU IRQ x (10,11..20)

These bits are direct interrupts from the GPU. They have been selected as interrupts which are most likely to be useful to the ARM. The GPU interrupt selected are 7, 9, 10, 18, 19, 53,54,55,56,57,62. For details see the *GPU interrupts table*.

Bits set in pending registers (8,9)

These bits indicates if there are bits set in the pending 1/2 registers. The pending 1/2 registers hold ALL interrupts 0..63 from the GPU side. Some of these 64 interrupts are also connected to the basic pending register. Any bit set in pending register 1/2 which is NOT connected to the basic pending register causes bit 8 or 9 to set. Status bits 8 and 9 should be seen as "There are some interrupts pending which you don't know about. They are in pending register 1 /2."

Illegal access type-0 IRQ (7)

This bit indicate that the address/access error line from the ARM processor has generated an interrupt. That signal is asserted when either an address bit 31 or 30 was high or when an access was



BCM2835 ARM Peripherals

seen on the ARM Peripheral bus. The status of that signal can be read from Error/HALT status register bit 2.

Illegal access type-1 IRQ (6)

This bit indicates that an address/access error is seen in the ARM control has generated an interrupt. That can either be an address bit 29..26 was high or when a burst access was seen on the GPU Peripheral bus. The status of that signal can be read from Error/HALT status register bits 0 and 1.

GPU-1 halted IRQ (5)

This bit indicate that the GPU-1 halted status bit has generated an interrupt. The status of that signal can be read from Error/HALT status register bits 4.

GPU-0 (or any GPU) halted IRQ (4)

This bit indicate that the GPU-0 halted status bit has generated an interrupt. The status of that signal can be read from Error/HALT status register bits 3.

In order to allow a fast interrupt (FIQ) routine to cope with GPU 0 OR GPU-1 there is a bit in control register 1 which, if set will also route a GPU-1 halted status on this bit.

Standard peripheral IRQs (0,1,2,3)

These bits indicate if an interrupt is pending for one of the ARM control peripherals.

GPU pending 1 register.

| Name: IRQ pend base | | Address: 0x204 | Reset: 0x000 |
|---------------------|-----|---|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R | IRQ pending source 31:0 (See IRQ table above) | |

This register holds ALL interrupts 0..31 from the GPU side. Some of these interrupts are also connected to the basic pending register. Any interrupt status bit in here which is NOT connected to the basic pending will also cause bit 8 of the basic pending register to be set. That is all bits except 7, 9, 10, 18, 19.

GPU pending 2 register.

| Name: IRQ pend base | | Address: 0x208 | Reset: 0x000 |
|---------------------|-----|--|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R | IRQ pending source 63:32 (See IRQ table above) | |

This register holds ALL interrupts 32..63 from the GPU side. Some of these interrupts are also connected to the basic pending register. Any interrupt status bit in here which is NOT connected to the basic pending will also cause bit 9 of the basic pending register to be set. That is all bits except . register bits 21..25, 30 (Interrupts 53..57,62).

FIQ register.

The FIQ register control which interrupt source can generate a FIQ to the ARM. Only a single interrupt can be selected.

| Name: FIQ | | Address: 0x20C | Reset: 0x000 |
|-----------|-----|---|--------------|
| Bit(s) | R/W | Function | |
| 31:8 | R | <unused> | |
| 7 | R | FIQ enable. Set this bit to 1 to enable FIQ generation. If set to 0 bits 6:0 are don't care. | |
| 6:0 | R/W | Select FIQ Source | |

FIQ Source.

The FIQ source values 0-63 correspond to the GPU interrupt table. (See above)

The following values can be used to route ARM specific interrupts to the FIQ vector/routine:

| FIQ index | Source |
|-----------|------------------------------------|
| 0-63 | GPU Interrupts (See GPU IRQ table) |
| 64 | ARM Timer interrupt |
| 65 | ARM Mailbox interrupt |
| 66 | ARM Doorbell 0 interrupt |
| 67 | ARM Doorbell 1 interrupt |
| 68 | GPU0 Halted interrupt (Or GPU1) |
| 69 | GPU1 Halted interrupt |
| 70 | Illegal access type-1 interrupt |
| 71 | Illegal access type-0 interrupt |
| 72-127 | Do Not Use |

Interrupt enable register 1.

| Name: IRQ enable 1 | | Address: 0x210 | Reset: 0x000 |
|--------------------|-------|---|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R/Wbs | Set to enable IRQ source 31:0 (See IRQ table above) | |

Writing a 1 to a bit will set the corresponding IRQ enable bit. All other IRQ enable bits are unaffected. Only bits which are enabled can be seen in the interrupt pending registers. There is no provision here to see if there are interrupts which are pending but not enabled.



Interrupt enable register 2.

| Name: IRQ enable 2 | | Address: 0x214 | Reset: 0x000 |
|--------------------|-------|--|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R/Wbs | Set to enable IRQ source 63:32 (See IRQ table above) | |

Writing a 1 to a bit will set the corresponding IRQ enable bit. All other IRQ enable bits are unaffected. Only bits which are enabled can be seen in the interrupt pending registers. There is no provision here to see if there are interrupts which are pending but not enabled.

Base Interrupt enable register.

| Name: IRQ enable 3 | | Address: 0x218 | Reset: 0x000 |
|--------------------|-------|--|--------------|
| Bit(s) | R/W | Function | |
| 31:8 | R/Wbs | <Unused> | |
| 7 | R/Wbs | Set to enable Access error type -0 IRQ | |
| 6 | R/Wbs | Set to enable Access error type -1 IRQ | |
| 5 | R/Wbs | Set to enable GPU 1 Halted IRQ | |
| 4 | R/Wbs | Set to enable GPU 0 Halted IRQ | |
| 3 | R/Wbs | Set to enable ARM Doorbell 1 IRQ | |
| 2 | R/Wbs | Set to enable ARM Doorbell 0 IRQ | |
| 1 | R/Wbs | Set to enable ARM Mailbox IRQ | |
| 0 | R/Wbs | Set to enable ARM Timer IRQ | |

Writing a 1 to a bit will set the corresponding IRQ enable bit. All other IRQ enable bits are unaffected. Again only bits which are enabled can be seen in the basic pending register. There is no provision here to see if there are interrupts which are pending but not enabled.

Interrupt disable register 1.

| Name: IRQ disable 1 | | Address: 0x21C | Reset: 0x000 |
|---------------------|-------|--|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R/Wbc | Set to disable IRQ source 31:0 (See IRQ table above) | |

Writing a 1 to a bit will clear the corresponding IRQ enable bit. All other IRQ enable bits are unaffected.



BCM2835 ARM Peripherals

Interrupt disable register 2.

| Name: IRQ disable 2 | | Address: 0x220 | Reset: 0x000 |
|---------------------|-------|---|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R/Wbc | Set to disable IRQ source 63:32 (See IRQ table above) | |

Writing a 1 to a bit will clear the corresponding IRQ enable bit. All other IRQ enable bits are unaffected.

Base disable register.

| Name: IRQ disable 3 | | Address: 0x224 | Reset: 0x000 |
|---------------------|-------|---|--------------|
| Bit(s) | R/W | Function | |
| 31:8 | - | <Unused> | |
| 7 | R/Wbc | Set to disable Access error type -0 IRQ | |
| 6 | R/Wbc | Set to disable Access error type -1 IRQ | |
| 5 | R/Wbc | Set to disable GPU 1 Halted IRQ | |
| 4 | R/Wbc | Set to disable GPU 0 Halted IRQ | |
| 3 | R/Wbc | Set to disable ARM Doorbell 1 IRQ | |
| 2 | R/Wbc | Set to disable ARM Doorbell 0 IRQ | |
| 1 | R/Wbc | Set to disable ARM Mailbox IRQ | |
| 0 | R/Wbc | Set to disable ARM Timer IRQ | |

Writing a 1 to a bit will clear the corresponding IRQ enable bit. All other IRQ enable bits are unaffected.

8 PCM / I2S Audio

The PCM audio interface is an APB peripheral providing input and output of telephony or high quality serial audio streams. It supports many classic PCM formats including I2S.

The PCM audio interface has 4 interface signals;

- PCM_CLK - bit clock.
- PCM_FS - frame sync signal.
- PCM_DIN - serial data input.
- PCM_DOUT - serial data output.

PCM is a serial format with a single bit data_in and single bit data_out. Data is always serialised MS-bit first.

The frame sync signal (PCM_FS) is used to delimit the serial data into individual frames. The length of the frame and the size and position of the frame sync are fully programmable.

Frames can contain 1 or 2 audio/data channels in each direction. Each channel can be between 8 and 32 bits wide and can be positioned anywhere within the frame as long as the two channels don't overlap. The channel format is separately programmable for transmit and receive directions.

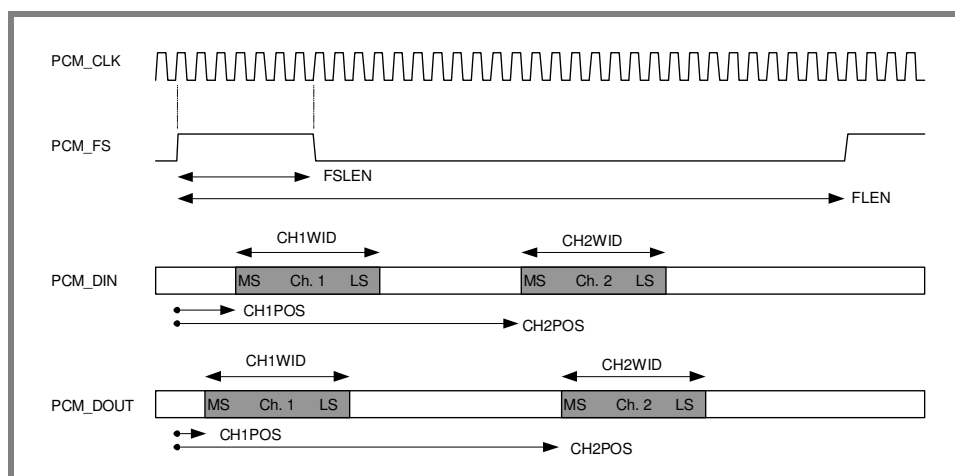


Figure 8-1 PCM Audio Interface Typical Timing

The PCM_CLK can be asynchronous to the bus APB clock and can be logically inverted if required.

The direction of the PCM_CLK and PCM_FS signals can be individually selected, allowing the interface to act as a master or slave device.

The input interface is also capable of supporting up to 2 PDM microphones, as an alternative to the classic PCM input format, in conjunction with a PCM output.

8.1 Block Diagram

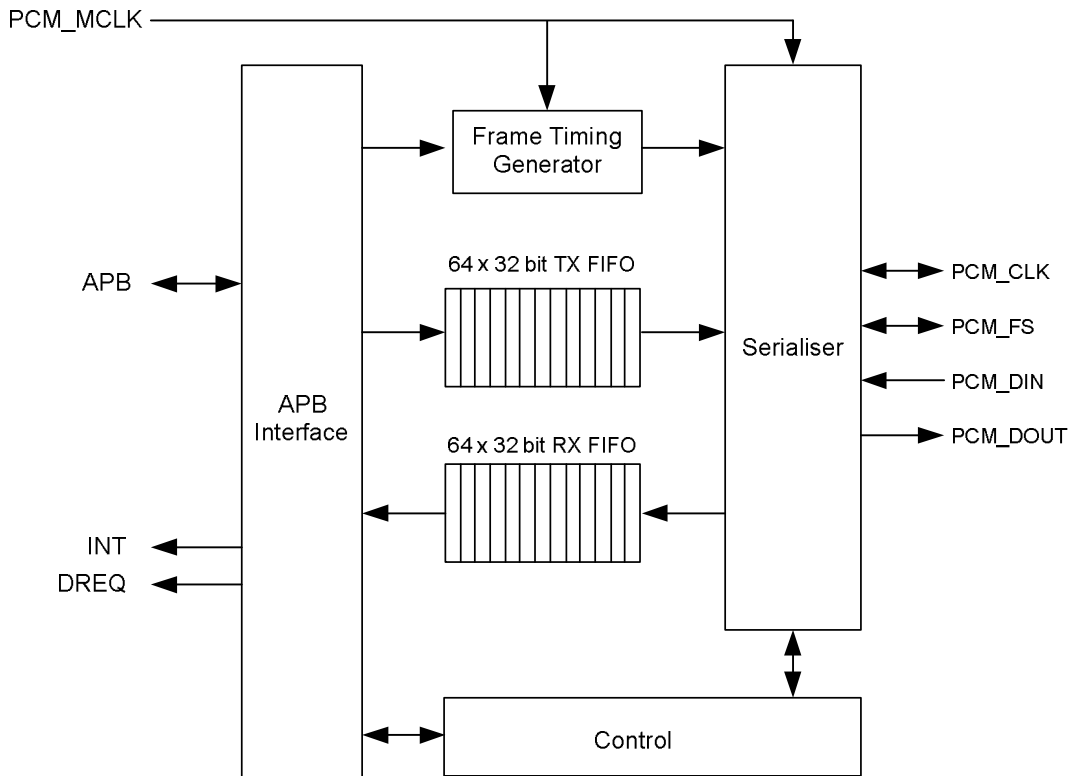


Figure 8-2 PCM Audio Interface Block Diagram

The PCM audio interface contains separate transmit and receive FIFOs. Note that if the frame contains two data channels, they must share the same FIFO and so the channel data will be interleaved. The block can be driven using simple polling, an interrupt based method or direct DMA control.

8.2 Typical Timing

Figure 8-1 shows typical interface timing and indicates the flexibility that the peripheral offers.

Normally PCM output signals change on the rising edge of PCM_CLK and input signals are sampled on its falling edge. The frame sync is considered as a data signal and sampled in the same way.

The front end of the PCM audio interface is run off the PCM_CLK and the PCM signals are timed against this clock. However, the polarity of the PCM_CLK can be physically inverted, in which case the edges are reversed.

In clock master mode (CLKM=0), the PCM_CLK is an output and is driven from the PCM_MCLK clock input.

In clock slave mode (CLKM=1), the PCM_CLK is an input, supplied by some external clock source.



In frame sync master mode (FSM=0), the PCM_FS is internally generated and is treated as a data output that changes on the positive edge of the clock. The length and polarity of the frame sync is fully programmable and it can be used as a standard frame sync signal, or as an L-R signal for I2S.

In frame sync slave mode (FSM=1), the PCM_FS is treated as a data input and is sampled on the negative edge of PCM_CLK. The first clock of a frame is taken as the first clock period where PCM_FS is sampled as a 1 following a period or periods where it was previously a 0. The PCM audio interface locks onto the incoming frame sync and uses this to indicate where the data channels are positioned. The precise timing at the start of frame is shown in Figure 8-3.

Note that in frame sync slave mode there are two synchronising methods. The legacy method is used when the frame length = 0. In this case the internal frame logic has to detect the incoming PCM_FS signal and reset the internal frame counter at the start of every frame. The logic relies on the PCM_FS to indicate the length of the frame and so can cope with adjacent frames of different lengths. However, this creates a short timing path that will corrupt the PCM_DOUT for one specific frame/channel setting.

The preferred method is to set the frame length to the expected length. Here the incoming PCM_FS is used to resynchronise the internal frame counter and this eliminates the short timing path.

8.3 Operation

The PCM interface runs asynchronously at the PCM_CLK rate and automatically transfers transmit and receive data across to the internal APB clock domain. The control registers are NOT synchronised and should be programmed before the device is enabled and should NOT be changed whilst the interface is running.

Only the EN, RXON and TXON bits of the PCMCS register are synchronised across the PCM - APB clock domain and are allowed to be changed whilst the interface is running.

The EN bit is a global power-saving enable. The TXON and RXON bits enable transmit and receive, and the interface is running whenever either TXON or RXON is enabled.

In operation, the PCM format is programmed by setting the appropriate frame length, frame sync, channel position values, and signal polarity controls. The transmit FIFO should be preloaded with data and the interface can then be enabled and started, and will run continuously until stopped. If the transmit FIFO becomes empty or the receive FIFO becomes full, the RXERR or TXERR error flags will be set, but the interface will just continue. If the RX FIFO overflows, new samples are discarded and if the TX FIFO underflows, zeros are transmitted.

Normally channel data is read or written into the appropriate FIFO as a single word. If the channel is less than 32 bits, the data is right justified and should be padded with zeros. If the RXSEX bit is set then the received data is sign extended up to the full 32 bits. When a frame is programmed to have two data channels, then each channel is written/read as a separate word in the FIFO, producing an interleaved data stream. When initialising the interface, the first word read out of the TX FIFO will be used for the first channel, and the data from the first channel on the first frame to be received will be the first word written into the RX FIFO.

If a FIFO error occurs in a two channel frame, then channel synchronisation may be lost which may result in a left right audio channel swap. RXSYNC and TXSYNC status bits are

provided to help determine if channel slip has occurred. They indicate if the number of words in the FIFO is a multiple of a full frame (taking into account where we are in the current frame being transferred). This assumes that an integer number of frames data has been sent/read from the FIFOs.

If a frame is programmed to have two data channels and the packed mode bits are set (FRXP FTXP) then the FIFOs are configured so that each word contains the data for both channels (2x 16 bit samples). In this mode each word written to the TX FIFO contains 2 16 bit samples, and the Least Significant sample is transmitted first. Each word read from the RX FIFO will contain the data received from 2 channels, the first channel received will be in the Least Significant half of the word. If the channels size is less than 16 bits, the TX data will be truncated and RX data will be padded to 16 bits with zeros.

Note that data is always serialised MS-bit first. This is well-established behaviour in both PCM and I2S.

If the PDM input mode is enabled then channel 1 is sampled on the negative edge of PCM_CLK whilst channel 2 is sampled on the positive edge of PCM_CLK.

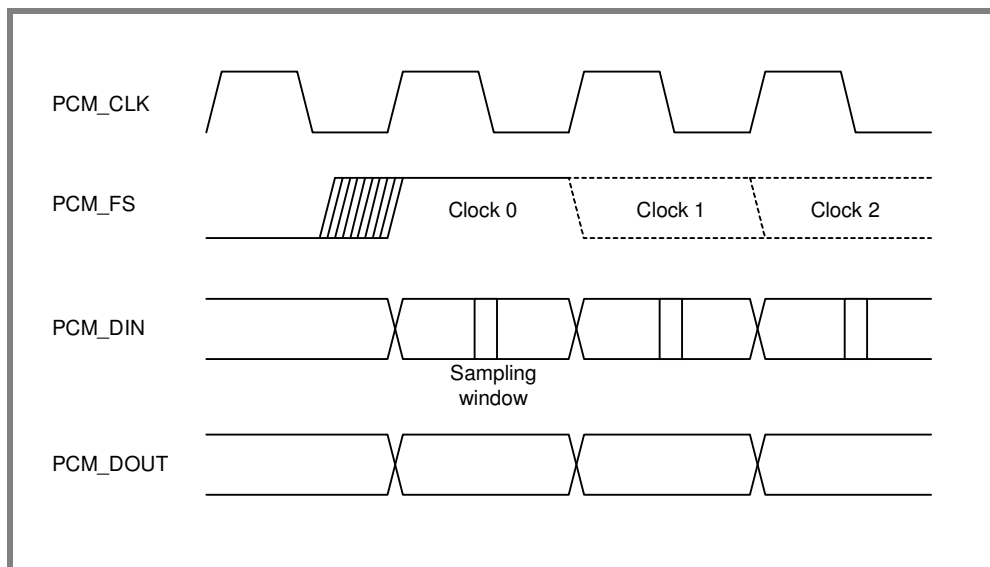


Figure 8-3 Timing at Start of Frame

Note that the precise timing of FS (when it is an input) is not clearly defined and it may change state before or after the positive edge of the clock. Here the first clock of the frame is defined as the clock period where the PCM_FS is sampled (negative edge) as a 1 where it was previously sampled as a 0.

8.4 Software Operation

8.4.1 Operating in Polled mode

Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed. Set RXTHR/TXTHR to determine the FIFO thresholds.



If transmitting, ensure that sufficient sample words have been written to PCMFIFO before transmission is started. Set TXON and/or RXON to begin operation. Poll TXW writing sample words to PCMFIFO and RXR reading sample words from PCMFIFO until all data is transferred.

8.4.2 Operating in Interrupt mode

- a) Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed. Set RXTHR/TXTHR to determine the FIFO thresholds.
- b) Set INTR and/or INTT to enable interrupts.
- c) If transmitting, ensure that sufficient sample words have been written to PCMFIFO before transmission is started. Set TXON and/or RXON to begin operation.
- d) When an interrupt occurs, check RXR. If this is set then one or more sample words are available in PCMFIFO. If TXW is set then one or more sample words can be sent to PCMFIFO.

8.4.3 DMA

- a) Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed.
- b) Set DMAEN to enable DMA DREQ generation and set RXREQ/TXREQ to determine the FIFO thresholds for the DREQs. If required, set TXPANIC and RXPANIC to determine the level at which the DMA should increase its AXI priority,
- c) In the DMA controllers set the correct DREQ channels, one for RX and one for TX. Start the DMA which should fill the TX FIFO.
- d) Set TXON and/or RXON to begin operation.

8.5 Error Handling.

In all software operational modes, the possibility of FIFO over or under run exists. Should this happen when using 2 channels per frame, there is a risk of losing sync with the channel data stored in the FIFO. If this happens and is not detected and corrected, then the data channels may become swapped.

The FIFO's will automatically detect an error condition caused by a FIFO over or under-run and this will set the appropriate latching error bit in the control/status register. Writing a '1' back to this error bit will clear the latched flag.

In a system using a polled operation, the error bits can be checked manually. For an interrupt or DMA based system, setting the INTE bit will cause the PCM interface to generate an interrupt when an error is detected.

If a FIFO error occurs during operation in which 2 data channels are being used then the synchronisation of the data may be lost. This can be recovered by either of these two methods:

- a) Disable transmit and receive (TXON and RXON =0). Clear the FIFO's (RXCLR and TXCLR =1). Note that it may take up to 2 PCM clocks for the FIFOs to be physically cleared after initiating a clear. Then preload the transmit FIFO and restart transmission. This of course loses the data in the FIFO and further interrupts the data flow to the external device.
- b) Examine the TXSYNC and RXSYNC flags. These flags indicate if the amount of data in the FIFO is a whole number of frames, automatically taking into account where we are in the current frame being transmitted or received. Thus, providing an even number of samples was read or written to the FIFOs, then if the flags are set then this indicates that a single word needs to be written or read to adjust the data. Normal exchange of data can then proceed (where the first word in a data pair is for channel 1). This method should cause less disruption to the data stream.

8.6 PDM Input Mode Operation

The PDM input mode is capable of interfacing with two digital half-cycle PDM microphones and implements a 4th order CIC decimation filter with a selectable decimation factor. The clock input of the microphones is shared with the PCM output codec and it should be configured to provide the correct clock rate for the microphones. As a result it may be necessary to add a number of padding bits into the PCM output and configure the output codec to allow for this.

When using the PDM input mode the bit width and the rate of the data received will depend on the decimation factor used. Once the data has been read from the peripheral a further decimation and filtering stage will be required and can be implemented in software. The software filter should also correct the droop introduced by the CIC filter stage. Similarly a DC correction stage should also be employed.

| PDMN | PCM_CLK (MHz) | Peripheral Output Format | OSR | Fs |
|----------|---------------|--------------------------|-----|-------|
| 0 (N=16) | 3.072 | 16 bits unsigned | 4 | 48kHz |
| 1 (N=32) | 3.072 | 20 bits unsigned | 2 | 48kHz |

Table 8-1 PDM Input Mode Configuration

8.7 GRAY Code Input Mode Operation

GRAY mode is used for an incoming data stream only. GRAY mode is selected by setting the enable bit (EN) in the PCM_GRAY register.

In this mode data is received on the PCM_DIN (data) and the PCM_FS (strobe) pins. The data is expected to be in data/strobe format. In this mode data is detected when either the data or the strobe change state. As each bit is received it is written into the RX buffer and when 32 bits are received they are written out to the RXFIFO as a 32 bit word. In order for this mode to work the user must program a PCM clock rate which is 4 times faster than the gray data rate. Also the gray coded data input signals should be clean.



BCM2835 ARM Peripherals

The normal RXREQ and RXTHR FIFO levels will apply as for normal PCM received data.

If a message is received that is not a multiple of 32 bits, any data in the RX Buffer can be flushed out by setting the flush bit (FLUSH). Once set, this bit will read back as zero until the flush operation has completed. This may take several cycles as the APB clock may be many times faster than the PCM clock. Once the flush has occurred, the bits are packed up to 32 bits with zeros and written out to the RXFIFO. The flushed field (FLUSHED) will indicate how many of bits of this word are valid.

Note that to get an accurate indication of the number of bits currently in the rx shift register (RXLEVEL) the APB clock must be at least 2x the PCM_CLK.

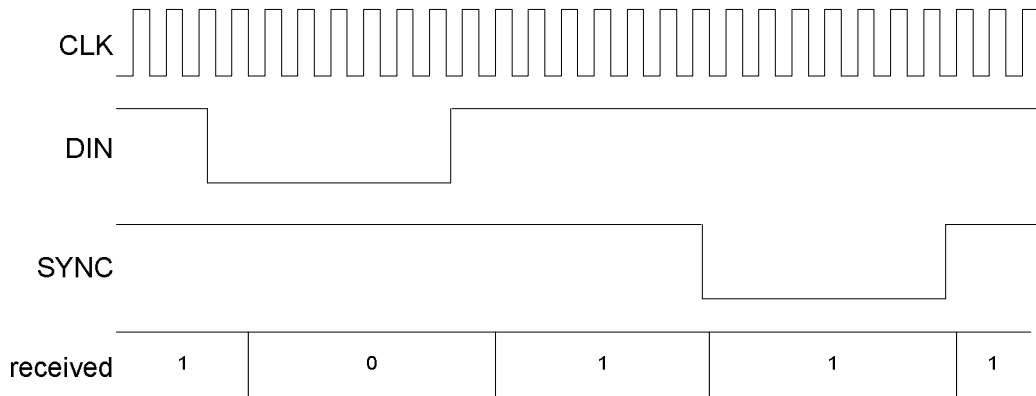


Figure 8-4 Gray mode input format

8.8 PCM Register Map

There is only PCM module in the BCM2835. The PCM base address for the registers is 0x7E203000.

| PCM Address Map | | | |
|-----------------|------------------------|----------------------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | CS_A | PCM Control and Status | 32 |
| 0x4 | FIFO_A | PCM FIFO Data | 32 |
| 0x8 | MODE_A | PCM Mode | 32 |
| 0xc | RXC_A | PCM Receive Configuration | 32 |
| 0x10 | TXC_A | PCM Transmit Configuration | 32 |
| 0x14 | DREQ_A | PCM DMA Request Level | 32 |



BCM2835 ARM Peripherals

| | | | |
|------|--------------------------|------------------------------|----|
| 0x18 | INTEN_A | PCM Interrupt Enables | 32 |
| 0x1c | INTSTC_A | PCM Interrupt Status & Clear | 32 |
| 0x20 | GRAY | PCM Gray Mode Control | 32 |

CS_A Register

Synopsis This register contains the main control and status bits for the PCM. The bottom 3 bits of this register can be written to whilst the PCM is running. The remaining bits cannot.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:26 | | Reserved - Write as 0, read as don't care | | |
| 25 | STBY | <u>RAM Standby</u> This bit is used to control the PCM Rams standby mode. By default this bit is 0 causing RAMs to start initially in standby mode. Rams should be released from standby prior to any transmit/receive operation. Allow for at least 4 PCM clock cycles to take effect. This may or may not be implemented, depending upon the RAM libraries being used. | RW | 0x0 |
| 24 | SYNC | <u>PCM Clock sync helper.</u> This bit provides a software synchronisation mechanism to allow the software to detect when 2 PCM clocks have occurred. It takes 2 PCM clocks before the value written to this bit will be echoed back in the read value. | RW | 0x0 |
| 23 | RXSEX | <u>RX Sign Extend</u> 0 = No sign extension. 1 = Sign extend the RX data. When set, the MSB of the received data channel (as set by the CHxWID parameter) is repeated in all the higher data bits up to the full 32 bit data width. | RW | 0x0 |
| 22 | RXF | <u>RX FIFO is Full</u> 0 = RX FIFO can accept more data. 1 = RX FIFO is full and will overflow if more data is received. | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-------|--------|--|----|-----|
| 21 | TXE | <u>TX FIFO is Empty</u> 0 = TX FIFO is not empty. 1 = TX FIFO is empty and underflow will take place if no more data is written. | RO | 0x1 |
| 20 | RXD | <u>Indicates that the RX FIFO contains data</u> 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 sample. | RO | 0x0 |
| 19 | TXD | <u>Indicates that the TX FIFO can accept data</u> 0 = TX FIFO is full and so cannot accept more data. 1 = TX FIFO has space for at least 1 sample. | RO | 0x1 |
| 18 | RXR | <u>Indicates that the RX FIFO needs reading</u> 0 = RX FIFO is less than RXTHR full. 1 = RX FIFO is RXTHR or more full. This is cleared by reading sufficient data from the RX FIFO. | RO | 0x0 |
| 17 | TXW | <u>Indicates that the TX FIFO needs Writing</u> 0 = TX FIFO is at least TXTHR full. 1 = TX FIFO is less than TXTHR full. This is cleared by writing sufficient data to the TX FIFO. | RO | 0x1 |
| 16 | RXERR | <u>RX FIFO Error</u> 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1. | RW | 0x0 |
| 15 | TXERR | <u>TX FIFO Error</u> 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1. | RW | 0x0 |
| 14 | RXSYNC | <u>RX FIFO Sync</u> 0 = FIFO is out of sync. The amount of data left in the FIFO is not a multiple of that required for a frame. This takes into account if we are halfway through the frame. 1 = FIFO is in sync. | RO | 0x0 |
| 13 | TXSYNC | <u>TX FIFO Sync</u> 0 = FIFO is out of sync. The amount of data left in the FIFO is not a multiple of that required for a frame. This takes into account if we are halfway through the frame. 1 = FIFO is in sync. | RO | 0x0 |
| 12:10 | | Reserved - Write as 0, read as don't care | | |



BCM2835 ARM Peripherals

| | | | | |
|-----|-------|--|----|-----|
| 9 | DMAEN | <p><u>DMA DREQ Enable</u> 0 = Don't generate DMA DREQ requests. 1 = Generates a TX DMA DREQ requests whenever the TX FIFO level is lower than TXREQ or generates a RX DMA DREQ when the RX FIFO level is higher than RXREQ.</p> | RW | 0x0 |
| 8:7 | RXTHR | <p><u>Sets the RX FIFO threshold at which point the RXR flag is set</u> 00 = set when we have a single sample in the RX FIFO 01 = set when the RX FIFO is at least full 10 = set when the RX FIFO is at least 11 = set when the RX FIFO is full</p> | RW | 0x0 |
| 6:5 | TXTHR | <p><u>Sets the TX FIFO threshold at which point the TXW flag is set</u> 00 = set when the TX FIFO is empty 01 = set when the TX FIFO is less than full 10 = set when the TX FIFO is less than full 11 = set when the TX FIFO is full but for one sample</p> | RW | 0x0 |
| 4 | RXCLR | <p><u>Clear the RX FIFO.</u> Assert to clear RX FIFO. This bit is self clearing and is always read as clear Note that it will take 2 PCM clocks for the FIFO to be physically cleared.</p> | WO | 0x0 |
| 3 | TXCLR | <p><u>Clear the TX FIFO</u> Assert to clear TX FIFO. This bit is self clearing and is always read as clear. Note that it will take 2 PCM clocks for the FIFO to be physically cleared.</p> | WO | 0x0 |
| 2 | TXON | <p><u>Enable transmission</u> 0 = Stop transmission. This will stop immediately if possible or else at the end of the next frame. The TX FIFO can still be written to to preload data. 1 = Start transmission. This will start transmitting at the start of the next frame. Once enabled, the first data read from the TX FIFO will be placed in the first channel of the frame, thus ensuring proper channel synchronisation. The frame counter will be started whenever TXON or RXON are set. This bit can be written whilst the interface is running.</p> | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|------|---|----|-----|
| 1 | RXON | <p><u>Enable reception.</u> 0 = Disable reception. This will stop on the next available frame end. RX FIFO data can still be read. 1 = Enable reception. This will be start receiving at the start of the next frame. The first channel to be received will be the first word written to the RX FIFO. This bit can be written whilst the interface is running.</p> | RW | 0x0 |
| 0 | EN | <p><u>Enable the PCM Audio Interface</u> 0 = The PCM interface is disabled and most logic is gated off to save power. 1 = The PCM Interface is enabled. This bit can be written whilst the interface is running.</p> | RW | 0x0 |

FIFO_A Register

Synopsis This is the FIFO port of the PCM. Data written here is transmitted, and received data is read from here.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | | <i>Reserved - Write as 0, read as don't care</i> | | |

MODE_A Register

Synopsis This register defines the basic PCM Operating Mode. It is used to configure the frame size and format and whether the PCM is in master or slave modes for its frame sync or clock. This register cannot be changed whilst the PCM is running.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:29 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|----|---------|---|----|-----|
| 28 | CLK_DIS | <u>PCM Clock Disable</u> 1 = Disable the PCM Clock. This cleanly disables the PCM clock. This enables glitch free clock switching between an internal and an uncontrollable external clock. The PCM clock can be disabled, and then the clock source switched, and then the clock re-enabled. 0 = Enable the PCM clock. | RW | 0x0 |
| 27 | PDMN | <u>PDM Decimation Factor (N)</u> 0 = Decimation factor 16. 1 = Decimation factor 32. Sets the decimation factor of the CIC decimation filter. | RW | 0x0 |
| 26 | PDME | <u>PDM Input Mode Enable</u> 0 = Disable PDM (classic PCM input). 1 = Enable PDM input filter. Enable CIC filter on input pin for PDM inputs. In order to receive data RXON must also be set. | RW | 0x0 |
| 25 | FRXP | <u>Receive Frame Packed Mode</u> 0 = The data from each channel is written into the RX FIFO. 1 = The data from both RX channels is merged (1st channel is in the LS half) and then written to the RX FIFO as a single 2x16 bit packed mode word. First received channel in the frame goes into the LS half word. If the received data is larger than 16 bits, the upper bits are truncated. The maximum channel size is 16 bits. | RW | 0x0 |
| 24 | FTXP | <u>Transmit Frame Packed Mode</u> 0 = Each TX FIFO word is written into a single channel. 1 = Each TX FIFO word is split into 2 16 bit words and used to fill both data channels in the same frame. The maximum channel size is 16 bits. The LS half of the word is used in the first channel of the frame. | RW | 0x0 |
| 23 | CLKM | <u>PCM Clock Mode</u> 0 = Master mode. The PCM CLK is an output and drives at the MCLK rate. 1 = Slave mode. The PCM CLK is an input. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-------|-------|---|----|-----|
| 22 | CLKI | <p><u>Clock Invert</u> this logically inverts the PCM_CLK signal.</p> <p>0 = Outputs change on rising edge of clock, inputs are sampled on falling edge.</p> <p>1 = Outputs change on falling edge of clock, inputs are sampled on rising edge.</p> | RW | 0x0 |
| 21 | FSM | <p><u>Frame Sync Mode</u></p> <p>0 = Master mode. The PCM_FS is an output and we generate the frame sync.</p> <p>1 = Slave mode. The PCM_FS is an input and we lock onto the incoming frame sync signal.</p> | RW | 0x0 |
| 20 | FSI | <p><u>Frame Sync Invert</u> This logically inverts the frame sync signal.</p> <p>0 = In master mode, FS is normally low and goes high to indicate frame sync. In slave mode, the frame starts with the clock where FS is a 1 after being a 0.</p> <p>1 = In master mode, FS is normally high and goes low to indicate frame sync. In slave mode, the frame starts with the clock where FS is a 0 after being a 1.</p> | RW | 0x0 |
| 19:10 | FLEN | <p><u>Frame Length</u></p> <p>Sets the frame length to (FLEN+1) clocks. Used only when FSM == 0.</p> <p>1 = frame length of 2 clocks.</p> <p>2 = frame length of 3 clocks. etc</p> | RW | 0x0 |
| 9:0 | FSLEN | <p><u>Frame Sync Length</u></p> <p>Sets the frame sync length to (FSLEN) clocks. This is only used when FSM == 0. PCM_FS will remain permanently active if FSLEN >= FLEN.</p> <p>0 = frame sync pulse is off.</p> <p>1 = frame sync pulse is 1 clock wide. etc</p> | RW | 0x0 |

RXC_A Register

Synopsis Sets the Channel configurations for Receiving. This sets the position and width of the 2 receive channels within the frame. The two channels cannot overlap, however they channel 1 can come after channel zero, although the first data will always be from the first channel in the frame. Channels can also straddle the frame begin end boundary as that is set by the frame sync position. This register cannot be changed whilst the PCM is running.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-------------|------|-------|
|--------|------------|-------------|------|-------|



BCM2835 ARM Peripherals

| | | | | |
|-------|--------|---|----|-----|
| 31 | CH1WEX | <u>Channel 1 Width Extension Bit</u> This is the MSB of the channel 1 width (CH1WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM | RW | 0x0 |
| 30 | CH1EN | <u>Channel 1 Enable</u> 0 = Channel 1 disabled and no data is received from channel 1 and written to the RX FIFO. 1 = Channel 1 enabled. | RW | 0x0 |
| 29:20 | CH1POS | <u>Channel 1 Position</u> This sets the bit clock at which the first bit (MS bit) of channel 1 data occurs in the frame. 0 indicates the first clock of frame. | RW | 0x0 |
| 19:16 | CH1WID | <u>Channel 1 Width</u> This sets the width of channel 1 in bit clocks. This field has been extended with the CH1WEX bit giving a total width of $(CH1WEX * 16) + CH1WID + 8$. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide | RW | 0x0 |
| 15 | CH2WEX | <u>Channel 2 Width Extension Bit</u> This is the MSB of the channel 2 width (CH2WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM | RW | 0x0 |
| 14 | CH2EN | <u>Channel 2 Enable</u> 0 = Channel 2 disabled and no data is received from channel 2 and written to the RX FIFO. 1 = Channel 2 enabled. | RW | 0x0 |
| 13:4 | CH2POS | <u>Channel 2 Position</u> This sets the bit clock at which the first bit (MS bit) of channel 2 data occurs in the frame. 0 indicates the first clock of frame. | RW | 0x0 |
| 3:0 | CH2WID | <u>Channel 2 Width</u> This sets the width of channel 2 in bit clocks. This field has been extended with the CH2WEX bit giving a total width of $(CH2WEX * 16) + CH2WID + 8$. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide | RW | 0x0 |



BCM2835 ARM Peripherals

TXC_A Register

Synopsis Sets the Channel configurations for Transmitting. This sets the position and width of the 2 transmit channels within the frame. The two channels cannot overlap, however they channel 1 can come after channel zero, although the first data will always be used in the first channel in the frame. Channels can also straddle the frame begin end boundary as that is set by the frame sync position. This register cannot be changed whilst the PCM is running.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31 | CH1WEX | <u>Channel 1 Width Extension Bit</u> This is the MSB of the channel 1 width (CH1WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM | RW | 0x0 |
| 30 | CH1EN | <u>Channel 1 Enable</u> 0 = Channel 1 disabled and no data is taken from the TX FIFO and transmitted on channel 1. 1 = Channel 1 enabled. | RW | 0x0 |
| 29:20 | CH1POS | <u>Channel 1 Position</u> This sets the bit clock at which the first bit (MS bit) of channel 1 data occurs in the frame. 0 indicates the first clock of frame. | RW | 0x0 |
| 19:16 | CH1WID | <u>Channel 1 Width</u> This sets the width of channel 1 in bit clocks. This field has been extended with the CH1WEX bit giving a total width of (CH1WEX* 16) + CH1WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide | RW | 0x0 |
| 15 | CH2WEX | <u>Channel 2 Width Extension Bit</u> This is the MSB of the channel 2 width (CH2WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM | RW | 0x0 |
| 14 | CH2EN | <u>Channel 2 Enable</u> 0 = Channel 2 disabled and no data is taken from the TX FIFO and transmitted on channel 2. 1 = Channel 2 enabled. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|------|--------|---|----|-----|
| 13:4 | CH2POS | <u>Channel 2 Position</u> This sets the bit clock at which the first bit (MS bit) of channel 2 data occurs in the frame. 0 indicates the first clock of frame. | RW | 0x0 |
| 3:0 | CH2WID | <u>Channel 2 Width</u> This sets the width of channel 2 in bit clocks. This field has been extended with the CH2WEX bit giving a total width of (CH2WEX* 16) + CH2WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide | RW | 0x0 |

DREQ_A Register

Synopsis Set the DMA DREQ and Panic thresholds. The PCM drives 2 DMA controls back to the DMA, one for the TX channel and one for the RX channel. DMA DREQ is used to request the DMA to perform another transfer, and DMA Panic is used to tell the DMA to use its panic level of priority when requesting things on the AXI bus. This register cannot be changed whilst the PCM is running.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31 | | Reserved - Write as 0, read as don't care | | |
| 30:24 | TX_PANIC | <u>TX Panic Level</u> This sets the TX FIFO Panic level. When the level is below this the PCM will assert its TX DMA Panic signal. | RW | 0x10 |
| 23 | | Reserved - Write as 0, read as don't care | | |
| 22:16 | RX_PANIC | <u>RX Panic Level</u> This sets the RX FIFO Panic level. When the level is above this the PCM will assert its RX DMA Panic signal. | RW | 0x30 |
| 15 | | Reserved - Write as 0, read as don't care | | |
| 14:8 | TX | <u>TX Request Level</u> This sets the TX FIFO DREQ level. When the level is below this the PCM will assert its DMA DREQ signal to request more data is written to the TX FIFO. | RW | 0x30 |
| 7 | | Reserved - Write as 0, read as don't care | | |



BCM2835 ARM Peripherals

| | | | | |
|-----|----|--|----|------|
| 6:0 | RX | <u>RX Request Level</u> This sets the RX FIFO DREQ level. When the level is above this the PCM will assert its DMA DREQ signal to request that some more data is read out of the RX FIFO. | RW | 0x20 |
|-----|----|--|----|------|

INTEN_A Register

Synopsis Set the reasons for generating an Interrupt. This register cannot be changed whilst the PCM is running.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | RXERR | <u>RX Error Interrupt</u> Setting this bit enables interrupts from PCM block when RX FIFO error occurs. | RW | 0x0 |
| 2 | TXERR | <u>TX Error Interrupt</u> Setting this bit enables interrupts from PCM block when TX FIFO error occurs. | RW | 0x0 |
| 1 | RXR | <u>RX Read Interrupt Enable</u> Setting this bit enables interrupts from PCM block when RX FIFO level is greater than or equal to the specified RXTHR level. | RW | 0x0 |
| 0 | TXW | <u>TX Write Interrupt Enable</u> Setting this bit enables interrupts from PCM block when TX FIFO level is less than the specified TXTHR level. | RW | 0x0 |

INTSTC_A Register

Synopsis This register is used to read and clear the PCM interrupt status. Writing a 1 to the asserted bit clears the bit. Writing a 0 has no effect.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|---|-------|---|----|-----|
| 3 | RXERR | <u>RX Error Interrupt Status / Clear</u> This bit indicates an interrupt occurred on RX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect. | RW | 0x0 |
| 2 | TXERR | <u>TX Error Interrupt Status / Clear</u> This bit indicates an interrupt occurred on TX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect. | RW | 0x0 |
| 1 | RXR | <u>RX Read Interrupt Status / Clear</u> This bit indicates an interrupt occurred on RX Read. Writing 1 to this bit clears it. Writing 0 has no effect. | RW | 0x0 |
| 0 | TXW | <u>TX Write Interrupt Status / Clear</u> This bit indicates an interrupt occurred on TX Write. Writing 1 to this bit clears it. Writing 0 has no effect. | RW | 0x0 |

GRAY Register

Synopsis This register is used to control the gray mode generation. This is used to put the PCM into a special data/strobe mode. This mode is under 'best effort' contract.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|-------------|---|------|-------|
| 31:22 | | Reserved - Write as 0, read as don't care | | |
| 21:16 | RXFIFOLEVEL | <u>The Current level of the RXFIFO</u> This indicates how many words are currently in the RXFIFO. | RO | 0x0 |
| 15:10 | FLUSHED | <u>The Number of bits that were flushed into the RXFIFO</u> This indicates how many bits were valid when the flush operation was performed. The valid bits are from bit 0 upwards. Non-valid bits are set to zero. | RO | 0x0 |
| 9:4 | RXLEVEL | <u>The Current fill level of the RX Buffer</u> This indicates how many GRAY coded bits have been received. When 32 bits are received, they are written out into the RXFIFO. | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|-------|---|----|-----|
| 3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2 | FLUSH | <u>Flush the RX Buffer into the RX FIFO</u> This forces the RX Buffer to do an early write. This is necessary if we have reached the end of the message and we have bits left in the RX Buffer. Flushing will write these bits as a single 32 bit word, starting at bit zero. Empty bits will be packed with zeros. The number of bits written will be recorded in the FLUSHED Field. This bit is written as a 1 to initiate a flush. It will read back as a zero until the flush operation has completed (as the PCM Clock may be very slow). | RW | 0x0 |
| 1 | CLR | <u>Clear the GRAY Mode Logic</u> This Bit will reset all the GRAY mode logic, and flush the RX buffer. It is not self clearing. | RW | 0x0 |
| 0 | EN | <u>Enable GRAY Mode</u> Setting this bit will put the PCM into GRAY mode. In gray mode the data is received on the data in and the frame sync pins. The data is expected to be in data/strobe format. | RW | 0x0 |

9 Pulse Width Modulator

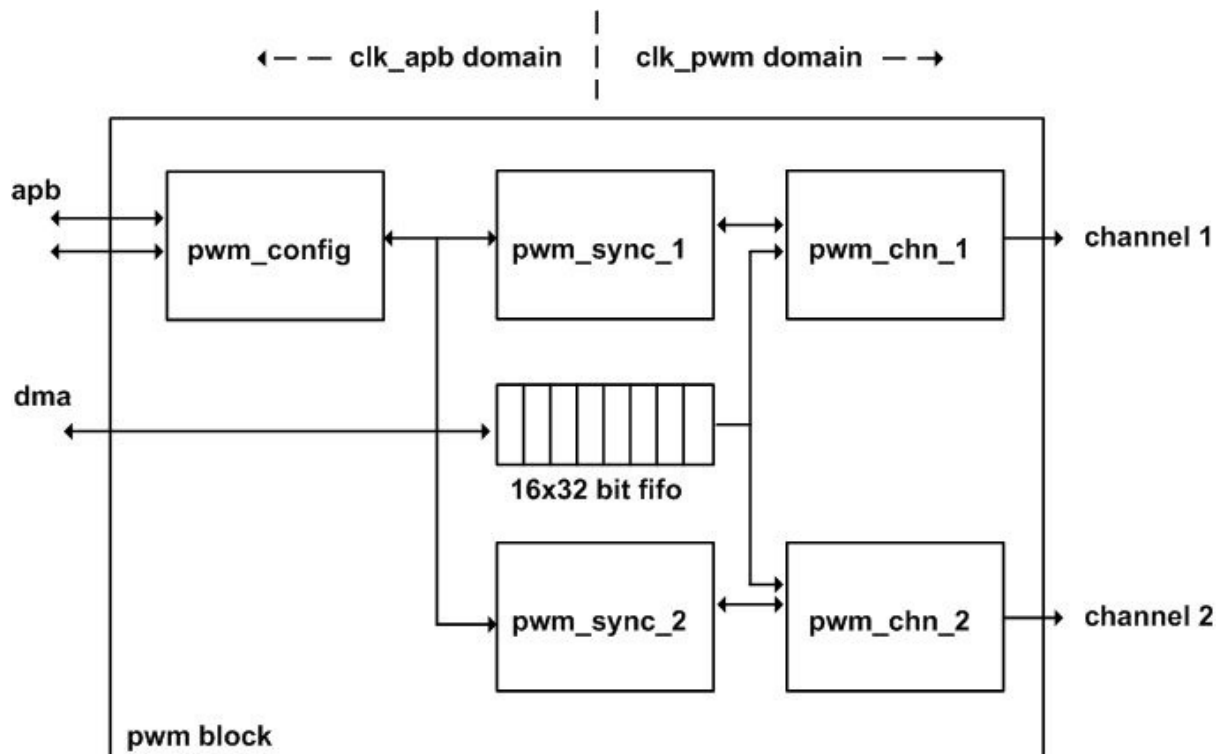
9.1 Overview

This section specifies in detail the functionality provided by the device Pulse Width Modulator (PWM) peripheral.

The PWM controller incorporates the following features:

- Two independent output bit-streams, clocked at a fixed frequency.
- Bit-streams configured individually to output either PWM or a serialised version of a 32-bit word.
- PWM outputs have variable input and output resolutions.
- Serialise mode configured to load data to and/or read data from a FIFO storage block, which can store up to eight 32-bit words.
- Both modes clocked by `clk_pwm` which is nominally 100MHz, but can be varied by the clock manager.

9.2 Block Diagram



9.3 PWM Implementation

A value represented as a ratio of N/M can be transmitted along a serial channel with pulse width modulation in which the value is represented by the duty cycle of the output signal. To send value N/M within a periodic sequence of M cycles, output should be 1 for N cycles and 0 for (M-N) cycles. The desired sequence should have 1's and 0's spread out as even as possible so that during any arbitrary period of time duty cycle achieves closest approximation of the value. This can be shown in the following table where 4/8 is modulated (N= 4, M= 8).

| | | | | | | | | | | | | |
|------|---|---|---|---|---|---|---|---|---|---|---|---|
| Bad | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Fair | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| Good | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Sequence which gives the 'good' approximation from the table above can be achieved by the following algorithm:

```

1. Set context = 0
2. context = context + N
3. if (context >= M)
    context = context - M
    send 1
else

```

where context is a register which stores the result of the addition/subtractions.

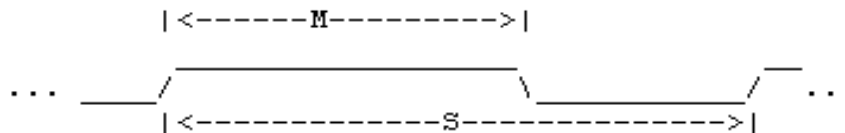
9.4 Modes of Operation

PWM controller consists of two independent channels (pwm_chn in block diagram) which implement the pwm algorithm explained in section 1.3. Each channel can operate in either pwm mode or serialiser mode.

PWM mode: There are two sub-modes in PWM mode: MSEN=0 and MSEN=1.

When MSEN=0, which is the default mode, data to be sent is interpreted as the value N of the algorithm explained above. Number of clock cycles (range) used to send data is the value M of the algorithm. Pulses are sent within this range so that the resulting duty cycle is N/M. Channel sends its output continuously as long as data register is used, or buffer is used and it is not empty.

When MSEN=1, PWM block does not use the algorithm explained above, instead it sends serial data with the M/S ratio as in the picture below. M is the data to be sent, and S is the range. This mode may be preferred if high frequency modulation is not required or has negative effects. Channel sends its output continuously as long as data register is used, or buffer is used and it is not empty.



Serial bit transmission when M/S Mode enabled

Serialiser mode: Each channel is also capable of working as a serialiser. In this mode data written in buffer or the data register is sent serially.

9.5 Quick Reference

- PWM DMA is mapped to DMA channel 5.
- GPIOs are assigned to PWM channels as below. Please refer to GPIO section for further details:

| | PWM0 | PWM1 |
|----------------|-----------|-----------|
| GPIO 12 | Alt Fun 0 | - |
| GPIO 13 | - | Alt Fun 0 |
| GPIO 18 | Alt Fun 5 | - |
| GPIO 19 | - | Alt Fun 5 |
| GPIO 40 | Alt Fun 0 | - |
| GPIO 41 | - | Alt Fun 0 |
| GPIO 45 | - | Alt Fun 0 |
| GPIO 52 | Alt Fun 1 | - |
| GPIO 53 | - | Alt Fun 1 |



- PWM clock source and frequency is controlled in CPRMAN.

9.6 Control and Status Registers

| PWM Address Map | | | |
|-----------------|----------------------|-----------------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | CTL | PWM Control | 32 |
| 0x4 | STA | PWM Status | 32 |
| 0x8 | DMAC | PWM DMA Configuration | 32 |
| 0x10 | RNG1 | PWM Channel 1 Range | 32 |
| 0x14 | DAT1 | PWM Channel 1 Data | 32 |
| 0x18 | FIF1 | PWM FIFO Input | 32 |
| 0x20 | RNG2 | PWM Channel 2 Range | 32 |
| 0x24 | DAT2 | PWM Channel 2 Data | 32 |

| CTL Register |
|--------------|
|--------------|



BCM2835 ARM Peripherals

Synopsis PWENi is used to enable/disable the corresponding channel. Setting this bit to 1 enables the channel and transmitter state machine. All registers and FIFO is writable without setting this bit.

MODEi bit is used to determine mode of operation. Setting this bit to 0 enables PWM mode. In this mode data stored in either PWM_DATi or FIFO is transmitted by pulse width modulation within the range defined by PWM_RNGi. When this mode is used MSENi defines whether to use PWM algorithm. Setting MODEi to 1 enables serial mode, in which data stored in either PWM_DATi or FIFO is transmitted serially within the range defined by PWM_RNGi. Data is transmitted MSB first and truncated or zero-padded depending on PWM_RNGi. Default mode is PWM.

RPTLi is used to enable/disable repeating of the last data available in the FIFO just before it empties. When this bit is 1 and FIFO is used, the last available data in the FIFO is repeatedly sent. This may be useful in PWM mode to avoid duty cycle gaps. If the FIFO is not used this bit does not have any effect. Default operation is do-not-repeat.

SBITi defines the state of the output when no transmission takes place. It also defines the zero polarity for the zero padding in serialiser mode. This bit is padded between two consecutive transfers as well as tail of the data when PWM_RNGi is larger than bit depth of data being transferred. this bit is zero by default.

POLAi is used to configure the polarity of the output bit. When set to high the final output is inverted. Default operation is no inversion.

USEFi bit is used to enable/disable FIFO transfer. When this bit is high data stored in the FIFO is used for transmission. When it is low, data written to PWM_DATi is transferred. This bit is 0 as default.

CLRF is used to clear the FIFO. Writing a 1 to this bit clears the FIFO. Writing 0 has no effect. This is a single shot operation and reading the bit always returns 0.

MSENi is used to determine whether to use PWM algorithm or simple M/S ratio transmission. When this bit is high M/S transmission is used. This bit is zero as default. When MODEi is 1, this configuration bit has no effect.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | Reserved - Write as 0, read as don't care | | |
| 15 | MSEN2 | <u>Channel 2 M/S Enable</u> 0: PWM algorithm is used 1: M/S transmission is used. | RW | 0x0 |
| 14 | | Reserved - Write as 0, read as don't care | | |
| 13 | USEF2 | <u>Channel 1 Use Fifo</u> 0: Data register is transmitted 1: Fifo is used for transmission | RW | 0x0 |
| 12 | POLA2 | <u>Channel 1 Polarity</u> 0 : 0=low 1=high 1: 1=low 0=high | RW | 0x0 |
| 11 | SBIT2 | <u>Channel 1 Silence Bit</u> Defines the state of the output when no transmission takes place | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|----|-------|--|----|-----|
| 10 | RPTL2 | <u>Channel 1 Repeat Last Data</u> 0: Transmission interrupts when FIFO is empty 1: Last data in FIFO is transmitted repeatedly until FIFO is not empty | RW | 0x0 |
| 9 | MODE2 | <u>Channel 1 Mode</u> 0: PWM mode 1: Serialiser mode | RW | 0x0 |
| 8 | PWEN2 | <u>Channel 1 Enable</u> 0: Channel is disabled 1: Channel is enabled | RW | 0x0 |
| 7 | MSEN1 | <u>Channel 1 M/S Enable</u> 0: PWM algorithm is used 1: M/S transmission is used. | RW | 0x0 |
| 6 | CLRF1 | <u>Clear Fifo</u> 1: Clears FIFO 0: Has no effect This is a single shot operation. This bit always reads 0 | RO | 0x0 |
| 5 | USEF1 | <u>Channel 1 Use Fifo</u> 0: Data register is transmitted 1: Fifo is used for transmission | RW | 0x0 |
| 4 | POLA1 | <u>Channel 1 Polarity</u> 0 : 0=low 1=high 1: 1=low 0=high | RW | 0x0 |
| 3 | SBIT1 | <u>Channel 1 Silence Bit</u> Defines the state of the output when no transmission takes place | RW | 0x0 |
| 2 | RPTL1 | <u>Channel 1 Repeat Last Data</u> 0: Transmission interrupts when FIFO is empty 1: Last data in FIFO is transmitted repeatedly until FIFO is not empty | RW | 0x0 |
| 1 | MODE1 | <u>Channel 1 Mode</u> 0: PWM mode 1: Serialiser mode | RW | 0x0 |
| 0 | PWEN1 | <u>Channel 1 Enable</u> 0: Channel is disabled 1: Channel is enabled | RW | 0x0 |

STA Register

Synopsis FULL1 bit indicates the full status of the FIFO. If this bit is high FIFO is full.
 EMPT1 bit indicates the empty status of the FIFO. If this bit is high FIFO is empty.
 WERR1 bit sets to high when a write when full error occurs. Software must clear this bit by writing 1. Writing 0 to this bit has no effect.
 RERR1 bit sets to high when a read when empty error occurs. Software must clear this bit by writing 1. Writing 0 to this bit has no effect.
 GAPOi. bit indicates that there has been a gap between transmission of two consecutive data from FIFO. This may happen when FIFO gets empty after state machine has sent a word and waits for the next. If control bit RPTLi is set to high this event will not occur. Software must clear this bit by writing 1. Writing 0 to this bit has no effect.
 BERR sets to high when an error has occurred while writing to registers via APB. This may happen if the bus tries to write successively to same set of registers faster than the synchroniser block can cope with. Multiple switching may occur and contaminate the data during synchronisation. Software should clear this bit by writing 1. Writing 0 to this bit has no effect.
 STAi bit indicates the current state of the channel which is useful for debugging purposes. 0 means the channel is not currently transmitting. 1 means channel is transmitting data.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:13 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 12 | STA4 | <u>Channel 4 State</u> | RW | 0x0 |
| 11 | STA3 | <u>Channel 3 State</u> | RW | 0x0 |
| 10 | STA2 | <u>Channel 2 State</u> | RW | 0x0 |
| 9 | STA1 | <u>Channel 1 State</u> | RW | 0x0 |
| 8 | BERR | <u>Bus Error Flag</u> | RW | 0x0 |
| 7 | GAPO4 | <u>Channel 4 Gap Occurred Flag</u> | RW | 0x0 |
| 6 | GAPO3 | <u>Channel 3 Gap Occurred Flag</u> | RW | 0x0 |
| 5 | GAPO2 | <u>Channel 2 Gap Occurred Flag</u> | RW | 0x0 |
| 4 | GAPO1 | <u>Channel 1 Gap Occurred Flag</u> | RW | 0x0 |
| 3 | RERR1 | <u>Fifo Read Error Flag</u> | RW | 0x0 |
| 2 | WERR1 | <u>Fifo Write Error Flag</u> | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|-------|------------------------|----|-----|
| 1 | EMPT1 | <u>Fifo Empty Flag</u> | RW | 0x1 |
| 0 | FULL1 | <u>Fifo Full Flag</u> | RW | 0x0 |

DMAC Register

Synopsis ENAB bit is used to start DMA.
 PANIC bits are used to determine the threshold level for PANIC signal going active.
 Default value is 7.
 DREQ bits are used to determine the threshold level for DREQ signal going active.
 Default value is 7.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31 | ENAB | <u>DMA Enable</u> 0: DMA disabled 1: DMA enabled | RW | 0x0 |
| 30:16 | | Reserved - Write as 0, read as don't care | | |
| 15:8 | PANIC | <u>DMA Threshold for PANIC signal</u> | RW | 0x7 |
| 7:0 | DREQ | <u>DMA Threshold for DREQ signal</u> | RW | 0x7 |

RNG1 Register

Synopsis This register is used to define the range for the corresponding channel. In PWM mode evenly distributed pulses are sent within a period of length defined by this register. In serial mode serialised data is transmitted within the same period. If the value in PWM_RNGi is less than 32, only the first PWM_RNGi bits are sent resulting in a truncation. If it is larger than 32 excess zero bits are padded at the end of data. Default value for this register is 32.
 Note: Channels 3 and 4 are not available in B0 and corresponding Channel Range Registers are ignored.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|------------------------|------|-------|
| 31:0 | PWM_RNGi | <u>Channel i Range</u> | RW | 0x20 |



BCM2835 ARM Peripherals

DAT1 Register

Synopsis This register stores the 32 bit data to be sent by the PWM Controller when USEFi is 0. In PWM mode data is sent by pulse width modulation: the value of this register defines the number of pulses which is sent within the period defined by PWM_RNGi. In serialiser mode data stored in this register is serialised and transmitted.
Note: Channels 3 and 4 are not available in B0 and corresponding Channel Data Registers are ignored.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-----------------------|------|-------|
| 31:0 | PWM_DATi | <u>Channel i Data</u> | RW | 0x0 |

FIF1 Register

Synopsis This register is the FIFO input for the all channels. Data written to this address is stored in channel FIFO and if USEFi is enabled for the channel i it is used as data to be sent. This register is write only, and reading this register will always return bus default return value, pwm0 .
When more than one channel is enabled for FIFO usage, the data written into the FIFO is shared between these channels in turn. For example if the word series A B C D E F G H I .. is written to FIFO and two channels are active and configured to use FIFO then channel 1 will transmit words A C E G I .. and channel 2 will transmit words B D F H .. .
Note that requesting data from the FIFO is in locked-step manner and therefore requires tight coupling of state machines of the channels. If any of the channel range (period) value is different than the others this will cause the channels with small range values to wait between words hence resulting in gaps between words. To avoid that, each channel sharing the FIFO should be configured to use the same range value. Also note that RPTLi are not meaningful when the FIFO is shared between channels as there is no defined channel to own the last data in the FIFO. Therefore sharing channels must have their RPTLi set to zero.
If the set of channels to share the FIFO has been modified after a configuration change, FIFO should be cleared before writing new data.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---------------------------|------|-------|
| 31:0 | PWM_FIFO | <u>Channel FIFO Input</u> | RW | 0x0 |

RNG2 Register



BCM2835 ARM Peripherals

Synopsis This register is used to define the range for the corresponding channel. In PWM mode evenly distributed pulses are sent within a period of length defined by this register. In serial mode serialised data is transmitted within the same period. If the value in PWM_RNGi is less than 32, only the first PWM_RNGi bits are sent resulting in a truncation. If it is larger than 32 excess zero bits are padded at the end of data. Default value for this register is 32.

Note: Channels 3 and 4 are not available in B0 and corresponding Channel Range Registers are ignored.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|------------------------|------|-------|
| 31:0 | PWM_RNGi | <u>Channel i Range</u> | RW | 0x20 |

DAT2 Register

Synopsis This register stores the 32 bit data to be sent by the PWM Controller when USEFi is 1. In PWM mode data is sent by pulse width modulation: the value of this register defines the number of pulses which is sent within the period defined by PWM_RNGi. In serialiser mode data stored in this register is serialised and transmitted.

Note: Channels 3 and 4 are not available in B0 and corresponding Channel Data Registers are ignored.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|-----------------------|------|-------|
| 31:0 | PWM_DATi | <u>Channel i Data</u> | RW | 0x0 |

10 SPI

10.1 Introduction

This Serial interface peripheral supports the following features:

- Implements a 3 wire serial protocol, variously called Serial Peripheral Interface (SPI) or Synchronous Serial Protocol (SSP).
- Implements a 2 wire version of SPI that uses a single wire as a bidirectional data wire instead of one for each direction as in standard SPI.
- Implements a LoSSI Master (Low Speed Serial Interface)
- Provides support for polled, interrupt or DMA operation.

10.2 SPI Master Mode

10.2.1 Standard mode

In Standard SPI master mode the peripheral implements the standard 3 wire serial protocol described below.

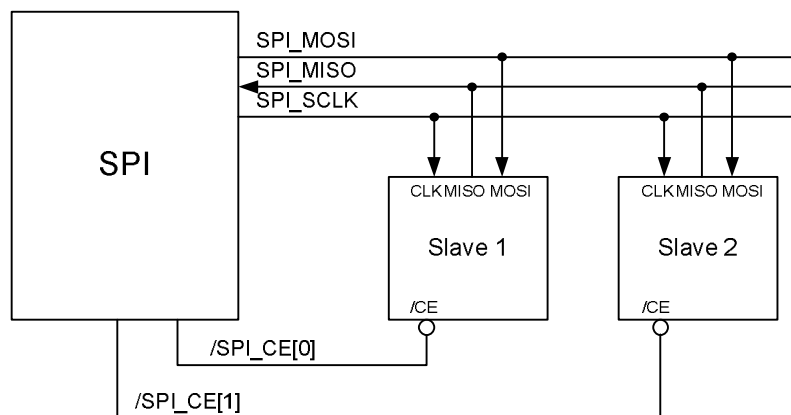
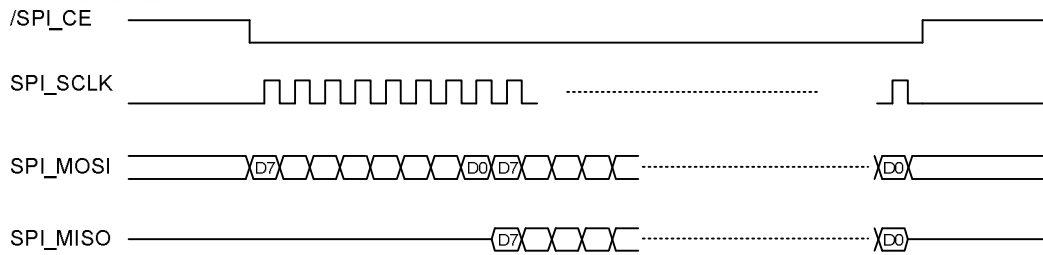


Figure 10-1 SPI Master Typical Usage



Notes

1. Slave enables itself onto SPI_MISO only when it is outputting data. At other times, output is tristate.
2. Different SCLK polarity and phase values are possible. Case illustrated is CPOL = 0, CPHA = 0.
3. Data is transmitted MSB first.
4. Transactions can be from a single byte to hundreds of bytes.

Figure 10-2 SPI Cycle

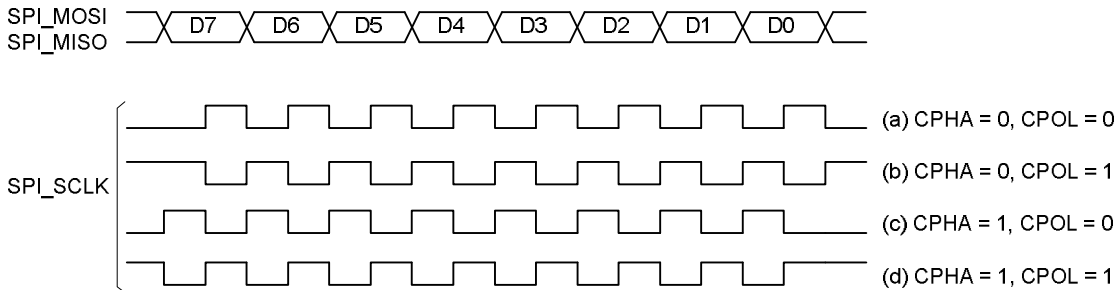


Figure 10-3 Different Clock Polarity/Phase

10.2.2 Bidirectional mode

In bidirectional SPI master mode the same SPI standard is implemented except that a single wire is used for the data (MIMO) instead of the two as in standard mode (MISO and MOSI). Bidirectional mode is used in a similar way to standard mode, the only difference is that before attempting to read data from the slave, you must set the read enable (SPI_REN) bit in the SPI control and status register (SPI_CS). This will turn the bus around, and when you write to the SPI_FIFO register (with junk) a read transaction will take place on the bus, and the read data will appear in the FIFO.

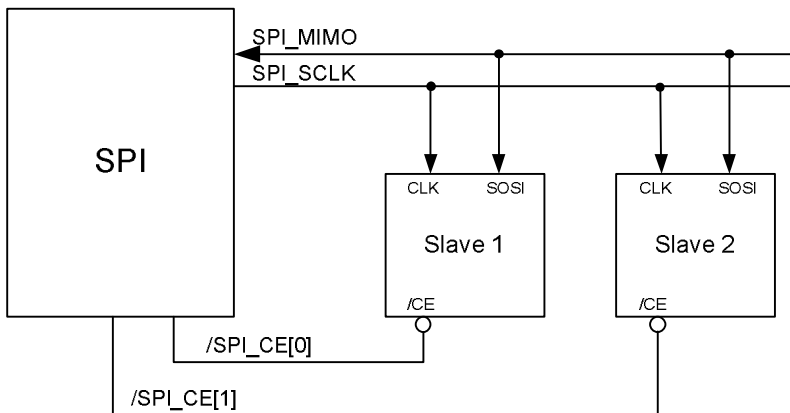


Figure 10-4 Bidirectional SPI Master Typical Usage

10.3 LoSSI mode

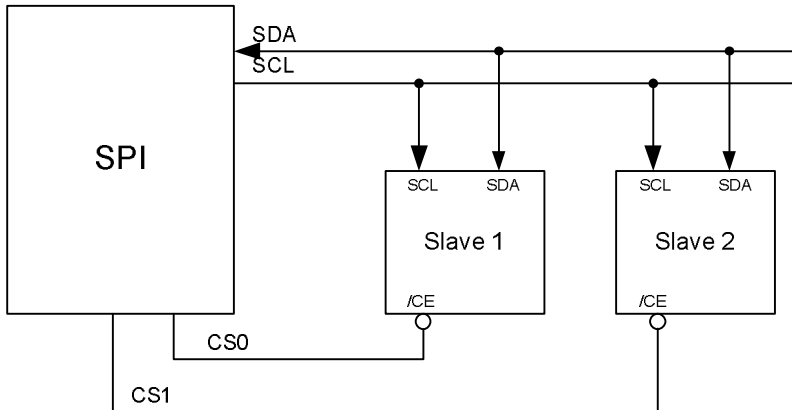
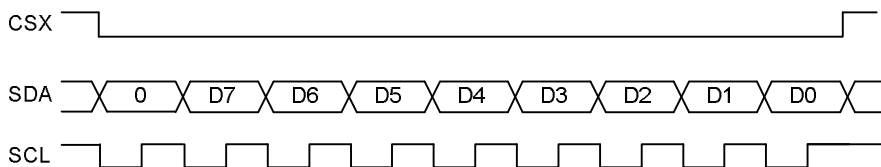


Figure 10-5 LoSSI mode Typical usage

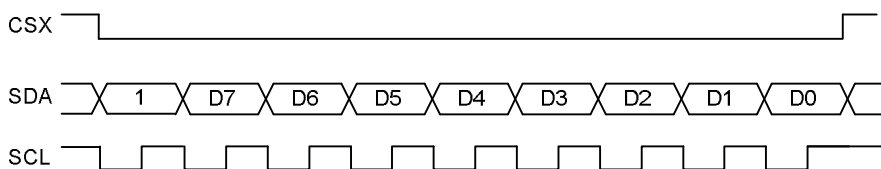
The LoSSI standard allows us to issue commands to peripherals and to transfer data to and from them. LoSSI commands and parameters are 8 bits long, but an extra bit is used to indicate whether the byte is a command or data. This extra bit is set high for a parameter and low for a command. The resulting 9-bit value is serialized to the output. When reading from a LoSSI peripheral the standard allows us to read bytes of data, as well as 24 and 32 bit words.

Commands and parameters are issued to a LoSSI peripheral by writing the 9-bit value of the command or data into the SPI_FIFO register as you would for SPI mode. Reads are automated in that if the serial interface peripheral detects a read command being issued, it will issue the command and complete the read transaction, putting the received data into the FIFO.

10.3.1 Command write

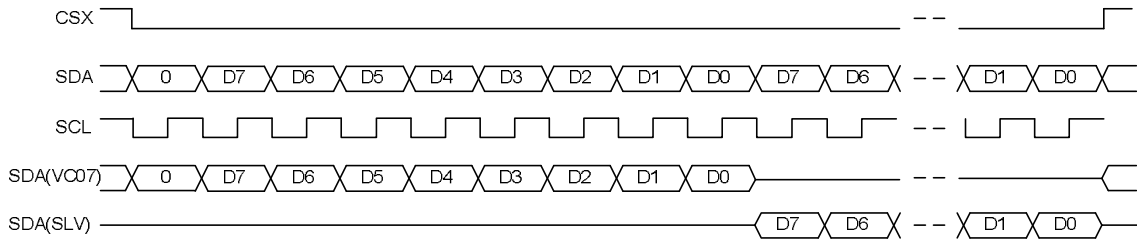


10.3.2 Parameter write



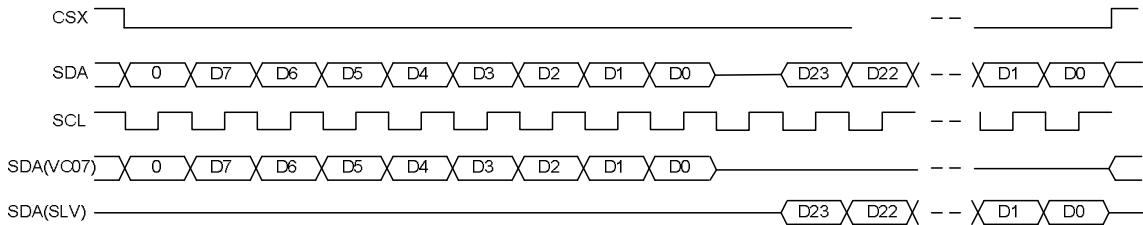
10.3.3 Byte read commands

Byte read commands are 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0xda, 0xdb, 0xdc.



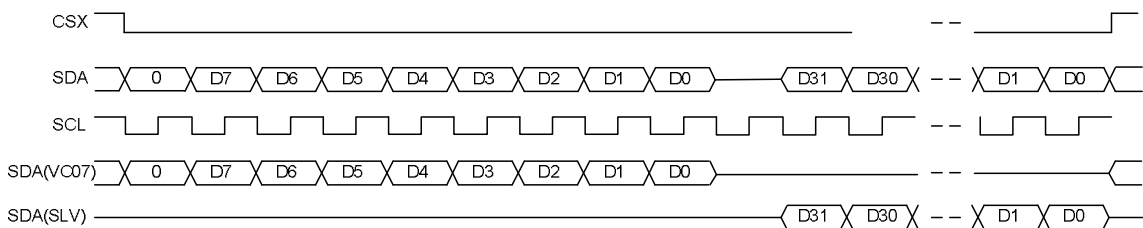
10.3.4 24bit read command

A 24 bit read can be achieved by using the command 0x04.



10.3.5 32bit read command

A 32bit read can be achieved by using the command 0x09.



10.4 Block Diagram

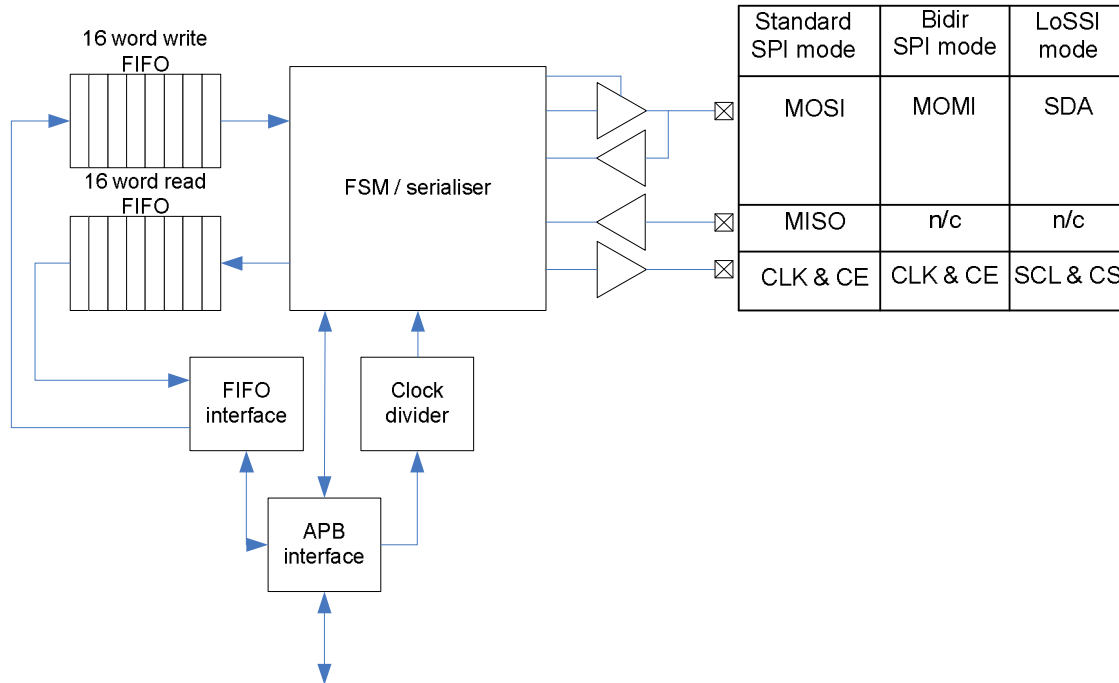


Figure 10-6 Serial interface Block Diagram

10.5 SPI Register Map

The BCM2835 devices has only one SPI interface of this type. It is referred to in all the documentation as SPI0. It has two additional mini SPI interfaces (SPI1 and SPI2). The specification of those can be found under 2.3 *Universal SPI Master (2x)*.

The base address of this SPI0 interface is 0x7E204000.

| SPI Address Map | | | |
|-----------------|----------------------|-------------------------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | CS | SPI Master Control and Status | 32 |
| 0x4 | FIFO | SPI Master TX and RX FIFOs | 32 |
| 0x8 | CLK | SPI Master Clock Divider | 32 |



BCM2835 ARM Peripherals

| | | | |
|------|----------------------|------------------------|----|
| 0xc | DLEN | SPI Master Data Length | 32 |
| 0x10 | LTOH | SPI LOSSI mode TOH | 32 |
| 0x14 | DC | SPI DMA DREQ Controls | 32 |

CS Register

Synopsis This register contains the main control and status bits for the SPI.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:26 | | Reserved - Write as 0, read as don't care | | |
| 25 | LEN_LONG | <u>Enable Long data word in Lossi mode if DMA_LEN is set</u> 0= writing to the FIFO will write a single byte 1= writing to the FIFO will write a 32 bit word | RW | 0x0 |
| 24 | DMA_LEN | <u>Enable DMA mode in Lossi mode</u> | RW | 0x0 |
| 23 | CSPOL2 | <u>Chip Select 2 Polarity</u> 0= Chip select is active low. 1= Chip select is active high. | RW | 0x0 |
| 22 | CSPOL1 | <u>Chip Select 1 Polarity</u> 0= Chip select is active low. 1= Chip select is active high. | RW | 0x0 |
| 21 | CSPOL0 | <u>Chip Select 0 Polarity</u> 0= Chip select is active low. 1= Chip select is active high. | RW | 0x0 |
| 20 | RXF | <u>RXF - RX FIFO Full</u> 0 = RXFIFO is not full. 1 = RX FIFO is full. No further serial data will be sent/ received until data is read from FIFO. | RO | 0x0 |
| 19 | RXR | <u>RXR RX FIFO needs Reading (full)</u> 0 = RX FIFO is less than full (or not active TA = 0). 1 = RX FIFO is or more full. Cleared by reading sufficient data from the RX FIFO or setting TA to 0. | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|----|-------|---|----|-----|
| 18 | TXD | <u>TXD TX FIFO can accept Data</u> 0 = TX FIFO is full and so cannot accept more data. 1 = TX FIFO has space for at least 1 byte. | RO | 0x1 |
| 17 | RXD | <u>RXD RX FIFO contains Data</u> 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 byte. | RO | 0x0 |
| 16 | DONE | <u>Done transfer Done</u> 0 = Transfer is in progress (or not active TA = 0). 1 = Transfer is complete. Cleared by writing more data to the TX FIFO or setting TA to 0. | RO | 0x0 |
| 15 | TE_EN | <u>Unused</u> | RW | 0x0 |
| 14 | LMONO | <u>Unused</u> | RW | 0x0 |
| 13 | LEN | <u>LEN LoSSI enable</u> The serial interface is configured as a LoSSI master. 0 = The serial interface will behave as an SPI master. 1 = The serial interface will behave as a LoSSI master. | RW | 0x0 |
| 12 | REN | <u>REN Read Enable</u> read enable if you are using bidirectional mode. If this bit is set, the SPI peripheral will be able to send data to this device. 0 = We intend to write to the SPI peripheral. 1 = We intend to read from the SPI peripheral. | RW | 0x1 |
| 11 | ADCS | <u>ADCS Automatically Deassert Chip Select</u> 0 = Don t automatically deassert chip select at the end of a DMA transfer chip select is manually controlled by software. 1 = Automatically deassert chip select at the end of a DMA transfer (as determined by SPIDLEN) | RW | 0x0 |
| 10 | INTR | <u>INTR Interrupt on RXR</u> 0 = Don t generate interrupts on RX FIFO condition. 1 = Generate interrupt while RXR = 1. | RW | 0x0 |
| 9 | INTD | <u>INTD Interrupt on Done</u> 0 = Don t generate interrupt on transfer complete. 1 = Generate interrupt when DONE = 1. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|-------|---|----|-----|
| 8 | DMAEN | <u>DMAEN DMA Enable</u> 0 = No DMA requests will be issued. 1 = Enable DMA operation. Peripheral generates data requests. These will be taken in four-byte words until the SPIDLEN has been reached. | RW | 0x0 |
| 7 | TA | <u>Transfer Active</u> 0 = Transfer not active./CS lines are all high (assuming CSPOL = 0). RXR and DONE are 0. Writes to SPIFIFO write data into bits -0 of SPICS allowing DMA data blocks to set mode before sending data. 1 = Transfer active. /CS lines are set according to CS bits and CSPOL. Writes to SPIFIFO write data to TX FIFO.TA is cleared by a dma_frame_end pulse from the DMA controller. | RW | 0x0 |
| 6 | CSPOL | <u>Chip Select Polarity</u> 0 = Chip select lines are active low 1 = Chip select lines are active high | RW | 0x0 |
| 5:4 | CLEAR | <u>CLEAR FIFO Clear</u> 00 = No action. x1 = Clear TX FIFO. One shot operation. 1x = Clear RX FIFO. One shot operation. If CLEAR and TA are both set in the same operation, the FIFOs are cleared before the new frame is started. Read back as 0. | RW | 0x0 |
| 3 | CPOL | <u>Clock Polarity</u> 0 = Rest state of clock = low. 1 = Rest state of clock = high. | RW | 0x0 |
| 2 | CPHA | <u>Clock Phase</u> 0 = First SCLK transition at middle of data bit. 1 = First SCLK transition at beginning of data bit. | RW | 0x0 |
| 1:0 | CS | <u>Chip Select</u> 00 = Chip select 0 01 = Chip select 1 10 = Chip select 2 11 = Reserved | RW | 0x0 |

FIFO Register

Synopsis This register allows TX data to be written to the TX FIFO and RX data to be read from the RX FIFO.



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | DATA | <u>DMA Mode (DMAEN set)</u> If TA is clear, the first 32-bit write to this register will control SPIDLEN and SPICS. Subsequent reads and writes will be taken as four-byte data words to be read/written to the FIFOs <u>Poll/Interrupt Mode (DMAEN clear, TA set)</u> Writes to the register write bytes to TX FIFO. Reads from register read bytes from the RX FIFO | RW | 0x0 |

CLK Register

Synopsis This register allows the SPI clock rate to be set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:16 | | Reserved - Write as 0, read as don't care | | |
| 15:0 | CDIV | <u>Clock Divider</u> $SCLK = \text{Core Clock} / CDIV$ If CDIV is set to 0, the divisor is 65536. The divisor must be a power of 2. Odd numbers rounded down. The maximum SPI clock rate is of the APB clock. | RW | 0x0 |

DLEN Register

Synopsis This register allows the SPI data length rate to be set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | Reserved - Write as 0, read as don't care | | |
| 15:0 | LEN | <u>Data Length</u> The number of bytes to transfer. This field is only valid for DMA mode (DMAEN set) and controls how many bytes to transmit (and therefore receive). | RW | 0x0 |



BCM2835 ARM Peripherals

LTOH Register

Synopsis This register allows the LoSSI output hold delay to be set.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved</i> - Write as 0, read as don't care | | |
| 3:0 | TOH | <u>This sets the Output Hold delay in APB clocks. A value of 0 causes a 1 clock delay.</u> | RW | 0x1 |

DC Register

Synopsis This register controls the generation of the DREQ and Panic signals to an external DMA engine. The DREQ signals are generated when the FIFOs reach their defined levels and need servicing. The Panic signals instruct the external DMA engine to raise the priority of its AXI requests.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:24 | RPANIC | <u>DMA Read Panic Threshold.</u> Generate the Panic signal to the RX DMA engine whenever the RX FIFO level is greater than this amount. | RW | 0x30 |
| 23:16 | RDREQ | <u>DMA Read Request Threshold.</u> Generate a DREQ to the RX DMA engine whenever the RX FIFO level is greater than this amount, (RX DREQ is also generated if the transfer has finished but the RXFIFO isn't empty). | RW | 0x20 |
| 15:8 | TPANIC | <u>DMA Write Panic Threshold.</u> Generate the Panic signal to the TX DMA engine whenever the TX FIFO level is less than or equal to this amount. | RW | 0x10 |
| 7:0 | TDREQ | <u>DMA Write Request Threshold.</u> Generate a DREQ signal to the TX DMA engine whenever the TX FIFO level is less than or equal to this amount. | RW | 0x20 |

10.6 Software Operation

10.6.1 Polled

- a) Set CS, CPOL, CPHA as required and set TA = 1.
- b) Poll TXD writing bytes to SPI_FIFO, RXD reading bytes from SPI_FIFO until all data written.
- c) Poll DONE until it goes to 1.
- d) Set TA = 0.

10.6.2 Interrupt

- e) Set INTR and INTD. These can be left set over multiple operations.
- f) Set CS, CPOL, CPHA as required and set TA = 1. This will immediately trigger a first interrupt with DONE == 1.
- g) On interrupt:
- h) If DONE is set and data to write (this means it is the first interrupt), write up to 16 bytes to SPI_FIFO. If DONE is set and no more data, set TA = 0. Read trailing data from SPI_FIFO until RXD is 0.
- i) If RXR is set read 12 bytes data from SPI_FIFO and if more data to write, write up to 12 bytes to SPI_FIFO.

10.6.3 DMA

Note: In order to function correctly, each DMA channel must be set to perform 32-bit transfers when communicating with the SPI. Either the Source or the Destination Transfer Width field in the DMA TI register must be set to 0 (i.e. 32-bit words) depending upon whether the channel is reading or writing to the SPI.

Two DMA channels are required, one to read from and one to write to the SPI.

- j) Enable DMA DREQ's by setting the DMAEN bit and ADCS if required.
- k) Program two DMA control blocks, one for each DMA controller.
- l) DMA channel 1 control block should have its PER_MAP set to x and should be set to write 'transfer length' + 1 words to SPI_FIFO. The data should comprise:
 - i) A word with the transfer length in bytes in the top sixteen bits, and the control register settings [7:0] in the bottom eight bits (i.e. TA = 1, CS, CPOL, CPHA as required.)
 - ii) 'Transfer length' number in words of data to send.
- m) DMA channel 2 control block should have its PER_MAP set to y and should be set to read 'transfer length' words from SPI_FIFO.
- n) Point each DMA channel at its CB and set its ACTIVE bit to 1.
- o) On receipt of an interrupt from DMA channel 2, the transfer is complete.

10.6.4 Notes

1. The SPI Master knows nothing of the peripherals it is connected to. It always both sends and receives bytes for every byte of the transaction.
2. SCLK is only generated during byte serial transfer. It pauses in the rest state if the next byte to send is not ready or RXF is set.
3. Setup and Hold times related to the automatic assertion and de-assertion of the CS lines when operating in DMA mode (DMAEN and ADCS set) are as follows:

The CS line will be asserted at least 3 core clock cycles before the msb of the first byte of the transfer.

The CS line will be de-asserted no earlier than 1 core clock cycle after the trailing edge of the final clock pulse.

If these parameters are insufficient, software control should alleviate the problem. ADCS should be 0 allowing software to manually control the assertion and de-assertion of the CS lines.



11 SPI/BSC SLAVE

11.1 Introduction

The BSC interface can be used as either a Broadcom Serial Controller (BSC) or a Serial Peripheral Interface (SPI) controller. The BSC bus is a proprietary bus compliant with the Philips® I2C bus/interface version 2.1 January 2000. Both BSC and SPI controllers work in the slave mode. The BSC slave controller has specially built in the Host Control and Software Registers for a Chip booting. The BCS controller supports fast-mode (400Kb/s) and it is compliant to the I²C bus specification version 2.1 January 2000 with the restrictions:

- I²C slave only operation
- clock stretching is not supported
- 7-bit addressing only
-

There is only one BSC/SPI slave. The registers base addresses is 0x7E21_4000.

11.2 Registers

The SPI controller implements 3 wire serial protocol variously called Serial Peripheral Interface (SPI) or Synchronous Serial Protocol (SSP). BSC and SPI controllers do not have DMA connected, hence DMA is not supported.

| I2C_SPI_SLV Address Map | | | |
|-------------------------|----------------------|--|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | DR | Data Register | 32 |
| 0x4 | RSR | The operation status register and error clear register | 32 |
| 0x8 | SLV | The I2C SPI Address Register holds the I2C slave address value | 32 |
| 0xc | CR | The Control register is used to configure the I2C or SPI operation | 32 |
| 0x10 | FR | Flag register | 32 |
| 0x14 | IFLS | Interrupt fifo level select register | 32 |
| 0x18 | IMSC | Interupt Mask Set Clear Register | 32 |



BCM2835 ARM Peripherals

| | | | |
|------|-------------------------|----------------------------------|----|
| 0x1c | RIS | Raw Interrupt Status Register | 32 |
| 0x20 | MIS | Masked Interrupt Status Register | 32 |
| 0x24 | ICR | Interrupt Clear Register | 32 |
| 0x28 | DMACR | DMA Control Register | 32 |
| 0x2c | TDR | FIFO Test Data | 32 |
| 0x30 | GPUSTAT | GPU Status Register | 32 |
| 0x34 | HCTRL | Host Control Register | 32 |
| 0x38 | DEBUG1 | I2C Debug Register | 32 |
| 0x3c | DEBUG2 | SPI Debug Register | 32 |

DR Register

Synopsis The I2C SPI Data Register is used to transfer/receive data characters and provide a Status and Flag information. Status and Flag information is also available via individual registers.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:27 | RXFLEVEL | <u>RXFLEVEL RX FIFO Level</u> Returns the current level of the RX FIFO use | RO | 0x0 |
| 26:22 | TXFLEVEL | <u>TXFLEVEL TX FIFO Level</u> Returns the current level of the TX FIFO use | RO | 0x0 |
| 21 | RXBUSY | <u>RXBUSY Receive Busy</u> 0 Receive operation inactive 1 Receive operation in operation | RO | 0x0 |
| 20 | TXFE | <u>TXFE TX FIFO Empty</u> 0 TX FIFO is not empty 1 When TX FIFO is empty | RO | 0x1 |
| 19 | RXFF | <u>RXFE RX FIFO Full</u> 0 FX FIFO is not full 1 When FX FIFO is full | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-------|--------|--|----|-----|
| 18 | TXFF | <u>TXFF TX FIFO Full</u> 0 TX FIFO is not full 1 When TX FIFO is full | RO | 0x0 |
| 17 | RXFE | <u>RXFE RX FIFO Empty</u> 0 FX FIFO is not empty 1 When FX FIFO is empty | RO | 0x1 |
| 16 | TXBUSY | <u>TXBUSY Transmit Busy</u> 0 Transmit operation inactive 1 Transmit operation in operation | RO | 0x0 |
| 15:10 | | Reserved - Write as 0, read as don't care | | |
| 9 | UE | <u>TXUE TX Underrun Error</u> 0 - No error case detected 1 Set when TX FIFO is empty and I2C master attempt to read a data character from I2C slave. Cleared by writing 0 to I2C SPI Status register . | RO | 0x0 |
| 8 | OE | <u>RXOE RX Overrun Error</u> 0 No error case detected 1 Set when RX FIFO is full and a new data character is received. Cleared by writing 0 to I2C SPI Status register . | RO | 0x0 |
| 7:0 | DATA | <u>DATA Received/Transferred data characters</u> Data written to this location is pushed into the TX FIFO. Data read from this location is fetched from the RX FIFO. | RW | 0x0 |

RSR Register

Synopsis The operation status register and error clear register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:6 | | Reserved - Write as 0, read as don't care | | |
| 5 | RXDMABREQ | <u>Unsupported, write zero, read as don't care</u> | RO | 0x0 |
| 4 | RXDMAPREQ | <u>Unsupported, write zero, read as don't care</u> | RO | 0x0 |
| 3 | TXDMABREQ | <u>Unsupported, write zero, read as don't care</u> | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|----------|--|----|-----|
| 2 | TXDMPREQ | <u>Unsupported, write zero, read as don't care</u> | RO | 0x0 |
| 1 | UE | <u>TXUE TX Underrun Error</u> 0 - No error case detected 1 Set when TX FIFO is empty and I2C master attempt to read a data character from I2C slave. Cleared by writing 0 to it. | RW | 0x0 |
| 0 | OE | <u>RXOE RX Overrun Error</u> 0 No error case detected 1 Set when RX FIFO is full and a new data character is received. Cleared by writing 0 to it. | RW | 0x0 |

SLV Register

Synopsis The I2C SPI Address Register holds the I2C slave address value. NOTE: It is of no use in SPI mode.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:7 | | Reserved - Write as 0, read as don't care | | |
| 6:0 | ADDR | <u>SLVADDR I2C Slave Address</u> Programmable I2C slave address Note: In case HOSTCTRLLEN bit is set from the I2C SPI Control Register bit SLVADDR[0] chooses the following: 0 - selects normal operation, i.e. accessing RX and TX FIFOs. 1 - selects access to I2C SPI SW Status Register or I2C SPI Host Control Register | RW | 0x0 |

CR Register

Synopsis The Control register is used to configure the I2C or SPI operation.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:14 | | Reserved - Write as 0, read as don't care | | |



BCM2835 ARM Peripherals

| | | | | |
|----|-------------|---|----|-----|
| 13 | INV_TXF | <u>INV-RX Inverse TX status flags</u> 0 = default status flags When this bit is 0, bit 6 (TXFE - TX FIFO Empty) will reset to a 1 1 = inverted status flags When this bit is set, bit 6 (TXFE - TX FIFO Full) will reset to a 0 * Note: INV_TX bit changes the default values of 6 bit as it is specified for I2C SPI GPU Host Status Register . | RW | 0x0 |
| 12 | HOSTCTRLLEN | <u>HOSTCTRLLEN Enable Control for Host</u> 0 = Host Control disabled 1 = Host Control enabled Note: HOSTCTRLLEN allows Host to request GPUSTAT or HCTRL register. The same behaviour is achieved from the GPU side using ENSTAT and ENCTRL. | RW | 0x0 |
| 11 | TESTFIFO | <u>TESTFIFO TEST FIFO</u> 0 = TESTT FIFO disabled 1 = TESTT FIFO enabled | RW | 0x0 |
| 10 | INV_RXF | <u>INV-RX Inverse RX status flags</u> 0 = default status flags When this bit is 0, bit 6 (RXFF - RX FIFO Full) will reset to a 0 1 = inverted status flags When this bit is 0, bit 6 (RXFF - RX FIFO Empty) will reset to a 1 * NOTE: INV_RX bit changes the default values of 7 bit as it is specified for I2C SPI GPU Host Status Register . | RW | 0x0 |
| 9 | RXE | <u>RXE Receive Enable</u> 0 = Receive mode disabled 1 = Receive mode enabled | RW | 0x0 |
| 8 | TXE | <u>TXE Transmit Enable</u> 0 = Transmit mode disabled 1 = Transmit mode enabled | RW | 0x0 |
| 7 | BRK | <u>BRK Break current operation</u> 0 = No effect. 1 = Stop operation and clear the FIFOs. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|--------|--|----|-----|
| 6 | ENCTRL | <u>ENCTRL ENABLE CONTROL 8bit register</u> 0 = Control register disabled. Implies ordinary I2C protocol. 1 = Control register enabled. When enabled the control register is received as a first data character on the I2C bus. NOTE: The same behaviour is achieved from the Host side by using bit SLVADDR[6] of the slave address. | RO | 0x0 |
| 5 | ENSTAT | <u>ENSTAT ENABLE STATUS 8bit register</u> 0 = Status register disabled. Implies ordinary I2C protocol. 1 = Status register enabled. When enabled the status register is transferred as a first data character on the I2C bus. Status register is transferred to the host. NOTE: The same behaviour is achieved from the Host side by using bit SLVADDR[6] of the slave address. | RW | 0x0 |
| 4 | CPOL | <u>CPOL Clock Polarity</u> 0 = 1 = SPI Related | RW | 0x0 |
| 3 | CPHA | <u>CPHA Clock Phase</u> 0 = 1 = SPI Related | RW | 0x0 |
| 2 | I2C | <u>SPI Mode</u> 0 = Disabled I2C mode 1 = Enabled I2C mode | RW | 0x0 |
| 1 | SPI | <u>SPI Mode</u> 0 = Disabled SPI mode 1 = Enabled SPI mode | RW | 0x0 |
| 0 | EN | <u>EN Enable Device</u> 1 = Enable I2C SPI Slave. 0 = Disable I2C SPI Slave. | RW | 0x0 |

FR Register

Synopsis The flag register indicates the current status of the operation.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | Reserved - Write as 0, read as don't care | | |



BCM2835 ARM Peripherals

| | | | | |
|-------|----------|---|----|-----|
| 15:11 | RXFLEVEL | <u>RXFLEVEL RX FIFO Level</u> Returns the current level of the RX FIFO use | RW | 0x0 |
| 10:6 | TXFLEVEL | <u>TXFLEVEL TX FIFO Level</u> Returns the current level of the TX FIFO use | RW | 0x0 |
| 5 | RXBUSY | <u>RXBUSY Receive Busy</u> 0 Receive operation inactive 1 Receive operation in operation | RW | 0x0 |
| 4 | TXFE | <u>TXFE TX FIFO Empty</u> 0 TX FIFO is not empty 1 When TX FIFO is empty | RW | 0x1 |
| 3 | RXFF | <u>RXFE RX FIFO Full</u> 0 FX FIFO is not full 1 When FX FIFO is full | RW | 0x0 |
| 2 | TXFF | <u>TXFF TX FIFO Full</u> 0 TX FIFO is not full 1 When TX FIFO is full | RW | 0x0 |
| 1 | RXFE | <u>RXFE RX FIFO Empty</u> 0 FX FIFO is not empty 1 When FX FIFO is empty | RW | 0x1 |
| 0 | TXBUSY | <u>TXBUSY Transmit Busy</u> 0 Transmit operation inactive 1 Transmit operation in operation | RW | 0x0 |

IFLS Register

Synopsis The flag register indicates the current status of the operation.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:12 | | Reserved - Write as 0, read as don't care | | |
| 11:9 | RXIFPSEL | <u>Unsupported, write zero, read as don't care</u> | RO | 0x0 |
| 8:6 | TXIFPSEL | <u>Unsupported, write zero, read as don't care</u> | RO | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|----------|--|----|-----|
| 5:3 | RXIFLSEL | <u>RXIFLSEL RX Interrupt FIFO Level Select</u> Interrupt is triggered when : 000 RX FIFO gets 1/8 full 001 RX FIFO gets 1/4 full 010 RX FIFO gets 1/2 full 011 RX FIFO gets 3/4 full 100 RX FIFO gets 7/8 full 101 111 not used | RW | 0x0 |
| 2:0 | TXIFLSEL | <u>TXIFLSEL TX Interrupt FIFO Level Select</u> Interrupt is triggered when : 000 TX FIFO gets 1/8 full 001 TX FIFO gets 1/4 full 010 TX FIFO gets 1/2 full 011 TX FIFO gets 3/4 full 100 TX FIFO gets 7/8 full 101 111 not used | RW | 0x0 |

IMSC Register

Synopsis Interrupt Mask Set/Clear Register. On a read this register returns the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | OEIM | Overrun error interrupt mask. A read returns the current mask for the interrupt. On a write of 1, the mask of the OEINTR interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 2 | BEIM | Break error interrupt mask. A read returns the current mask for the BEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 1 | TXIM | Transmit interrupt mask. A read returns the current mask for the TXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 0 | RXIM | Receive interrupt mask. A read returns the current mask for the RXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |



BCM2835 ARM Peripherals

RIS Register

Synopsis The Raw Interrupt Status Register returns the current raw status value, prior to masking, of the corresponding interrupt.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | OERIS | Overrun error interrupt status. Returns the raw interrupt state of the OEINTR interrupt. | RW | 0x0 |
| 2 | BERIS | Break error interrupt status. Returns the raw interrupt state of the BEINTR interrupt. | RW | 0x0 |
| 1 | TXRIS | Transmit interrupt status. Returns the raw interrupt state of the TXINTR interrupt. | RW | 0x0 |
| 0 | RXRIS | Receive interrupt status. Returns the raw interrupt state of the RXINTR interrupt. | RW | 0x0 |

MIS Register

Synopsis The Masked Interrupt Status Register returns the current masked status value of the corresponding interrupt.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | OEMIS | Overrun error masked interrupt status. Returns the masked interrupt state of the OEINTR interrupt. | RW | 0x0 |
| 2 | BEMIS | Break error masked interrupt status. Returns the masked interrupt state of the BEINTR interrupt. | RW | 0x0 |
| 1 | TXMIS | Transmit masked interrupt status. Returns the masked interrupt state of the TXINTR interrupt. | RW | 0x0 |
| 0 | RXMIS | Receive masked interrupt status. Returns the masked interrupt state of the RXINTR interrupt. | RW | 0x0 |



BCM2835 ARM Peripherals

ICR Register

Synopsis The Interrupt Clear Register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | OEIC | Overrun error interrupt clear. Clears the OEINTR interrupt. | RW | 0x0 |
| 2 | BEIC | Break error interrupt clear. Clears the BEINTR interrupt. | RW | 0x0 |
| 1 | TXIC | Transmit interrupt clear. Clears the TXINTR interrupt. | RW | 0x0 |
| 0 | RXIC | Receive masked interrupt status. Returns the masked interrupt state of the RXINTR interrupt. | RW | 0x0 |

DMACR Register

Synopsis The DMA Control register is not supported in this version.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2 | DMAONERR | Unsupported, write zero, read as don't care | RW | 0x0 |
| 1 | TXDMAE | Unsupported, write zero, read as don't care | RW | 0x0 |
| 0 | RXDMAE | Unsupported, write zero, read as don't care | RW | 0x0 |

TDR Register

Synopsis The Test Data Register enables data to be written into the receive FIFO and read out from the transmit FIFO for test purposes.



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:8 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 7:0 | DATA | Test data is written into the receive FIFO and read out of the transmit FIFO. | RW | 0x0 |

GPUSTAT Register

Synopsis The GPU SW Status Register to be passed via I2C bus to a Host.
NOTE: GPU SW Status Register is combined with the status bit coming from within I2C SPI Slave device. Hence, the I2C SPI GPU Host Status Register as it is seen by a Host is depicted on Table 1 14.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3:0 | DATA | <u>GPUSTAT GPU to Host Status Register</u> SW controllable | RW | 0x0 |

HCTRL Register

Synopsis The Host Control register is received from the host side via I2C bus. When ENCTRL - enable control register bit is set, the host control register is received as the first data character after the I2C address.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:8 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 7:0 | DATA | <u>HCTRL Host Control Register</u> SW processing received via I2C bus | RW | 0x0 |

DEBUG1 Register

Synopsis I2C Debug Register



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:26 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 25:0 | DATA | | RW | 0xe |

DEBUG2 Register

Synopsis SPI Debug Register

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|----------|
| 31:24 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 23:0 | DATA | | RW | 0x400000 |

12 System Timer

The System Timer peripheral provides four 32-bit timer channels and a single 64-bit free running counter. Each channel has an output compare register, which is compared against the 32 least significant bits of the free running counter values. When the two values match, the system timer peripheral generates a signal to indicate a match for the appropriate channel. The match signal is then fed into the interrupt controller. The interrupt service routine then reads the output compare register and adds the appropriate offset for the next timer tick. The free running counter is driven by the timer clock and stopped whenever the processor is stopped in debug mode.

The Physical (hardware) base address for the system timers is 0x7E003000.

12.1 System Timer Registers

| ST Address Map | | | |
|----------------|---------------------|-------------------------------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | CS | System Timer Control/Status | 32 |
| 0x4 | CLO | System Timer Counter Lower 32 bits | 32 |
| 0x8 | CHI | System Timer Counter Higher 32 bits | 32 |
| 0xc | C0 | System Timer Compare 0 | 32 |
| 0x10 | C1 | System Timer Compare 1 | 32 |
| 0x14 | C2 | System Timer Compare 2 | 32 |
| 0x18 | C3 | System Timer Compare 3 | 32 |

CS Register

Synopsis System Timer Control / Status.

This register is used to record and clear timer channel comparator matches. The system timer match bits are routed to the interrupt controller where they can generate an interrupt.

The M0-3 fields contain the free-running counter match status. Write a one to the relevant bit to clear the match detect status bit and the corresponding interrupt request line.

© 2012 Broadcom Corporation.
All rights reserved

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | M3 | <u>System Timer Match 3</u> 0 = No Timer 3 match since last cleared. 1 = Timer 3 match detected. | RW | 0x0 |
| 2 | M2 | <u>System Timer Match 2</u> 0 = No Timer 2 match since last cleared. 1 = Timer 2 match detected. | RW | 0x0 |
| 1 | M1 | <u>System Timer Match 1</u> 0 = No Timer 1 match since last cleared. 1 = Timer 1 match detected. | RW | 0x0 |
| 0 | M0 | <u>System Timer Match 0</u> 0 = No Timer 0 match since last cleared. 1 = Timer 0 match detected. | RW | 0x0 |

CLO Register

Synopsis System Timer Counter Lower bits.

The system timer free-running counter lower register is a read-only register that returns the current value of the lower 32-bits of the free running counter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | CNT | <u>Lower 32-bits of the free running counter value.</u> | RW | 0x0 |

CHI Register

Synopsis System Timer Counter Higher bits.

The system timer free-running counter higher register is a read-only register that returns the current value of the higher 32-bits of the free running counter.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | CNT | <u>Higher 32-bits of the free running counter value.</u> | RW | 0x0 |

C0 C1 C2 C3 Register

Synopsis System Timer Compare.

The system timer compare registers hold the compare value for each of the four timer channels. Whenever the lower 32-bits of the free-running counter matches one of the compare values the corresponding bit in the system timer control/status register is set.

Each timer peripheral (minirun and run) has a set of four compare registers.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:0 | CMP | <u>Compare value for match channel n.</u> | RW | 0x0 |

13 UART

The BCM2835 device has two UARTS. One mini UART and one PL011 UART. This section describes the PL011 UART. For details of the mini UART see 2.2 *Mini UART*.

The PL011 UART is a Universal Asynchronous Receiver/Transmitter. This is the ARM UART (PL011) implementation. The UART performs serial-to-parallel conversion on data characters received from an external peripheral device or modem, and parallel-to-serial conversion on data characters received from the Advanced Peripheral Bus (APB).

The ARM PL011 UART has some optional functionality which can be included or left out.

The following functionality is *not supported* :

- Infrared Data Association (IrDA)
- Serial InfraRed (SIR) protocol Encoder/Decoder (ENDEC)
- Direct Memory Access (DMA).

The UART provides:

- Separate 16x8 transmit and 16x12 receive FIFO memory.
- Programmable baud rate generator.
- Standard asynchronous communication bits (start, stop and parity). These are added prior to transmission and removed on reception.
- False start bit detection.
- Line break generation and detection.
- Support of the modem control functions CTS and RTS. However DCD, DSR, DTR, and RI are not supported.
- Programmable hardware flow control.
- Fully-programmable serial interface characteristics:
 - data can be 5, 6, 7, or 8 bits
 - even, odd, stick, or no-parity bit generation and detection
 - 1 or 2 stop bit generation
 - baud rate generation, dc up to $UARTCLK/16$

The UART clock source and associated dividers are controlled by the Clock Manager.

For the in-depth UART overview, please, refer to the ARM PrimeCell UART (PL011) Revision: r1p5 Technical Reference Manual.

13.1 Variations from the 16C650 UART

The UART varies from the industry-standard 16C650 UART device as follows:

- Receive FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, and 7/8
- Transmit FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, and 7/8
- The internal register map address space, and the bit function of each register differ



- The deltas of the modem status signals are not available.

The following 16C650 UART features are not supported:

- 1.5 stop bits (1 or 2 stop bits only are supported)
- Independent receive clock.

13.2 Primary UART Inputs and Outputs

The UART has two primary inputs RXD, nCTS and two primary outputs TXD, nRTS. The remaining signals like SRIN, SROUT, OUT1, OUT2, DSR, DTR, and RI are not supported in this implementation. The following table shows the UART signals map on the General Purpose I/O (GPIO). For the insight on how to program alternate function refer to the GPIO paragraph.

| | Pull | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 |
|--------|------|------|------|------|------|------|------|
| GPIO14 | Low | TXD0 | | | | | |
| GPIO15 | Low | RXD0 | | | | | |
| GPIO16 | Low | | | | CTS0 | | |
| GPIO17 | Low | | | | RTS0 | | |
| GPIO30 | Low | | | | CTS0 | | |
| GPIO31 | Low | | | | RTS0 | | |
| GPIO32 | Low | | | | TXD0 | | |
| GPIO33 | Low | | | | RXD0 | | |
| GPIO36 | High | | | TXD0 | | | |
| GPIO37 | Low | | | RXD0 | | | |
| GPIO38 | Low | | | RTS0 | | | |
| GPIO39 | Low | | | CTS0 | | | |

Table 13-1 UART Assignment on the GPIO Pin map

13.3 UART Interrupts

The UART has one intra-chip interrupt UARTINTR generated as the OR-ed function of the five individual interrupts.

- UARTINTR, this is an OR function of the five individual masked outputs:
 - UARTRXINTR
 - UARTRTXINTR
 - UARTRTINTR
 - UARTMSINTR, that can be caused by:
 - UARTCTSINTR, because of a change in the nUARTCTS modem status
 - UARTDSRINTR, because of a change in the nUARTDSR modem status.
 - UARTEINTR, that can be caused by an error in the reception:



BCM2835 ARM Peripherals

- UARTOEINTR, because of an overrun error
- UARTBEINTR, because of a break in the reception
- UARTPEINTR, because of a parity error in the received character
- UARTFEINTR, because of a framing error in the received character.

One can enable or disable the individual interrupts by changing the mask bits in the Interrupt Mask Set/Clear Register, UART_IMSC. Setting the appropriate mask bit HIGH enables the interrupt.

UARTRXINTR:

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO is equal to or lower than the programmed trigger level then the transmit interrupt is asserted HIGH. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the transmit interrupt is asserted HIGH. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt.

UARTRTINTR:

The receive interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level. When this happens, the receive interrupt is asserted HIGH. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the receive interrupt is asserted HIGH. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt.

13.4 Register View

The PL011 USRT is mapped on base address 0x7E20100. It has the following memory-mapped registers.

| UART Address Map | | | |
|------------------|------------------------|---------------|------|
| Address Offset | Register Name | Description | Size |
| 0x0 | DR | Data Register | 32 |
| 0x4 | RSRECR | | 32 |
| 0x18 | FR | Flag register | 32 |



BCM2835 ARM Peripherals

| | | | |
|------|-----------------------|-------------------------------------|----|
| 0x20 | ILPR | not in use | 32 |
| 0x24 | IBRD | Integer Baud rate divisor | 32 |
| 0x28 | FBRD | Fractional Baud rate divisor | 32 |
| 0x2c | LCRH | Line Control register | 32 |
| 0x30 | CR | Control register | 32 |
| 0x34 | IFLS | Interupt FIFO Level Select Register | 32 |
| 0x38 | IMSC | Interupt Mask Set Clear Register | 32 |
| 0x3c | RIS | Raw Interupt Status Register | 32 |
| 0x40 | MIS | Masked Interupt Status Register | 32 |
| 0x44 | ICR | Interupt Clear Register | 32 |
| 0x48 | DMACR | DMA Control Register | 32 |
| 0x80 | ITCR | Test Control register | 32 |
| 0x84 | ITIP | Integration test input reg | 32 |
| 0x88 | ITOP | Integration test output reg | 32 |
| 0x8c | TDR | Test Data reg | 32 |

DR Register



BCM2835 ARM Peripherals

Synopsis The UART_DR Register is the data register. For words to be transmitted:
if the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO.
if the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO).
The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted.
For received words:
if the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO
if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:12 | | Reserved - Write as 0, read as don't care | | |
| 11 | OE | Overrun error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it. | RW | 0x0 |
| 10 | BE | Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received. | RW | 0x0 |
| 9 | PE | Parity error. When set to 1, it indicates that the parity of the received data character does not match the parity that the EPS and SPS bits in the Line Control Register, UART_LCRH select. In FIFO mode, this error is associated with the character at the top of the FIFO. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|------|---|----|-----|
| 8 | FE | Framing error. When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO. | RW | 0x0 |
| 7:0 | DATA | Receive (read) data character. Transmit (write) data character. | RW | 0x0 |

RSRECR Register

Synopsis The UART_RSRECR Register is the receive status register/error clear register. If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from the Data Register, UART_DR. The status information for overrun is set immediately when an overrun condition occurs. NOTE: The received data character must be read first from the Data Register, UART_DR on before reading the error status associated with that data character from this register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | OE | Overrun error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it. | RW | 0x0 |

| | | | | |
|---|----|--|----|-----|
| 2 | BE | Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received. | RW | 0x0 |
| 1 | PE | Parity error. When set to 1, it indicates that the parity of the received data character does not match the parity that the EPS and SPS bits in the Line Control Register, UART_LCRH select. In FIFO mode, this error is associated with the character at the top of the FIFO. | RW | 0x0 |
| 0 | FE | Framing error. When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO. | RW | 0x0 |

FR Register

Synopsis The UART_FR Register is the flag register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:9 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 8 | RI | Unsupported, write zero, read as don't care | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|------|--|----|-----|
| 7 | TXFE | <p>Transmit FIFO empty. The meaning of this bit depends on the state of the FEN bit in the Line Control Register, UARTLCR_LCRH.</p> <p>If the FIFO is disabled, this bit is set when the transmit holding register is empty.</p> <p>If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty. This bit does not indicate if there is data in the transmit shift register.</p> | RW | 0x1 |
| 6 | RXFF | <p>Receive FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_LCRH Register.</p> <p>If the FIFO is disabled, this bit is set when the receive holding register is full.</p> <p>If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.</p> | RW | 0x0 |
| 5 | TXFF | <p>Transmit FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_LCRH Register.</p> <p>If the FIFO is disabled, this bit is set when the transmit holding register is full.</p> <p>If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.</p> | RW | 0x0 |
| 4 | RXFE | <p>Receive FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H Register.</p> <p>If the FIFO is disabled, this bit is set when the receive holding register is empty.</p> <p>If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.</p> | RW | 0x0 |
| 3 | BUSY | <p>UART busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register.</p> <p>This bit is set as soon as the transmit FIFO becomes non-empty, regardless of whether the UART is enabled or not.</p> | RW | 0x0 |
| 2 | DCD | Unsupported, write zero, read as don't care | RW | 0x0 |
| 1 | DSR | Unsupported, write zero, read as don't care | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|-----|--|----|-----|
| 0 | CTS | Clear to send. This bit is the complement of the UART clear to send, nUARTCTS, modem status input. That is, the bit is 1 when nUARTCTS is LOW. | RW | 0x0 |
|---|-----|--|----|-----|

ILPR Register

Synopsis This is the disabled IrDA register, writing to it has not effect and reading returns 0.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:0 | ILPR | Reserved - write zero, read as don't care. | RW | 0x0 |

IBRD Register

Synopsis The UART_IBRD Register is the integer part of the baud rate divisor value.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 15:0 | IBRD | The integer baud rate divisor. | RW | 0x0 |

FBRD Register

Synopsis The UART_FBRD Register is the fractional part of the baud rate divisor value. The baud rate divisor is calculated as follows:
 Baud rate divisor BAUDDIV = (FUARTCLK/(16 Baud rate))
 where FUARTCLK is the UART reference clock frequency. The BAUDDIV is comprised of the integer value IBRD and the fractional value FBRD. NOTE: The contents of the IBRD and FBRD registers are not updated until transmission or reception of the current character is complete.



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:6 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 5:0 | FBRD | The fractional baud rate divisor. | RW | 0x0 |

LCRH Register

Synopsis The UARTLCR_LCRH Register is the line control register.
 NOTE: The UART_LCRH, UART_IBRD, and UART_FBRD registers must not be changed:
 when the UART is enabled
 when completing a transmission or a reception when it has been programmed to become disabled.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:8 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 7 | SPS | Stick parity select. 0 = stick parity is disabled 1 = either: if the EPS bit is 0 then the parity bit is transmitted and checked as a 1 if the EPS bit is 1 then the parity bit is transmitted and checked as a 0. See Table 25 9. | RO | 0x0 |
| 6:5 | WLEN | Word length. These bits indicate the number of data bits transmitted or received in a frame as follows: b11 = 8 bits b10 = 7 bits b01 = 6 bits b00 = 5 bits. | RW | 0x0 |
| 4 | FEN | Enable FIFOs: 0 = FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers 1 = transmit and receive FIFO buffers are enabled (FIFO mode). | RW | 0x0 |

| | | | | |
|---|------|--|----|-----|
| 3 | STP2 | Two stop bits select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received. | RW | 0x0 |
| 2 | EPS | Even parity select. Controls the type of parity the UART uses during transmission and reception: 0 = odd parity. The UART generates or checks for an odd number of 1s in the data and parity bits. 1 = even parity. The UART generates or checks for an even number of 1s in the data and parity bits. This bit has no effect when the PEN bit disables parity checking and generation. See Table 25 9. | RW | 0x0 |
| 1 | PEN | Parity enable: 0 = parity is disabled and no parity bit added to the data frame 1 = parity checking and generation is enabled. See Table 25 9. | RW | 0x0 |
| 0 | BRK | Send break. If this bit is set to 1, a low-level is continually output on the TXD output, after completing transmission of the current character. | RW | 0x0 |

CR Register

Synopsis The UART_CR Register is the control register.

NOTE:

To enable transmission, the TXE bit and UARTEN bit must be set to 1. Similarly, to enable reception, the RXE bit and UARTEN bit, must be set to 1.

NOTE:

Program the control registers as follows:

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by setting the FEN bit to 0 in the Line Control Register, UART_LCRH.
4. Reprogram the Control Register, UART_CR.
5. Enable the UART.



BCM2835 ARM Peripherals

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:16 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 15 | CTSEN | CTS hardware flow control enable. If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the nUARTCTS signal is asserted. | RW | 0x0 |
| 14 | RTSEN | RTS hardware flow control enable. If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received. | RW | 0x0 |
| 13 | OUT2 | Unsupported, write zero, read as don't care | RO | 0x0 |
| 12 | OUT1 | Unsupported, write zero, read as don't care | RO | 0x0 |
| 11 | RTS | Request to send. This bit is the complement of the UART request to send, nUARTRTS, modem status output. That is, when the bit is programmed to a 1 then nUARTRTS is LOW. | RW | 0x0 |
| 10 | DTR | Unsupported, write zero, read as don't care | RO | 0x0 |
| 9 | RXE | Receive enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for UART signals. When the UART is disabled in the middle of reception, it completes the current character before stopping. | RW | 0x1 |
| 8 | TXE | Transmit enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for UART signals. When the UART is disabled in the middle of transmission, it completes the current character before stopping. | RW | 0x1 |
| 7 | LBE | Loopback enable. If this bit is set to 1, the UARTTXD path is fed through to the UARTRXD path. In UART mode, when this bit is set, the modem outputs are also fed through to the modem inputs. This bit is cleared to 0 on reset, to disable loopback. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|--------|--|----|-----|
| 6:3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2 | SIRLP | Unsupported, write zero, read as don't care | RO | 0x0 |
| 1 | SIREN | Unsupported, write zero, read as don't care | RO | 0x0 |
| 0 | UARTEN | <u>UART enable:</u> 0 = UART is disabled. If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping. 1 = the UART is enabled. | RW | 0x0 |

IFLS Register

Synopsis The UART_IFLS Register is the interrupt FIFO level select register. You can use this register to define the FIFO level that triggers the assertion of the combined interrupt signal. The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. The bits are reset so that the trigger level is when the FIFOs are at the half-way mark.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:12 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 11:9 | RXIFPSEL | Unsupported, write zero, read as don't care | RO | 0x0 |
| 8:6 | TXIFPSEL | Unsupported, write zero, read as don't care | RO | 0x0 |
| 5:3 | RXIFLSEL | Receive interrupt FIFO level select. The trigger points for the receive interrupt are as follows: b000 = Receive FIFO becomes 1/8 full b001 = Receive FIFO becomes 1/4 full b010 = Receive FIFO becomes 1/2 full b011 = Receive FIFO becomes 3/4 full b100 = Receive FIFO becomes 7/8 full b101-b111 = reserved. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|----------|---|----|-----|
| 2:0 | TXIFLSEL | Transmit interrupt FIFO level select. The trigger points for the transmit interrupt are as follows: b000 = Transmit FIFO becomes 1/8 full b001 = Transmit FIFO becomes 1/4 full b010 = Transmit FIFO becomes 1/2 full b011 = Transmit FIFO becomes 3/4 full b100 = Transmit FIFO becomes 7/8 full b101-b111 = reserved. | RW | 0x0 |
|-----|----------|---|----|-----|

IMSC Register

Synopsis The UART_IMSC Register is the interrupt mask set/clear register. It is a read/write register. On a read this register returns the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:11 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 10 | OEIM | Overrun error interrupt mask. A read returns the current mask for the interrupt. On a write of 1, the mask of the UARTOEINTR interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 9 | BEIM | Break error interrupt mask. A read returns the current mask for the UARTBEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 8 | PEIM | Parity error interrupt mask. A read returns the current mask for the UARTPEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 7 | FEIM | Framing error interrupt mask. A read returns the current mask for the UARTFEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|---|--------|--|----|-----|
| 6 | RTIM | Receive timeout interrupt mask. A read returns the current mask for the UARTRTINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 5 | TXIM | Transmit interrupt mask. A read returns the current mask for the UARTTXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 4 | RXIM | Receive interrupt mask. A read returns the current mask for the UARTRXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 3 | DSRMIM | Unsupported, write zero, read as don't care | RO | 0x0 |
| 2 | DCDMIM | Unsupported, write zero, read as don't care | RO | 0x0 |
| 1 | CTSMIM | nUARTCTS modem interrupt mask. A read returns the current mask for the UARTCTSINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask. | RW | 0x0 |
| 0 | RIMIM | Unsupported, write zero, read as don't care | RO | 0x0 |

RIS Register

Synopsis The UART_RIS Register is the raw interrupt status register. It is a read-only register. This register returns the current raw status value, prior to masking, of the corresponding interrupt.
 NOTE: All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:11 | | <i>Reserved - Write as 0, read as don't care</i> | | |



BCM2835 ARM Peripherals

| | | | | |
|----|---------|--|----|-----|
| 10 | OERIS | Overrun error interrupt status. Returns the raw interrupt state of the UARTOEINTR interrupt. | RW | 0x0 |
| 9 | BERIS | Break error interrupt status. Returns the raw interrupt state of the UARTBEINTR interrupt. | RW | 0x0 |
| 8 | PERIS | Parity error interrupt status. Returns the raw interrupt state of the UARTPEINTR interrupt. | RW | 0x0 |
| 7 | FERIS | Framing error interrupt status. Returns the raw interrupt state of the UARTFEINTR interrupt. | RW | 0x0 |
| 6 | RTRIS | Receive timeout interrupt status. Returns the raw interrupt state of the UARTRTINTR interrupt. | RW | 0x0 |
| 5 | TXRIS | Transmit interrupt status. Returns the raw interrupt state of the UARTRXINTR interrupt. | RW | 0x0 |
| 4 | RXRIS | Receive interrupt status. Returns the raw interrupt state of the UARTRXINTR interrupt. | RW | 0x0 |
| 3 | DSRRMIS | Unsupported, write zero, read as don't care | RW | 0x0 |
| 2 | DCDRMIS | Unsupported, write zero, read as don't care | RW | 0x0 |
| 1 | CTSRMIS | nUARTCTS modem interrupt status. Returns the raw interrupt state of the UARTCTSINTR interrupt. | RW | 0x0 |
| 0 | RIRMIS | Unsupported, write zero, read as don't care | RW | 0x0 |

MIS Register



BCM2835 ARM Peripherals

Synopsis The UART_MIS Register is the masked interrupt status register. This register returns the current masked status value of the corresponding interrupt.

NOTE: All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:11 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 10 | OEMIS | Overrun error masked interrupt status. Returns the masked interrupt state of the UARTOEINTR interrupt. | RW | 0x0 |
| 9 | BEMIS | Break error masked interrupt status. Returns the masked interrupt state of the UARTBEINTR interrupt. | RW | 0x0 |
| 8 | PEMIS | Parity error masked interrupt status. Returns the masked interrupt state of the UARTPEINTR interrupt. | RW | 0x0 |
| 7 | FEMIS | Framing error masked interrupt status. Returns the masked interrupt state of the UARTFEINTR interrupt. | RW | 0x0 |
| 6 | RTMIS | Receive timeout masked interrupt status. Returns the masked interrupt state of the UARTRTINTR interrupt. | RW | 0x0 |
| 5 | TXMIS | Transmit masked interrupt status. Returns the masked interrupt state of the UARTTXINTR interrupt. | RW | 0x0 |
| 4 | RXMIS | Receive masked interrupt status. Returns the masked interrupt state of the UARTRXINTR interrupt. | RW | 0x0 |
| 3 | DSRMMIS | Unsupported, write zero, read as don't care | RW | 0x0 |
| 2 | DCDMMIS | Unsupported, write zero, read as don't care | RW | 0x0 |
| 1 | CTSMMIS | nUARTCTS modem masked interrupt status. Returns the masked interrupt state of the UARTCTSINTR interrupt. | RW | 0x0 |
| 0 | RIMMIS | Unsupported, write zero, read as don't care | RW | 0x0 |



BCM2835 ARM Peripherals

ICR Register

Synopsis The UART_ICR Register is the interrupt clear register.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:11 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 10 | OEIC | Overrun error interrupt clear. Clears the UARTOEINTR interrupt. | RW | 0x0 |
| 9 | BEIC | Break error interrupt clear. Clears the UARTBEINTR interrupt. | RW | 0x0 |
| 8 | PEIC | Parity error interrupt clear. Clears the UARTPEINTR interrupt. | RW | 0x0 |
| 7 | FEIC | Framing error interrupt clear. Clears the UARTFEINTR interrupt.. | RW | 0x0 |
| 6 | RTIC | Receive timeout interrupt clear. Clears the UARTRTINTR interrupt. | RW | 0x0 |
| 5 | TXIC | Transmit interrupt clear. Clears the UARTTXINTR interrupt. | RW | 0x0 |
| 4 | RXIC | Receive masked interrupt status. Returns the masked interrupt state of the UARTRXINTR interrupt. | RW | 0x0 |
| 3 | DSRMIC | Unsupported, write zero, read as don't care | RW | 0x0 |
| 2 | DCDMIC | Unsupported, write zero, read as don't care | RW | 0x0 |
| 1 | CTSMIC | nUARTCTS modem masked interrupt status. Returns the masked interrupt state of the UARTCTSINTR interrupt. | RW | 0x0 |
| 0 | RIMIC | Unsupported, write zero, read as don't care | RW | 0x0 |



BCM2835 ARM Peripherals

DMACR Register

Synopsis This is the disabled DMA Control Register, writing to it has not effect and reading returns 0.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:3 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 2 | DMAONERR | Unsupported, write zero, read as don't care | RW | 0x0 |
| 1 | TXDMAE | Unsupported, write zero, read as don't care | RW | 0x0 |
| 0 | RXDMAE | Unsupported, write zero, read as don't care | RW | 0x0 |

ITCR Register

Synopsis This is the Test Control Register UART_ITCR.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:2 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 1 | ITCR1 | Test FIFO enable. When this bit is 1, a write to the Test Data Register, UART_DR writes data into the receive FIFO, and reads from the UART_DR register reads data out of the transmit FIFO. When this bit is 0, data cannot be read directly from the transmit FIFO or written directly to the receive FIFO (normal operation). | RW | 0x0 |
| 0 | ITCR0 | Integration test enable. When this bit is 1, the UART is placed in integration test mode, otherwise it is in normal operation. | RW | 0x0 |



BCM2835 ARM Peripherals

ITIP Register

Synopsis This is the Test Control Register UART_ITIP.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:4 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 3 | ITIP3 | Reads return the value of the nUARTCTS primary input. | RW | 0x0 |
| 2:1 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 0 | ITIP0 | Reads return the value of the UARTRXD primary input. | RW | 0x0 |

ITOP Register

Synopsis This is the Test Control Register UART_ITOP.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|---|------|-------|
| 31:12 | | <i>Reserved - Write as 0, read as don't care</i> | | |
| 11 | ITOP11 | Intra-chip output. Writes specify the value to be driven on UARTMSINTR. Reads return the value of UARTMSINTR at the output of the test multiplexor. | RW | 0x0 |
| 10 | ITOP10 | Intra-chip output. Writes specify the value to be driven on UARTRXINTR. Reads return the value of UARTRXINTR at the output of the test multiplexor. | RW | 0x0 |
| 9 | ITOP9 | Intra-chip output. Writes specify the value to be driven on UARTTXINTR. Reads return the value of UARTTXINTR at the output of the test multiplexor. | RW | 0x0 |



BCM2835 ARM Peripherals

| | | | | |
|-----|-------|--|----|-----|
| 8 | ITOP8 | Intra-chip output. Writes specify the value to be driven on UARTRTINTR. Reads return the value of UARTRTINTR at the output of the test multiplexor. | RW | 0x0 |
| 7 | ITOP7 | Intra-chip output. Writes specify the value to be driven on UARTEINTR. Reads return the value of UARTEINTR at the output of the test multiplexor. | RW | 0x0 |
| 6 | ITIP6 | Intra-chip output. Writes specify the value to be driven on UARTINTR. Reads return the value of UARTINTR at the output of the test multiplexor. | RW | 0x0 |
| 5:4 | | Reserved - Write as 0, read as don't care | | |
| 3 | ITIP3 | Primary output. Writes specify the value to be driven on nUARTRTS. | RW | 0x0 |
| 2:1 | | Reserved - Write as 0, read as don't care | | |
| 0 | ITIP0 | Primary output. Writes specify the value to be driven on UARTTXD. | RW | 0x0 |

TDR Register

Synopsis UART_TDR is the test data register. It enables data to be written into the receive FIFO and read out from the transmit FIFO for test purposes. This test function is enabled by the ITCR1 bit in the Test Control Register, UART_ITCR.

| Bit(s) | Field Name | Description | Type | Reset |
|--------|------------|--|------|-------|
| 31:11 | | Reserved - Write as 0, read as don't care | | |
| 10:0 | TDR10_0 | When the ITCR1 bit is set to 1, data is written into the receive FIFO and read out of the transmit FIFO. | RW | 0x0 |

14 Timer (ARM side)

14.1 Introduction

The ARM Timer is *based* on a ARM AP804, but it has a number of differences with the standard SP804:

- There is only one timer.
- It only runs in continuous mode.
- It has a extra clock pre-divider register.
- It has a extra stop-in-debug-mode control bit.
- It also has a 32-bit free running counter.

The clock from the ARM timer is derived from the system clock. This clock can change dynamically e.g. if the system goes into reduced power or in low power mode. Thus the clock speed adapts to the overall system performance capabilities. For accurate timing it is recommended to use the system timers.

14.2 Timer Registers:

The base address for the ARM timer register is 0x7E00B000.

| Address offset ⁸ | Description |
|-----------------------------|--|
| 0x400 | Load |
| 0x404 | Value (Read Only) |
| 0x408 | Control |
| 0x40C | IRQ Clear/Ack (Write only) |
| 0x410 | RAW IRQ (Read Only) |
| 0x414 | Masked IRQ (Read Only) |
| 0x418 | Reload |
| 0x41C | <i>Pre-divider (Not in real 804!)</i> |
| 0x420 | <i>Free running counter (Not in real 804!)</i> |

Timer Load register

The timer load register sets the time for the timer to count down. This value is loaded into the timer value register after the load register has been written or if the timer-value register has counted down to 0.

⁸ This is the offset which needs to be added to the base address to get the full hardware address.



Timer Value register:

This register holds the current timer value and is counted down when the counter is running. It is counted down each timer clock until the value 0 is reached. Then the value register is re-loaded from the timer load register and the interrupt pending bit is set. The timer count down speed is set by the timer pre-divide register.

Timer control register:

The standard SP804 timer control register consist of 8 bits but in the BCM implementation there are more control bits for the extra features. Control bits 0-7 are identical to the SP804 bits, albeit some functionality of the SP804 is not implemented. All new control bits start from bit 8 upwards. Differences between a real 804 and the BCM implementation are shown in italics.

| Name: Timer control | | Address: base + 0x40C | Reset: 0x3E0020 |
|---------------------|-----|---|-----------------|
| Bit(s) | R/W | Function | |
| 31:10 | - | <Unused> | |
| 23:16 | R/W | Free running counter pre-scaler. Freq is $sys_clk/(prescale+1)$ <i>These bits do not exists in a standard 804! Reset value is 0x3E</i> | |
| 15:10 | - | <Unused> | |
| 9 | R/W | 0 : Free running counter Disabled 1 : Free running counter Enabled <i>This bit does not exists in a standard 804 timer!</i> | |
| 8 | R/W | 0 : Timers keeps running if ARM is in debug halted mode 1 : Timers halted if ARM is in debug halted mode <i>This bit does not exists in a standard 804 timer!</i> | |
| 7 | R/W | 0 : Timer disabled 1 : Timer enabled | |
| 6 | R/W | <i>Not used, The timer is always in free running mode.</i> <i>If this bit is set it enables periodic mode in a standard 804. That mode is not supported in the BC2835M.</i> | |
| 5 | R/W | 0 : Timer interrupt disabled 1 : Timer interrupt enabled | |
| 4 | R/W | <Not used> | |
| 3:2 | R/W | Pre-scale bits: 00 : pre-scale is clock / 1 (No pre-scale) 01 : pre-scale is clock / 16 10 : pre-scale is clock / 256 11 : pre-scale is clock / 1 (<i>Undefined in 804</i>) | |
| 1 | R/W | 0 : 16-bit counters 1 : 23-bit counter | |
| 0 | R/W | <i>Not used, The timer is always in wrapping mode.</i> <i>If this bit is set it enables one-shot mode in real 804. That mode is not supported in the BCM2835.</i> | |



Timer IRQ clear register:

The timer IRQ clear register is write only. When writing this register the interrupt-pending bit is cleared.

When reading this register it returns 0x544D5241 which is the ASCII reversed value for "ARMT".

Timer Raw IRQ register

The raw IRQ register is a read-only register. It shows the status of the interrupt pending bit.

| Name: Raw IRQ | | Address: base + 0x40C | Reset: 0x3E0020 |
|---------------|-----|--|-----------------|
| Bit(s) | R/W | Function | |
| 31:0 | R | 0 | |
| 0 | R | 0 : The interrupt pending bits is clear 1 : The interrupt pending bit is set. | |

The interrupt pending bits is set each time the value register is counted down to zero. The interrupt pending bit can not by itself generates interrupts. Interrupts can only be generated if the interrupt enable bit is set.

Timer Masked IRQ register:

The masked IRQ register is a read-only register. It shows the status of the interrupt signal. It is simply a logical AND of the interrupt pending bit and the interrupt enable bit.

| Name: Masked IRQ | | Address: base + 0x40C | Reset: 0x3E0020 |
|------------------|-----|--|-----------------|
| Bit(s) | R/W | Function | |
| 31:0 | R | 0 | |
| 0 | R | 0 : Interrupt line not asserted. 1 :Interrupt line is asserted, (the interrupt pending and the interrupt enable bit are set.) | |

Timer Reload register:

This register is a copy of the timer load register. The difference is that a write to this register does not trigger an immediate reload of the timer value register. Instead the timer load register value is only accessed if the value register has finished counting down to zero.

The timer pre-divider register:

| Name: pre-divide | | Address: base + 0x41C | Reset: 0x07D |
|------------------|-----|-----------------------|--------------|
| Bit(s) | R/W | Function | |
| 31:10 | - | <Unused> | |
| 9:0 | R/W | Pre-divider value. | |

The Pre-divider register is not present in the SP804.



BCM2835 ARM Peripherals

The pre-divider register is 10 bits wide and can be written or read from. This register has been added as the SP804 expects a 1MHz clock which we do not have. Instead the pre-divider takes the APB clock and divides it down according to:

$$\text{timer_clock} = \text{apb_clock}/(\text{pre_divider}+1)$$

The reset value of this register is 0x7D so gives a divide by 126.

Free running counter

| Name: Free running | | Address: base + 0x420 | Reset: 0x000 |
|--------------------|-----|-----------------------|--------------|
| Bit(s) | R/W | Function | |
| 31:0 | R | Counter value | |

The free running counter is not present in the SP804.

The free running counter is a 32 bits wide read only register. The register is enabled by setting bit 9 of the Timer control register. The free running counter is incremented immediately after it is enabled. The timer can not be reset but when enabled, will always increment and roll-over. The free running counter is also running from the APB clock and has its own clock pre-divider controlled by bits 16-23 of the timer control register.

This register will be halted too if bit 8 of the control register is set and the ARM is in Debug Halt mode.



15 USB

The USB core used in the Videocore is build from Synopsys IP. Details about the block can be found in [DWC_otg_databook.pdf](https://www.synopsys.com/dw/ipdir.php?ds=dwc_usb_2_0_hs_otg) (Which can also be downloaded from https://www.synopsys.com/dw/ipdir.php?ds=dwc_usb_2_0_hs_otg).

15.1 Configuration

A number of features of the block are specified before the block is build and thus can not be changed using software. The above mentioned document has a list of these under the chapter "Configuration Parameters". The following table list all configuration parameters mentioned in that chapter and the values which have been chosen.

| Feature/Parameter | Selected value |
|---|---|
| Mode of Operation | 0: HNP- and SRP-Capable OTG (Device and Host) |
| LPM Mode of Operation | 0: Non-LPM-capable core |
| HSIC Mode of Operation | 0: Non-HSIC-capable core |
| Architecture | 2: Internal DMA |
| Point-to-Point Application Only | 0: No |
| High-Speed PHY Interfaces | 1: UTMI+ |
| USB 1.1 Full-Speed Serial Transceiver Interface | 1: Dedicated FS |
| USB IC_USB Transceiver Interface | 0: Non-IC_USB-capable |
| Default (Power on) Interface selection: FS_USB/IC_USB | 0 : FS_USB interface |
| Data Width of the UTMI+ Interface | 0: 8 bits |
| Enable I2C Interface | 0: None |
| Enable ULPI Carkit | 0: No |
| Enable PHY Vendor Control Interface | 0: No |



BCM2835 ARM Peripherals

| Feature/Parameter | Selected value |
|--|----------------|
| Number of Device Mode Endpoints in Addition to Control Endpoint 0 | 7 |
| Enable Dedicated Transmit FIFOs for Device IN Endpoints | 1: Yes |
| Enable descriptor based scatter/gather DMA | 0: No |
| Enable Option for Endpoint- Specific Interrupt | 0: No |
| Number of Device Mode Periodic IN Endpoints | 0 |
| Number of Device Mode IN Endpoints including Control Endpo 0 | 8 |
| Number of Device Mode Control Endpoints in Addition to Endpoint 0 | 0 |
| Number of Host Mode Channels | 8 |
| Is Periodic OUT Channel Support Needed in Host Mode | 1: Yes |
| Total Data FIFO RAM Depth | 4096 |
| Enable Dynamic FIFO Sizing | 1: Yes |
| Largest Rx Data FIFO Depth | 4096 |
| Largest Non-Periodic Host Tx Data FIFO Depth | 1024 |
| Largest Non-periodic Tx Data FIFO Depth | 4096 |
| Largest Host Mode Tx Periodic Data FIFO Dept | 4096 |
| Non-periodic Request Queue Depth | 8 |
| Host Mode Periodic Request Queue Depth | 8 |
| Device Mode IN Token Sequence Learning Queue Depth | 8 |
| Width of Transfer Size Counters | 19 |



BCM2835 ARM Peripherals

| Feature/Parameter | Selected value |
|--|--|
| Width of Packet Counters | 10 |
| Remove Optional Features | 0: No |
| Power-on Value of User ID Register | 0x2708A000 |
| Enable Power Optimization | 0: No |
| Is Minimum AHB Operating Frequency Less than 60 MHz | 1: Yes |
| Reset Style of Clocked always Blocks in RTL | 0: Asynchronous |
| Instantiate Double- Synchronization Flops | 1: Yes |
| Enable Filter on "iddig" Signal from PHY | 1: Yes |
| Enable Filter on "vbus_valid" Signal from PHY | 1: Yes |
| Enable Filter on "a_valid" Signal from PHY | 1: Yes |
| Enable Filter on "b_valid" Signal from PHY | 1: Yes |
| Enable Filter on "session_end" Signal from PHY | 1: Yes |
| Direction of Endpoints | Mode is {IN and OUT} for all endpoints |
| Largest Device Mode Periodic Tx Data FIFO n Depth | 768 for all endpoints (Except 0) |
| Largest Device Mode IN Endpoint Tx FIFO n Depth (n = 0 to 15) when using dynamic FIFO sizing | 0=32 1..5=512 6,7=768 |

15.2 Extra / Adapted registers.

Besides the registers as specified in the documentation of Synopsys a number of extra registers have been added. These control the Analogue USB Phy and the connections of the USB block into the Video core bus structure. Also the USB_GAHBCFG register has an alternative function for the bits [4:1].

Base Address of the USB block – 0x7E98_0000



| Offset Address | Description | | Size | Read/Write |
|----------------|---------------|---------------------------------------|------|------------|
| 0x080 | USB_MDIO_CNTL | MDIO interface control | | R/W |
| 0x084 | USB_MDIO_GEN | Data for MDIO interface | 32 | R/W |
| 0x088 | USB_VBUS_DRV | Vbus and other Miscellaneous controls | | R/W |

USB MDIO Control (USB_MDIO_CNTL)

Address 0x 7E98 0080

| Bit Number | Field Name | Description | Read/Write | Reset |
|------------|------------|--|------------|-------|
| 31 | mdio_busy | 1= MDIO read or write in progress 0= MDIO Idle | R | 0 |
| 30-24 | - | Unused | - | 0 |
| 23 | bb_mdo | Direct write (bitbash) MDO output | R/W | 0 |
| 22 | bb_mdc | Direct write (bitbash) MDC output | R/W | 0 |
| 21 | bb_enbl | 1= MDIO bitbash enable 0= MDIO under control of the phy | R/W | 0 |
| 20 | freerun | 1= MDC is continous active 0 = MDC only active during data transfer | R/W | 0 |
| 19:16 | mdc_ratio | MDC clock freq is sysclk/mdc_ratio | R/W | 0 |
| 15:0 | mdi | 16-bit read of MDIO input shift register. Updates on falling edge of MDC | RO | 0 |

Table 15-1 MDIO Control

USB MDIO Data (USB_MDIO_DATA)

Address 0x 7E98 0084

| Bit Number | Field Name | Description | Read/Write | Reset |
|------------|------------|--|------------|-------|
| 31-0 | mdio_data | 32-bit sequence to send over MDIO bus | W | 0 |
| 31-0 | mdio_data | 32-bit sequence received from MDIO bus | R | 0 |

Table 15-2 USB MDIO data

A Preamble is not auto-generated so any MDIO access must be preceded by a write to this register of 0xFFFFFFFF. Furthermore, a bug in the USB PHY requires an extra clock edge so a write of 0x00000000 must follow the actual access.



BCM2835 ARM Peripherals

USB VBUS (USB_VBUS)

Address 0x 7E98 0088

| Bit Number | Field Name | Description | Read/Write | Reset |
|------------|---------------------|---|------------|-------|
| 31-20 | - | Unused | - | 0 |
| 19-16 | axi_priority | Sets the USB AXI priority level | R/W | 0 |
| 15:10 | - | Unused | - | 0 |
| 9 | vbus_irq | 1=one or more bits of [6:4] have changed since last read. This bit is cleared when the register is read. | RC | 0 |
| 8 | vbus_irq_en | 1=Enable IRQ on VBUS status change | R/W | 0 |
| 7 | afe_non_driving | 1=USB PHY AFE pull ups/pull downs are off 0=Normal USB AFE operation (Has no effect if MDIO mode is enabled in the phy) | R/W | 0 |
| 6 | utmisrp_dischrgvbus | Drive VBUS | R | 0 |
| 5 | utmisrp_chrgvbuses | Charge VBUS | R | 0 |
| 4 | utmio_tg_drvvbus | Discharge VBUS | R | 0 |
| 3 | utmio_tg_avalid | A session Valid | R/W | 0 |
| 2 | utmio_tg_bvalid | B session Valid | R/W | 0 |
| 1 | utmio_tg_vbusvalid | VBUS valid | R/W | 0 |
| 0 | utmisrp_sessend | Session end | R/W | 0 |

Table 15-3 USB MDIO data

The RW bits in this register are fed into the USB2.0 controller and the RO bits are coming out of it. In the real device, it will be up to the software to communicate this information between the USB2.0 controller and external VBUS device (some of these have I2C control, others will have to interface via GPIO).

USB AHB configuration (USB_GAHBCFG)

Address 0x 7E98 0008

The USB_GAHBCFG register has been adapted. Bits [4:1] which are marked in the Synopsys documentation as "Burst Length/Type (HBstLen)" have been used differently.

- [4] 1 = Wait for all outstanding AXI writes to complete before signalling (internally) that DMA is done.
0 = don't wait.
- [3] Not used
- [2:1] Sets the maximum AXI burst length, but the bits are inverted,
00 = maximum AXI burst length of 4,
01 = maximum AXI burst length of 3,
10 = maximum AXI burst length of 2
11 = maximum AXI burst length of 1



BCM2835 ARM Peripherals

Personal Notes: