BCM2711 ARM Peripherals

Colophon

BCM2711 ARM Peripherals, based in large part on the earlier BCM2835 ARM Peripherals documentation.

© 2012 Broadcom Europe Ltd., 2020-2022 Raspberry Pi Ltd (formerly Raspberry Pi (Trading) Ltd.)
All rights reserved.

build-date: 2022-01-18
build-version: githash: cfcff44-clean

Table 1. Release History

Release	Date	Description
1	05 Feb 2020	First release.
2	24 Sep 2020	Corrected GPIO base address. Updated styling.
3	16 Oct 2020	First public release.
4	18 Jan 2022	Added information about switching DMA DREQ channels. Updated GPIO_PUP_PDN_CNTRL register reset values. Updated UART GPIO mapping table.

The latest release can be found at https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf

Legal Disclaimer Notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME ("RESOURCES") ARE PROVIDED BY RASPBERRY PI LTD ("RPL") "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation pf nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL's Standard Terms. RPL's provision of the RESOURCES does not expand or otherwise modify RPL's Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Table of Contents

Chapter 1. Introduction

1.1. Overview

BCM2711 contains the following peripherals which may safely be accessed by the ARM:

- Timers
- Interrupt controller
- GPIO
- USB
- PCM / I2S
- DMA controller
- I2C masters
- SPI masters
- PWM
- UARTs

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 BCM2711. Not all of these peripherals have been fully documented yet.

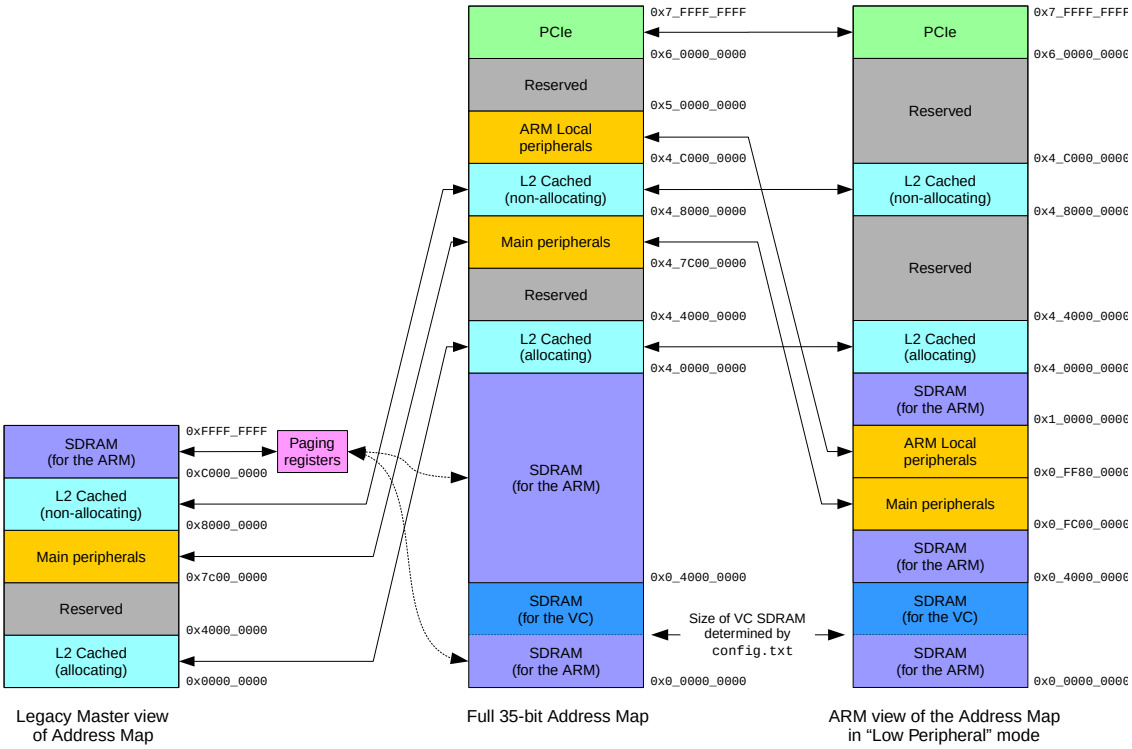
There are a number of peripherals which are intended to be controlled by the GPU. These are omitted from this datasheet. Accessing these peripherals from the ARM is not recommended.

1.2. Address map

1.2.1. Diagrammatic overview

The BCM2711 has two main addressing schemes: a "Full" 35-bit address bus and a 32-bit "Legacy Master" view as seen by the peripherals (except for "large address" masters). There's also a "Low Peripherals" mode which modifies the ARM's view of the peripheral addresses. Figure 1 shows how these address maps inter-relate. Note that the relative sizes of the address blocks in the diagram are definitely **not** to scale! (The PCIe address range covers 8GB, but the Main peripherals address range only covers 64MB.)

Figure 1. BCM2711 Address Maps



Addresses in ARM Linux are:

1. Issued as virtual addresses by the ARM core, then
2. Mapped into a physical address by the ARM MMU, then
3. Used to select the appropriate peripheral or location in RAM

1.2.2. Full 35-bit address map

The full 35-bit address map is shown in Figure 1. This is seen by both "large address" masters (e.g. the DMA4 engines) and the ARM CPU.

It has two L2 cache aliases (one allocating, one non-allocating) which cache (only) the first 1GB of SDRAM.

1.2.3. ARM physical addresses

Physical addresses start at 0x0_0000_0000 for RAM.

- The ARM section of the RAM starts at 0x0_0000_0000 and extends up to the size of installed SDRAM.
- The VideoCore section of the RAM is mapped in from 0x0_4000_0000 downwards. The size of the VideoCore RAM is determined by a setting in `config.txt` - refer to raspberrypi.com documentation for further details.

The VideoCore maps the ARM physical address space directly to the bus address space seen by VideoCore. The bus addresses for RAM are set up to map onto the uncached bus address range on the VideoCore starting at 0x0_0000_0000.

NOTE

BCM2711 provides a 1MB system L2 cache, which is used primarily by the GPU. Accesses to memory are routed either via or around the L2 cache depending on the address range being used.

When running in 32-bit mode, the ARM uses LPAE mode to enable it to access the full 32GB address space.

Physical addresses range from 0x4_7C00_0000 to 0x4_7FFF_FFFF for Main peripherals, and from 0x4_C000_0000 to 0x4_FFFF_FFFF for ARM Local peripherals.

If the VPU enables "Low Peripheral" mode then the ARM (only) has Main peripherals available from 0x0_FC00_0000 to 0x0_FF7F_FFFF and ARM Local peripherals available from 0x0_FF80_0000 to 0x0_FFFF_FFFF.

1.2.4. Legacy master addresses

The peripheral addresses specified in this document are legacy master addresses. Software accessing peripherals using the DMA engines must use 32-bit legacy master addresses. The Main peripherals are available from 0x7C00_0000 to 0x7FFF_FFFF. Behind the scenes, the VideoCore transparently translates these addresses to the 35-bit 0x4_7nnn_nnnn addresses.

So a peripheral described in this document as being at legacy address 0x7Enn_nnnn is available in the 35-bit address space at 0x4_7Enn_nnnn, and visible to the ARM at 0x0_FEnn_nnnn if Low Peripheral mode is enabled.

Software accessing RAM using the DMA engines must use legacy addresses (between 0xC000_0000 and 0xFFFF_FFFF). This accesses a 1GB window within the full 16GB SDRAM address space. If the DMA engine needs to access RAM above the first 1GB, this window can be moved using the PAGE or PAGELITE bits - see Chapter 4 for more details. Behind the scenes, the VideoCore transparently translates these addresses to the 35-bit 0x0_nnnn_nnnn addresses.

Software accessing the VPU L2 cache using the DMA engines must use legacy addresses starting at 0x0000_0000 (for allocating cache) or 0x8000_0000 (for non-allocating cache). Behind the scenes, the VideoCore transparently translates these addresses to the corresponding 35-bit 0x4_nnnn_nnnn addresses. These 1GB windows can't be moved, and are limited to the first 1GB of SDRAM.

1.3. Peripheral access precautions for correct memory ordering

The BCM2711 system uses an AMBA AXI-compatible interface structure. In order to keep the system complexity low and data throughput high, the BCM2711 AXI system does not always return read data in-order. 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, you should also 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.

NOTE

Normally 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 theoretically 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.

Chapter 2. Auxiliaries: UART1, SPI1 & SPI2

2.1. Overview

The BCM2711 device has three Auxiliary peripherals: One mini UART (UART1) and two SPI masters (SPI1 & SPI2). These three peripherals 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. The Auxiliary register base address is `0x7e215000`.

Table 2. Auxiliary peripherals Address Map

Offset	Name	Description
0x00	AUX_IRQ	Auxiliary Interrupt status
0x04	AUX_ENABLES	Auxiliary enables
0x40	AUX_MU_IO_REG	Mini UART I/O Data
0x44	AUX_MU_IER_REG	Mini UART Interrupt Enable
0x48	AUX_MU_IIR_REG	Mini UART Interrupt Identify
0x4c	AUX_MU_LCR_REG	Mini UART Line Control
0x50	AUX_MU_MCR_REG	Mini UART Modem Control
0x54	AUX_MU_LSR_REG	Mini UART Line Status
0x58	AUX_MU_MSR_REG	Mini UART Modem Status
0x5c	AUX_MU_SCRATCH	Mini UART Scratch
0x60	AUX_MU_CNTL_REG	Mini UART Extra Control
0x64	AUX_MU_STAT_REG	Mini UART Extra Status
0x68	AUX_MU_BAUD_REG	Mini UART Baudrate
0x80	AUX_SPI1_CNTL0_REG	SPI 1 Control register 0
0x84	AUX_SPI1_CNTL1_REG	SPI 1 Control register 1
0x88	AUX_SPI1_STAT_REG	SPI 1 Status
0x8c	AUX_SPI1_PEEK_REG	SPI 1 Peek
0xa0	AUX_SPI1_IO_REGa	SPI 1 Data
0xa4	AUX_SPI1_IO_REGb	SPI 1 Data
0xa8	AUX_SPI1_IO_REGc	SPI 1 Data
0xac	AUX_SPI1_IO_REGd	SPI 1 Data
0xb0	AUX_SPI1_TXHOLD_REGa	SPI 1 Extended Data
0xb4	AUX_SPI1_TXHOLD_REGb	SPI 1 Extended Data
0xb8	AUX_SPI1_TXHOLD_REGc	SPI 1 Extended Data
0xbc	AUX_SPI1_TXHOLD_REGd	SPI 1 Extended Data
0xc0	AUX_SPI2_CNTL0_REG	SPI 2 Control register 0

Offset	Name	Description
0xc4	AUX_SPI2_CNTL1_REG	SPI 2 Control register 1
0xc8	AUX_SPI2_STAT_REG	SPI 2 Status
0xcc	AUX_SPI2_PEEK_REG	SPI 2 Peek
0xe0	AUX_SPI2_IO_REGa	SPI 2 Data
0xe4	AUX_SPI2_IO_REGb	SPI 2 Data
0xe8	AUX_SPI2_IO_REGc	SPI 2 Data
0xec	AUX_SPI2_IO_REGd	SPI 2 Data
0xf0	AUX_SPI2_TXHOLD_REGa	SPI 2 Extended Data
0xf4	AUX_SPI2_TXHOLD_REGb	SPI 2 Extended Data
0xf8	AUX_SPI2_TXHOLD_REGc	SPI 2 Extended Data
0xfc	AUX_SPI2_TXHOLD_REGd	SPI 2 Extended Data

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.

AUX_IRQ Register

Description

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

Table 3. AUX_IRQ Register

Bits	Name	Description	Type	Reset
31:3	Reserved.	-	-	-
2	SPI 2 IRQ	If set the SPI 2 module has an interrupt pending.	RO	0x0
1	SPI 1 IRQ	If set the SPI1 module has an interrupt pending.	RO	0x0
0	Mini UART IRQ	If set the mini UART has an interrupt pending.	RO	0x0

AUX_ENABLES Register

Description

The AUX_ENABLES register is used to enable the three modules: UART1, SPI1, SPI2.

Table 4. AUX_ENABLES Register

Bits	Name	Description	Type	Reset
31:3	Reserved.	-	-	-
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	RW	0x0
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	RW	0x0

Bits	Name	Description	Type	Reset
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 *low*. If clear the mini UART is disabled. That also disables any mini UART register access	RW	0x0

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 before enabling the UART. The UART core is built 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 a stop bit, and is only used so there is a clean timing start for the next bit.

Immediately after a reset, the baudrate register 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 if the UART is enabled without changing the default configuration, the FIFO will be full and overflowing in no time flat.

2.2. Mini UART

The mini UART is a secondary low throughput 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 (see Chapter 5).

NOTE

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.

The mini UART has the following features:

- 7-bit 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. All 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.

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

Description

The AUX_MU_IO_REG register is primarily 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 in AUX_MU_BAUD_REG)

Table 5. AUX_MU_IO_REG Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
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)	RW	0x00
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)	WO	0x00
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)	RO	0x00

AUX_MU_IER_REG Register

Description

The AUX_MU_IER_REG register is primarily 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 in AUX_MU_BAUD_REG)

Table 6. AUX_MU_IER_REG Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
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)	RW	0x00
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.	RW	0x0
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.	RW	0x0

AUX_MU_IIR_REG Register

Description

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

Table 7. AUX_MU_IIR_REG Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
7:6	FIFO enables	Both bits always read as 1 as the FIFOs are always enabled	RO	0x3
5:4	-	Always read as zero	RO	0x0
3	-	Always read as zero as the mini UART has no timeout function	RO	0x0
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	RW	0x0
0	Interrupt pending	This bit is clear whenever an interrupt is pending	RO	0x1

AUX_MU_LCR_REG Register

Description

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

Table 8. AUX_MU_LCR_REG Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
7	DLAB access	If set the first two Mini UART registers give access to the Baudrate register. During operation this bit must be cleared.	RW	0x0
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.	RW	0x0
5:1	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
0	Data size	If clear the UART works in 7-bit mode If set the UART works in 8-bit mode	RW	0x0

AUX_MU_MCR_REG Register

Description

The AUX_MU_MCR_REG register controls the 'modem' signals.

Table 9. AUX_MU_MCR_REG Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
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)	RW	0x0
0	Reserved.	-	-	-

AUX_MU_LSR_REG Register

Description

The AUX_MU_LSR_REG register shows the data status.

Table 10. AUX_MU_LSR_REG Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6	Transmitter idle	This bit is set if the transmit FIFO is empty and the transmitter is idle. (Finished shifting out the last bit).	RO	0x1
5	Transmitter empty	This bit is set if the transmit FIFO can accept at least one byte.	RO	0x0
4:2	Reserved.	-	-	-
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.	RC	0x0
0	Data ready	This bit is set if the receive FIFO holds at least 1 symbol.	RO	0x0

AUX_MU_MSR_REG Register

Description

The AUX_MU_MSR_REG register shows the 'modem' status.

Table 11. AUX_MU_MSR_REG Register

Bits	Name	Description	Type	Reset
31:5	Reserved.	-	-	-
4	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	RO	0x1
3:0	Reserved.	-	-	-

AUX_MU_SCRATCH Register

Description

The AUX_MU_SCRATCH is a single byte of temporary storage.

Table 12. AUX_MU_SCRATCH Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
7:0	Scratch	A byte of temporary storage	RW	0x00

AUX_MU_CNTL_REG Register

Description

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

Table 13. AUX_MU_CNTL_REG Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
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*	RW	0x0
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*	RW	0x0
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.	RW	0x0
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	RW	0x0
2	Enable receive Auto flow-control using RTS	If this bit is set the RTS line will de-assert if the receive FIFO reaches its 'auto flow' level. In fact the RTS line will behave as an RTR (Ready To Receive) line. If this bit is clear the RTS line is controlled by the AUX_MU_MCR_REG register bit 1.	RW	0x0
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	RW	0x1
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	RW	0x1

Receiver enable

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

Transmitter enable

If this bit is clear no new symbols will be sent by 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-assert 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.

AUX_MU_STAT_REG Register

Description

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.

Table 14. AUX_MU_STAT_REG Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
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	RO	0x0
23:20	Reserved.	-	-	-
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	RO	0x0
15:10	Reserved.	-	-	-
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 3 and 8	RO	0x1
8	Transmit FIFO is empty	If this bit is set the transmitter FIFO is empty. Thus it can accept 8 symbols.	RO	0x1
7	CTS line	This bit shows the status of the UART1_CTS line.	RO	0x0
6	RTS status	This bit shows the status of the UART1_RTS line.	RO	0x0
5	Transmit FIFO is full	This is the inverse of bit 1	RO	0x0

Bits	Name	Description	Type	Reset
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.	RO	0x0
3	Transmitter is idle	If this bit is set the transmitter is idle. If this bit is clear the transmitter is busy.	RO	0x1
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	RO	0x1
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	RO	0x0
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	RO	0x0

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_REG Register

Description

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

Table 15. AUX_MU_BAUD_REG Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	Baudrate	mini UART baudrate counter	RW	0x0000

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

2.3. Universal SPI Master (2x)

The two universal SPI masters are secondary low throughput SPI interfaces. Like the mini UART the devices need to be enabled before they can be used.

NOTE

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.

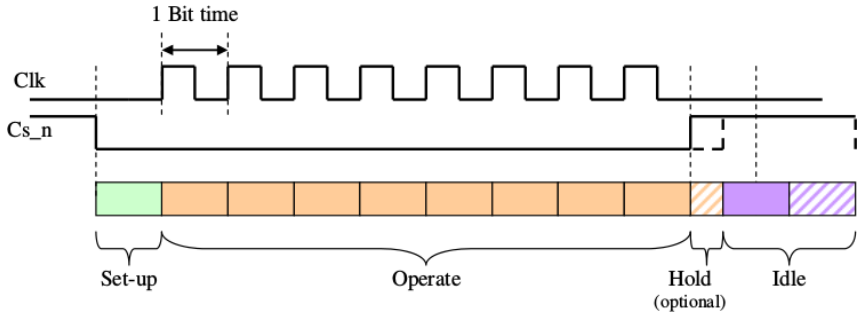
Each SPI master has the following features:

- 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 diagrams 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).

Lastly 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. There is a RX FIFO level field which tells you 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 as the number of entries received can never be more than 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 received all data as well.

2.3.3. Long bit streams

The SPI module works in bursts of up to 32 bits. Some SPI devices require data which is longer than 32 bits. To do this the user must make use of the two different data TX addresses: TX data written to one address causes the CS to remain asserted. TX data written to the other address causes 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

AUX_SPI1_CNTL0_REG, AUX_SPI2_CNTL0_REG Registers

Description

The AUX_SPIx_CNTL0_REG registers control many features of the SPI interfaces.

Table 16. AUX_SPI1_CNTL0_REG, AUX_SPI2_CNTL0_REG Registers

Bits	Name	Description	Type	Reset
31:20	Speed	Sets the SPI clock speed. spi_clk_freq = system_clock_freq/2*(speed+1)	RW	0x000
19:17	Chip Selects	The pattern output on the CS pins when active.	RW	0x7
16	Post-input mode	If set the SPI input works in post-input mode. For details see text further down	RW	0x0
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 bit 14 (variable width) is also set	RW	0x0

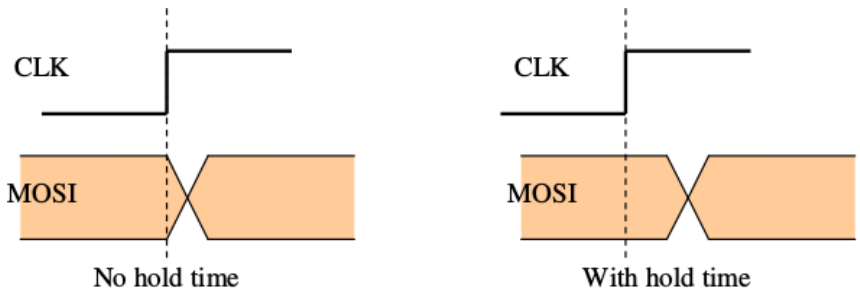
Bits	Name	Description	Type	Reset
14	Variable width	If 1 the SPI takes the shift length and the data from the TX FIFO If 0 the SPI takes the shift length from bits 0-5 of this register	RW	0x0
13:12	DOUT Hold time	Controls the extra DOUT hold time in system clock cycles. 00 : No extra hold time 01 : 1 system clock extra hold time 10 : 4 system clocks extra hold time 11 : 7 system clocks extra hold time	RW	0x0
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.	RW	0x0
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	RW	0x0
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.	RW	0x0
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	RW	0x0
7	Invert SPI CLK	If 1 the 'idle' clock line state is high. If 0 the 'idle' clock line state is low.	RW	0x0
6	Shift out MS bit first	If 1 the data is shifted out starting with the MS bit. (bit 31 or bit 23) If 0 the data is shifted out starting with the LS bit. (bit 0)	RW	0x0
5:0	Shift length	Specifies the number of bits to shift This field is ignored when using 'variable width' mode	RW	0x00

Invert SPI CLK

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

DOUT hold time

Because the interface runs off 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

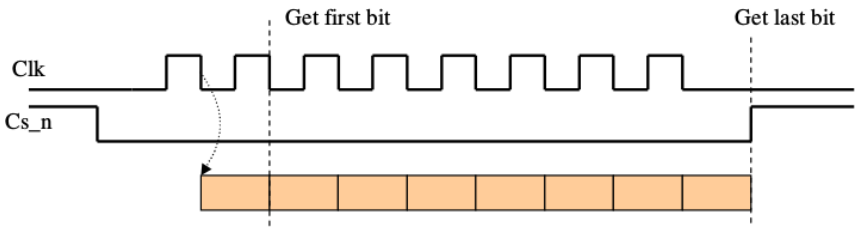
This mode is used together with the variable width mode. In this mode the CS pattern is taken from the transmit FIFO. The transmit data bits 31:29 are used as CS and the data bits 23:0 are the actual transmit data. This allows the CPU to write to different SPI devices without having to change the CS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

AUX_SPI1_CNTL1_REG, AUX_SPI2_CNTL1_REG Registers

Description

The AUX_SPIx_CNTL1_REG registers control more features of the SPI interfaces.

Table 17. AUX_SPI1_CNTL1_REG, AUX_SPI2_CNTL1_REG Registers

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-
10:8	CS high time	Additional SPI clock cycles where the CS is high.	RW	0x0
7	TX empty IRQ	If 1 the interrupt line is high when the transmit FIFO is empty	RW	0x0
6	Done IRQ	If 1 the interrupt line is high when the interface is idle	RW	0x0
5:2	Reserved.	-	-	-
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)	RW	0x0
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.	RW	0x0

Keep input

Setting the 'Keep input' bit will prevent the input shift register being 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).

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).

AUX_SPI1_STAT_REG, AUX_SPI2_STAT_REG Registers

Description

The AUX_SPIx_STAT_REG registers show the status of the SPI interfaces.

Table 18. AUX_SPI1_STAT_REG, AUX_SPI2_STAT_REG Registers

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27:24	TX FIFO level	The number of data units in the transmit data FIFO	RO	0x0
23:20	Reserved.	-	-	-
19:16	RX FIFO level	The number of data units in the receive data FIFO.	RO	0x0
15:11	Reserved.	-	-	-
10	TX Full	If 1 the transmit FIFO is full If 0 the transmit FIFO can accept at least 1 data unit.	RO	0x0
9	TX Empty	If 1 the transmit FIFO is empty If 0 the transmit FIFO holds at least 1 data unit.	RO	0x0
8	RX Full	If 1 the receiver FIFO is full If 0 the receiver FIFO can accept at least 1 data unit.	RO	0x0
7	RX Empty	If 1 the receiver FIFO is empty If 0 the receiver FIFO holds at least 1 data unit.	RO	0x0
6	Busy	Indicates the module is busy transferring data.	RO	0x0
5:0	Bit count	The number of bits still to be processed. Starts with 'shift-length' and counts down.	RO	0x00

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.

AUX_SPI1_PEEK_REG, AUX_SPI2_PEEK_REG Registers

Description

The AUX_SPIx_PEEK_REG registers show received data of the SPI interfaces.

Table 19. AUX_SPI1_PEEK_REG, AUX_SPI2_PEEK_REG Registers

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
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	0x0000

AUX_SPI1_IO_REGa, AUX_SPI1_IO_REGb, AUX_SPI1_IO_REGc, AUX_SPI1_IO_REGd Registers

Description

The AUX_SPI1_IO_REG registers are the primary data port of the SPI 1 interface.
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.

Table 20. AUX_SPI1_IO_REGa, AUX_SPI1_IO_REGb, AUX_SPI1_IO_REGc, AUX_SPI1_IO_REGd Registers

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
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 empty will return the last data received.	RW	0x0000

AUX_SPI1_TXHOLD_REGa, AUX_SPI1_TXHOLD_REGb, AUX_SPI1_TXHOLD_REGc, AUX_SPI1_TXHOLD_REGd Registers

Description

The AUX_SPI1_TXHOLD_REG registers are the extended CS port of the SPI 1 interface.
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.

Table 21. AUX_SPI1_TXHOLD_REGa, AUX_SPI1_TXHOLD_REGb, AUX_SPI1_TXHOLD_REGc, AUX_SPI1_TXHOLD_REGd Registers

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
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 empty will return the last data received.	RW	0x0000

AUX_SPI2_IO_REGa, AUX_SPI2_IO_REGb, AUX_SPI2_IO_REGc, AUX_SPI2_IO_REGd Registers

Description

The AUX_SPI2_IO_REG registers are the primary data port of the SPI 2 interface.
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.

Table 22. AUX_SPI2_IO_REGa, AUX_SPI2_IO_REGb, AUX_SPI2_IO_REGc, AUX_SPI2_IO_REGd Registers

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
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 empty will return the last data received.	RW	0x0000

AUX_SPI2_TXHOLD_REGa, AUX_SPI2_TXHOLD_REGb, AUX_SPI2_TXHOLD_REGc, AUX_SPI2_TXHOLD_REGd Registers

Description

The AUX_SPI2_TXHOLD_REG registers are the extended CS port of the SPI 2 interface.
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.

Table 23. AUX_SPI2_TXHOLD_REGa, AUX_SPI2_TXHOLD_REGb, AUX_SPI2_TXHOLD_REGc, AUX_SPI2_TXHOLD_REGd Registers

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
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 empty will return the last data received.	RW	0x0000

Chapter 3. BSC

3.1. Overview

The Broadcom Serial Control (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
- The BSC controller in the BCM2711 fixes the clock-strectching bug that was present in BCM283x devices

3.2. Register View

The BSC controller has eight memory-mapped registers. All accesses are assumed to be 32-bit. Note that the BSC2 and BSC7 masters are dedicated for use by the HDMI interfaces and should not be accessed by user programs.

There are eight BSC masters inside BCM2711. The user-accessible register addresses start from

- BSC0: `0x7e205000`
- BSC1: `0x7e804000`
- BSC3: `0x7e205600`
- BSC4: `0x7e205800`
- BSC5: `0x7e205a80`
- BSC6: `0x7e205c00`

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

Table 24. I2C Address Map

Offset	Name	Description
0x00	C	Control
0x04	S	Status
0x08	DLEN	Data Length
0x0c	A	Slave Address
0x10	FIFO	Data FIFO
0x14	DIV	Clock Divider
0x18	DEL	Data Delay
0x1c	CLKT	Clock Stretch Timeout

C Register

Description

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 is 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 ¼ 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 ¾ 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 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.

Table 25. C Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15	I2CEN	I2C Enable 0 = BSC controller is disabled 1 = BSC controller is enabled	RW	0x0
14:11	Reserved.	-	-	-
10	INTR	Interrupt on RX 0 = Don't generate interrupts on RXR condition. 1 = Generate interrupt while RXR = 1.	RW	0x0
9	INTT	Interrupt on TX 0 = Don't generate interrupts on TXW condition. 1 = Generate interrupt while TXW = 1.	RW	0x0
8	INTD	Interrupt on DONE 0 = Don't generate interrupts on DONE condition. 1 = Generate interrupt while DONE = 1.	RW	0x0
7	ST	Start Transfer 0 = No action. 1 = Start a new transfer. One-shot operation. Read back as 0.	W1SC	0x0
6	Reserved.	-	-	-
5:4	CLEAR	FIFO Clear 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 with the previous version.	W1SC	0x0
3:1	Reserved.	-	-	-
0	READ	Read Transfer 0 = Write Packet Transfer. 1 = Read Packet Transfer.	RW	0x0

S Register

Description

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 ¼ full and needs writing. Writing sufficient data (i.e. enough data to either fill the FIFO more than ¼ 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 ¾ or more full and needs reading. Reading sufficient data to bring the depth below ¾ 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.

Table 26. S Register

Bits	Name	Description	Type	Reset
31:10	Reserved.	-	-	-
9	CLKT	Clock Stretch Timeout 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.	W1C	0x0
8	ERR	ACK Error 0 = No errors detected. 1 = Slave has not acknowledged its address. Cleared by writing 1 to the field.	W1C	0x0
7	RXF	FIFO Full 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
6	TXE	FIFO Empty 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

Bits	Name	Description	Type	Reset
5	RXD	FIFO contains Data 0 = FIFO is empty. 1 = FIFO contains at least 1 byte. Cleared by reading sufficient data from FIFO.	RO	0x0
4	TXD	FIFO can accept Data 0 = FIFO is full. The FIFO cannot accept more data. 1 = FIFO has space for at least 1 byte.	RO	0x1
3	RXR	FIFO needs Reading (¾ full) 0 = FIFO is less than ¾ full and a read is underway. 1 = FIFO is ¾ or more full and a read is underway. Cleared by reading sufficient data from the FIFO.	RO	0x0
2	TXW	FIFO needs Writing (¼ full) 0 = FIFO is at least ¼ full and a write is underway (or sufficient data to send). 1 = FIFO is less than ¼ full and a write is underway. Cleared by writing sufficient data to the FIFO.	RO	0x0
1	DONE	Transfer Done 0 = Transfer not completed. 1 = Transfer complete. Cleared by writing 1 to the field.	W1C	0x0
0	TA	Transfer Active 0 = Transfer not active. 1 = Transfer active.	RO	0x0

DLEN Register

Description

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.

Table 27. DLEN Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	DLEN	Data Length. 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	0x0000

A Register

Description

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.

Table 28. A Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:0	ADDR	Slave Address.	RW	0x00

FIFO Register

Description

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.

Table 29. FIFO Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
7:0	DATA	Writes to the register write transmit data to the FIFO. Reads from register read received data from the FIFO.	RW	0x00

DIV Register

Description

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.

Table 30. DIV Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	CDIV	Clock Divider 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	0x05dc

DEL Register

Description

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.

Table 31. DEL Register

Bits	Name	Description	Type	Reset
31:16	FEDL	Falling Edge Delay Number of core clock cycles to wait after the falling edge of SCL before outputting next bit of data.	RW	0x0030
15:0	REDL	Rising Edge Delay Number of core clock cycles to wait after the rising edge of SCL before reading the next bit of data.	RW	0x0030

CLKT Register

Description

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.

Table 32. CLKT Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	TOUT	Clock Stretch Timeout Value Number of SCL clock cycles to wait after the rising edge of SCL before deciding that the slave is not responding.	RW	0x0040

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 I^2C 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.

3.3.1. Writing

Figure 2. Write to a slave with 10-bit address

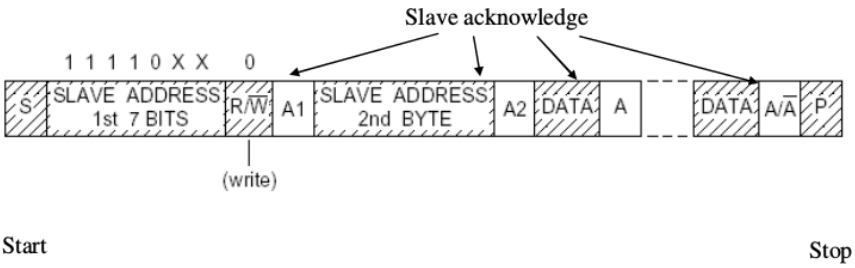


Figure 2 shows a write to a slave with a 10-bit address, to perform this using the controller one must do the following:

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
5. Set I2CC.READ = 0 and I2CC.ST = 1, this will start a write transfer

3.3.2. Reading

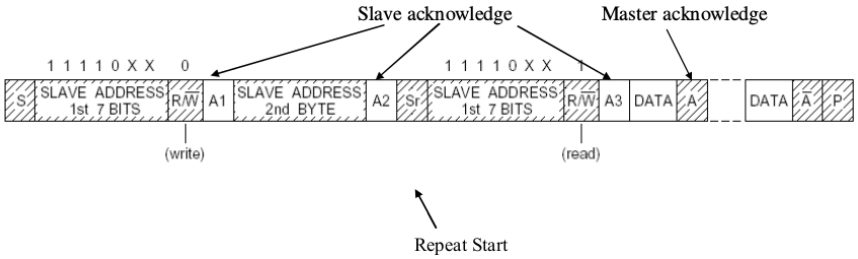


Figure 3. Read from slave with 10-bit address

Figure 3 shows how a read from a slave with a 10-bit address is performed. The following procedure shows how to perform 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
4. Set I2CC.READ = 0 and I2CC.ST = 1, this will start a write transfer
5. Poll the I2CS.TA bit, waiting for the transfer to start
6. Write the number of data bytes to read to the I2CDLEN register
7. Set I2CC.READ = 1 and I2CC.ST = 1, this will send the repeat start bit, the slave address and the R/W bit (which is '1'), initiating the read

Chapter 4. DMA Controller

4.1. Overview

The majority of hardware pipelines and peripherals within the BCM2711 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.

Note that the DMA controller is directly connected to the peripherals. Therefore the DMA controller must be set-up to use the Legacy Master addresses of the peripherals.

The BCM2711 DMA Controller provides a total of 16 DMA channels. Four of these are *DMA Lite* channels (with reduced performance and features), and four of them are *DMA4* channels (with increased performance and a wider address range). Each channel operates independently from the others and is internally arbitrated onto one of the three system busses. This means that the amount of bandwidth that a DMA channel may consume can be controlled by the arbiter settings (although these are not publicly exposed).

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.

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 fewer registers and DMA4 channels have more 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 15 is exclusively used by the VPU.

Table 33. DMA Controller Register Address Map

Base Address	DMA Channel
DMA0_BASE + 0x000	DMA Channel 0 Register Set
DMA0_BASE + 0x100	DMA Channel 1 Register Set

Base Address	DMA Channel
DMA0_BASE + 0x200	DMA Channel 2 Register Set
DMA0_BASE + 0x300	DMA Channel 3 Register Set
DMA0_BASE + 0x400	DMA Channel 4 Register Set
DMA0_BASE + 0x500	DMA Channel 5 Register Set
DMA0_BASE + 0x600	DMA Channel 6 Register Set
DMA0_BASE + 0x700	DMA Channel 7 Register Set
DMA0_BASE + 0x800	DMA Channel 8 Register Set
DMA0_BASE + 0x900	DMA Channel 9 Register Set
DMA0_BASE + 0xa00	DMA Channel 10 Register Set
DMA0_BASE + 0xb00	DMA Channel 11 Register Set
DMA0_BASE + 0xc00	DMA Channel 12 Register Set
DMA0_BASE + 0xd00	DMA Channel 13 Register Set
DMA0_BASE + 0xe00	DMA Channel 14 Register Set
DMA15_BASE + 0x000	DMA Channel 15 Register Set

4.2.1. DMA Channel Register Address Map

Each DMA channel of a particular type 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 channel's 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 different CB data structures in memory, are 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 define the corresponding bit locations in the CB data structure in memory.

Table 34. DMA Control Block Definition

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	Reserved – set to zero.	N/A

Table 35. DMA Lite Control Block Definition

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	Reserved – set to zero.	N/A
5	Next Control Block Address	NEXTCONBK
6-7	Reserved – set to zero.	N/A

Table 36. DMA4 Control Block Definition

32-bit Word Offset	Description	Associated Read-Only Register
0	Transfer Information	TI
1	Source Address	SRC
2	Source Information	SRCI
3	Destination Address	DEST
4	Destination Information	DESTI
5	Transfer Length	LEN
6	Next Control Block Address	NEXT_CB
7	Reserved – set to zero.	N/A

The DMA is started by writing the address of a CB structure into the CONBLK_AD register (or the CB register in the DMA4 channels) and then setting the ACTIVE bit. The DMA will fetch the CB from the address set in the SCB_ADDR field of the CONBLK_AD register (or the ADDR field of the CB register in the DMA4 channels) 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 (or the NEXT_CB register in the DMA4 channels), 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 register and then stop.

Most of the Control Block registers cannot be written to directly as they are 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 / NEXT_CB 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

Table 37. DMA Controller Register Map

Offset	Name	Description
0x000	0_CS	DMA Channel 0 Control and Status
0x004	0_CONBLK_AD	DMA Channel 0 Control Block Address
0x008	0_TI	DMA Channel 0 CB Word 0 (Transfer Information)

Offset	Name	Description
0x00c	0_SOURCE_AD	DMA Channel 0 CB Word 1 (Source Address)
0x010	0_DEST_AD	DMA Channel 0 CB Word 2 (Destination Address)
0x014	0_TXFR_LEN	DMA Channel 0 CB Word 3 (Transfer Length)
0x018	0_STRIDE	DMA Channel 0 CB Word 4 (2D Stride)
0x01c	0_NEXTCONBK	DMA Channel 0 CB Word 5 (Next CB Address)
0x020	0_DEBUG	DMA Channel 0 Debug
0x100	1_CS	DMA Channel 1 Control and Status
0x104	1_CONBLK_AD	DMA Channel 1 Control Block Address
0x108	1_TI	DMA Channel 1 CB Word 0 (Transfer Information)
0x10c	1_SOURCE_AD	DMA Channel 1 CB Word 1 (Source Address)
0x110	1_DEST_AD	DMA Channel 1 CB Word 2 (Destination Address)
0x114	1_TXFR_LEN	DMA Channel 1 CB Word 3 (Transfer Length)
0x118	1_STRIDE	DMA Channel 1 CB Word 4 (2D Stride)
0x11c	1_NEXTCONBK	DMA Channel 1 CB Word 5 (Next CB Address)
0x120	1_DEBUG	DMA Channel 1 Debug
0x200	2_CS	DMA Channel 2 Control and Status
0x204	2_CONBLK_AD	DMA Channel 2 Control Block Address
0x208	2_TI	DMA Channel 2 CB Word 0 (Transfer Information)
0x20c	2_SOURCE_AD	DMA Channel 2 CB Word 1 (Source Address)
0x210	2_DEST_AD	DMA Channel 2 CB Word 2 (Destination Address)
0x214	2_TXFR_LEN	DMA Channel 2 CB Word 3 (Transfer Length)
0x218	2_STRIDE	DMA Channel 2 CB Word 4 (2D Stride)
0x21c	2_NEXTCONBK	DMA Channel 2 CB Word 5 (Next CB Address)
0x220	2_DEBUG	DMA Channel 2 Debug
0x300	3_CS	DMA Channel 3 Control and Status

Offset	Name	Description
0x304	3_CONBLK_AD	DMA Channel 3 Control Block Address
0x308	3_TI	DMA Channel 3 CB Word 0 (Transfer Information)
0x30c	3_SOURCE_AD	DMA Channel 3 CB Word 1 (Source Address)
0x310	3_DEST_AD	DMA Channel 3 CB Word 2 (Destination Address)
0x314	3_TXFR_LEN	DMA Channel 3 CB Word 3 (Transfer Length)
0x318	3_STRIDE	DMA Channel 3 CB Word 4 (2D Stride)
0x31c	3_NEXTCONBK	DMA Channel 3 CB Word 5 (Next CB Address)
0x320	3_DEBUG	DMA Channel 0 Debug
0x400	4_CS	DMA Channel 4 Control and Status
0x404	4_CONBLK_AD	DMA Channel 4 Control Block Address
0x408	4_TI	DMA Channel 4 CB Word 0 (Transfer Information)
0x40c	4_SOURCE_AD	DMA Channel 4 CB Word 1 (Source Address)
0x410	4_DEST_AD	DMA Channel 4 CB Word 2 (Destination Address)
0x414	4_TXFR_LEN	DMA Channel 4 CB Word 3 (Transfer Length)
0x418	4_STRIDE	DMA Channel 4 CB Word 4 (2D Stride)
0x41c	4_NEXTCONBK	DMA Channel 4 CB Word 5 (Next CB Address)
0x420	4_DEBUG	DMA Channel 0 Debug
0x500	5_CS	DMA Channel 5 Control and Status
0x504	5_CONBLK_AD	DMA Channel 5 Control Block Address
0x508	5_TI	DMA Channel 5 CB Word 0 (Transfer Information)
0x50c	5_SOURCE_AD	DMA Channel 5 CB Word 1 (Source Address)
0x510	5_DEST_AD	DMA Channel 5 CB Word 2 (Destination Address)
0x514	5_TXFR_LEN	DMA Channel 5 CB Word 3 (Transfer Length)
0x518	5_STRIDE	DMA Channel 5 CB Word 4 (2D Stride)

Offset	Name	Description
0x51c	5_NEXTCONBK	DMA Channel 5 CB Word 5 (Next CB Address)
0x520	5_DEBUG	DMA Channel 5 Debug
0x600	6_CS	DMA Channel 6 Control and Status
0x604	6_CONBLK_AD	DMA Channel 6 Control Block Address
0x608	6_TI	DMA Channel 6 CB Word 0 (Transfer Information)
0x60c	6_SOURCE_AD	DMA Channel 6 CB Word 1 (Source Address)
0x610	6_DEST_AD	DMA Channel 6 CB Word 2 (Destination Address)
0x614	6_TXFR_LEN	DMA Channel 6 CB Word 3 (Transfer Length)
0x618	6_STRIDE	DMA Channel 6 CB Word 4 (2D Stride)
0x61c	6_NEXTCONBK	DMA Channel 6 CB Word 5 (Next CB Address)
0x620	6_DEBUG	DMA Channel 6 Debug
0x700	7_CS	DMA Lite Channel 7 Control and Status
0x704	7_CONBLK_AD	DMA Lite Channel 7 Control Block Address
0x708	7_TI	DMA Lite Channel 7 CB Word 0 (Transfer Information)
0x70c	7_SOURCE_AD	DMA Lite Channel 7 CB Word 1 (Source Address)
0x710	7_DEST_AD	DMA Lite Channel 7 CB Word 2 (Destination Address)
0x714	7_TXFR_LEN	DMA Lite Channel 7 CB Word 3 (Transfer Length)
0x71c	7_NEXTCONBK	DMA Lite Channel 7 CB Word 5 (Next CB Address)
0x720	7_DEBUG	DMA Lite Channel 7 Debug
0x800	8_CS	DMA Lite Channel 8 Control and Status
0x804	8_CONBLK_AD	DMA Lite Channel 8 Control Block Address
0x808	8_TI	DMA Lite Channel 8 CB Word 0 (Transfer Information)
0x80c	8_SOURCE_AD	DMA Lite Channel 8 CB Word 1 (Source Address)
0x810	8_DEST_AD	DMA Lite Channel 8 CB Word 2 (Destination Address)
0x814	8_TXFR_LEN	DMA Lite Channel 8 CB Word 3 (Transfer Length)

Offset	Name	Description
0x81c	8_NEXTCONBK	DMA Lite Channel 8 CB Word 5 (Next CB Address)
0x820	8_DEBUG	DMA Lite Channel 8 Debug
0x900	9_CS	DMA Lite Channel 9 Control and Status
0x904	9_CONBLK_AD	DMA Lite Channel 9 Control Block Address
0x908	9_TI	DMA Lite Channel 9 CB Word 0 (Transfer Information)
0x90c	9_SOURCE_AD	DMA Lite Channel 9 CB Word 1 (Source Address)
0x910	9_DEST_AD	DMA Lite Channel 9 CB Word 2 (Destination Address)
0x914	9_TXFR_LEN	DMA Lite Channel 9 CB Word 3 (Transfer Length)
0x91c	9_NEXTCONBK	DMA Lite Channel 9 CB Word 5 (Next CB Address)
0x920	9_DEBUG	DMA Lite Channel 9 Debug
0xa00	10_CS	DMA Lite Channel 10 Control and Status
0xa04	10_CONBLK_AD	DMA Lite Channel 10 Control Block Address
0xa08	10_TI	DMA Lite Channel 10 CB Word 0 (Transfer Information)
0xa0c	10_SOURCE_AD	DMA Lite Channel 10 CB Word 1 (Source Address)
0xa10	10_DEST_AD	DMA Lite Channel 10 CB Word 2 (Destination Address)
0xa14	10_TXFR_LEN	DMA Lite Channel 10 CB Word 3 (Transfer Length)
0xa1c	10_NEXTCONBK	DMA Lite Channel 10 CB Word 5 (Next CB Address)
0xa20	10_DEBUG	DMA Lite Channel 10 Debug
0xb00	11_CS	DMA4 Channel 11 Control and Status
0xb04	11_CB	DMA4 Channel 11 Control Block Address
0xb0c	11_DEBUG	DMA4 Channel 11 Debug
0xb10	11_TI	DMA4 Channel 11 CB Word 0 (Transfer Information)
0xb14	11_SRC	DMA4 Channel 11 CB Word 1 (Source Address [31:0])
0xb18	11_SRCI	DMA4 Channel 11 CB Word 2 (Source Address [40:32] and Info)
0xb1c	11_DEST	DMA4 Channel 11 CB Word 3 (Destination Address[31:0])

Offset	Name	Description
0xb20	11_DESTI	DMA4 Channel 11 CB Word 4 (Destination Address[40:32] and Info)
0xb24	11_LEN	DMA4 Channel 11 CB Word 5 (Transfer Length)
0xb28	11_NEXT_CB	DMA4 Channel 11 CB Word 6 (Next CB Address)
0xb2c	11_DEBUG2	DMA4 Channel 11 More Debug
0xc00	12_CS	DMA4 Channel 12 Control and Status
0xc04	12_CB	DMA4 Channel 12 Control Block Address
0xc0c	12_DEBUG	DMA4 Channel 12 Debug
0xc10	12_TI	DMA4 Channel 12 CB Word 0 (Transfer Information)
0xc14	12_SRC	DMA4 Channel 12 CB Word 1 (Source Address [31:0])
0xc18	12_SRCI	DMA4 Channel 12 CB Word 2 (Source Address [40:32] and Info)
0xc1c	12_DEST	DMA4 Channel 12 CB Word 3 (Destination Address[31:0])
0xc20	12_DESTI	DMA4 Channel 12 CB Word 4 (Destination Address[40:32] and Info)
0xc24	12_LEN	DMA4 Channel 12 CB Word 5 (Transfer Length)
0xc28	12_NEXT_CB	DMA4 Channel 12 CB Word 6 (Next CB Address)
0xc2c	12_DEBUG2	DMA4 Channel 12 More Debug
0xd00	13_CS	DMA4 Channel 13 Control and Status
0xd04	13_CB	DMA4 Channel 13 Control Block Address
0xd0c	13_DEBUG	DMA4 Channel 13 Debug
0xd10	13_TI	DMA4 Channel 13 CB Word 0 (Transfer Information)
0xd14	13_SRC	DMA4 Channel 13 CB Word 1 (Source Address [31:0])
0xd18	13_SRCI	DMA4 Channel 13 CB Word 2 (Source Address [40:32] and Info)
0xd1c	13_DEST	DMA4 Channel 13 CB Word 3 (Destination Address[31:0])
0xd20	13_DESTI	DMA4 Channel 13 CB Word 4 (Destination Address[40:32] and Info)
0xd24	13_LEN	DMA4 Channel 13 CB Word 5 (Transfer Length)
0xd28	13_NEXT_CB	DMA4 Channel 13 CB Word 6 (Next CB Address)

Offset	Name	Description
0xd2c	13_DEBUG2	DMA4 Channel 13 More Debug
0xe00	14_CS	DMA4 Channel 14 Control and Status
0xe04	14_CB	DMA4 Channel 14 Control Block Address
0xe0c	14_DEBUG	DMA4 Channel 14 Debug
0xe10	14_TI	DMA4 Channel 14 CB Word 0 (Transfer Information)
0xe14	14_SRC	DMA4 Channel 14 CB Word 1 (Source Address [31:0])
0xe18	14_SRCI	DMA4 Channel 14 CB Word 2 (Source Address [40:32] and Info)
0xe1c	14_DEST	DMA4 Channel 14 CB Word 3 (Destination Address[31:0])
0xe20	14_DESTI	DMA4 Channel 14 CB Word 4 (Destination Address[40:32] and Info)
0xe24	14_LEN	DMA4 Channel 14 CB Word 5 (Transfer Length)
0xe28	14_NEXT_CB	DMA4 Channel 14 CB Word 6 (Next CB Address)
0xe2c	14_DEBUG2	DMA4 Channel 14 More Debug
0xfe0	INT_STATUS	Interrupt status of each DMA channel
0xff0	ENABLE	Global enable bits for each DMA channel

0_CS, 1_CS, …, 9_CS, 10_CS Registers

Description

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

Table 38. 0_CS, 1_CS, …, 9_CS, 10_CS Registers

Bits	Name	Description	Type	Reset
31	RESET	DMA Channel Reset Writing a 1 to this bit will reset the DMA. The bit cannot be read, and will self clear.	W1SC	0x0
30	ABORT	Abort DMA 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	Disable debug pause signal When set to 1, the DMA will not stop when the debug pause signal is asserted.	RW	0x0

Bits	Name	Description	Type	Reset
28	WAIT_FOR_OUTSTANDING_WRITES	Wait for outstanding writes 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. In this mode, the DMA will pause if it has more than 13 outstanding writes at any one time.	RW	0x0
27:24	Reserved.	-	-	-
23:20	PANIC_PRIORITY	AXI Panic Priority Level 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.	RW	0x0
19:16	PRIORITY	AXI Priority Level 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.	RW	0x0
15:9	Reserved.	-	-	-
8	ERROR	DMA Error 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.	RO	0x0
7	Reserved.	-	-	-
6	WAITING_FOR_OUTSTANDING_WRITES	DMA is Waiting for the Last Write to be Received Indicates if the DMA is currently waiting for any outstanding writes to be received, and is not transferring data. 1 = DMA channel is waiting.	RO	0x0
5	DREQ_STOPS_DMA	DMA Paused by DREQ State 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.	RO	0x0
4	PAUSED	DMA Paused State Indicates if the DMA is currently paused and not transferring data. This will occur if: the active bit has been cleared, the DMA is currently executing wait cycles, 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.	RO	0x0

Bits	Name	Description	Type	Reset
3	DREQ	DREQ State Indicates the state of the selected DREQ (Data Request) signal, i.e. 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.	RO	0x0
2	INT	Interrupt Status 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.	W1C	0x0
1	END	DMA End Flag Set when the transfer described by the current Control Block is complete. Write 1 to clear.	W1C	0x0
0	ACTIVE	Activate the DMA 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, i.e. after a NEXTCONBK = 0x0000_0000 has been loaded.	RW	0x0

0_CONBLK_AD, 1_CONBLK_AD, ..., 9_CONBLK_AD, 10_CONBLK_AD Registers

Description

DMA Control Block Address register.

Table 39. 0_CONBLK_AD, 1_CONBLK_AD, ..., 9_CONBLK_AD, 10_CONBLK_AD Registers

Bits	Name	Description	Type	Reset
31:0	SCB_ADDR	Control Block Address 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 CBs). The address must be 256-bit aligned, so the bottom 5 bits of the address must be zero.	RW	0x00000000

0_TI, 1_TI, ..., 5_TI, 6_TI Registers

Description

DMA Transfer Information.

Table 40. 0_TI, 1_TI, ..., 5_TI, 6_TI Registers

Bits	Name	Description	Type	Reset
31:27	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
26	NO_WIDE_BURSTS	Don't do wide writes as a 2 beat burst 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.	RW	0x0
25:21	WAITS	Add Wait Cycles 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	0x00
20:16	PERMAP	Peripheral Mapping 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	0x00
15:12	BURST_LENGTH	Burst Transfer Length 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	Ignore Reads 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.	RW	0x0
10	SRC_DREQ	Control Source Reads with DREQ 1 = The DREQ selected by PERMAP will gate the source reads. 0 = DREQ has no effect.	RW	0x0
9	SRC_WIDTH	Source Transfer Width 1 = Use 128-bit source read width. 0 = Use 32-bit source read width.	RW	0x0
8	SRC_INC	Source Address Increment 1 = Source address increments after each read. The address will increment by 4, if SRC_WIDTH=0 else by 32. 0 = Source address does not change.	RW	0x0
7	DEST_IGNORE	Ignore Writes 1 = Do not perform destination writes. 0 = Write data to destination.	RW	0x0
6	DEST_DREQ	Control Destination Writes with DREQ 1 = The DREQ selected by PERMAP will gate the destination writes. 0 = DREQ has no effect.	RW	0x0
5	DEST_WIDTH	Destination Transfer Width 1 = Use 128-bit destination write width. 0 = Use 32-bit destination write width.	RW	0x0

Bits	Name	Description	Type	Reset
4	DEST_INC	Destination Address Increment 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	Wait for a Write Response 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.	-	-	-
1	TDMODE	2D Mode 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_LEN register as a single transfer of total length {YLENGTH, XLENGTH}.	RW	0x0
0	INTEN	Interrupt Enable 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, …, 9_SOURCE_AD, 10_SOURCE_AD Registers

Description

DMA Source Address

Table 41. 0_SOURCE_AD, 1_SOURCE_AD, …, 9_SOURCE_AD, 10_SOURCE_AD Registers

Bits	Name	Description	Type	Reset
31:0	S_ADDR	DMA Source Address Source address for the DMA operation. Updated by the DMA engine as the transfer progresses.	RW	0x00000000

0_DEST_AD, 1_DEST_AD, …, 9_DEST_AD, 10_DEST_AD Registers

Description

DMA Destination Address

Table 42. 0_DEST_AD, 1_DEST_AD, …, 9_DEST_AD, 10_DEST_AD Registers

Bits	Name	Description	Type	Reset
31:0	D_ADDR	DMA Destination Address Destination address for the DMA operation. Updated by the DMA engine as the transfer progresses.	RW	0x00000000

0_TXFR_LEN, 1_TXFR_LEN, …, 5_TXFR_LEN, 6_TXFR_LEN Registers

Description

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.

Table 43. 0_TXFR_LEN, 1_TXFR_LEN, ..., 5_TXFR_LEN, 6_TXFR_LEN Registers

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
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	0x0000
15:0	XLENGTH	Transfer Length in bytes.	RW	0x0000

0_STRIDE, 1_STRIDE, ..., 5_STRIDE, 6_STRIDE Registers

Description

DMA 2D Stride

Table 44. 0_STRIDE, 1_STRIDE, ..., 5_STRIDE, 6_STRIDE Registers

Bits	Name	Description	Type	Reset
31:16	D_STRIDE	Destination Stride (2D Mode) Signed (2 s complement) byte increment to apply to the destination address at the end of each row in 2D mode.	RW	0x0000
15:0	S_STRIDE	Source Stride (2D Mode) Signed (2 s complement) byte increment to apply to the source address at the end of each row in 2D mode.	RW	0x0000

0_NEXTCONBK, 1_NEXTCONBK, ..., 9_NEXTCONBK, 10_NEXTCONBK Registers

Description

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.

Table 45. 0_NEXTCONBK, 1_NEXTCONBK, ..., 9_NEXTCONBK, 10_NEXTCONBK Registers

Bits	Name	Description	Type	Reset
31:0	ADDR	Address of next CB for chained DMA operations.	RW	0x00000000

0_DEBUG, 1_DEBUG, ..., 5_DEBUG, 6_DEBUG Registers

Description

DMA Debug register.

Table 46. 0_DEBUG, 1_DEBUG, ..., 5_DEBUG, 6_DEBUG Registers

Bits	Name	Description	Type	Reset
31:29	Reserved.	-	-	-
28	LITE	DMA Lite Set if the DMA is a reduced performance LITE engine.	RO	0x0
27:25	VERSION	DMA Version DMA version number, indicating control bit field changes.	RO	0x2
24:16	DMA_STATE	DMA State Machine State Returns the value of the DMA engine's state machine for this channel.	RO	0x000
15:8	DMA_ID	DMA ID Returns the DMA AXI ID of this DMA channel.	RO	0x00

Bits	Name	Description	Type	Reset
7:4	OUTSTANDING_WRITES	DMA Outstanding Writes Counter 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.	-	-	-
2	READ_ERROR	Slave Read Response Error Set if the read operation returned an error value on the read response bus. It can be cleared by writing a 1.	W1C	0x0
1	FIFO_ERROR	FIFO Error Set if the optional read FIFO records an error condition. It can be cleared by writing a 1.	W1C	0x0
0	READ_LAST_NOT_SET_ERROR	Read Last Not Set Error 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.	W1C	0x0

7_TI, 8_TI, 9_TI, 10_TI Registers

Description

DMA Lite Transfer Information.

Table 47. 7_TI, 8_TI, 9_TI, 10_TI Registers

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:21	WAITS	Add Wait Cycles 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	0x00
20:16	PERMAP	Peripheral Mapping 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	0x00
15:12	BURST_LENGTH	Burst Transfer Length 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	Reserved.	-	-	-
10	SRC_DREQ	Control Source Reads with DREQ 1 = The DREQ selected by PERMAP will gate the source reads. 0 = DREQ has no effect.	RW	0x0
9	SRC_WIDTH	Source Transfer Width 1 = Use 128-bit source read width. 0 = Use 32-bit source read width.	RW	0x0

Bits	Name	Description	Type	Reset
8	SRC_INC	Source Address Increment 1 = Source address increments after each read. The address will increment by 4, if SRC_WIDTH=0 else by 32. 0 = Source address does not change.	RW	0x0
7	Reserved.	-	-	-
6	DEST_DREQ	Control Destination Writes with DREQ 1 = The DREQ selected by PERMAP will gate the destination writes. 0 = DREQ has no effect.	RW	0x0
5	DEST_WIDTH	Destination Transfer Width 1 = Use 128-bit destination write width. 0 = Use 32-bit destination write width.	RW	0x0
4	DEST_INC	Destination Address Increment 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	Wait for a Write Response 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	Reserved.	-	-	-
0	INTEN	Interrupt Enable 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 Registers

Description

DMA Lite Transfer Length

Table 48. 7_TXFR_LEN, 8_TXFR_LEN, 9_TXFR_LEN, 10_TXFR_LEN Registers

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	XLENGTH	Transfer Length Length of transfer, in bytes. Updated by the DMA engine as the transfer progresses.	RW	0x0000

7_DEBUG, 8_DEBUG, 9_DEBUG, 10_DEBUG Registers

Description

DMA Lite Debug register.

Table 49. 7_DEBUG, 8_DEBUG, 9_DEBUG, 10_DEBUG Registers

Bits	Name	Description	Type	Reset
31:29	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
28	LITE	DMA Lite Set if the DMA is a reduced performance LITE engine.	RO	0x1
27:25	VERSION	DMA Version DMA version number, indicating control bit field changes.	RO	0x2
24:16	DMA_STATE	DMA State Machine State Returns the value of the DMA engine's state machine for this channel.	RO	0x000
15:8	DMA_ID	DMA ID Returns the DMA AXI ID of this DMA channel.	RO	0x00
7:4	OUTSTANDING_WRITES	DMA Outstanding Writes Counter 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.	-	-	-
2	READ_ERROR	Slave Read Response Error Set if the read operation returned an error value on the read response bus. It can be cleared by writing a 1.	W1C	0x0
1	FIFO_ERROR	FIFO Error Set if the optional read FIFO records an error condition. It can be cleared by writing a 1.	W1C	0x0
0	READ_LAST_NOT_SET_ERROR	Read Last Not Set Error 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.	W1C	0x0

11_CS, 12_CS, 13_CS, 14_CS Registers

Description

DMA4 Control and Status register contains the main control and status bits for this DMA4 channel.

Table 50. 11_CS, 12_CS, 13_CS, 14_CS Registers

Bits	Name	Description	Type	Reset
31	HALT	Writing a 1 to this bit will cleanly halt the current DMA transfer. The halt will cause the DMA4 to zero its length counters and thus it will complete the current transfer and wait until all outstanding bus activity has finished. The DMA4 will then zero the active flag and return to idle, leaving the address of the aborted CB in the CB reg. The halt bit will self clear when the DMA4 has fully stopped. The Halt bit can be automatically set if the DMA4 detects an error and the debug HALT_ON_ERROR bit is set.	W1SC	0x0

Bits	Name	Description	Type	Reset
30	ABORT	Abort DMA Writing a 1 to this bit will cleanly abort the current DMA transfer. The abort will cause the DMA4 to zero its length counters and thus it will complete the current transfer and wait until all outstanding bus activity has finished. The DMA4 will then check the NEXT_CB address and if it is non zero it will load it into the CB and attempt to continue. The abort bit will self clear when the abort has completed. The abort bit can be automatically set if the DMA4 detects an error and the debug ABORT_ON_ERROR bit is set.	W1SC	0x0
29	DISDEBUG	Disable Debug Pause Signal When set to 1, the DMA4 will not pause when the debug pause signal is asserted. Normally the DMA4 will pause when the debugger asserts the debug_pause DMA control signal. This prevents the DMA4 from running on ahead when the processor is stopped by the debugger. Debug_pause will cleanly pause the DMA4 by preventing it from issuing new commands. Releasing the debug_pause will allow the DMA4 to carry on where it left off.	RW	0x0
28	WAIT_FOR_OUTSTANDING_WRITES	Wait for outstanding writes. The DMA4 keeps a tally of the AXI writes requests going out and the write responses coming in. When set to 1, the DMA4 will complete the current transfer and then wait until the last outstanding write response has been received and the tally has returned to zero. Only then will it indicate that the transfer is complete and set the END flag or if required the INT and move on to the next CB. The number of outstanding writes will be limited by the FIFO size as set by the instantiation parameters.	RW	0x0
27:26	Reserved.	-	-	-
25	OUTSTANDING_TRANSACTIONS	Indicates that there are outstanding AXI transfers, either outstanding read data or outstanding write responses This just indicates that the outstanding counters in DEBUG2 are >0	RO	0x0
24	DMA_BUSY	Indicates the DMA4 is BUSY. This indicates that the DMA4 is operating or waiting for outstanding data or otherwise in use. It can be used as an indicator of when it is safe to powersave the DMA4 and turn off all the clocks by using the global DMA_EN bits	RO	0x0
23:20	PANIC_QOS	AXI Panic QOS Level Sets the QOS level of AXI bus transactions when the DMA4 is panicking. This value is used when the panic bit of the selected peripheral channel is 1 indicating that the peripheral is in panic mode. Zero is the lowest QOS.	RW	0x0

Bits	Name	Description	Type	Reset
19:16	QOS	AXI QOS Level Sets the QOS level of normal AXI bus transactions. This value is used when the panic bit of the selected peripheral channel is zero or when no peripheral is selected as in a memory to memory transfer. Zero is the lowest QOS.	RW	0x0
15:11	Reserved.	-	-	-
10	ERROR	DMA Error Indicates if the DMA4 has detected an error. The error flags are available in the debug register, and are cleared by reading that register. 1 = there is error flag set. 0 = no errors.	RO	0x0
9:8	Reserved.	-	-	-
7	WAITING_FOR_OUTSTANDING_WRITES	The DMA4 is Waiting for all the Write Response to be returned. If WAIT_FOR_OUTSTANDING_WRITES is enabled, the DMA4 will complete its transfer and then enter a waiting state where it waits for all the outstanding write responses to be returned. When they are all accounted for, the DMA4 will indicate that the transfer is complete and set the END or INT flags and move on to the next CB. 1 = The DMA4 is waiting for outstanding bresponses.	RO	0x0
6	DREQ_STOPS_DMA	DMA Paused by DREQ State This indicates that the DMA4 is currently paused and not transferring data due to the selected DREQ being inactive. The DMA4 has either src_dreq or dest_dreq set in its CB and the permap value will be indicating which of the DREQ line it should select. If this DREQ line is low then the DMA4 will be paused waiting for the peripheral to request more data. 1 = DMA is paused. 0 = DMA is running.	RO	0x0
5	WR_PAUSED	DMA Write Paused State Indicates that the DMA4 is currently paused and not writing data. This will occur if: the active bit has been cleared, if the debug_pause signal has been set by the debug block, or the selected peripheral dreq input isn't set and writes are gated by dreq. 1 = paused for writes. 0 = running.	RO	0x0

Bits	Name	Description	Type	Reset
4	RD_PAUSED	DMA read Paused State Indicates that the DMA4 is currently paused and not reading data. This will occur if: the active bit has been cleared or if the debug_pause signal has been set by the debug block or the selected peripheral dreq isn't set and reads are gated by dreq. 1 = paused for reads. 0 = running.	RO	0x0
3	DREQ	DREQ State Indicates the state of the selected DREQ (Data Request) signal, i.e. 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.	RO	0x1
2	INT	Interrupt Status If interrupts are enabled (INTEN is set to 1) the interrupt is set when the transfer for the current CB is completed. If WAIT_FOR_OUTSTANDING_WRITES is enabled, the DMA4 will wait for all the outstanding bresponses before setting the INT bit. Once set it must be manually cleared by writing a 1 to this bit, even if the next CB has INTEN = 0. The interrupt can also be set if the INT_ON_ERROR debug bit is set and an error is detected. An error interrupt won't be set until the current CB has completed. Write 1 to clear.	W1C	0x0
1	END	End Flag Set when the transfer described by the current Control Block is complete. If WAIT_FOR_OUTSTANDING_WRITES is enabled, the DMA4 will wait for all the outstanding bresponses before setting the end bit. Once set it must be manually cleared by writing a 1 to this bit. Write 1 to clear.	W1C	0x0
0	ACTIVE	Activate the DMA4 This bit enables the DMA4 to start transferring data. The DMA4 will start operating if this bit is set and the CB is non zero. The DMA transfer can be cleanly paused and re-started in mid transfer by clearing and setting this active bit. The DMA4 will pause at a safe AXI transaction point. This bit is automatically cleared at the end of the CB linked List, i.e. after a CB with a NEXTCONBK = 0x0000_0000 has been executed.	RW	0x0

11_CB, 12_CB, 13_CB, 14_CB Registers

Description

DMA4 Control Block Address register.

Table 51. 11_CB, 12_CB, 13_CB, 14_CB Registers

Bits	Name	Description	Type	Reset
31:0	ADDR	Control Block Address [36:5] This tells the DMA4 where to find a Control Block (CB) stored in memory. The address must be 256-bit aligned, i.e. the bottom 5 address bits are 0. To support a larger address range, the bottom 5 bits of the CB address are not written here i.e. you should write CB_byte_addr>>5 The DMA4 will not start unless this address is non zero and the ACTIVE bit has been set. Once the CB and Active are set, the DMA4 will start by reading the CB from the given address and loading the data into the relevant CB registers It will then execute the DMA described by the CB regardless of what it is, so if garbage is read then it will execute it. At the end of the DMA transfer described by the CB, the NEXTCONBK field of the CB will be loaded into to this CB_ADDR register and if it is non zero another DMA sequence will begin anew. Reading this register will return the address of the currently active CB.	RW	0x00000000

11_DEBUG, 12_DEBUG, 13_DEBUG, 14_DEBUG Registers

Description

DMA4 Debug register.

Table 52. 11_DEBUG, 12_DEBUG, 13_DEBUG, 14_DEBUG Registers

Bits	Name	Description	Type	Reset
31:28	VERSION	DMA Version DMA version number, indicating control bit field changes.	RO	0x1
27:24	ID	ID Returns the ID of this DMA4. This is also used as the AXI subid	RO	0x0
23	RESET	DMA Reset This is a hard reset of the DMA4 state machine and certain internal registers. Writing a 1 to this bit will reset the DMA4. The bit cannot be read, and will self clear. Using this in the middle of a DMA transfer or when the AXI bus is active or has outstanding transactions will probably be fatal.	W1SC	0x0
22	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
21:18	W_STATE	Write State Machine State Returns the value of the DMA4 engine's write state machine. W_IDLE = 0 W_PRELOAD = 1 W_CALC = 2 W_WRITE4K = 3 W_READFIFO_EMPTY = 4 W_WAIT_OUTSTANDING = 5	RO	0x0
17:14	R_STATE	Read State Machine State Returns the value of the DMA4 engine's read state machine. R_IDLE = 0 R_WAIT_CB_DATA = 1 R_CALC = 2 R_READ4K = 3 R_READING = 4 R_READFIFO_FULL = 5 R_WAIT_WRITE_COMPLETE = 6	RO	0x0
13:12	Reserved.	-	-	-
11	DISABLE_CLK_GATE	Disable the clock gating logic.	RW	0x0
10	ABORT_ON_ERROR	Instruct the DMA4 to ABORT if it detects an error. If any of the error conditions are met then the CS_ABORT bit will be set causing the DMA4 to terminate the current CB and attempt to start the next one. Clearing this bit will allow it to plough on regardless and probably trample over the entire memory.	RW	0x1
9	HALT_ON_ERROR	Instruct the DMA4 to HALT if it detects an error. If any of the error conditions are met then the CS_HALT bit will be set causing the DMA4 to stop. This will override the abort on error behaviour.	RW	0x0
8	INT_ON_ERROR	Generate an interrupt if an error is detected This forces the DMA4 to generate an error even if the inten bit in the TI isn't set The int will be generated when the DMA4 finishes the current CB	RW	0x0
7:4	Reserved.	-	-	-
3	READ_CB_ERROR	Slave Read Response Error During Control Block Read Set if the read operation returned an error value on the read response bus whilst reading the CB. It is cleared by reading.	RC	0x0
2	READ_ERROR	Slave Read Response Error Set if the read operation returned an error value on the read response bus during a data read. It is cleared by reading.	RC	0x0

Bits	Name	Description	Type	Reset
1	FIFO_ERROR	FIFO Error Set if the optional read FIFO records an error condition (read when empty or write when full). It is cleared by reading.	RC	0x0
0	WRITE_ERROR	Slave Write Response Error Set if a write operation returned an error value on the write response bus. It is cleared by reading.	RC	0x0

11_TI, 12_TI, 13_TI, 14_TI Registers

Description

DMA4 Transfer Information.

Table 53. 11_TI, 12_TI, 13_TI, 14_TI Registers

Bits	Name	Description	Type	Reset
31:24	D_WAITS	Write Wait Cycles This slows down the DMA throughput by setting the number of dummy cycles before each AXI Write operation is started. A value of 0 means that no wait cycles are to be added. Waits are counted down when the DMA4 wants to do a wait and the bus is available and the writes aren't paused because of DREQS or some other reason.	RW	0x00
23:16	S_WAITS	Read Wait Cycles This slows down the DMA throughput by setting the number of dummy cycles burnt before each DMA AXI read operation is started. A value of 0 means that no wait cycles are to be added. Waits are counted down when the DMA4 wants to do a read and the bus is available and the reads aren't paused because of DREQS or some other reason.	RW	0x00

Bits	Name	Description	Type	Reset
15	D_DREQ	Control Destination Writes with DREQ. This is used when writing to a peripheral that has a DREQ flow control available to control the data flow. The DMA4 will observe the DREQ input selected by the PERMAP value and pause writes when it is low. Care must be taken when using this as the DMA4 will only stop writing peripheral data a clock cycle or two after it sees a low DREQ at its input. However there may still be outstanding write data in the pipeline formed by the infrastructure between the DMA and the peripheral. The peripheral must take this into account when deciding when to drop its DREQ signal, and must have spare FIFO room to accommodate the data that's still in flight. The WAIT_RESP feature can be used to ensure there is only ever 1 outstanding write at any time for use with peripherals that can't provide any spare storage for any in-flight data. The D_WAITS feature can be used to add a delay before each write to allow the DREQ more time to make it back to the DMA. 1 = The DREQ selected by PERMAP will gate the Destination writes. 0 = DREQ has no effect.	RW	0x0
14	S_DREQ	Control Source Reads with DREQ This is used when reading from a peripheral that has a DREQ flow control available. The DMA will observe the DREQ input selected by the permap value and pause reads when it is low. Care must be taken when using this as the DMA4 will only stop issuing peripheral read requests a clock cycle or two after it sees a low DREQ at its input. The AXI infrastructure will allow several read requests to become queued outside of the DMA4 engine so it's possible to request far more data may than a peripheral can immediately supply. If this happens then the infrastructure may become locked until the data is available and this will adversely affect system performance. The WAIT_RD_RESP option prevents the DMA4 from issuing more than 1 read request at a time, so the amount of data requested can be governed by the burst size, and this allows more time for the peripheral to retract its DREQ when it runs out of data. The S_WAITS feature can be used to add a delay before each read to allow the DREQ more time to make it back to the DMA. 1 = The DREQ selected by PERMAP will gate the source reads. 0 = DREQ has no effect.	RW	0x0

Bits	Name	Description	Type	Reset
13:09	PERMAP	Peripheral Mapping Indicates the DREQ of selected peripheral (1-31). The DMA4 will select the DREQ from this peripheral and use that to control the rate of read or write transfers. The DMA4 will also select the panic signals from this peripheral and use that to set the QOS level on the AXI bus. Setting a PERMAP of 0 selects a dummy peripheral that is always active for a continuous un-paced transfer.	RW	0x00
8:4	Reserved.	-	-	-
3	WAIT_RD_RESP	Wait for a Read Response When set this makes the DMA4 wait until it receives all the data from each read. This ensures that multiple reads cannot get stacked in the AXI bus pipeline. This allows the amount of data to be controlled by the burst size, e.g. when reading for a peripheral FIFO 1 = Wait for the read data to be received before proceeding. 0 = Don't wait; allow multiple reads to be queued.	RW	0x0
2	WAIT_RESP	Wait for a Write Response When set this makes the DMA4 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
1	TDMODE	2D Mode - perform a 2D transfer instead of a normal linear transfer. In 2D mode the DMA4 will interpret the length field as an X and a Y length. It will execute Y+1 transfers each of length X. After each X transfer, the DMA4 will add the value in the STRIDE registers to the source and destination address. If 2D mode isn't selected then the DMA4 interprets the X&Y lengths as a single 30bit length and performs one transfer of that number of bytes. 1 = 2D mode - perform Y+1 transfers of X bytes 0 = Linear mode interpret the LEN register as a single transfer of total length {YLENGTH, XLENGTH} bytes.	RW	0x0
0	INTEN	Interrupt Enable 1 = Generate an interrupt when the transfer described by the current Control Block completes. 0 = Do not generate an interrupt.	RW	0x0

11_SRC, 12_SRC, 13_SRC, 14_SRC Registers

Description

Lower 32 bits of the DMA4 Source Address
The DMA4 can handle up to 40bit addresses so the full source address is split over 2 registers.

Table 54. 11_SRC, 12_SRC, 13_SRC, 14_SRC Registers

Bits	Name	Description	Type	Reset
31:0	ADDR	Lower bits of the Source Address [31:0] This specifies the BYTE address that the DMA4 should read source data from. The address is BYTE aligned allowing transfers from any byte address to any other byte address. This reg value is automatically updated by the DMA4 engine as the transfer progresses, so it indicates the current address being read.	RW	0x00000000

11_SRCI, 12_SRCI, 13_SRCI, 14_SRCI Registers

Description

DMA4 Source Information

This contains the high bits of the source address[40:32] as well as other source control bits

Table 55. 11_SRCI, 12_SRCI, 13_SRCI, 14_SRCI Registers

Bits	Name	Description	Type	Reset
31:16	STRIDE	Source Stride This is only used in 2D transfer mode (TDMODE). In a 2D transfer the DMA4 will perform Y transfers each of X bytes. At the end of each X row, the source stride is added to the source address and this is used as the start address of the source data for the next X row. The source stride is a signed (2 s complement) byte increment so negative values are allowed.	RW	0x0000
15	IGNORE	Ignore Reads. The DMA4 will perform a normal transfer except that it will not produce any reads. The DMA4 will write zero data. 1 = Do not perform source reads. 0 = Perform source reads.	RW	0x0
14:13	SIZE	Source Transfer Width The DMA4 will perform all AXI source reads with this AXI transfer width. Data will be fetched in bursts of this width and assembled into the correct data size inside the DMA4. On the BCM2711 the width cannot be set larger than 128 bits. 3 = 256 2 = 128 1 = 64 0 = 32	RW	0x0
12	INC	Increment the Source Address 1 = Source address increments after each read. The address will increment by by the number of bytes in the transfer width. 0 = Source address does not change. Data will always be read from the same source address with an AXI "Fixed" transfer. This is intended to be used to read from a peripheral FIFO type of source.	RW	0x0

Bits	Name	Description	Type	Reset
11:08	BURST_LENGTH	Burst Transfer Length Indicates the maximum burst length of the source reads. The DMA4 will attempt to transfer data as bursts of this number of words unless it will cause a 4k crossing or there isn't enough data required. A value of zero will produce a single-beat transfer.	RW	0x0
7:0	ADDR	High Bits of the Source Address [40:32] The source address is split over 2 registers, and together they give a 40-bit address	RW	0x00

11_DEST, 12_DEST, 13_DEST, 14_DEST Registers

Description

Lower 32 bits of the DMA4 Destination Address

The DMA4 can handle up to 40bit addresses so the full address is split over 2 registers.

Table 56. 11_DEST, 12_DEST, 13_DEST, 14_DEST Registers

Bits	Name	Description	Type	Reset
31:0	ADDR	Destination Address This specifies the BYTE address that the DMA4 should write data to. The address is BYTE aligned allowing transfers from any byte address to any other byte address. This register value is automatically updated by the DMA4 engine as the transfer progresses, so it indicates the current address being written.	RW	0x00000000

11_DESTI, 12_DESTI, 13_DESTI, 14_DESTI Registers

Description

DMA4 Destination Information

This contains the high bits of the destination address [40:32] and other information bits for the destination

Table 57. 11_DESTI, 12_DESTI, 13_DESTI, 14_DESTI Registers

Bits	Name	Description	Type	Reset
31:16	STRIDE	Destination Stride This is only used in 2D transfer mode. In a 2D transfer the DMA4 will perform Y transfers each of X bytes. At the end of each X row, the destination stride is added to the destination address and this is used as the start address of the destination for the next X row. The destination stride is a signed (2 s complement) byte increment so negative values are allowed.	RW	0x0000
15	IGNORE	Ignore Destination Writes 1 = Do not perform destination Writes. The DMA4 will read the source data but not write it. 0 = Perform destination Writes.	RW	0x0

Bits	Name	Description	Type	Reset
14:13	SIZE	Destination Transfer Width The DMA4 will perform all AXI destination writes with this AXI transfer width. Data will be written in bursts of this width. On the BCM2711 the width cannot be set larger than 128 bits. 3 = 256 2 = 128 1 = 64 0 = 32	RW	0x0
12	INC	Destination Address Increment 1 = Destination address increments after each write. The address will increment by by the number bytes in the transfer width. 0 = Destination address does not change.	RW	0x0
11:08	BURST_LENGTH	Burst Transfer Length Indicates the maximum burst length of the destination writes. The DMA4 will attempt to transfer data as bursts of this number of words unless it will cause a 4k crossing or there isn't enough data required. A value of zero will produce a single transfer.	RW	0x0
7:0	ADDR	High Bits of the Destination Address [40:32] The destination address is split over 2 registers, and together they give a 40-bit address	RW	0x00

11_LEN, 12_LEN, 13_LEN, 14_LEN Registers

Description

DMA4 Transfer Length.
This specifies the amount of data to be transferred in bytes.
In normal (non 2D) mode the X&Y are combined to specifies the number of bytes to be transferred up to a max of 2^30-1.
In 2D mode it is interpreted as an X and a Y length, and the DMA4 will perform Y+1 transfers, each of length X bytes.
In 2D mode the source and destination strides are added onto the source and destination addresses after each X leg of the transfer.
The length register is updated by the DMA4 engine as the transfer progresses, so it will indicate the data left to transfer.

Table 58. 11_LEN, 12_LEN, 13_LEN, 14_LEN Registers

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
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 In 2D mode a value of 0 will result in a single XLENGTH transfer and a value of 1 will result in 2 XLENGTH transfers.	RW	0x0000
15:0	XLENGTH	Transfer Length in bytes. A value of 1 will transfer 1 byte A value of 0 is illegal	RW	0x0000

11_NEXT_CB, 12_NEXT_CB, 13_NEXT_CB, 14_NEXT_CB Registers

Description

DMA4 Next Control Block Address
When the current DMA transfer has completed, the Next Control Block address is transferred to the CB address register and if the active bit is still set the next DMA in the linked list of CBs is begun.
A CB with a Next Control Block Address of 0 indicates the end of the list. Once that CB is executed the zero next CB will be loaded and the DMA will stop (as the start condition for the DMA4 is (ACTIVE & CB!=0).
The value loaded into this 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 DMA4 is paused.
The address must be 256-bit aligned and so the bottom 5 bits of the byte address are discarded, i.e. write cb_byte_address[39:0]>>5 into the CB.

Table 59. 11_NEXT_CB, 12_NEXT_CB, 13_NEXT_CB, 14_NEXT_CB Registers

Bits	Name	Description	Type	Reset
31:0	ADDR	Address of next CB for chained DMA operations.	RW	0x00000000

11_DEBUG2, 12_DEBUG2, 13_DEBUG2, 14_DEBUG2 Registers

Description

DMA4 Debug2 register.

Table 60. 11_DEBUG2, 12_DEBUG2, 13_DEBUG2, 14_DEBUG2 Registers

Bits	Name	Description	Type	Reset
31:25	Reserved.	-	-	-
24:16	OUTSTANDING_READS	Outstanding read Words Count This indicates the number of outstanding read words. This keeps count of the number of read words that have been requested and the number that have actually been returned. This should be zero at the end of every transfer	RO	0x000
15:9	Reserved.	-	-	-
8:0	OUTSTANDING_WRITES	Outstanding Write Response Count This indicates the number of outstanding write responses. This keeps count of the number of write requests that have been generated and the number of bresponses that have been returned. This should be zero at the end of every transfer	RO	0x000

INT_STATUS Register

Description

Interrupt status of each DMA engine

Table 61. INT_STATUS Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15	INT15	Interrupt status of DMA engine 15	RO	0x0
14	INT14	Interrupt status of DMA engine 14	RO	0x0
13	INT13	Interrupt status of DMA engine 13	RO	0x0
12	INT12	Interrupt status of DMA engine 12	RO	0x0
11	INT11	Interrupt status of DMA engine 11	RO	0x0
10	INT10	Interrupt status of DMA engine 10	RO	0x0
9	INT9	Interrupt status of DMA engine 9	RO	0x0

Bits	Name	Description	Type	Reset
8	INT8	Interrupt status of DMA engine 8	RO	0x0
7	INT7	Interrupt status of DMA engine 7	RO	0x0
6	INT6	Interrupt status of DMA engine 6	RO	0x0
5	INT5	Interrupt status of DMA engine 5	RO	0x0
4	INT4	Interrupt status of DMA engine 4	RO	0x0
3	INT3	Interrupt status of DMA engine 3	RO	0x0
2	INT2	Interrupt status of DMA engine 2	RO	0x0
1	INT1	Interrupt status of DMA engine 1	RO	0x0
0	INT0	Interrupt status of DMA engine 0	RO	0x0

ENABLE Register

Description

Global enable bits for each channel.

Setting these to 0 will disable the DMA for power saving reasons. Disabling whilst the DMA is operating will be fatal.

Table 62. ENABLE Register

Bits	Name	Description	Type	Reset
31:28	PAGELITE	Set the 1G SDRAM ram page that the DMA Lite engines (DMA7-10) will access when addressing the 1G uncached range C000_0000->ffff_ffff E.g. setting this to 1 will mean that when the DMA writes to C000_0000 (uncached) the final address in SDRAM will be 4000_0000 (pagelite<<30 | addr[29:0]) This allows the 1G uncached page to be moved around the 16G SDRAM space	RW	0x0
27:24	PAGE	Set the 1G SDRAM ram page that the 30-bit DMA engines (DMA0-6) will access when addressing the 1G uncached range C000_0000->ffff_ffff E.g. setting this to 1 will mean that when the DMA writes to C000_0000 (uncached) the final address in SDRAM will be 4000_0000 (page<<30 | addr[29:0]) This allows the 1G uncached page to be moved around the 16G SDRAM space	RW	0x0
23:15	Reserved.	-	-	-
14	EN14	Enable DMA engine 14	RW	0x1
13	EN13	Enable DMA engine 13	RW	0x1
12	EN12	Enable DMA engine 12	RW	0x1
11	EN11	Enable DMA engine 11	RW	0x1
10	EN10	Enable DMA engine 10	RW	0x1
9	EN9	Enable DMA engine 9	RW	0x1
8	EN8	Enable DMA engine 8	RW	0x1
7	EN7	Enable DMA engine 7	RW	0x1
6	EN6	Enable DMA engine 6	RW	0x1

Bits	Name	Description	Type	Reset
5	EN5	Enable DMA engine 5	RW	0x1
4	EN4	Enable DMA engine 4	RW	0x1
3	EN3	Enable DMA engine 3	RW	0x1
2	EN2	Enable DMA engine 2	RW	0x1
1	EN1	Enable DMA engine 1	RW	0x1
0	EN0	Enable DMA engine 0	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.

DREQs 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 is does not have room for, so no pacing of the data flow is required.

DREQs 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 a less efficient transfer mechanism.

The mapping of peripherals to DREQs is as follows:

DREQ	Peripheral
0	DREQ = 1 This is always on so use this channel if no DREQ is required.
1	DSI0 / PWM1 **
2	PCM TX
3	PCM RX
4	SMI
5	PWM0
6	SPI0 TX
7	SPI0 RX
8	BSC/SPI Slave TX

DREQ	Peripheral
9	BSC/SPI Slave RX
10	HDMI0
11	e.MMC
12	UART0 TX
13	SD HOST
14	UART0 RX
15	DSI1
16	SPI1 TX
17	HDMI1
18	SPI1 RX
19	UART3 TX / SPI4 TX **
20	UART3 RX / SPI4 RX **
21	UART5 TX / SPI5 TX **
22	UART5 RX / SPI5 RX **
23	SPI6 TX
24	Scaler FIFO 0 & SMI *
25	Scaler FIFO 1 & SMI *
26	Scaler FIFO 2 & SMI *
27	SPI6 RX
28	UART2 TX
29	UART2 RX
30	UART4 TX
31	UART4 RX

* 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.
** The alternate DREQs are available by changing the DMA_CNTRL_MUX bits in the PACTL_CS register (at address `0x7E20 4E00`) as detailed in the following table:

Bits	Name	Description	Type	Reset
25	DMA_CNTRL_MUX_1	This controls the source of DMA DREQ channel 21 and 22: 0 selects UART5 TX on channel 21 and RX on channel 22 1 selects SPI5 TX on channel 21 and RX on channel 22	RW	0x0
24	DMA_CNTRL_MUX_0	This controls the source of DMA DREQ channel 19 and 20: 0 selects UART3 TX on channel 19 and RX on channel 20 1 selects SPI4 TX on channel 19 and RX on channel 20	RW	0x0
23	DMA_CNTRL_MUX_2	This controls the source of DMA DREQ channel 1: 0 selects DSI0 1 selects PWM1	RW	0x1

Other bits in this register should be treated as reserved, and only written back with the previously read value.

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, source 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 DMA's 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 signal being 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-ed 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 transferable 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.

4.6. DMA4 Engines

Several of the DMA engines are of the DMA4 design. These have higher performance due to their uncoupled read/write design and can access up to 40 address bits. Unlike the other DMA engines they are also capable of performing write bursts. Note that they directly access the full 35-bit address bus of the BCM2711 and so bypass the paging registers of the DMA and DMA Lite engines.

DMA channel 11 is additionally able to access the PCIe interface.

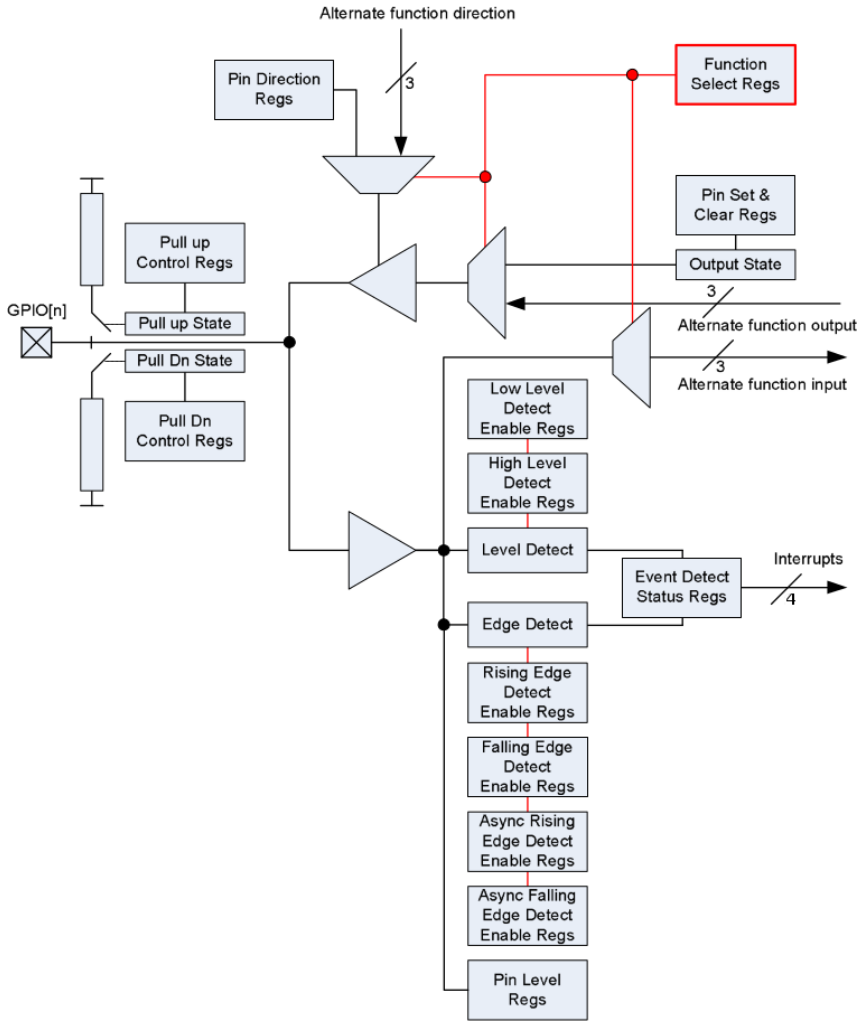
Chapter 5. General Purpose I/O (GPIO)

5.1. Overview

There are 58 General-Purpose Input/Output (GPIO) lines split into three banks. Bank 0 contains GPIOs 0 to 27, bank 1 contains GPIOs 28 to 45, and bank 2 contains GPIOs 46 to 57. All GPIO pins have at least two alternative functions within BCM2711. The alternate functions are usually peripheral IO, and a single peripheral may appear in multiple banks to allow flexibility on the choice of IO voltage (as each bank has a selectable IO voltage). Details of alternative functions are given in Section 5.3.

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

Figure 4. GPIO Block Diagram



The GPIO peripheral has four 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 fourth line shared between all bits.

The Alternate function table (Table 94) also has the pull state (pull-up/pull-down) which is applied after a power down.

5.2. Register View

The GPIO has the following registers. All accesses are assumed to be 32-bit. The GPIO register base address is `0x7e200000`.

Table 63. GPIO Register Assignment

Offset	Name	Description
0x00	GPFSEL0	GPIO Function Select 0
0x04	GPFSEL1	GPIO Function Select 1
0x08	GPFSEL2	GPIO Function Select 2
0x0c	GPFSEL3	GPIO Function Select 3
0x10	GPFSEL4	GPIO Function Select 4
0x14	GPFSEL5	GPIO Function Select 5
0x1c	GPSET0	GPIO Pin Output Set 0
0x20	GPSET1	GPIO Pin Output Set 1
0x28	GPCLR0	GPIO Pin Output Clear 0
0x2c	GPCLR1	GPIO Pin Output Clear 1
0x34	GPLEV0	GPIO Pin Level 0
0x38	GPLEV1	GPIO Pin Level 1
0x40	GPEDS0	GPIO Pin Event Detect Status 0
0x44	GPEDS1	GPIO Pin Event Detect Status 1
0x4c	GPREN0	GPIO Pin Rising Edge Detect Enable 0
0x50	GPREN1	GPIO Pin Rising Edge Detect Enable 1
0x58	GPFEN0	GPIO Pin Falling Edge Detect Enable 0
0x5c	GPFEN1	GPIO Pin Falling Edge Detect Enable 1
0x64	GPHEN0	GPIO Pin High Detect Enable 0
0x68	GPHEN1	GPIO Pin High Detect Enable 1
0x70	GPLEN0	GPIO Pin Low Detect Enable 0
0x74	GPLEN1	GPIO Pin Low Detect Enable 1
0x7c	GPAREN0	GPIO Pin Async. Rising Edge Detect 0
0x80	GPAREN1	GPIO Pin Async. Rising Edge Detect 1
0x88	GPAFEN0	GPIO Pin Async. Falling Edge Detect 0
0x8c	GPAFEN1	GPIO Pin Async. Falling Edge Detect 1
0xe4	GPIO_PUP_PDN_CNTRL_REG0	GPIO Pull-up / Pull-down Register 0
0xe8	GPIO_PUP_PDN_CNTRL_REG1	GPIO Pull-up / Pull-down Register 1
0xec	GPIO_PUP_PDN_CNTRL_REG2	GPIO Pull-up / Pull-down Register 2
0xf0	GPIO_PUP_PDN_CNTRL_REG3	GPIO Pull-up / Pull-down Register 3

GPFSEL0 Register

Description

The function select registers are used to define the operation of the general-purpose I/O pins. Each of the 58 GPIO pins has at least two alternative functions as defined in Section 5.3. The FSEL*n* field determines the functionality of the *n*th 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.

Table 64. GPIO Alternate function select register 0

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
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	RW	0x0
26:24	FSEL8	FSEL8 - Function Select 8	RW	0x0
23:21	FSEL7	FSEL7 - Function Select 7	RW	0x0
20:18	FSEL6	FSEL6 - Function Select 6	RW	0x0
17:15	FSEL5	FSEL5 - Function Select 5	RW	0x0
14:12	FSEL4	FSEL4 - Function Select 4	RW	0x0
11:9	FSEL3	FSEL3 - Function Select 3	RW	0x0
8:6	FSEL2	FSEL2 - Function Select 2	RW	0x0
5:3	FSEL1	FSEL1 - Function Select 1	RW	0x0
2:0	FSEL0	FSEL0 - Function Select 0	RW	0x0

GPFSEL1 Register

Table 65. GPIO Alternate function select register 1

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
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	RW	0x0
26:24	FSEL18	FSEL18 - Function Select 18	RW	0x0
23:21	FSEL17	FSEL17 - Function Select 17	RW	0x0
20:18	FSEL16	FSEL16 - Function Select 16	RW	0x0
17:15	FSEL15	FSEL15 - Function Select 15	RW	0x0
14:12	FSEL14	FSEL14 - Function Select 14	RW	0x0
11:9	FSEL13	FSEL13 - Function Select 13	RW	0x0

Bits	Name	Description	Type	Reset
8:6	FSEL12	FSEL12 - Function Select 12	RW	0x0
5:3	FSEL11	FSEL11 - Function Select 11	RW	0x0
2:0	FSEL10	FSEL10 - Function Select 10	RW	0x0

GPFSEL2 Register

Table 66. GPIO Alternate function select register 2

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
29:27	FSEL29	FSEL29 - Function Select 29 000 = 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	RW	0x0
26:24	FSEL28	FSEL28 - Function Select 28	RW	0x0
23:21	FSEL27	FSEL27 - Function Select 27	RW	0x0
20:18	FSEL26	FSEL26 - Function Select 26	RW	0x0
17:15	FSEL25	FSEL25 - Function Select 25	RW	0x0
14:12	FSEL24	FSEL24 - Function Select 24	RW	0x0
11:9	FSEL23	FSEL23 - Function Select 23	RW	0x0
8:6	FSEL22	FSEL22 - Function Select 22	RW	0x0
5:3	FSEL21	FSEL21 - Function Select 21	RW	0x0
2:0	FSEL20	FSEL20 - Function Select 20	RW	0x0

GPFSEL3 Register

Table 67. GPIO Alternate function select register 3

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
29:27	FSEL39	FSEL39 - Function Select 39 000 = 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	RW	0x0
26:24	FSEL38	FSEL38 - Function Select 38	RW	0x0
23:21	FSEL37	FSEL37 - Function Select 37	RW	0x0
20:18	FSEL36	FSEL36 - Function Select 36	RW	0x0
17:15	FSEL35	FSEL35 - Function Select 35	RW	0x0

Bits	Name	Description	Type	Reset
14:12	FSEL34	FSEL34 - Function Select 34	RW	0x0
11:9	FSEL33	FSEL33 - Function Select 33	RW	0x0
8:6	FSEL32	FSEL32 - Function Select 32	RW	0x0
5:3	FSEL31	FSEL31 - Function Select 31	RW	0x0
2:0	FSEL30	FSEL30 - Function Select 30	RW	0x0

GPFSEL4 Register

Table 68. GPIO Alternate function select register 4

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
29:27	FSEL49	FSEL49 - Function Select 49 000 = 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	RW	0x0
26:24	FSEL48	FSEL48 - Function Select 48	RW	0x0
23:21	FSEL47	FSEL47 - Function Select 47	RW	0x0
20:18	FSEL46	FSEL46 - Function Select 46	RW	0x0
17:15	FSEL45	FSEL45 - Function Select 45	RW	0x0
14:12	FSEL44	FSEL44 - Function Select 44	RW	0x0
11:9	FSEL43	FSEL43 - Function Select 43	RW	0x0
8:6	FSEL42	FSEL42 - Function Select 42	RW	0x0
5:3	FSEL41	FSEL41 - Function Select 41	RW	0x0
2:0	FSEL40	FSEL40 - Function Select 40	RW	0x0

GPFSEL5 Register

Table 69. GPIO Alternate function select register 5

Bits	Name	Description	Type	Reset
31:24	Reserved.	-	-	-
23:21	FSEL57	FSEL57 - Function Select 57 000 = GPIO Pin 57 is an input 001 = GPIO Pin 57 is an output 100 = GPIO Pin 57 takes alternate function 0 101 = GPIO Pin 57 takes alternate function 1 110 = GPIO Pin 57 takes alternate function 2 111 = GPIO Pin 57 takes alternate function 3 011 = GPIO Pin 57 takes alternate function 4 010 = GPIO Pin 57 takes alternate function 5	RW	0x0
20:18	FSEL56	FSEL56 - Function Select 56	RW	0x0
17:15	FSEL55	FSEL55 - Function Select 55	RW	0x0

Bits	Name	Description	Type	Reset
14:12	FSEL54	FSEL54 - Function Select 54	RW	0x0
11:9	FSEL53	FSEL53 - Function Select 53	RW	0x0
8:6	FSEL52	FSEL52 - Function Select 52	RW	0x0
5:3	FSEL51	FSEL51 - Function Select 51	RW	0x0
2:0	FSEL50	FSEL50 - Function Select 50	RW	0x0

GPSET0 Register

Description

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 an 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

Table 70. GPIO Output Set Register 0

Bits	Name	Description	Type	Reset
31:0	SET*n* (*n*=0..31)	0 = No effect 1 = Set GPIO pin *n*	WO	0x00000000

GPSET1 Register

Table 71. GPIO Output Set Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	SET*n* (*n*=32..57)	0 = No effect 1 = Set GPIO pin *n*.	WO	0x0000000

GPCLR0 Register

Description

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 an 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.

Table 72. GPIO Output Clear Register 0

Bits	Name	Description	Type	Reset
31:0	CLR*n* (*n*=0..31)	0 = No effect 1 = Clear GPIO pin *n*	WO	0x00000000

GPCLR1 Register

Table 73. GPIO Output Clear Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	CLR*n* (*n*=32..57)	0 = No effect 1 = Clear GPIO pin *n*	WO	0x0000000

GPLEV0 Register

Description

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

Table 74. GPIO Level Register 0

Bits	Name	Description	Type	Reset
31:0	LEV*n* (*n*=0..31)	0 = GPIO pin *n* is low 1 = GPIO pin *n* is high	RO	0x00000000

GPLEV1 Register

Table 75. GPIO Level Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	LEV*n* (*n*=32..57)	0 = GPIO pin *n* is low 1 = GPIO pin *n* is high	RO	0x0000000

GPEDS0 Register

Description

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 four dedicated interrupt lines.

Each GPIO bank can generate an independent interrupt. The fourth line generates a single interrupt whenever any bit is set.

Table 76. GPIO Event Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	EDS*n* (*n*=0..31)	0 = Event not detected on GPIO pin *n* 1 = Event detected on GPIO pin *n*	W1C	0x00000000

GPEDS1 Register

Table 77. GPIO Event Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	EDS*n* (*n*=32..57)	0 = Event not detected on GPIO pin *n* 1 = Event detected on GPIO pin *n*	W1C	0x0000000

GPREN0 Register

Description

The rising edge detect enable registers define the pins for which a rising edge transition sets a bit in the event detect status registers (GPEDS*n*). When the relevant bits are set in both the GPREN*n* and GPFEN*n* registers, any transition (1 to 0 and 0 to 1) will set a bit in the GPEDS*n* registers. The GPREN*n* 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.

Table 78. GPIO Rising Edge Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	REN*n* (*n*=0..31)	0 = Rising edge detect disabled on GPIO pin *n* 1 = Rising edge on GPIO pin *n* sets corresponding bit in GPEDS0	RW	0x00000000

GPREN1 Register

Table 79. GPIO Rising Edge Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	REN*n* (*n*=32..57)	0 = Rising edge detect disabled on GPIO pin *n* 1 = Rising edge on GPIO pin *n* sets corresponding bit in GPEDS1	RW	0x0000000

GPFEN0 Register

Description

The falling edge detect enable registers define the pins for which a falling edge transition sets a bit in the event detect status registers (GPEDS*n*). When the relevant bits are set in both the GPREN*n* and GPFEN*n* registers, any transition (1 to 0 and 0 to 1) will set a bit in the GPEDS*n* registers. The GPFEN*n* 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.

Table 80. GPIO Falling Edge Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	FEN*n* (*n*=0..31)	0 = Falling edge detect disabled on GPIO pin *n* 1 = Falling edge on GPIO pin *n* sets corresponding bit in GPEDS0	RW	0x00000000

GPFEN1 Register

Table 81. GPIO Falling Edge Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	FEN*n* (*n*=32..57)	0 = Falling edge detect disabled on GPIO pin *n* 1 = Falling edge on GPIO pin *n* sets corresponding bit in GPEDS1	RW	0x0000000

GPHEN0 Register

Description

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

Table 82. GPIO High Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	HEN*n* (*n*=0..31)	0 = High detect disabled on GPIO pin *n* 1 = High on GPIO pin *n* sets corresponding bit in GPEDS0	RW	0x00000000

GPHEN1 Register

Table 83. GPIO High Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	HEN*n* (*n*=32..57)	0 = High detect disabled on GPIO pin *n* 1 = High on GPIO pin *n* sets corresponding bit in GPEDS1	RW	0x0000000

GPLEN0 Register

Description

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

Table 84. GPIO Low Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	LEN*n* (*n*=0..31)	0 = Low detect disabled on GPIO pin *n* 1 = Low on GPIO pin *n* sets corresponding bit in GPEDS0	RW	0x00000000

GPLEN1 Register

Table 85. GPIO Low Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	LEN*n* (*n*=32..57)	0 = Low detect disabled on GPIO pin *n* 1 = Low on GPIO pin *n* sets corresponding bit in GPEDS1	RW	0x0000000

GPAREN0 Register

Description

The asynchronous rising edge detect enable registers define the pins for which an asynchronous rising edge transition sets a bit in the event detect status registers (GPEDS*n*).

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

Table 86. GPIO Asynchronous rising Edge Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	AREN*n* (*n*=0..31)	0 = Asynchronous rising edge detect disabled on GPIO pin *n* 1 = Asynchronous rising edge on GPIO pin *n* sets corresponding bit in GPEDS0	RW	0x00000000

GPAREN1 Register

Table 87. GPIO Asynchronous rising Edge Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	AREN*n* (*n*=32..57)	0 = Asynchronous rising edge detect disabled on GPIO pin *n* 1 = Asynchronous rising edge on GPIO pin *n* sets corresponding bit in GPEDS1	RW	0x0000000

GPAFEN0 Register

Description

The asynchronous falling edge detect enable registers define the pins for which an asynchronous falling edge transition sets a bit in the event detect status registers (GPEDS*n*). Asynchronous means the incoming signal is not

sampled by the system clock. As such falling edges of very short duration can be detected.

Table 88. GPIO Asynchronous Falling Edge Detect Status Register 0

Bits	Name	Description	Type	Reset
31:0	AFEN*n* (*n*=0..31)	0 = Asynchronous falling edge detect disabled on GPIO pin *n* 1 = Asynchronous falling edge on GPIO pin *n* sets corresponding bit in GPEDS0	RW	0x00000000

GPAFEN1 Register

Table 89. GPIO Asynchronous Falling Edge Detect Status Register 1

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	AFEN*n* (*n*=32..57)	0 = Asynchronous falling edge detect disabled on GPIO pin *n* 1 = Asynchronous falling edge on GPIO pin *n* sets corresponding bit in GPEDS1	RW	0x0000000

GPIO_PUP_PDN_CNTRL_REG0 Register

Description

The GPIO Pull-up / Pull-down Registers control the actuation of the internal pull-up/down resistors. Reading these registers gives the current pull-state.
The Alternate function table also has the pull state which is applied after a power down.

Table 90. GPIO Pull-up / Pull-down Register 0

Bits	Name	Description	Type	Reset
31:30	GPIO_PUP_PDN_CNTRL15	Resistor Select for GPIO15 00 = No resistor is selected 01 = Pull up resistor is selected 10 = Pull down resistor is selected 11 = Reserved	RW	0x2
29:28	GPIO_PUP_PDN_CNTRL14	Resistor Select for GPIO14	RW	0x2
27:26	GPIO_PUP_PDN_CNTRL13	Resistor Select for GPIO13	RW	0x2
25:24	GPIO_PUP_PDN_CNTRL12	Resistor Select for GPIO12	RW	0x2
23:22	GPIO_PUP_PDN_CNTRL11	Resistor Select for GPIO11	RW	0x2
21:20	GPIO_PUP_PDN_CNTRL10	Resistor Select for GPIO10	RW	0x2
19:18	GPIO_PUP_PDN_CNTRL09	Resistor Select for GPIO09	RW	0x2
17:16	GPIO_PUP_PDN_CNTRL08	Resistor Select for GPIO08	RW	0x1
15:14	GPIO_PUP_PDN_CNTRL07	Resistor Select for GPIO07	RW	0x1
13:12	GPIO_PUP_PDN_CNTRL06	Resistor Select for GPIO06	RW	0x1

Bits	Name	Description	Type	Reset
11:10	GPIO_PUP_PDN_CNTRL05	Resistor Select for GPIO05	RW	0x1
09:08	GPIO_PUP_PDN_CNTRL04	Resistor Select for GPIO04	RW	0x1
07:06	GPIO_PUP_PDN_CNTRL03	Resistor Select for GPIO03	RW	0x1
05:04	GPIO_PUP_PDN_CNTRL02	Resistor Select for GPIO02	RW	0x1
03:02	GPIO_PUP_PDN_CNTRL01	Resistor Select for GPIO01	RW	0x1
01:00	GPIO_PUP_PDN_CNTRL00	Resistor Select for GPIO00	RW	0x1

GPIO_PUP_PDN_CNTRL_REG1 Register

Table 91. GPIO Pull-up / Pull-down Register 1

Bits	Name	Description	Type	Reset
31:30	GPIO_PUP_PDN_CNTRL31	Resistor Select for GPIO31 00 = No resistor is selected 01 = Pull up resistor is selected 10 = Pull down resistor is selected 11 = Reserved	RW	0x2
29:28	GPIO_PUP_PDN_CNTRL30	Resistor Select for GPIO30	RW	0x2
27:26	GPIO_PUP_PDN_CNTRL29	Resistor Select for GPIO29	RW	0x0
25:24	GPIO_PUP_PDN_CNTRL28	Resistor Select for GPIO28	RW	0x0
23:22	GPIO_PUP_PDN_CNTRL27	Resistor Select for GPIO27	RW	0x2
21:20	GPIO_PUP_PDN_CNTRL26	Resistor Select for GPIO26	RW	0x2
19:18	GPIO_PUP_PDN_CNTRL25	Resistor Select for GPIO25	RW	0x2
17:16	GPIO_PUP_PDN_CNTRL24	Resistor Select for GPIO24	RW	0x2
15:14	GPIO_PUP_PDN_CNTRL23	Resistor Select for GPIO23	RW	0x2
13:12	GPIO_PUP_PDN_CNTRL22	Resistor Select for GPIO22	RW	0x2
11:10	GPIO_PUP_PDN_CNTRL21	Resistor Select for GPIO21	RW	0x2
09:08	GPIO_PUP_PDN_CNTRL20	Resistor Select for GPIO20	RW	0x2

Bits	Name	Description	Type	Reset
07:06	GPIO_PUP_PDN_CNTRL19	Resistor Select for GPIO19	RW	0x2
05:04	GPIO_PUP_PDN_CNTRL18	Resistor Select for GPIO18	RW	0x2
03:02	GPIO_PUP_PDN_CNTRL17	Resistor Select for GPIO17	RW	0x2
01:00	GPIO_PUP_PDN_CNTRL16	Resistor Select for GPIO16	RW	0x2

GPIO_PUP_PDN_CNTRL_REG2 Register

Table 92. GPIO Pull-up / Pull-down Register 2

Bits	Name	Description	Type	Reset
31:30	GPIO_PUP_PDN_CNTRL47	Resistor Select for GPIO47 00 = No resistor is selected 01 = Pull up resistor is selected 10 = Pull down resistor is selected 11 = Reserved	RW	0x1
29:28	GPIO_PUP_PDN_CNTRL46	Resistor Select for GPIO46	RW	0x1
27:26	GPIO_PUP_PDN_CNTRL45	Resistor Select for GPIO45	RW	0x0
25:24	GPIO_PUP_PDN_CNTRL44	Resistor Select for GPIO44	RW	0x0
23:22	GPIO_PUP_PDN_CNTRL43	Resistor Select for GPIO43	RW	0x2
21:20	GPIO_PUP_PDN_CNTRL42	Resistor Select for GPIO42	RW	0x2
19:18	GPIO_PUP_PDN_CNTRL41	Resistor Select for GPIO41	RW	0x2
17:16	GPIO_PUP_PDN_CNTRL40	Resistor Select for GPIO40	RW	0x2
15:14	GPIO_PUP_PDN_CNTRL39	Resistor Select for GPIO39	RW	0x2
13:12	GPIO_PUP_PDN_CNTRL38	Resistor Select for GPIO38	RW	0x2
11:10	GPIO_PUP_PDN_CNTRL37	Resistor Select for GPIO37	RW	0x2
09:08	GPIO_PUP_PDN_CNTRL36	Resistor Select for GPIO36	RW	0x1
07:06	GPIO_PUP_PDN_CNTRL35	Resistor Select for GPIO35	RW	0x1
05:04	GPIO_PUP_PDN_CNTRL34	Resistor Select for GPIO34	RW	0x1

Bits	Name	Description	Type	Reset
03:02	GPIO_PUP_PDN_CNTRL33	Resistor Select for GPIO33	RW	0x2
01:00	GPIO_PUP_PDN_CNTRL32	Resistor Select for GPIO32	RW	0x2

GPIO_PUP_PDN_CNTRL_REG3 Register

Table 93. GPIO Pull-up / Pull-down Register 3

Bits	Name	Description	Type	Reset
31:20	Reserved.	-	-	-
19:18	GPIO_PUP_PDN_CNTRL57	Resistor Select for GPIO57 00 = No resistor is selected 01 = Pull up resistor is selected 10 = Pull down resistor is selected 11 = Reserved	RW	0x1
17:16	GPIO_PUP_PDN_CNTRL56	Resistor Select for GPIO56	RW	0x1
15:14	GPIO_PUP_PDN_CNTRL55	Resistor Select for GPIO55	RW	0x1
13:12	GPIO_PUP_PDN_CNTRL54	Resistor Select for GPIO54	RW	0x1
11:10	GPIO_PUP_PDN_CNTRL53	Resistor Select for GPIO53	RW	0x1
09:08	GPIO_PUP_PDN_CNTRL52	Resistor Select for GPIO52	RW	0x1
07:06	GPIO_PUP_PDN_CNTRL51	Resistor Select for GPIO51	RW	0x1
05:04	GPIO_PUP_PDN_CNTRL50	Resistor Select for GPIO50	RW	0x1
03:02	GPIO_PUP_PDN_CNTRL49	Resistor Select for GPIO49	RW	0x1
01:00	GPIO_PUP_PDN_CNTRL48	Resistor Select for GPIO48	RW	0x1

5.3. Alternative Function Assignments

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

Table 94. GPIO Pins Alternative Function Assignment

GPIO	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
GPIO1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
GPIO2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
GPIO3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
GPIO4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3

GPIO	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
GPIO6	High	GPCLK2	SOE_N / SE	DPI_D2	SPI4_MOSI	CTS3	SDA4
GPIO7	High	SPI0_CE1_N	SWE_N / SRW_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
GPIO8	High	SPI0_CE0_N	SD0	DPI_D4	BSCSL / CE_N	TXD4	SDA4
GPIO9	Low	SPI0_MISO	SD1	DPI_D5	BSCSL / MISO	RXD4	SCL4
GPIO10	Low	SPI0_MOSI	SD2	DPI_D6	BSCSL SDA / MOSI	CTS4	SDA5
GPIO11	Low	SPI0_SCLK	SD3	DPI_D7	BSCSL SCL / SCLK	RTS4	SCL5
GPIO12	Low	PWM0_0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
GPIO13	Low	PWM0_1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
GPIO14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
GPIO15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
GPIO16	Low	<reserved>	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
GPIO17	Low	<reserved>	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
GPIO18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0_0
GPIO19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM0_1
GPIO20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
GPIO21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
GPIO22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
GPIO23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
GPIO24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
GPIO25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
GPIO26	Low	SD0_DAT2	<reserved>	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
GPIO27	Low	SD0_DAT3	<reserved>	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N
GPIO28	-	SDA0	SA5	PCM_CLK	<reserved>	MII_A_RX_ERR	RGMII_MDIO
GPIO29	-	SCL0	SA4	PCM_FS	<reserved>	MII_A_TX_ERR	RGMII_MDC
GPIO30	Low	<reserved>	SA3	PCM_DIN	CTS0	MII_A_CRS	CTS1
GPIO31	Low	<reserved>	SA2	PCM_DOUT	RTS0	MII_A_COL	RTS1
GPIO32	Low	GPCLK0	SA1	<reserved>	TXD0	SD_CARD_PRES	TXD1
GPIO33	Low	<reserved>	SA0	<reserved>	RXD0	SD_CARD_WRPROT	RXD1
GPIO34	High	GPCLK0	SOE_N / SE	<reserved>	SD1_CLK	SD_CARD_LED	RGMII_IRQ
GPIO35	High	SPI0_CE1_N	SWE_N / SRW_N		SD1_CMD	RGMII_START_STOP	
GPIO36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	RGMII_RX_OK	MII_A_RX_ERR

GPIO	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	RGMII_MDIO	MII_A_TX_ERR
GPIO38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	RGMII_MDC	MII_A_CRS
GPIO39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	RGMII_IRQ	MII_A_COL
GPIO40	Low	PWM1_0	SD4		SD1_DAT4	SPI0_MISO	TXD1
GPIO41	Low	PWM1_1	SD5	<reserved>	SD1_DAT5	SPI0_MOSI	RXD1
GPIO42	Low	GPCLK1	SD6	<reserved>	SD1_DAT6	SPI0_SCLK	RTS1
GPIO43	Low	GPCLK2	SD7	<reserved>	SD1_DAT7	SPI0_CE0_N	CTS1
GPIO44	-	GPCLK1	SDA0	SDA1	<reserved>	SPI0_CE1_N	SD_CARD_VOLT
GPIO45	-	PWM0_1	SCL0	SCL1	<reserved>	SPI0_CE2_N	SD_CARD_PWR0
GPIO46	High	<Internal>					
GPIO47	High	<Internal>					
GPIO48	High	<Internal>					
GPIO49	High	<Internal>					
GPIO50	High	<Internal>					
GPIO51	High	<Internal>					
GPIO52	High	<Internal>					
GPIO53	High	<Internal>					
GPIO54	High	<Internal>					
GPIO55	High	<Internal>					
GPIO56	High	<Internal>					
GPIO57	High	<Internal>					

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:

Table 95. GPIO Pins Alternative Function Legend

Name	Function	See section
SDA0	BSC[a] master 0 data line	BSC
SCL0	BSC master 0 clock line	BSC
SDAx	BSC master 1,3,4,5,6[b] data line	BSC
SCLx	BSC master 1,3,4,5,6 clock line	BSC
GPCLKx	General purpose Clock 0,1,2	General Purpose GPIO Clocks
SPIx_CE2_N	SPI 0,3,4,5,6 Chip select 2	SPI
SPIx_CE1_N	SPI 0,3,4,5,6 Chip select 1	SPI
SPIx_CE0_N	SPI 0,3,4,5,6 Chip select 0	SPI
SPIx_MISO	SPI 0,3,4,5,6 MISO	SPI

Name	Function	See section
SPIx_MOSI	SPI 0,3,4,5,6 MOSI	SPI
SPIx_SCLK	SPI 0,3,4,5,6 Serial clock	SPI
PWMx_0	PWM 0,1 channel 0	Pulse Width Modulator
PWMx_1	PWM 0,1 channel 1	Pulse Width Modulator
TXDx	UART 0,2,3,4,5 Transmit Data	UART
RXDx	UART 0,2,3,4,5 Receive Data	UART
CTSx	UART 0,2,3,4,5 Clear To Send	UART
RTSx	UART 0,2,3,4,5 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/SPI slave
BSCSL_SCL / SCLK	BSC slave Clock, SPI slave clock	BSC/SPI slave
BSCSL - / MISO	BSC <not used>, SPI MISO	BSC/SPI slave
BSCSL - / CE_N	BSC <not used>, SPI CSn	BSC/SPI slave
SPI1_CE2_N	SPI 1[c] Chip select 2	Auxiliary I/O
SPI1_CE1_N	SPI 1 Chip select 1	Auxiliary I/O
SPI1_CE0_N	SPI 1 Chip select 0	Auxiliary I/O
SPI1_MISO	SPI 1 MISO	Auxiliary I/O
SPI1_MOSI	SPI 1 MOSI	Auxiliary I/O
SPI1_SCLK	SPI 1 Serial clock	Auxiliary I/O
TXD1	UART 1 Transmit Data	Auxiliary I/O
RXD1	UART 1 Receive Data	Auxiliary I/O
CTS1	UART 1 Clear To Send	Auxiliary I/O
RTS1	UART 1 Request To Send	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>

Name	Function	See section
PCLK	Display Parallel Interface	<TBD>
DE	Display Parallel Interface	<TBD>
LCD_VSYNC	Display Parallel Interface	<TBD>
LCD_HSYNC	Display Parallel Interface	<TBD>
DPI_Dx	Display Parallel Interface	<TBD>

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

[b] BSC master 2 & 7 are not user-accessible

[c] SPI 2 is not user-accessible

5.4. 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:

$$\frac{source\_frequency}{DIVI}$$

and

$$\frac{source\_frequency}{DIVI+1}$$

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. When using the 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.

Table 96. Effect of MASH Filter on Frequency

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 + 1)	source / (DIVI + DIVF / 1024)	source / (DIVI)
2	3	source / (DIVI + 2)	source / (DIVI + DIVF / 1024)	source / (DIVI - 1)
3	5	source / (DIVI + 4)	source / (DIVI + DIVF / 1024)	source / (DIVI - 3)

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.

Table 97. Example of Frequency Spread when using MASH Filtering

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

PLL freq (MHz)	target freq (MHz)	MASH	divisor	DIVI	DIVF	min freq (MHz)	ave freq (MHz)	max freq (MHz)	error
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

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.

5.4.1. 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.

5.4.2. Register Definitions

The General Purpose clocks register base address is `0x7e101000`.

Table 98. General Purpose Clocks Registers

Offset	Name	Description
0x70	CM_GP0CTL	Clock Manager General Purpose Clocks Control
0x74	CM_GP0DIV	Clock Manager General Purpose Clock Divisors
0x78	CM_GP1CTL	Clock Manager General Purpose Clocks Control
0x7c	CM_GP1DIV	Clock Manager General Purpose Clock Divisors
0x80	CM_GP2CTL	Clock Manager General Purpose Clocks Control
0x84	CM_GP2DIV	Clock Manager General Purpose Clock Divisors

CM_GP0CTL, CM_GP1CTL, CM_GP2CTL Registers

Table 99. General Purpose Clocks Control

Bits	Name	Description	Type	Reset
31:24	PASSWD	Clock Manager password "5a"	WO	0x00
23:11	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
10:9	MASH	MASH control 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.	RW	0x0
8	FLIP	Invert the clock generator output 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.	RW	0x0
7	BUSY	Clock generator is running 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.	RO	0x0
6	Reserved.	-	-	-
5	KILL	Kill the clock generator 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.	RW	0x0
4	ENAB	Enable the clock generator 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.	RW	0x0
3:0	SRC	Clock source 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.	RW	0x0

CM_GP0DIV, CM_GP1DIV, CM_GP2DIV Registers

Table 100. General Purpose Clock Divisors

Bits	Name	Description	Type	Reset
31:24	PASSWD	Clock Manager password "5a"	WO	0x00
23:12	DIVI	Integer part of divisor 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.	RW	0x000

Bits	Name	Description	Type	Reset
11:0	DIVF	Fractional part of divisor To avoid lock-ups and glitches do not change this control while BUSY=1.	RW	0x000

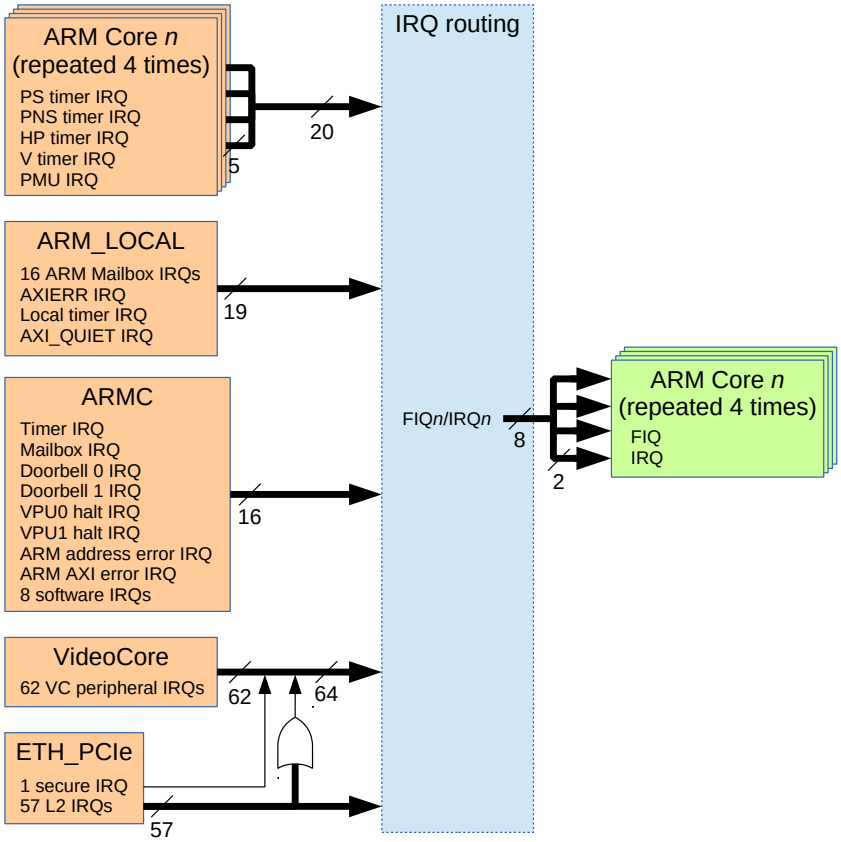
Chapter 6. Interrupts

6.1. Overview

The BCM2711 has a large number of interrupts from various sources, and a choice of two interrupt controllers. The GIC-400 interrupt controller is selected by default, but the legacy interrupt controller can be selected with a setting in `config.txt` - refer to raspberrypi.com documentation for further details.

In Figure 5 the orange boxes illustrate the various interrupt source blocks, the blue box covers the interrupt controller routing (explained later), and the green box shows the final interrupt destinations. The number underneath each slash through the thick arrows indicates how many signals that arrow contains (thin arrows without a number only contain one signal). The "ARM Core *n*" blocks in orange are actually the same as the "ARM Core *n*" blocks in green, they're just drawn as separate source and destination blocks for clarity. ARM_LOCAL and ARMC are different hardware blocks within the chip, each with their own set of registers; ARMC is visible to both the VPU and CPU, but ARM_LOCAL is only visible to the CPU (and corresponds to the "ARM Local peripherals" in Chapter 1).

Figure 5. Interrupt sources and destinations



The final output from each interrupt controller is 8 separate signals - a FIQ (Fast Interrupt reQuest) and an IRQ (Interrupt ReQuest) for each of the 4 ARM cores, i.e. FIQ0 and IRQ0 connected to ARM core 0, FIQ1 and IRQ1 connected to ARM core 1, FIQ2 and IRQ2 connected to ARM core 2, and FIQ3 and IRQ3 connected to ARM core 3. For convenience, this document will refer to those 8 signals as FIQ*n*/IRQ*n*.

To avoid confusion, note that the "ARM Mailbox IRQs" in the ARM_LOCAL block are different from the "Mailbox IRQ" in the ARMC block. Similarly, the "Local timer IRQ" in the ARM_LOCAL block is different to the "Timer IRQ" in the ARMC block, which are both different from the 4 timer IRQs in the "ARM Core *n*" block. The "AXIERR IRQ" in the ARM_LOCAL block is also different from the "ARM AXI error IRQ" in the ARMC block.

6.2. Interrupt sources

6.2.1. ARM Core *n* interrupts

Each of the ARM Cores can raise a Secure Physical (PS) timer interrupt, a Non-Secure Physical (PNS) timer interrupt, a Hypervisor (HP) timer interrupt, a Virtual (V) timer interrupt and a Performance Monitoring Unit (PMU) interrupt. For more information, please refer to the ARM Cortex-A72 documentation on the ARM Developer website.

6.2.2. ARM_LOCAL interrupts

Further information about the ARM Mailboxes can be found in Chapter 13. The AXIERR output is asserted by the ARM's L2 cache if an error response is received. Further information about the Local Timer and AXI_QUIET can be found in the Registers section of this chapter.

6.2.3. ARMC interrupts

Table 101. ARMC peripheral IRQs

#	IRQ
0	Timer
1	Mailbox
2	Doorbell 0
3	Doorbell 1
4	VPU0 halted
5	VPU1 halted
6	ARM address error
7	ARM AXI error
8	Software Interrupt 0
9	Software Interrupt 1
10	Software Interrupt 2
11	Software Interrupt 3
12	Software Interrupt 4
13	Software Interrupt 5
14	Software Interrupt 6
15	Software Interrupt 7

The Timer interrupt in Table 101 comes from the "Timer (ARM side)" described in Chapter 12.

The eight general-purpose software interrupts can be set by writing to the SWIRQ_SET register and cleared by writing to the SWIRQ_CLEAR register.

6.2.4. VideoCore interrupts

Table 102. VC peripheral IRQs

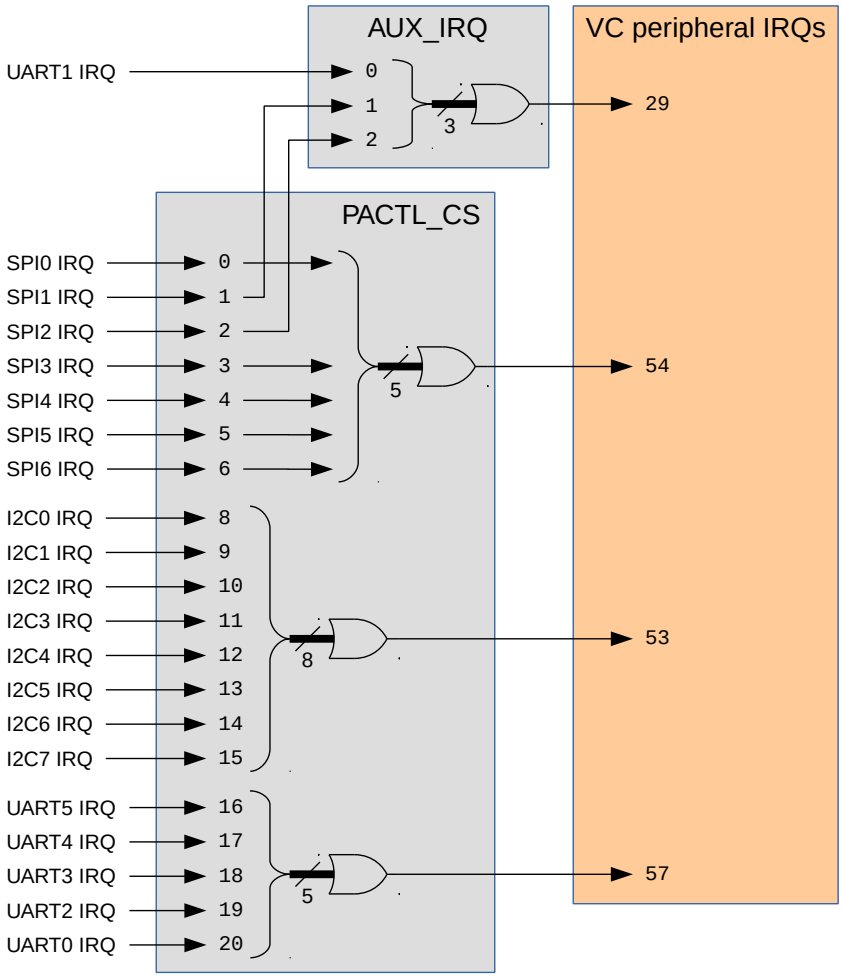
#	IRQ 0-15	#	IRQ 16-31	#	IRQ 32-47	#	IRQ 48-63
0	Timer 0	16	DMA 0	32	HDMI CEC	48	SMI
1	Timer 1	17	DMA 1	33	HVS	49	GPIO 0

#	IRQ 0-15	#	IRQ 16-31	#	IRQ 32-47	#	IRQ 48-63
2	Timer 2	18	DMA 2	34	RPIVID	50	GPIO 1
3	Timer 3	19	DMA 3	35	SDC	51	GPIO 2
4	H264 0	20	DMA 4	36	DSI 0	52	GPIO 3
5	H264 1	21	DMA 5	37	Pixel Valve 2	53	**OR of all I2C**
6	H264 2	22	DMA 6	38	Camera 0	54	**OR of all SPI**
7	JPEG	23	**DMA 7 & 8**	39	Camera 1	55	PCM/I2S
8	ISP	24	**DMA 9 & 10**	40	HDMI 0	56	SDHOST
9	USB	25	DMA 11	41	HDMI 1	57	**OR of all PL011 UART**
10	V3D	26	DMA 12	42	Pixel Valve 3	58	**OR of all ETH_PCIe L2**
11	Transposer	27	DMA 13	43	SPI/BSC Slave	59	VEC
12	Multicore Sync 0	28	DMA 14	44	DSI 1	60	CPG
13	MultiCore Sync 1	29	**AUX**	45	Pixel Valve 0	61	RNG
14	MultiCore Sync 2	30	ARM	46	**Pixel Valve 1 & 4**	62	**EMMC & EMMC2**
15	MultiCore Sync 3	31	DMA 15	47	CPR	63	ETH_PCIe secure

The 4 timer interrupts in Table 102 come from the "System Timer" described in Chapter 10.

Because there are more peripherals than available VC peripheral IRQs, some of the VC peripheral interrupts (highlighted in **bold** in Table 102) are the OR-ed version of multiple peripheral interrupts.
The per-peripheral interrupt statuses for VC peripheral IRQs 29, 53, 54 & 57 can in turn be read from the AUX_IRQ (documented in Chapter 2) and PACTL_CS (at address `0x7E20 4E00`) registers. Figure 6 shows how this is logically connected, with the vertically-aligned numbers inside the grey boxes indicating bit-positions within the registers.

Figure 6. Peripheral IRQ OR-ing



For example if VC peripheral IRQ 53 is triggered, then you know at least one of the I2C peripherals has caused an interrupt. To find out exactly *which* I2C peripherals have interrupts pending, you can read bits 8 to 15 inclusive of PACTL_CS (alternatively, you could simply read the Status register for each of the I2C peripherals).

There are also some VC peripheral interrupts (23, 24, 46 and 62) that are an OR-ed version of two peripheral interrupt signals - if these interrupts are received the only option is to read the status register for each of the peripherals concerned.

6.2.5. ETH_PCIe interrupts

Table 103. ETH_PCIe L2 IRQs

#	IRQ
9	AVS
15	PCIE_0_INTA
16	PCIE_0_INTB
17	PCIE_0_INTC
18	PCIE_0_INTD
20	PCIE_0_MSI
29	GENET_0_A
30	GENET_0_B
48	USB0_XHCI_0

Any IRQ numbers not listed in the table above are reserved.

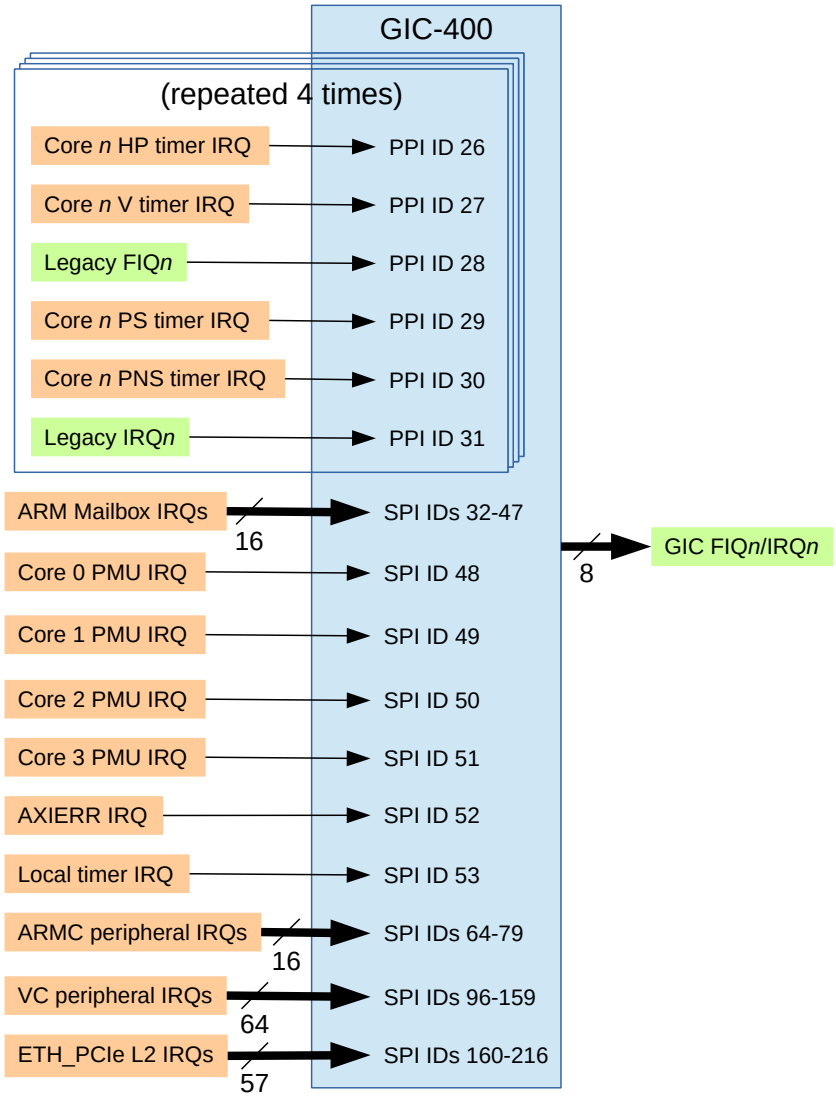
The secure IRQ output (which is only useful for the VPU and not the CPU) from the ETH_PCIe block is routed to VC peripheral IRQ 63, and all 57 ETH_PCIe L2 IRQs are OR-ed together and routed to VC peripheral IRQ 58 - see Figure 5 and Table 102.
Note that the 57 individual ETH_PCIe interrupts aren't routed to the legacy interrupt controller, only VC peripheral IRQ 58 (the OR-ed version) is available.

6.3. GIC-400 interrupt controller

The BCM2711 contains an ARM GIC-400 interrupt controller, which is enabled by default. For more information, please refer to the ARM GIC-400 documentation on the ARM Developer website.

Figure 7 shows how the interrupt sources described earlier are connected to the GIC. When the GIC-400 is selected as the interrupt controller, the eight "GIC FIQ*n*/IRQ*n*" outputs are routed to the FIQ*n*/IRQ*n* inputs of the ARM cores.
Note that even when the GIC-400 is selected as the interrupt controller, the outputs of the legacy interrupt controller (described later) are available as PPIs within the GIC.

Figure 7. GIC IRQ routing

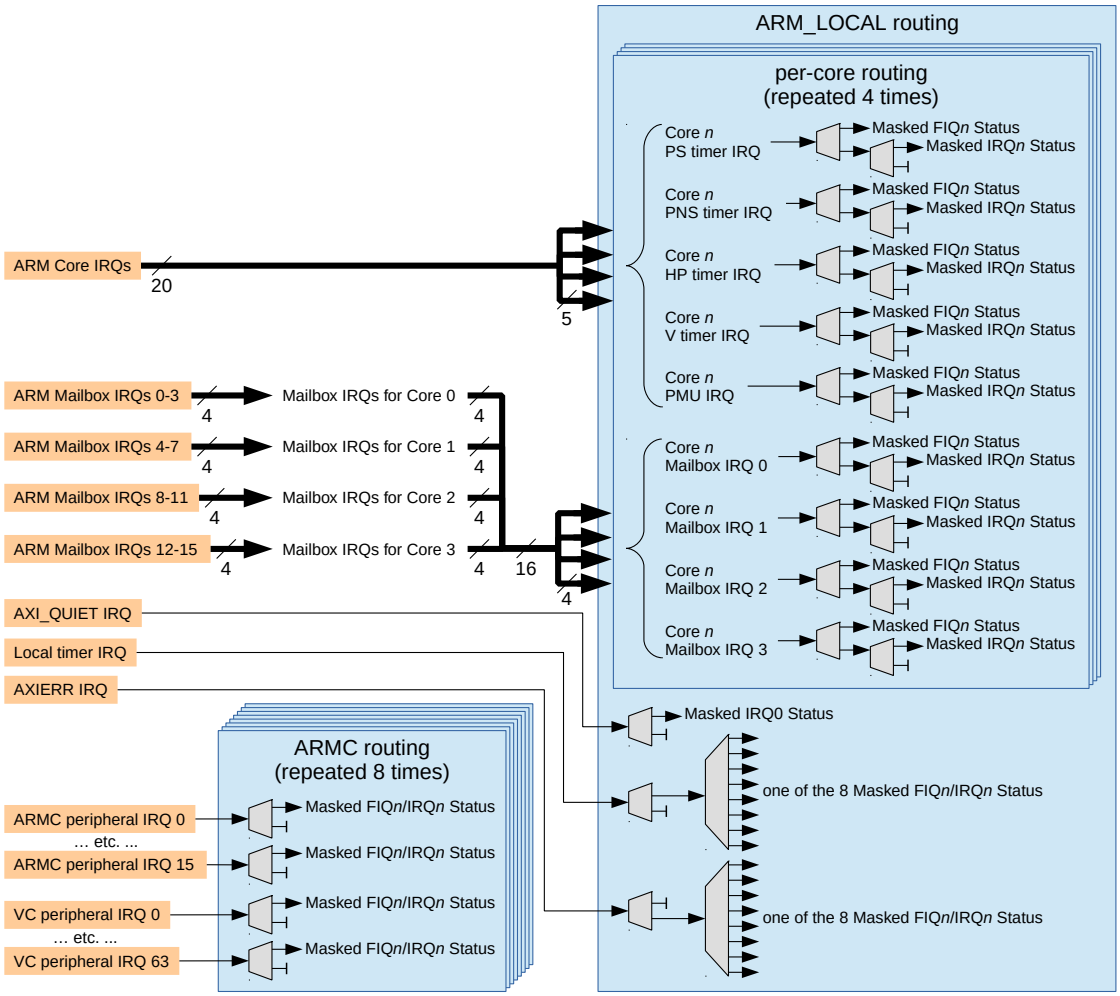


The GIC-400 also connects to the VFIQ (Virtual FIQ) and VIRQ (Virtual IRQ) input of each ARM core, but for brevity these signals are not shown here.

6.4. Legacy interrupt controller

The legacy interrupt controller in the BCM2711 has some similarities with the interrupt controllers used in earlier BCM283x chips, but also several differences. When the legacy interrupt controller is selected, the eight "Legacy FIQ*n*/IRQ *n*" outputs (shown in Figure 9) are routed to the FIQ*n*/IRQ*n* inputs of the ARM cores.

Figure 8. Legacy IRQ routing



The interrupts coming directly from each of the ARM cores (PS timer, PNS timer, HP timer, V timer and PMU) can only be routed to either the FIQ or IRQ of the core from which they originate. For example the PS timer and PMU IRQs from core 3 could be routed to FIQ3 and the PNS timer IRQ from core 2 could be routed to IRQ2. The masking of the ARM timer IRQs is controlled by the 4 TIMER_CNTRL registers (one for each core) and the masking of the PMU IRQs is controlled by the PMU_CONTROL_SET and PMU_CONTROL_CLR registers.

The sixteen ARM Mailbox interrupts are allocated so that four go to each core - ARM Mailbox IRQs 0 to 3 are routed to the four Mailbox IRQs on ARM core 0, and ARM Mailbox IRQs 12 to 15 are routed to the four Mailbox IRQs on ARM core 3, i.e. ARM Mailbox IRQ 13 appears to ARM Core 3 as Mailbox IRQ 1. Like the ARM Core interrupts, the ARM Mailbox IRQs can only be routed to the FIQ or IRQ of the core for which they are intended, for example the Mailbox 4 and 5 IRQs could be routed to FIQ1 and the Mailbox 10 IRQ could be routed to IRQ2. The masking of the ARM Mailbox IRQs is controlled by the four MAILBOX_CNTRL registers (one for each core).

The AXI_QUIET IRQ is only available to the IRQ input on ARM core 0, and its masking is controlled by the AXI_QUIET_TIME register.

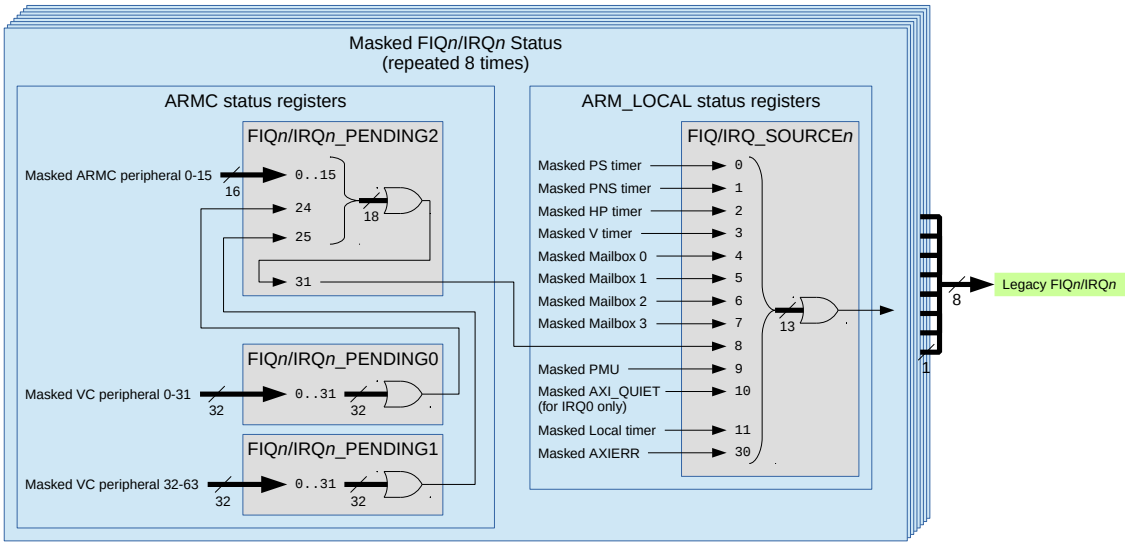
The Local timer and AXIERR IRQs can be routed to any one of the 8 FIQ*n*/IRQ*n* signals.
The masking of the Local timer IRQ is controlled by the LOCAL_TIMER_CONTROL and PERI_IRQ_ROUTE0 registers.
The masking of the AXIERR IRQ is controlled by the ARM_CONTROL and CORE_IRQ_CONTROL registers.

The unmasked inputs to the "ARMC routing" block are readable from the IRQ_STATUS0, IRQ_STATUS1 and IRQ_STATUS2

registers.
The masking within the "ARMC routing" block is controlled by the SET_EN_0, SET_EN_1, SET_EN_2, CLR_EN_0, CLR_EN_1 and CLR_EN_2 registers. Each of these registers is repeated for each of the eight FIQ*n*/IRQ*n* signals (48 registers in total).

Once the interrupts have been masked and routed, their statuses can be read from the 3 PENDING and 1 SOURCE registers (repeated 8 times for each of the FIQ*n*/IRQ*n* signals, for a total of 32 registers).

Figure 9. Legacy IRQ status registers



These are "nested" status registers, which means if bit 8 in the SOURCE register is set, you also need to read PENDING2 to see which bits are set there. If bit 24 in the PENDING2 register is set, then you also need to read PENDING0 to see which bits there are set.

As a more complete example, if the interrupt routing and masking is set up so that an interrupt from UART4 triggers a FIQ interrupt to ARM Core 3, the sequence (on ARM Core 3) would be:

1. Enter FIQ handler
2. Read FIQ_SOURCE3
3. Find that FIQ_SOURCE3[8] is set, so read FIQ3_PENDING2
4. Find that FIQ3_PENDING2[25] is set, so read FIQ3_PENDING1
5. Find that FIQ3_PENDING1[25] (i.e. VC peripheral IRQ 57) is set, so read PACTL_CS[20:16] (see Figure 6) to see which UART triggered it
6. Find that PACTL_CS[17] is set, so read UART4_MIS to (finally) determine what caused the interrupt

6.5. Registers

To allow atomic operations (where only particular bits are modified, without modifying any of the other bits in the register), some registers are split into a write-set register and a write-clear register.

A **write-set register** allows you to set particular bits **high** (change them to 1). You set a bit high by writing a '1' to its bit-position - bits that were low get changed to high, and bits that were already high remain high. Any bit-positions written with a '0' retain their previous value.

Old bit value	Write bit value	Result bit value
0	0	0
0	1	1
1	0	1
1	1	1

Thus writing 0xFC060014 to a write-set register containing 0x30840008 changes it to 0xFC86001C.

A **write-clear register** allows you to set particular bits **low** (change them to 0). You set a bit low by writing a '1' to its bit-position - bits that were low remain low, and bits that were high get changed to low. Any bit-positions written with a '0' retain their previous value. Note that you write a **one** to change a bit to **zero**!

Old bit value	Write bit value	Result bit value
0	0	0
0	1	0
1	0	1
1	1	0

Thus writing 0xFC060014 to a write-clear register containing 0x30840008 changes it to 0x00800008.

6.5.1. GIC-400

The base address of the GIC-400 is `0x4c0040000`. Note that, unlike other peripheral addresses in this document, this is an ARM-only address and not a legacy master address. If Low Peripheral mode is enabled this base address becomes `0xff840000`.

The GIC-400 is configured with "NUM_CPUS=4" and "NUM_SPIS=192". For full register details, please refer to the ARM GIC-400 documentation on the ARM Developer website.

6.5.2. ARM_LOCAL

The ARM_LOCAL register base address is `0x4c0000000`. Note that, unlike other peripheral addresses in this document, this is an ARM-only address and not a legacy master address. If Low Peripheral mode is enabled this base address becomes `0xff800000`.

The PMU_CONTROL_SET / PMU_CONTROL_CLR registers are write-set / write-clear registers as described earlier.

Table 104. ARM_LOCAL Interrupt Registers

Offset	Name	Description
0x00	ARM_CONTROL	ARM Timer and AXI Error IRQ control
0x0c	CORE_IRQ_CONTROL	VideoCore Interrupt Control
0x10	PMU_CONTROL_SET	PMU Bit Set
0x14	PMU_CONTROL_CLR	PMU Bit Clear
0x24	PERI_IRQ_ROUTE0	Peripheral Interrupt Routing (Bank 0)
0x30	AXI_QUIET_TIME	AXI Outstanding Transaction Time and IRQ Control
0x34	LOCAL_TIMER_CONTROL	Local Timer Control
0x38	LOCAL_TIMER_IRQ	Local Timer Reload and Interrupt
0x40	TIMER_CNTRL0	Timer Interrupt Control for ARM Core 0
0x44	TIMER_CNTRL1	Timer Interrupt Control for ARM Core 1
0x48	TIMER_CNTRL2	Timer Interrupt Control for ARM Core 2
0x4c	TIMER_CNTRL3	Timer Interrupt Control for ARM Core 3
0x50	MAILBOX_CNTRL0	Mailbox Interrupt Control for ARM Core 0
0x54	MAILBOX_CNTRL1	Mailbox Interrupt Control for ARM Core 1
0x58	MAILBOX_CNTRL2	Mailbox Interrupt Control for ARM Core 2

Offset	Name	Description
0x5c	MAILBOX_CNTRL3	Mailbox Interrupt Control for ARM Core 3
0x60	IRQ_SOURCE0	IRQ Source flags for ARM Core 0
0x64	IRQ_SOURCE1	IRQ Source flags for ARM Core 1
0x68	IRQ_SOURCE2	IRQ Source flags for ARM Core 2
0x6c	IRQ_SOURCE3	IRQ Source flags for ARM Core 3
0x70	FIQ_SOURCE0	FIQ Source flags for ARM Core 0
0x74	FIQ_SOURCE1	FIQ Source flags for ARM Core 1
0x78	FIQ_SOURCE2	FIQ Source flags for ARM Core 2
0x7c	FIQ_SOURCE3	FIQ Source flags for ARM Core 3

ARM_CONTROL Register

Description

Main Timer and AXI Error Control.

Table 105. ARM_CONTROL Register

Bits	Name	Description	Type	Reset
31:9	Reserved.	-	-	-
08	TIMER_INCREMENT	Main timer increment value selection The main timer (driving the ARM core 'global system counter') is incremented by this amount each time the prescaler output is asserted. The ability to set the increment value to two allows the main timer to count ARM core clock cycles in the case where the AXI/APB clock frequency is half of the ARM core clock frequency and the prescaler ratio is unity. 1 = increment count by two. 0 = increment count by one.	RW	0x0
07	PROC_CLK_TIMER	Main timer clock selection The main timer (driving the ARM core 'global system counter') may be driven either from the fast but variable AXI/APB bus clock or from the fixed-frequency but slower crystal reference clock. 1 = select AXI/APB clock. 0 = select crystal clock.	RW	0x0
06	AXIERRIRQ_EN	When set to '1', this bit masks the AXI Error interrupt. An AXI error output is asserted by the ARM's L2 cache if an error response is received. If not masked, this causes an interrupt to be raised. If this bit is set, the interrupt is not raised. Interrupt routing for this is controlled by the AXI_ERR_CORE field in the CORE_IRQ_CONTROL register.	RW	0x0
5:0	Reserved.	-	-	-

CORE_IRQ_CONTROL Register

Description

VideoCore Interrupt Routing Control

Table 106. CORE_IRQ_CONTROL Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
06:04	AXI_ERR_CORE	Controls to which ARM core interrupt request pin the external error interrupt request signal from the ARM L2 cache is routed. This interrupt is enabled in the AXIERRIRQ_EN field in the ARM_CONTROL register. 0 = CORE0_IRQ 1 = CORE1_IRQ 2 = CORE2_IRQ 3 = CORE3_IRQ 4 = CORE0_FIQ 5 = CORE1_FIQ 6 = CORE2_FIQ 7 = CORE3_FIQ	RW	0x0
3:0	Reserved.	-	-	-

PMU_CONTROL_SET Register

Description

Performance Monitoring Unit (PMU) control word. Each ARM core provides a PMUIRQ output; this control word specifies to which interrupt pins they are routed.

Writing a '1' to a bit position in this register causes the corresponding bit in the PMU control word to be set to 1.

Table 107. PMU_CONTROL_SET Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
07:04	PMU_FIQ	PMU to FIQ routing mask. Setting bit N within this four bit field routes the PMUIRQ output from ARM core N to the FIQ interrupt request on that ARM core.	RW	0x0
03:00	PMU_IRQ	PMU to IRQ routing mask Setting bit N within this four bit field routes the PMUIRQ output from ARM core N to the IRQ interrupt request on that ARM core. Note that it is not possible to route the PMU interrupt to both the FIQ and IRQ request pins: setting a bit in the PMU_FIQ field causes the request to go to the core's FIQ pin only, irrespective of this field.	RW	0x0

PMU_CONTROL_CLR Register

Description

Performance Monitoring Unit (PMU) control word. Each ARM core provides a PMUIRQ output; this control word specifies to which interrupt pins they are routed.

Writing a '1' to a bit position in this register causes the corresponding bit in the PMU control word to be cleared to 0.

Table 108. PMU_CONTROL_CLR Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
07:04	PMU_FIQ	PMU to FIQ routing mask. Setting bit N within this four bit field routes the PMUIRQ output from ARM core N to the FIQ interrupt request on that ARM core.	W1C	0x0
03:00	PMU_IRQ	PMU to IRQ routing mask Setting bit N within this four bit field routes the PMUIRQ output from ARM core N to the IRQ interrupt request on that ARM core. Note that it is not possible to route the PMU interrupt to both the FIQ and IRQ request pins: setting a bit in the PMU_FIQ field causes the request to go to the core's FIQ pin only, irrespective of this field.	W1C	0x0

PERI_IRQ_ROUTE0 Register

Description

This register controls the routing of the Local timer interrupts.

Table 109. PERI_IRQ_ROUTE0 Register

Bits	Name	Description	Type	Reset
31:24	WRITE_MASKS	Interrupt routing field write mask bits. This field must be written with 0x01, otherwise changes to LOCAL_TIMER_IRQ will be ignored.	RW	0x00
23:3	Reserved.	-	-	-
02:00	LOCAL_TIMER_IRQ	Local timer Routing 0 = CORE0_IRQ 1 = CORE1_IRQ 2 = CORE2_IRQ 3 = CORE3_IRQ 4 = CORE0_FIQ 5 = CORE1_FIQ 6 = CORE2_FIQ 7 = CORE3_FIQ	RW	0x0

AXI_QUIET_TIME Register

Description

No outstanding AXI transactions for a while.

This register controls logic that is able to generate an interrupt to the IRQ interrupt pin of ARM core 0 if there has been no AXI bus traffic for a programmable time. The intention is that software can use this to have reasonable confidence that the bus traffic from the ARM cluster to VideoCore has ceased.

A 24-bit timer is loaded with a value equal to

16 x AXI_QUIET_TIME.AXI_QUIET_TIME + 15

whenever one or more AXI transactions are outstanding. The counter decrements on each AXI/APB clock rising edge when no transactions are outstanding. When the counter reaches zero, the interrupt request is generated if enabled.

Table 110. AXI_QUIET_TIME Register

Bits	Name	Description	Type	Reset
31:21	Reserved.	-	-	-
20	AXI_QUIET_IRQ_ENB	1: Enable Core 0 IRQ on AXI quiet timer expiry 0: Disable Core 0 IRQ on AXI quiet timer expiry	RW	0x0
19:00	AXI_QUIET_TIME	Timer load value, in units of 16 AXI/APB clock cycles.	RW	0x00000

LOCAL_TIMER_CONTROL Register

Description

Local Timer Configuration.

A free-running secondary timer is provided that can generate an interrupt each time it crosses zero. When it is enabled, the timer is decremented on each edge (positive or negative) of the crystal reference clock. It is automatically reloaded with the TIMER_TIMEOUT value when it reaches zero and then continues to decrement.

Routing of the timer interrupt is controlled by the PERI_IRQ_ROUTE0 register.

Table 111. LOCAL_TIMER_CONTROL Register

Bits	Name	Description	Type	Reset
31	TIMER_IRQ_FLAG	This read-only field allows software to see the current state of the timer interrupt request. A '1' indicates a valid interrupt request.	RO	0x0
30	Reserved.	-	-	-
29	TIMER_IRQ_EN	Interrupt request enable. When set to '1', this bit causes the timer to request an interrupt as the timer crosses zero.	RW	0x0
28	TIMER_EN	Timer Enable When set to '1', this bit enables to the timer. When cleared to '0', timer operation is completely disabled: the timer does not decrement or raise interrupt requests.	RW	0x0
27:00	TIMER_TIMEOUT	Timer load value.	RW	0x0000000

LOCAL_TIMER_IRQ Register

Description

Local Timer Interrupt Control

Table 112. LOCAL_TIMER_IRQ Register

Bits	Name	Description	Type	Reset
31	IRQ_CLEAR	Write a '1' to this field to clear a timer interrupt request. If the timer crosses zero at the same time as the write, the clear operation will fail; interrupt request will remain asserted. This bit self-clears.	W1SC	0x0
30	RELOAD	Write a '1' to this field to (re)load the timer with the timeout value. This bit self-clears.	W1SC	0x0
29:0	Reserved.	-	-	-

TIMER_CNTRL0, TIMER_CNTRL1, TIMER_CNTRL2, TIMER_CNTRL3 Registers

Description

This register allows software to determine the cause of a FIQ interrupt request received by an ARM core.

Table 113. TIMER_CNTRL0, TIMER_CNTRL1, TIMER_CNTRL2, TIMER_CNTRL3 Registers

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
07	CNT_V_IRQ_FIQ	When set to '1', this bit causes the 'Virtual Timer Event' output to be routed to the FIQ interrupt request.	RW	0x0
06	CNT_HP_IRQ_FIQ	When set to '1', this bit causes the 'Hypervisor Physical Timer Event' output to be routed to the FIQ interrupt request.	RW	0x0

Bits	Name	Description	Type	Reset
05	CNT_PNS_IRQ_FIQ	When set to '1', this bit causes the 'Nonsecure Physical Timer Event' output to be routed to the FIQ interrupt request.	RW	0x0
04	CNT_PS_IRQ_FIQ	When set to '1', this bit causes the 'Secure Physical Timer Event' output to be routed to the FIQ interrupt request.	RW	0x0
03	CNT_V_IRQ	When set to '1', this bit causes the 'Virtual Timer Event' output to be routed to the IRQ interrupt request. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0
02	CNT_HP_IRQ	When set to '1', this bit causes the 'Hypervisor Physical Timer Event' output to be routed to the IRQ interrupt request. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0
01	CNT_PNS_IRQ	When set to '1', this bit causes the 'Nonsecure Physical Timer Event' output to be routed to the IRQ interrupt request. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0
00	CNT_PS_IRQ	When set to '1', this bit causes the 'Secure Physical Timer Event' output to be routed to the IRQ interrupt request. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0

MAILBOX_CNTRL0, MAILBOX_CNTRL1, MAILBOX_CNTRL2, MAILBOX_CNTRL3 Registers

Description

This register controls the routing of the mailbox interrupts to an ARM core's IRQ or FIQ interrupt request pins. Each ARM can receive interrupts from four of the sixteen mailbox registers. For ARM core 0, these are mailboxes 0-3; for ARM core 1, mailboxes 4-7 and so on.

Table 114. MAILBOX_CNTRL0, MAILBOX_CNTRL1, MAILBOX_CNTRL2, MAILBOX_CNTRL3 Registers

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
07	MBOX3_FIQ	When set to '1', this bit causes the fourth mailbox, i.e. mailbox 4C+3 for ARM core number C, (so mailbox 3 for ARM core 0, 7 for ARM core 1, etc.) to trigger a FIQ interrupt when any bit is set in the mailbox.	RW	0x0

Bits	Name	Description	Type	Reset
06	MBOX2_FIQ	When set to '1', this bit causes the third mailbox, i.e. mailbox 4C+2 for ARM core number C, (so mailbox 2 for ARM core 0, 6 for ARM core 1, etc.) to trigger a FIQ interrupt when any bit is set in the mailbox.	RW	0x0
05	MBOX1_FIQ	When set to '1', this bit causes the second mailbox, i.e. mailbox 4C+1 for ARM core number C, (so mailbox 1 for ARM core 0, 5 for ARM core 1, etc.) to trigger a FIQ interrupt when any bit is set in the mailbox.	RW	0x0
04	MBOX0_FIQ	When set to '1', this bit causes the first mailbox, i.e. mailbox 4C for ARM core number C, (so mailbox 0 for ARM core 0, 4 for ARM core 1, etc.) to trigger a FIQ interrupt when any bit is set in the mailbox.	RW	0x0
03	MBOX3_IRQ	When set to '1', this bit causes the fourth mailbox, i.e. mailbox 4C+3 for ARM core number C, (so mailbox 3 for ARM core 0, 7 for ARM core 1, etc.) to trigger an IRQ interrupt when any bit is set in the mailbox. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0
02	MBOX2_IRQ	When set to '1', this bit causes the third mailbox, i.e. mailbox 4C+2 for ARM core number C, (so mailbox 2 for ARM core 0, 6 for ARM core 1, etc.) to trigger an IRQ interrupt when any bit is set in the mailbox. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0
01	MBOX1_IRQ	When set to '1', this bit causes the second mailbox, i.e. mailbox 4C+1 for ARM core number C, (so mailbox 1 for ARM core 0, 5 for ARM core 1, etc.) to trigger an IRQ interrupt when any bit is set in the mailbox. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0
00	MBOX0_IRQ	When set to '1', this bit causes the first mailbox, i.e. mailbox 4C for ARM core number C, (so mailbox 0 for ARM core 0, 4 for ARM core 1, etc.) to trigger an IRQ interrupt when any bit is set in the mailbox. Note that this is overridden by the corresponding FIQ bit: a particular event may be routed either to the FIQ or IRQ request pin, not both. If the FIQ bit is set, then the event will be routed to the FIQ request pin only, irrespective of the state of this bit.	RW	0x0

IRQ_SOURCE0, IRQ_SOURCE1, IRQ_SOURCE2, IRQ_SOURCE3 Registers

Description

This register allows software to determine the cause of an IRQ interrupt request received by an ARM core.

Table 115. IRQ_SOURCE0, IRQ_SOURCE1, IRQ_SOURCE2, IRQ_SOURCE3 Registers

Bits	Name	Description	Type	Reset
31	Reserved.	-	-	-
30	AXI_IRQ	AXI error, as reported by the ARM L2 cache.	RO	0x0
29:12	Reserved.	-	-	-
11	TIMER_IRQ	Local timer interrupt.	RO	0x0
10	AXI_QUIET	No AXI outstanding requests have been seen for the time-out period. **Present for Core 0 only. Reserved for others.**	RO	0x0
09	PMU_IRQ	Performance measurement unit interrupt.	RO	0x0
08	CORE_IRQ	VideoCore interrupt request.	RO	0x0
07:04	MAILBOX_IRQ	Mailbox interrupts: bit 4 is the first of the core's mailboxes, bit 7 is the fourth.	RO	0x0
03	CNT_V_IRQ	Virtual Timer Event interrupt.	RO	0x0
02	CNT_HP_IRQ	Hypervisor Physical Timer Event interrupt.	RO	0x0
01	CNT_PNS_IRQ	Nonsecure Physical Timer Event interrupt.	RO	0x0
00	CNT_PS_IRQ	Secure Physical Timer Event interrupt.	RO	0x0

FIQ_SOURCE0, FIQ_SOURCE1, FIQ_SOURCE2, FIQ_SOURCE3 Registers

Description

This register allows software to determine the cause of a FIQ interrupt request received by an ARM core.

Table 116. FIQ_SOURCE0, FIQ_SOURCE1, FIQ_SOURCE2, FIQ_SOURCE3 Registers

Bits	Name	Description	Type	Reset
31	Reserved.	-	-	-
30	AXI_FIQ	AXI error, as reported by the ARM L2 cache.	RO	0x0
29:12	Reserved.	-	-	-
11	LOCAL_TIMER_FIQ	Local timer interrupt.	RO	0x0
10	Reserved.	-	-	-
09	PMU_FIQ	Performance measurement unit interrupt.	RO	0x0
08	CORE_FIQ	VideoCore interrupt request.	RO	0x0
07:04	MAILBOX_FIQ	Mailbox interrupts: bit 4 is the first of the core's mailboxes, bit 7 is the fourth.	RO	0x0
03	CNT_V_FIQ	Virtual Timer Event interrupt.	RO	0x0
02	CNT_HP_FIQ	Hypervisor Physical Timer Event interrupt.	RO	0x0
01	CNT_PNS_FIQ	Nonsecure Physical Timer Event interrupt.	RO	0x0
00	CNT_PS_FIQ	Secure Physical Timer Event interrupt.	RO	0x0

6.5.3. ARMC

The IRQn_SET_EN_x / IRQn_CLR_EN_x, FIQn_SET_EN_x / FIQn_CLR_EN_x and SWIRQ_SET / SWIRQ_CLEAR registers are write-set / write-clear registers as described earlier.

The ARMC register base address is 0x7e00b000.

Table 117. ARMC Interrupt Registers

Offset	Name	Description
0x200	IRQ0_PENDING0	ARM Core 0 IRQ Enabled Interrupt Pending bits [31:0]
0x204	IRQ0_PENDING1	ARM Core 0 IRQ Enabled Interrupt pending bits [63:32]
0x208	IRQ0_PENDING2	ARM Core 0 IRQ Enabled Interrupt pending bits [79:64]
0x210	IRQ0_SET_EN_0	Write to Set ARM Core 0 IRQ enable bits [31:0]
0x214	IRQ0_SET_EN_1	Write to Set ARM Core 0 IRQ enable bits [63:32]
0x218	IRQ0_SET_EN_2	Write to Set ARM Core 0 IRQ enable bits[79:64]
0x220	IRQ0_CLR_EN_0	Write to Clear ARM Core 0 IRQ enable bits [31:0]
0x224	IRQ0_CLR_EN_1	Write to Clear ARM Core 0 IRQ enable bits [63:32]
0x228	IRQ0_CLR_EN_2	Write to Clear ARM Core 0 IRQ enable bits [79:64]
0x230	IRQ_STATUS0	Interrupt Line bits [31:0]
0x234	IRQ_STATUS1	Interrupt Line bits [63:32]
0x238	IRQ_STATUS2	Interrupt Line bits [79:64]
0x240	IRQ1_PENDING0	ARM Core 1 IRQ Enabled Interrupt pending bits [31:0]
0x244	IRQ1_PENDING1	ARM Core 1 IRQ Enabled Interrupt pending bits [63:32]
0x248	IRQ1_PENDING2	ARM Core 1 IRQ Enabled Interrupt pending bits [79:64]
0x250	IRQ1_SET_EN_0	Write to Set ARM Core 1 IRQ enable bits [31:0]
0x254	IRQ1_SET_EN_1	Write to Set ARM Core 1 IRQ enable bits [63:32]
0x258	IRQ1_SET_EN_2	Write to Set ARM Core 1 IRQ enable bits[79:64]
0x260	IRQ1_CLR_EN_0	Write to Clear ARM Core 1 IRQ enable bits [31:0]
0x264	IRQ1_CLR_EN_1	Write to Clear ARM Core 1 IRQ enable bits [63:32]
0x268	IRQ1_CLR_EN_2	Write to Clear ARM Core 1 IRQ enable bits [79:64]
0x280	IRQ2_PENDING0	ARM Core 2 IRQ Enabled Interrupt pending bits [31:0]
0x284	IRQ2_PENDING1	ARM Core 2 IRQ Enabled Interrupt pending bits [63:32]
0x288	IRQ2_PENDING2	ARM Core 2 IRQ Enabled Interrupt pending bits [79:64]
0x290	IRQ2_SET_EN_0	Write to Set ARM Core 2 IRQ enable bits [31:0]
0x294	IRQ2_SET_EN_1	Write to Set ARM Core 2 IRQ enable bits [63:32]
0x298	IRQ2_SET_EN_2	Write to Set ARM Core 2 IRQ enable bits[79:64]
0x2a0	IRQ2_CLR_EN_0	Write to Clear ARM Core 2 IRQ enable bits [31:0]
0x2a4	IRQ2_CLR_EN_1	Write to Clear ARM Core 2 IRQ enable bits [63:32]
0x2a8	IRQ2_CLR_EN_2	Write to Clear ARM Core 2 IRQ enable bits [79:64]
0x2c0	IRQ3_PENDING0	ARM Core 3 IRQ Enabled Interrupt pending bits [31:0]
0x2c4	IRQ3_PENDING1	ARM Core 3 IRQ Enabled Interrupt pending bits [63:32]
0x2c8	IRQ3_PENDING2	ARM Core 3 IRQ Enabled Interrupt pending bits [79:64]
0x2d0	IRQ3_SET_EN_0	Write to Set ARM Core 3 IRQ enable bits [31:0]
0x2d4	IRQ3_SET_EN_1	Write to Set ARM Core 3 IRQ enable bits [63:32]

Offset	Name	Description
0x2d8	IRQ3_SET_EN_2	Write to Set ARM Core 3 IRQ enable bits[79:64]
0x2e0	IRQ3_CLR_EN_0	Write to Clear ARM Core 3 IRQ enable bits [31:0]
0x2e4	IRQ3_CLR_EN_1	Write to Clear ARM Core 3 IRQ enable bits [63:32]
0x2e8	IRQ3_CLR_EN_2	Write to Clear ARM Core 3 IRQ enable bits [79:64]
0x300	FIQ0_PENDING0	ARM Core 0 FIQ Enabled Interrupt pending bits [31:0]
0x304	FIQ0_PENDING1	ARM Core 0 FIQ Enabled Interrupt pending bits [63:32]
0x308	FIQ0_PENDING2	ARM Core 0 FIQ Enabled Interrupt pending bits [79:64]
0x310	FIQ0_SET_EN_0	Write to Set ARM Core 0 FIQ enable bits [31:0]
0x314	FIQ0_SET_EN_1	Write to Set ARM Core 0 FIQ enable bits [63:32]
0x318	FIQ0_SET_EN_2	Write to Set ARM Core 0 FIQ enable bits[79:64]
0x320	FIQ0_CLR_EN_0	Write to Clear ARM Core 0 FIQ enable bits [31:0]
0x324	FIQ0_CLR_EN_1	Write to Clear ARM Core 0 FIQ enable bits [63:32]
0x328	FIQ0_CLR_EN_2	Write to Clear ARM Core 0 FIQ enable bits [79:64]
0x340	FIQ1_PENDING0	ARM Core 1 FIQ Enabled Interrupt pending bits [31:0]
0x344	FIQ1_PENDING1	ARM Core 1 FIQ Enabled Interrupt pending bits [63:32]
0x348	FIQ1_PENDING2	ARM Core 1 FIQ Enabled Interrupt pending bits [79:64]
0x350	FIQ1_SET_EN_0	Write to Set ARM Core 1 FIQ enable bits [31:0]
0x354	FIQ1_SET_EN_1	Write to Set ARM Core 1 FIQ enable bits [63:32]
0x358	FIQ1_SET_EN_2	Write to Set ARM Core 1 FIQ enable bits[79:64]
0x360	FIQ1_CLR_EN_0	Write to Clear ARM Core 1 FIQ enable bits [31:0]
0x364	FIQ1_CLR_EN_1	Write to Clear ARM Core 1 FIQ enable bits [63:32]
0x368	FIQ1_CLR_EN_2	Write to Clear ARM Core 1 FIQ enable bits [79:64]
0x380	FIQ2_PENDING0	ARM Core 2 FIQ Enabled Interrupt pending bits [31:0]
0x384	FIQ2_PENDING1	ARM Core 2 FIQ Enabled Interrupt pending bits [63:32]
0x388	FIQ2_PENDING2	ARM Core 2 FIQ Enabled Interrupt pending bits [79:64]
0x390	FIQ2_SET_EN_0	Write to Set ARM Core 2 FIQ enable bits [31:0]
0x394	FIQ2_SET_EN_1	Write to Set ARM Core 2 FIQ enable bits [63:32]
0x398	FIQ2_SET_EN_2	Write to Set ARM Core 2 FIQ enable bits[79:64]
0x3a0	FIQ2_CLR_EN_0	Write to Clear ARM Core 2 FIQ enable bits [31:0]
0x3a4	FIQ2_CLR_EN_1	Write to Clear ARM Core 2 FIQ enable bits [63:32]
0x3a8	FIQ2_CLR_EN_2	Write to Clear ARM Core 2 FIQ enable bits [79:64]
0x3c0	FIQ3_PENDING0	ARM Core 3 FIQ Enabled Interrupt pending bits [31:0]
0x3c4	FIQ3_PENDING1	ARM Core 3 FIQ Enabled Interrupt pending bits [63:32]
0x3c8	FIQ3_PENDING2	ARM Core 3 FIQ Enabled Interrupt pending bits [79:64]
0x3d0	FIQ3_SET_EN_0	Write to Set ARM Core 3 FIQ enable bits [31:0]
0x3d4	FIQ3_SET_EN_1	Write to Set ARM Core 3 FIQ enable bits [63:32]

Offset	Name	Description
0x3d8	FIQ3_SET_EN_2	Write to Set ARM Core 3 FIQ enable bits[79:64]
0x3e0	FIQ3_CLR_EN_0	Write to Clear ARM Core 3 FIQ enable bits [31:0]
0x3e4	FIQ3_CLR_EN_1	Write to Clear ARM Core 3 FIQ enable bits [63:32]
0x3e8	FIQ3_CLR_EN_2	Write to Clear ARM Core 3 FIQ enable bits [79:64]
0x3f0	SWIRQ_SET	Write to Set Software Interrupt sources
0x3f4	SWIRQ_CLEAR	Write to Clear Software Interrupt sources

IRQ0_PENDING0, IRQ1_PENDING0, IRQ2_PENDING0, IRQ3_PENDING0 Registers

Description

Shows the status of the Enabled interrupts [31:0] (that will be OR-ed into the Core's interrupt line)
Only Interrupts that are enabled will show up here

Table 118. IRQ0_PENDING0, IRQ1_PENDING0, IRQ2_PENDING0, IRQ3_PENDING0 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	RO	0x00000000

IRQ0_PENDING1, IRQ1_PENDING1, IRQ2_PENDING1, IRQ3_PENDING1 Registers

Description

Shows the status of the Enabled interrupts [63:32] (that will be OR-ed into the Core's interrupt line)
Only Interrupts that are enabled will show up here

Table 119. IRQ0_PENDING1, IRQ1_PENDING1, IRQ2_PENDING1, IRQ3_PENDING1 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	RO	0x00000000

IRQ0_PENDING2, IRQ1_PENDING2, IRQ2_PENDING2, IRQ3_PENDING2 Registers

Description

Shows the status of the Enabled interrupts [79:64] (that will be OR-ed into the Core's interrupt line)
Only Interrupts that are enabled will show up here

Table 120. IRQ0_PENDING2, IRQ1_PENDING2, IRQ2_PENDING2, IRQ3_PENDING2 Registers

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	RO	0x0
30:26	Reserved.	-	-	-
25	INT63_32	This bit is the logical OR of all the interrupt pending bits for interrupts 63 to 32. If set, read the PENDING1 register to determine which interrupts are pending from this set.	RO	0x0
24	INT31_0	This bit is the logical OR of all the interrupt pending bits for interrupts 31 to 0. If set, read the PENDING0 register to determine which interrupts are pending from this set.	RO	0x0
23:16	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits. These interrupts can only be cleared by clearing the trigger bit by writing to the SWIRQ_CLEAR register.	RO	0x00
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred. This interrupt cannot be cleared other than by resetting the ARM complex.	RO	0x0
06	ARM_ADDR_ERROR	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	RO	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	RO	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	RO	0x0
03	BELL_IRQ1	Doorbell 1 interrupt. This interrupt can be cleared by reading the relevant doorbell register.	RO	0x0
02	BELL_IRQ0	Doorbell 0 interrupt. This interrupt can be cleared by reading the relevant doorbell register.	RO	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	RO	0x0
00	TIMER_IRQ	Timer interrupt. This interrupt can be cleared by writing to the IRQCNTL register.	RO	0x0

IRQ0_SET_EN_0, IRQ1_SET_EN_0, IRQ2_SET_EN_0, IRQ3_SET_EN_0 Registers

Description

Writing a '1' to a bit position in this register enables the corresponding interrupt.
A read returns the current state of this enable register.

Table 121. IRQ0_SET_EN_0, IRQ1_SET_EN_0, IRQ2_SET_EN_0, IRQ3_SET_EN_0 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	RW	0x00000000

IRQ0_SET_EN_1, IRQ1_SET_EN_1, IRQ2_SET_EN_1, IRQ3_SET_EN_1 Registers

Description

Writing a '1' to a bit position in this register enables the corresponding interrupt.
A read returns the current state of this enable register.

Table 122. IRQ0_SET_EN_1, IRQ1_SET_EN_1, IRQ2_SET_EN_1, IRQ3_SET_EN_1 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	RW	0x00000000

IRQ0_SET_EN_2, IRQ1_SET_EN_2, IRQ2_SET_EN_2, IRQ3_SET_EN_2 Registers

Description

Writing a '1' to a bit position in this register enables the corresponding interrupt.
A read returns the current state of this enable register.

Table 123. IRQ0_SET_EN_2, IRQ1_SET_EN_2, IRQ2_SET_EN_2, IRQ3_SET_EN_2 Registers

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	RW	0x0
30:16	Reserved.	-	-	-
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits.	RW	0x00
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred.	RW	0x0
06	ARM_ADDR_ERROR	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	RW	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	RW	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	RW	0x0
03	BELL_IRQ1	Doorbell 1 interrupt.	RW	0x0
02	BELL_IRQ0	Doorbell 0 interrupt.	RW	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	RW	0x0
00	TIMER_IRQ	Timer interrupt.	RW	0x0

IRQ0_CLR_EN_0, IRQ1_CLR_EN_0, IRQ2_CLR_EN_0, IRQ3_CLR_EN_0 Registers

Description

Writing a '1' to a bit position in this register disables the corresponding interrupt.
A read returns the current state of the IRQ enable register.

Table 124. IRQ0_CLR_EN_0, IRQ1_CLR_EN_0, IRQ2_CLR_EN_0, IRQ3_CLR_EN_0 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	W1C	0x00000000

IRQ0_CLR_EN_1, IRQ1_CLR_EN_1, IRQ2_CLR_EN_1, IRQ3_CLR_EN_1 Registers

Description

Writing a '1' to a bit position in this register disables the corresponding interrupt.
A read returns the current state of the IRQ enable register.

Table 125. IRQ0_CLR_EN_1, IRQ1_CLR_EN_1, IRQ2_CLR_EN_1, IRQ3_CLR_EN_1 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	W1C	0x00000000

IRQ0_CLR_EN_2, IRQ1_CLR_EN_2, IRQ2_CLR_EN_2, IRQ3_CLR_EN_2 Registers

Description

Writing a '1' to a bit position in this register disables the corresponding interrupt.
A read returns the current state of the IRQ enable register.

Table 126. IRQ0_CLR_EN_2, IRQ1_CLR_EN_2, IRQ2_CLR_EN_2, IRQ3_CLR_EN_2 Registers

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	W1C	0x0
30:16	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits.	W1C	0x00
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred.	W1C	0x0
06	ARM_ADDR_ERROR	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	W1C	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	W1C	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	W1C	0x0
03	BELL_IRQ1	Doorbell 1 interrupt.	W1C	0x0
02	BELL_IRQ0	Doorbell 0 interrupt.	W1C	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	W1C	0x0
00	TIMER_IRQ	Timer interrupt.	W1C	0x0

IRQ_STATUS0 Register

Description

Shows the status of the actual Interrupts [31:0] before they are masked

Table 127. IRQ_STATUS0 Register

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	RO	0x00000000

IRQ_STATUS1 Register

Description

Shows the status of the actual Interrupts [63:32] before they are masked

Table 128. IRQ_STATUS1 Register

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	RO	0x00000000

IRQ_STATUS2 Register

Description

Shows the status of the actual Interrupts [79:64] before they are masked

Table 129. IRQ_STATUS2 Register

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	RO	0x0
30:16	Reserved.	-	-	-
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits. These interrupts can only be cleared by clearing the trigger bit by writing to the SWIRQ_CLEAR register.	RO	0x00

Bits	Name	Description	Type	Reset
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred. This interrupt cannot be cleared other than by resetting the ARM complex.	RO	0x0
06	ARM_ADDR_ERRO R	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	RO	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	RO	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	RO	0x0
03	BELL_IRQ1	Doorbell 1 interrupt. This interrupt can be cleared by reading the relevant doorbell register.	RO	0x0
02	BELL_IRQ0	Doorbell 0 interrupt. This interrupt can be cleared by reading the relevant doorbell register.	RO	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	RO	0x0
00	TIMER_IRQ	Timer interrupt. This interrupt can be cleared by writing to the IRQCNTL register.	RO	0x0

FIQ0_PENDING0, FIQ1_PENDING0, FIQ2_PENDING0, FIQ3_PENDING0 Registers

Description

Shows the status of the Enabled interrupts [31:0] (that will be OR-ed into the Core's interrupt line)
Only Interrupts that are enabled will show up here

Table 130. FIQ0_PENDING0, FIQ1_PENDING0, FIQ2_PENDING0, FIQ3_PENDING0 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	RO	0x00000000

FIQ0_PENDING1, FIQ1_PENDING1, FIQ2_PENDING1, FIQ3_PENDING1 Registers

Description

Shows the status of the Enabled interrupts [63:32] (that will be OR-ed into the Core's interrupt line)
Only Interrupts that are enabled will show up here

Table 131. FIQ0_PENDING1, FIQ1_PENDING1, FIQ2_PENDING1, FIQ3_PENDING1 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	RO	0x00000000

FIQ0_PENDING2, FIQ1_PENDING2, FIQ2_PENDING2, FIQ3_PENDING2 Registers

Description

Shows the status of the Enabled interrupts [79:64] (that will be OR-ed into the Core's interrupt line)
Only Interrupts that are enabled will show up here

Table 132. FIQ0_PENDING2, FIQ1_PENDING2, FIQ2_PENDING2, FIQ3_PENDING2 Registers

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	RO	0x0
30:26	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
25	INT63_32	This bit is the logical OR of all the interrupt pending bits for interrupts 63 to 32. If set, read the PENDING1 register to determine which interrupts are pending from this set.	RO	0x0
24	INT31_0	This bit is the logical OR of all the interrupt pending bits for interrupts 31 to 0. If set, read the PENDING0 register to determine which interrupts are pending from this set.	RO	0x0
23:16	Reserved.	-	-	-
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits. These interrupts can only be cleared by clearing the trigger bit by writing to the SWIRQ_CLEAR register.	RO	0x00
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred. This interrupt cannot be cleared other than by resetting the ARM complex.	RO	0x0
06	ARM_ADDR_ERROR	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	RO	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	RO	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	RO	0x0
03	BELL_IRQ1	Doorbell 1 interrupt. This interrupt can be cleared by reading the relevant doorbell register.	RO	0x0
02	BELL_IRQ0	Doorbell 0 interrupt. This interrupt can be cleared by reading the relevant doorbell register.	RO	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	RO	0x0
00	TIMER_IRQ	Timer interrupt. This interrupt can be cleared by writing to the IRQCNTL register.	RO	0x0

FIQ0_SET_EN_0, FIQ1_SET_EN_0, FIQ2_SET_EN_0, FIQ3_SET_EN_0 Registers

Description

Writing a '1' to a bit position in this register enables the corresponding interrupt.
A read returns the current state of the FIQ enable register.

Table 133. FIQ0_SET_EN_0, FIQ1_SET_EN_0, FIQ2_SET_EN_0, FIQ3_SET_EN_0 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	RW	0x00000000

FIQ0_SET_EN_1, FIQ1_SET_EN_1, FIQ2_SET_EN_1, FIQ3_SET_EN_1 Registers

Description

Writing a '1' to a bit position in this register enables the corresponding interrupt.
A read returns the current state of the FIQ enable register.

Table 134. FIQ0_SET_EN_1, FIQ1_SET_EN_1, FIQ2_SET_EN_1, FIQ3_SET_EN_1 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	RW	0x00000000

FIQ0_SET_EN_2, FIQ1_SET_EN_2, FIQ2_SET_EN_2, FIQ3_SET_EN_2 Registers

Description

Writing a '1' to a bit position in this register enables the corresponding interrupt.
A read returns the current state of the FIQ enable register.

Table 135. FIQ0_SET_EN_2, FIQ1_SET_EN_2, FIQ2_SET_EN_2, FIQ3_SET_EN_2 Registers

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	RW	0x0
30:16	Reserved.	-	-	-
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits.	RW	0x00
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred.	RW	0x0
06	ARM_ADDR_ERROR	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	RW	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	RW	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	RW	0x0
03	BELL_IRQ1	Doorbell 1 interrupt.	RW	0x0
02	BELL_IRQ0	Doorbell 0 interrupt.	RW	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	RW	0x0
00	TIMER_IRQ	Timer interrupt.	RW	0x0

FIQ0_CLR_EN_0, FIQ1_CLR_EN_0, FIQ2_CLR_EN_0, FIQ3_CLR_EN_0 Registers

Description

Writing a '1' to a bit position in this register disables the corresponding interrupt.
A read returns the current state of the FIQ enable register.

Table 136. FIQ0_CLR_EN_0, FIQ1_CLR_EN_0, FIQ2_CLR_EN_0, FIQ3_CLR_EN_0 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_31_0	VideoCore interrupts 31 to 0.	W1C	0x00000000

FIQ0_CLR_EN_1, FIQ1_CLR_EN_1, FIQ2_CLR_EN_1, FIQ3_CLR_EN_1 Registers

Description

Writing a '1' to a bit position in this register disables the corresponding interrupt.
A read returns the current state of the FIQ enable register.

Table 137. FIQ0_CLR_EN_1, FIQ1_CLR_EN_1, FIQ2_CLR_EN_1, FIQ3_CLR_EN_1 Registers

Bits	Name	Description	Type	Reset
31:00	VC_IRQ_63_32	VideoCore interrupts 63 to 32.	W1C	0x00000000

FIQ0_CLR_EN_2, FIQ1_CLR_EN_2, FIQ2_CLR_EN_2, FIQ3_CLR_EN_2 Registers

Description

Writing a '1' to a bit position in this register disables the corresponding interrupt.
A read returns the current state of the FIQ enable register.

Table 138. FIQ0_CLR_EN_2, FIQ1_CLR_EN_2, FIQ2_CLR_EN_2, FIQ3_CLR_EN_2 Registers

Bits	Name	Description	Type	Reset
31	IRQ	This is the value of the ARM interrupt input	W1C	0x0
30:16	Reserved.	-	-	-
15:08	SW_TRIG_INT	These eight bits are software-triggered interrupts. By writing to the SWIRQ_SET register, software may set interrupt trigger bits.	W1C	0x00
07	ARM_AXI_ERROR	ARM AXI error interrupt. This is set if the logic in the ARM block detects that an AXI error has occurred.	W1C	0x0
06	ARM_ADDR_ERROR	ARM address range error. This interrupt is set if the ARM attempts an AXI burst (ALEN > 0) access to VideoCore peripheral space.	W1C	0x0
05	VPU_C1_HALT	VPU Core 1 halted in debug mode.	W1C	0x0
04	VPU_C0_C1_HALT	VPU Core 0 halted in debug mode, or (if enabled by bit 10 of the config register) VPU Core 1 halted in debug mode.	W1C	0x0
03	BELL_IRQ1	Doorbell 1 interrupt.	W1C	0x0
02	BELL_IRQ0	Doorbell 0 interrupt.	W1C	0x0
01	MAILBOX_IRQ0	Mailbox 0 interrupt.	W1C	0x0
00	TIMER_IRQ	Timer interrupt.	W1C	0x0

SWIRQ_SET Register

Description

Software-triggered interrupts.
Writing a '1' to a bit position in this register sets the corresponding software interrupt source bit. A read returns the current state of the software interrupt bits.

Table 139. SWIRQ_SET Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
07:00	SW_INT	Eight software-triggered interrupt bits.	RW	0x00

SWIRQ_CLEAR Register

Description

Software-triggered interrupts.
Writing a '1' to a bit position in this register clears the corresponding software interrupt source bit. A read returns the current state of the software interrupt bits.

Table 140. SWIRQ_CLEAR Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
07:00	SW_INT	Eight software-triggered interrupt bits.	W1C	0x00

Chapter 7. PCM / I2S Audio

7.1. Overview

The PCM (Pulse Code Modulation) 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:

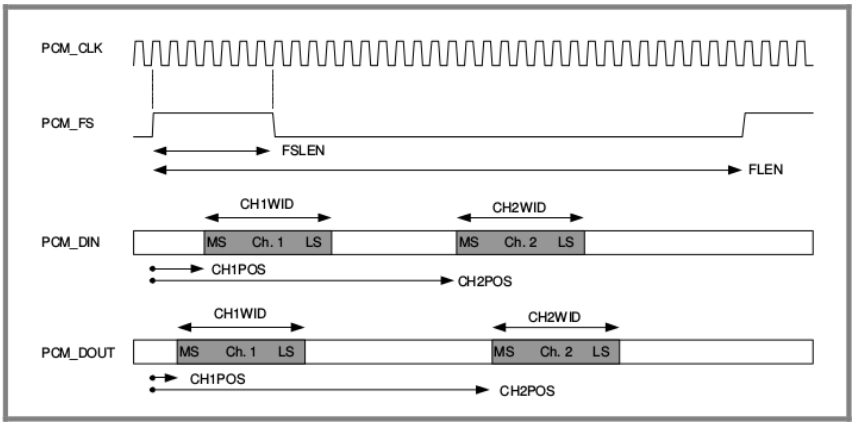
- PCM_CLK - bit clock
- PCM_FS - frame sync signal
- PCM_DIN - serial data input
- PCM_DOUT - serial data output

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 polarity 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 10. PCM Audio Interface Typical Timing



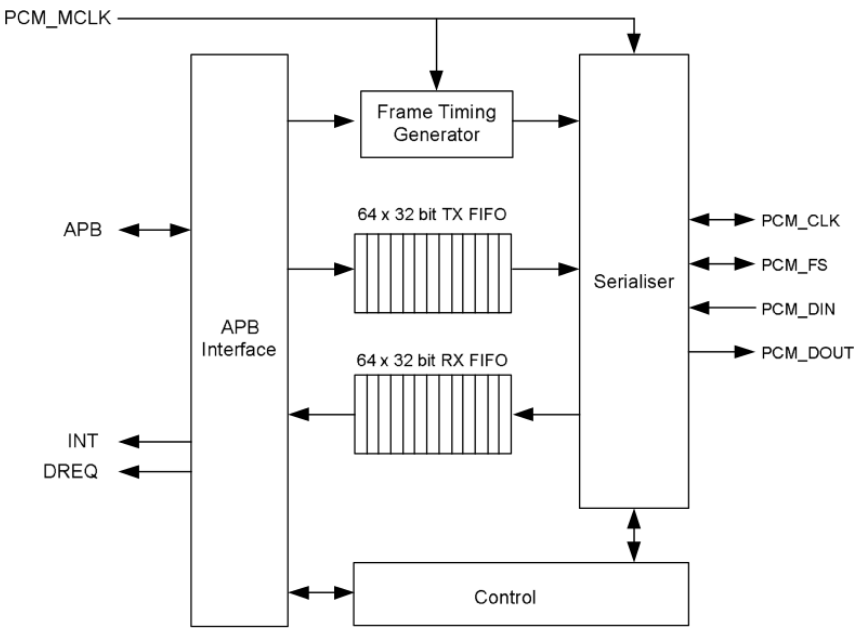
The PCM_CLK can be asynchronous to the bus APB clock and can be logically inverted if required.

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 (Pulse Density Modulation) microphones, as an alternative to the classic PCM input format, in conjunction with a PCM output.

7.2. Block Diagram

Figure 11. 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.

7.3. Typical Timing

Figure 10 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 12.

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.

7.4. 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 (with the exception of INTSTC_A and GRAY) 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 receive FIFO becomes full or the transmit FIFO becomes empty, 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.

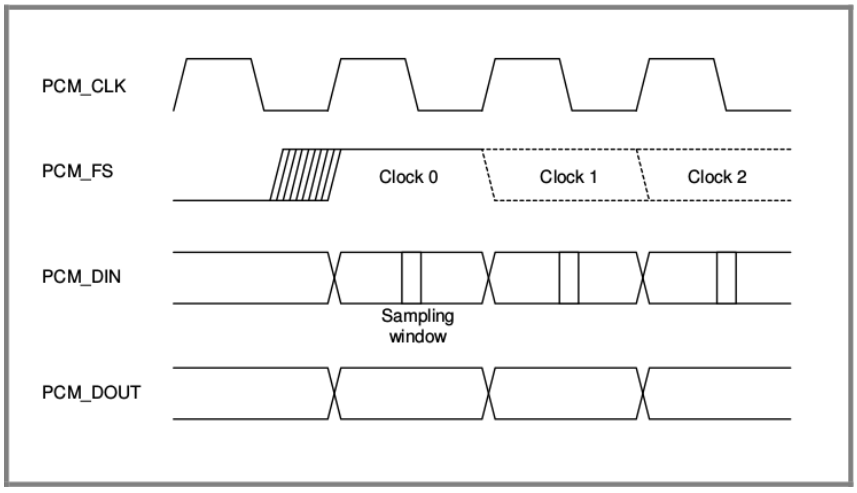
If a FIFO error occurs in a two channel frame, then channel synchronisation may be lost which may result in a left-right audio channel swap. RXSYNC and TXSYNC status bits are provided to help determine if channel slip has occurred. They indicate if the number of words in the FIFO is a multiple of a full 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 (2 x 16-bit samples). In this mode each word written to the TX FIFO contains two 16-bit samples, and the Least Significant sample is transmitted first. Each word read from the RX FIFO will contain the data received from two channels, the first channel received will be in the Least Significant half of the word. If the channel's 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 12. Timing at Start of Frame



Note that the precise timing of PCM_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 (on a negative edge of PCM_CLK) as a 1 where it was previously sampled as a 0.

7.5. Software Operation

7.5.1. Operating in Polled mode

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

7.5.2. Operating in Interrupt mode

1. 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 and 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.
2. Set INTR and/or INTT to enable interrupts.
3. If transmitting, ensure that sufficient sample words have been written to PCM FIFO before transmission is started. Set TXON and/or RXON to begin operation.
4. When an interrupt occurs, check RXR. If this is set then one or more sample words are available in PCM FIFO. If TXW is set then one or more sample words can be sent to PCM FIFO.

7.5.3. DMA

1. 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 and wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed.
2. Set DMAEN to enable DMA DREQ generation and set RX_REQ/TX_REQ to determine the FIFO thresholds for the DREQs. If required, set TX_PANIC and RX_PANIC to determine the level at which the DMA should increase its AXI priority,
3. 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.
4. Set TXON and/or RXON to begin operation.

7.6. 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 FIFOs 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 RXERR and/or TXERR bits in INTEN_A 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:

- Disable transmit and receive (set TXON and RXON to 0). Clear the FIFOs (set RXCLR and TXCLR to 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.
- 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.

7.7. PDM Input Mode Operation

The PDM input mode is capable of interfacing with two digital half-cycle PDM microphones and implements a 4^{th} 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.

Table 141. PDM Input Mode Configuration

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

7.8. 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.

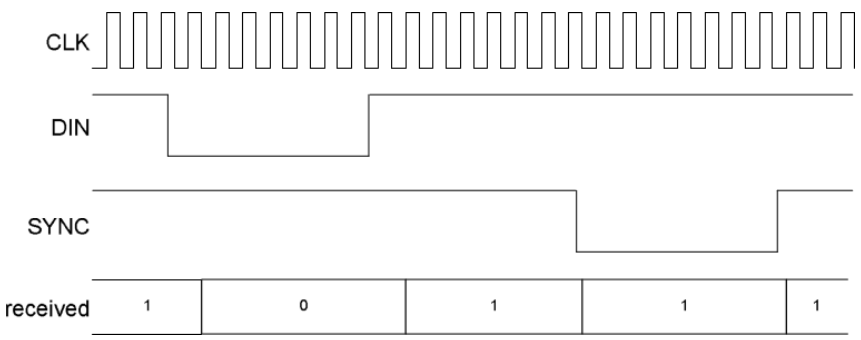
In this mode data is received on the PCM_DIN (data) and the PCM_FS (strobe) pins. The data is expected to be in data/strobe format. In this mode data is detected when either the data or the strobe change state. As each bit is received it is written into the RX buffer and when 32 bits are received they are written out to the RX FIFO 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 then the gray data rate. Also the gray coded data input signals should be clean.

The normal RX_REQ 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 RX FIFO. 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 twice the PCM_CLK.

Figure 13. Gray mode input format



7.9. PCM Register Map

There is only one PCM module in the BCM2711. The PCM base address for the registers is `0x7e203000`.

Table 142. PCM Register Map

Offset	Name	Description
0x00	CS_A	PCM Control and Status
0x04	FIFO_A	PCM FIFO Data
0x08	MODE_A	PCM Mode
0x0c	RXC_A	PCM Receive Configuration
0x10	TXC_A	PCM Transmit Configuration
0x14	DREQ_A	PCM DMA Request Level
0x18	INTEN_A	PCM Interrupt Enables
0x1c	INTSTC_A	PCM Interrupt Status & Clear
0x20	GRAY	PCM Gray Mode Control

CS_A Register

Description

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.

Table 143. CS_A Register

Bits	Name	Description	Type	Reset
31:25	Reserved.	-	-	-
24	SYNC	PCM Clock sync helper. 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	RX Sign Extend 0 = No sign extension. 1 = Sign extend the RX data. When set, the MSB of the received data channel (as set by the CH*x*WID parameter) is repeated in all the higher data bits up to the full 32-bit data width.	RW	0x0
22	RXF	RX FIFO is Full 0 = RX FIFO can accept more data. 1 = RX FIFO is full and will overflow if more data is received.	RO	0x0

Bits	Name	Description	Type	Reset
21	TXE	TX FIFO is Empty 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	Indicates that the RX FIFO contains Data 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 sample.	RO	0x0
19	TXD	Indicates that the TX FIFO can accept Data 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	Indicates that the RX FIFO needs Reading 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	Indicates that the TX FIFO needs Writing 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	RX FIFO Error 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1.	W1C	0x0
15	TXERR	TX FIFO Error 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1.	W1C	0x0
14	RXSYNC	RX FIFO Sync 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	TX FIFO Sync 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.	-	-	-
9	DMAEN	DMA DREQ Enable 0 = Don't generate DMA DREQ requests. 1 = Generates a TX DMA DREQ request whenever the TX FIFO level is lower than TX_REQ or generates a RX DMA DREQ when the RX FIFO level is higher than RX_REQ.	RW	0x0
8:7	RXTHR	Sets the RX FIFO threshold at which point the RXR flag is set 00 = set when we have a single sample in the RX FIFO 01 = set when the RX FIFO is at least ¼ full 10 = set when the RX FIFO is at least ¾ full 11 = set when the RX FIFO is full	RW	0x0

Bits	Name	Description	Type	Reset
6:5	TXTHR	Sets the TX FIFO threshold at which point the TXW flag is set 00 = set when the TX FIFO is empty 01 = set when the TX FIFO is less than ¼ full 10 = set when the TX FIFO is less than ¾ full 11 = set when the TX FIFO is full but for one sample	RW	0x0
4	RXCLR	Clear the RX FIFO. 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.	W1SC	0x0
3	TXCLR	Clear the TX FIFO 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.	W1SC	0x0
2	TXON	Enable transmission 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.	RW	0x0
1	RXON	Enable reception. 0 = Disable reception. This will stop on the next available frame end. RX FIFO data can still be read. 1 = Enable reception. This will 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.	RW	0x0
0	EN	Enable the PCM Audio Interface 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.	RW	0x0

FIFO_A Register

Description

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

Table 144. FIFO_A Register

Bits	Name	Description	Type	Reset
31:0	FIFO	Data written here is transmitted, and received data is read from here.	RW	0x00000000

MODE_A Register

Description

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.

Table 145. MODE_A Register

Bits	Name	Description	Type	Reset
31:29	Reserved.	-	-	-
28	CLK_DIS	PCM Clock Disable 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	PDM Decimation Factor (N) 0 = Decimation factor 16. 1 = Decimation factor 32. Sets the decimation factor of the CIC decimation filter.	RW	0x0
26	PDME	PDM Input Mode Enable 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	Receive Frame Packed Mode 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	Transmit Frame Packed Mode 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	PCM Clock Mode 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

Bits	Name	Description	Type	Reset
22	CLKI	Clock Invert This logically inverts the PCM_CLK signal. 0 = Outputs change on rising edge of clock, inputs are sampled on falling edge. 1 = Outputs change on falling edge of clock, inputs are sampled on rising edge.	RW	0x0
21	FSM	Frame Sync Mode 0 = Master mode. The PCM_FS is an output and we generate the frame sync. 1 = Slave mode. The PCM_FS is an input and we lock onto the incoming frame sync signal.	RW	0x0
20	FSI	Frame Sync Invert This logically inverts the frame sync signal. 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. 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.	RW	0x0
19:10	FLEN	Frame Length Sets the frame length to (FLEN+1) clocks. 1 = frame length of 2 clocks. 2 = frame length of 3 clocks. etc.	RW	0x000
9:0	FSLEN	Frame Sync Length Sets the frame sync length to (FSLEN) clocks. This is only used when FSM = 0. PCM_FS will remain permanently active if FSLEN >= FLEN. 0 = frame sync pulse is off. 1 = frame sync pulse is 1 clock wide. etc.	RW	0x000

RXC_A Register

Description

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 channel 2 can come after channel 1, although the first data will always be from the first channel in the frame. Channels can also straddle the frame begin-end boundary (as set by the frame sync position). This register cannot be changed whilst the PCM is running.

Table 146. RXC_A Register

Bits	Name	Description	Type	Reset
31	CH1WEX	Channel 1 Width Extension Bit 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	Channel 1 Enable 0 = Channel 1 disabled and no data is received from channel 1 and written to the RX FIFO. 1 = Channel 1 enabled.	RW	0x0

Bits	Name	Description	Type	Reset
29:20	CH1POS	Channel 1 Position 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	0x000
19:16	CH1WID	Channel 1 Width 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 etc.	RW	0x0
15	CH2WEX	Channel 2 Width Extension Bit 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	Channel 2 Enable 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	Channel 2 Position 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	0x000
3:0	CH2WID	Channel 2 Width 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 etc.	RW	0x0

TXC_A Register

Description

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 channel 2 can come after channel 1, 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 set by the frame sync position). This register cannot be changed whilst the PCM is running.

Table 147. TXC_A Register

Bits	Name	Description	Type	Reset
31	CH1WEX	Channel 1 Width Extension Bit 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

Bits	Name	Description	Type	Reset
30	CH1EN	Channel 1 Enable 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	Channel 1 Position 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	0x000
19:16	CH1WID	Channel 1 Width 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 etc.	RW	0x0
15	CH2WEX	Channel 2 Width Extension Bit 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	Channel 2 Enable 0 = Channel 2 disabled and no data is taken from the TX FIFO and transmitted on channel 2. 1 = Channel 2 enabled.	RW	0x0
13:4	CH2POS	Channel 2 Position 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	0x000
3:0	CH2WID	Channel 2 Width 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 etc.	RW	0x0

DREQ_A Register

Description

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.

Table 148. DREQ_A Register

Bits	Name	Description	Type	Reset
31	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
30:24	TX_PANIC	TX Panic Level 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.	-	-	-
22:16	RX_PANIC	RX Panic Level 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.	-	-	-
14:8	TX_REQ	TX Request Level This sets the TX FIFO DREQ level. When the level is below this the PCM will assert its DMA DREQ signal to request that more data is written to the TX FIFO.	RW	0x30
7	Reserved.	-	-	-
6:0	RX_REQ	RX Request Level 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

Description

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

Table 149. INTEN_A Register

Bits	Name	Description	Type	Reset
31:4	Reserved.	-	-	-
3	RXERR	RX Error Interrupt Setting this bit enables interrupts from PCM block when RX FIFO error occurs.	RW	0x0
2	TXERR	TX Error Interrupt Setting this bit enables interrupts from PCM block when TX FIFO error occurs.	RW	0x0
1	RXR	RX Read Interrupt Enable 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	TX Write Interrupt Enable Setting this bit enables interrupts from PCM block when TX FIFO level is less than the specified TXTHR level.	RW	0x0

INTSTC_A Register

Description

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.

Table 150. INTSTC_A Register

Bits	Name	Description	Type	Reset
31:4	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
3	RXERR	RX Error Interrupt Status / Clear This bit indicates an interrupt occurred on RX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect.	W1C	0x0
2	TXERR	TX Error Interrupt Status / Clear This bit indicates an interrupt occurred on TX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect.	W1C	0x0
1	RXR	RX Read Interrupt Status / Clear This bit indicates an interrupt occurred on RX Read. Writing 1 to this bit clears it. Writing 0 has no effect.	W1C	0x0
0	TXW	TX Write Interrupt Status / Clear This bit indicates an interrupt occurred on TX Write. Writing 1 to this bit clears it. Writing 0 has no effect.	W1C	0x0

GRAY Register

Description

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.

Table 151. GRAY Register

Bits	Name	Description	Type	Reset
31:22	Reserved.	-	-	-
21:16	RXFIFOLEVEL	The current level of the RX FIFO This indicates how many words are currently in the RX FIFO.	RO	0x00
15:10	FLUSHED	The number of bits that were flushed into the RX FIFO 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	0x00
9:4	RXLEVEL	The current fill level of the RX Buffer This indicates how many GRAY coded bits have been received. When 32 bits are received, they are written out into the RX FIFO.	RO	0x00
3	Reserved.	-	-	-
2	FLUSH	Flush the RX Buffer into the RX FIFO 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	Clear the GRAY Mode Logic This bit will reset all the GRAY mode logic, and flush the RX buffer. It is not self clearing.	RW	0x0

Bits	Name	Description	Type	Reset
0	EN	Enable GRAY Mode 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

Chapter 8. Pulse Width Modulator

8.1. Overview

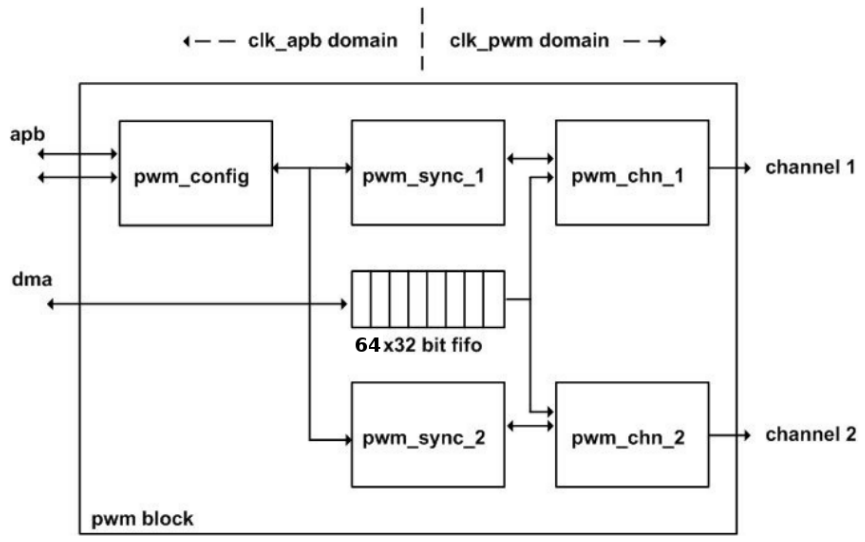
This section specifies in detail the functionality provided by the device Pulse Width Modulator (PWM) peripherals.

Each 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 output resolutions
- Serialise mode configured to read data from a FIFO storage block, which can store up to sixty-four 32-bit words
- Both modes clocked by clk_pwm which is nominally 100MHz, but can be varied by the clock manager

8.2. Block Diagram

Figure 14. PWM block diagram



The BCM2711 device has two instances of this block, named PWM0 and PWM1 (each with two output channels).

8.3. 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 1s and 0s spread out as evenly as possible, so that during any arbitrary period of time the duty cycle achieves the 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
       send 0
4. Repeat from step 2
```

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

8.4. Modes of Operation

The PWM controller consists of two independent channels (pwm_chn_*i* in Figure 14) which implement the PWM algorithm explained in the previous section. 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. The 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. The channel sends its output continuously as long as the data register is used (USEF*i*=0), or the FIFO is used and it is not empty.

When MSEN=1, the PWM channel does not use the algorithm explained above, instead it sends serial data with the M/S ratio as in Figure 15. 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. The channel sends its output continuously as long as the data register is used (USEF*i*=0), or the FIFO is used and it is not empty.

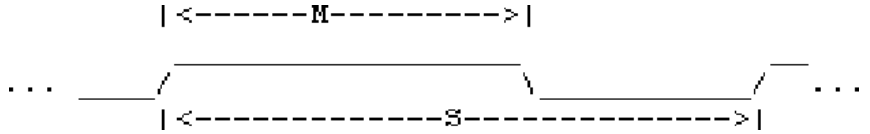


Figure 15. Serial bit transmission when M/S Mode enabled

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

8.5. Quick Reference

- PWM0 DMA is mapped to DMA channel 5.
- PWM1 DMA is mapped to DMA channel 1 (muxed with DSI0).
- GPIOs are assigned to PWM channels as below. Please refer to the GPIO chapter for further details:

GPIO	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO12	PWM0_0					
GPIO13	PWM0_1					
GPIO18						PWM0_0
GPIO19						PWM0_1
GPIO40	PWM1_0					
GPIO41	PWM1_1					
GPIO45	PWM0_1					

- PWM clock source and frequency is controlled in CPRMAN.

8.6. Control and Status Registers

The PWM0 register base address is `0x7e20c000` and the PWM1 register base address is `0x7e20c800`.

Table 152. PWM Register Map

Offset	Name	Description
0x00	CTL	PWM Control
0x04	STA	PWM Status
0x08	DMAC	PWM DMA Configuration
0x10	RNG1	PWM Channel 1 Range
0x14	DAT1	PWM Channel 1 Data
0x18	FIF1	PWM FIFO Input
0x20	RNG2	PWM Channel 2 Range
0x24	DAT2	PWM Channel 2 Data

CTL Register

Description

PWEN*i* is used to enable/disable the corresponding channel. Setting this bit to 1 enables the channel and transmitter state machine. All registers and FIFOs are writeable without setting this bit.
MODE*i* bit is used to determine mode of operation. Setting this bit to 0 (the default) enables PWM mode. In this mode data stored in either PWM_DAT*i* or FIFO is transmitted by pulse width modulation within the range defined by PWM_RNG*i*. When this mode is used, MSEN*i* defines whether to use PWM algorithm or M/S transmission. Setting MODE*i* to 1 enables serial mode, in which data stored in either PWM_DAT*i* or FIFO is transmitted serially within the range defined by PWM_RNG*i*. Data is transmitted MSB first and truncated or zero-padded depending on PWM_RNG*i*.
RPTL*i* 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.
SBIT*i* 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_RNG*i* is larger than bit depth of data being transferred. This bit is zero by default.
POLA*i* is used to configure the polarity of the output bit. When set to high the final output is inverted. Default operation is no inversion.
USEF*i* 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_DAT*i* is transferred. This bit is 0 by 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 one-shot operation and reading the bit always returns 0.
MSEN*i* 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 by default. When MODE*i* is 1, this configuration bit has no effect.

Table 153. CTL Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15	MSEN2	Channel 2 M/S Enable 0: PWM algorithm is used 1: M/S transmission is used.	RW	0x0
14	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
13	USEF2	Channel 2 Use FIFO 0: Data register is transmitted 1: FIFO is used for transmission	RW	0x0
12	POLA2	Channel 2 Polarity 0 : 0=low 1=high 1: 1=low 0=high	RW	0x0
11	SBIT2	Channel 2 Silence Bit Defines the state of the output when no transmission takes place	RW	0x0
10	RPTL2	Channel 2 Repeat Last Data 0: Transmission interrupts when FIFO is empty 1: Last data in FIFO is transmitted repeatedly until FIFO is not empty	RW	0x0
9	MODE2	Channel 2 Mode 0: PWM mode 1: Serialiser mode	RW	0x0
8	PWEN2	Channel 2 Enable 0: Channel is disabled 1: Channel is enabled	RW	0x0
7	MSEN1	Channel 1 M/S Enable 0: PWM algorithm is used 1: M/S transmission is used.	RW	0x0
6	CLRF	Clear FIFO 1: Clears FIFO 0: Has no effect This is a one-shot operation. This bit always reads 0	W1SC	0x0
5	USEF1	Channel 1 Use FIFO 0: Data register is transmitted 1: FIFO is used for transmission	RW	0x0
4	POLA1	Channel 1 Polarity 0 : 0=low 1=high 1: 1=low 0=high	RW	0x0
3	SBIT1	Channel 1 Silence Bit Defines the state of the output when no transmission takes place	RW	0x0
2	RPTL1	Channel 1 Repeat Last Data 0: Transmission interrupts when FIFO is empty 1: Last data in FIFO is transmitted repeatedly until FIFO is not empty	RW	0x0
1	MODE1	Channel 1 Mode 0: PWM mode 1: Serialiser mode	RW	0x0
0	PWEN1	Channel 1 Enable 0: Channel is disabled 1: Channel is enabled	RW	0x0

STA Register

Description

FULL1 bit indicates the full status of the FIFO. If this bit is high the FIFO is full.
EMPT1 bit indicates the empty status of the FIFO. If this bit is high the FIFO is empty.
WERR1 bit is set 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 is set 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.
GAPO*i* bit indicates that there has been a gap between transmission of two consecutive data from FIFO. This may happen when the FIFO becomes empty after the state machine has sent a word and is waiting for the next word. If control bit RPTL*i* 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 is set 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.
STA*i* 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.

Table 154. STA Register

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-
10	STA2	Channel 2 State	RO	0x0
9	STA1	Channel 1 State	RO	0x0
8	BERR	Bus Error Flag	W1C	0x0
7:6	Reserved.	-	-	-
5	GAPO2	Channel 2 Gap Occurred Flag	W1C	0x0
4	GAPO1	Channel 1 Gap Occurred Flag	W1C	0x0
3	RERR1	FIFO Read Error Flag	W1C	0x0
2	WERR1	FIFO Write Error Flag	W1C	0x0
1	EMPT1	FIFO Empty Flag	RO	0x1
0	FULL1	FIFO Full Flag	RO	0x0

DMAC Register

Description

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.

Table 155. DMAC Register

Bits	Name	Description	Type	Reset
31	ENAB	DMA Enable 0: DMA disabled 1: DMA enabled	RW	0x0
30:16	Reserved.	-	-	-
15:8	PANIC	DMA Threshold for PANIC signal	RW	0x07
7:0	DREQ	DMA Threshold for DREQ signal	RW	0x07

RNG1, RNG2 Registers

Description

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_RNG*i* is less than 32, only the first PWM_RNG*i* 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.

Table 156. RNG1, RNG2 Registers

Bits	Name	Description	Type	Reset
31:0	PWM_RNG*i*	Channel *i* Range	RW	0x00000020

DAT1, DAT2 Registers

Description

This register stores the 32-bit data to be sent by the PWM Controller when USEF*i* is 0. In PWM mode, data is sent by pulse width modulation: the value of this register defines the number of pulses which are sent within the period defined by PWM_RNG*i*. In serialiser mode, data stored in this register is serialised and transmitted.

Table 157. DAT1, DAT2 Registers

Bits	Name	Description	Type	Reset
31:0	PWM_DAT*i*	Channel *i* Data	RW	0x00000000

FIF1 Register

Description

This register is the FIFO input for both the channels. Data written to this address is stored in the FIFO and if USEF*i* is enabled for 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 the FIFO and both 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 the channel range (period) value of one channel is different to the other, this will cause the channel with the smaller range value 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 the RPTL*i* bits 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 RPTL*i* bit set to zero.

If the set of channels sharing the FIFO has been modified after a configuration change, the FIFO should be cleared before writing new data.

Table 158. FIF1 Register

Bits	Name	Description	Type	Reset
31:0	PWM_FIFO	Channel FIFO Input	WO	0x00000000

Chapter 9. SPI

9.1. Overview

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.

9.2. SPI Master Mode

9.2.1. Standard mode

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

Figure 16. SPI Master Typical Usage

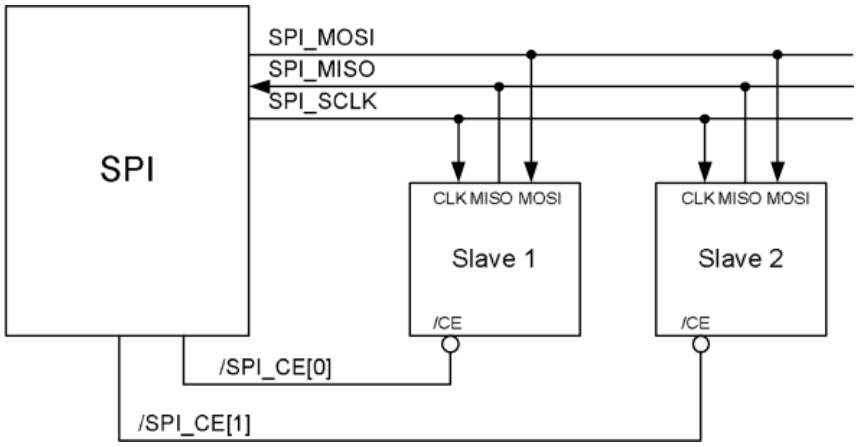
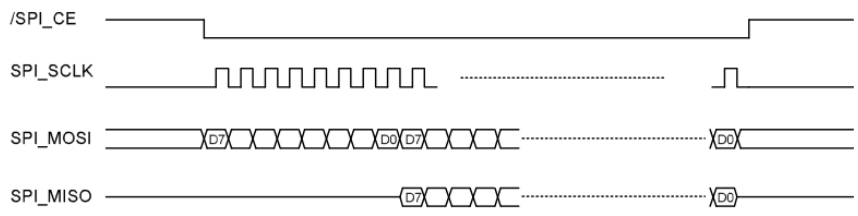
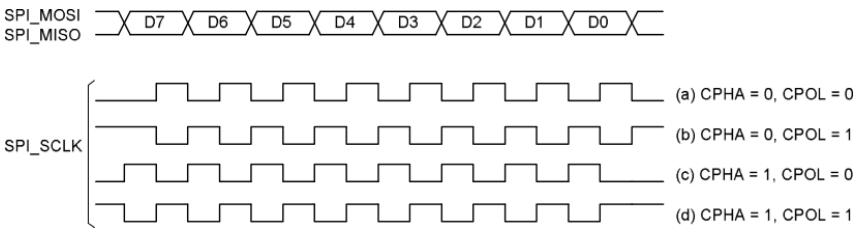


Figure 17. SPI Cycle



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.

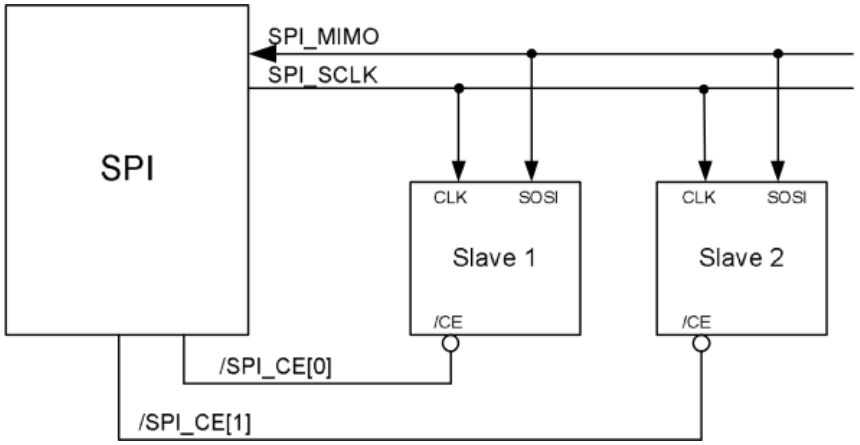
Figure 18. Different Clock Polarity/Phase



9.2.2. Bidirectional mode

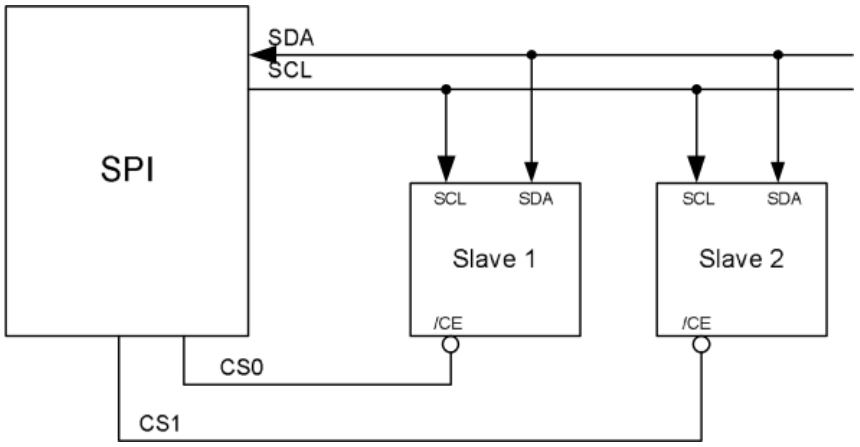
In bidirectional SPI master mode the same SPI standard is implemented except that a single wire is used for the data (MIMO) instead of two as in standard mode (MISO and MOSI). Bidirectional mode is used in a similar way to standard mode, the only difference is that before attempting to read data from 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 19. Bidirectional SPI Master Typical Usage



9.3. LoSSI mode

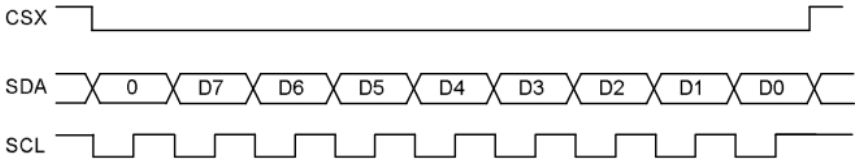
Figure 20. LoSSI mode Typical usage



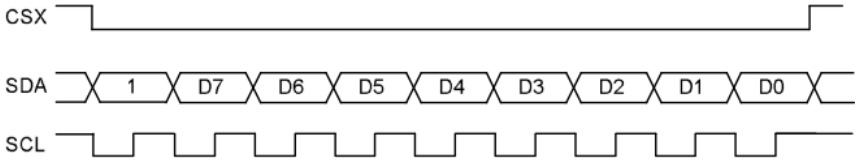
The LoSSI standard allows us to issue commands to peripherals and to transfer data to and from them. LoSSI commands and parameters are 8 bits long, but an extra bit is used to indicate whether the byte is a command or data. This extra bit is set high for a parameter and low for a command. The resulting 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.

9.3.1. Command write

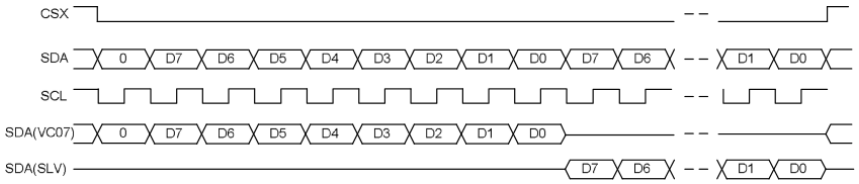


9.3.2. Parameter write



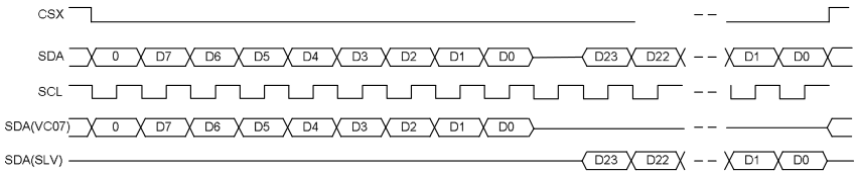
9.3.3. Byte read commands

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



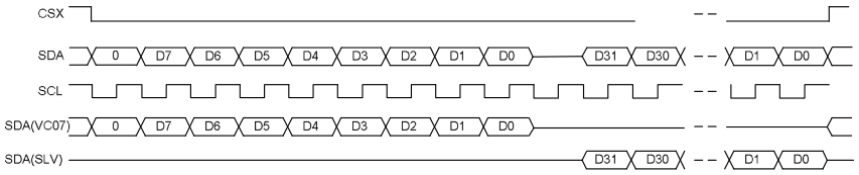
9.3.4. 24-bit read command

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



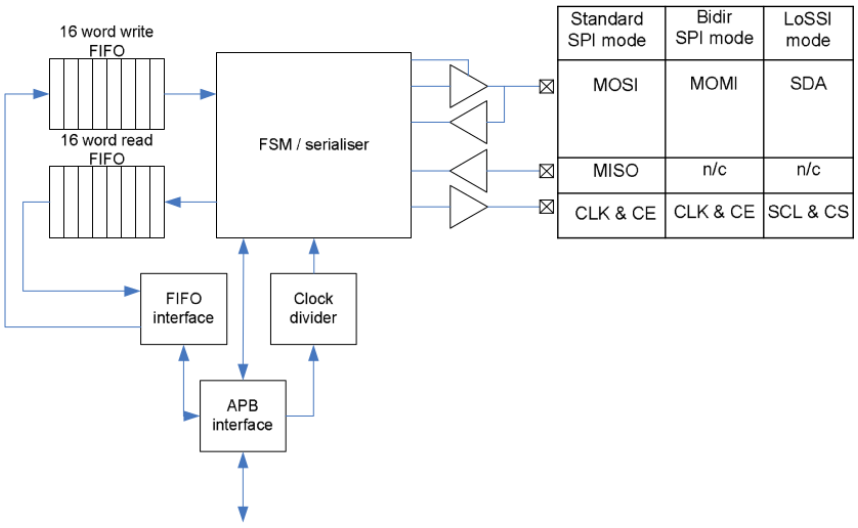
9.3.5. 32-bit read command

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



9.4. Block Diagram

Figure 21. Serial interface Block Diagram



9.5. SPI Register Map

The BCM2711 device has five SPI interfaces of this type: SPI0, SPI3, SPI4, SPI5 & SPI6. It has two additional mini SPI interfaces (SPI1 and SPI2). The specification of those can be found under Section 2.3.

The base addresses of these SPI interfaces are

- SPI0: `0x7e204000`
- SPI3: `0x7e204600`
- SPI4: `0x7e204800`
- SPI5: `0x7e204a00`
- SPI6: `0x7e204c00`

Table 159. SPI Register Map

Offset	Name	Description
0x00	CS	SPI Master Control and Status
0x04	FIFO	SPI Master TX and RX FIFOs
0x08	CLK	SPI Master Clock Divider
0x0c	DLEN	SPI Master Data Length
0x10	LTOH	SPI LoSSI mode TOH
0x14	DC	SPI DMA DREQ Controls

CS Register

Description

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

Table 160. CS Register

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25	LEN_LONG	Enable Long data word in LoSSI mode if DMA_LEN is set 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	Enable DMA mode in LoSSI mode	RW	0x0

Bits	Name	Description	Type	Reset
23	CSPOL2	Chip Select 2 Polarity 0= Chip select is active low. 1= Chip select is active high.	RW	0x0
22	CSPOL1	Chip Select 1 Polarity 0= Chip select is active low. 1= Chip select is active high.	RW	0x0
21	CSPOL0	Chip Select 0 Polarity 0= Chip select is active low. 1= Chip select is active high.	RW	0x0
20	RXF	RX FIFO Full 0 = RX FIFO 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	RX FIFO needs Reading (¾ full) 0 = RX FIFO is less than ¾ full (or not active TA = 0). 1 = RX FIFO is ¾ or more full. Cleared by reading sufficient data from the RX FIFO or setting TA to 0.	RO	0x0
18	TXD	TX FIFO can accept Data 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	RX FIFO contains Data 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 byte.	RO	0x0
16	DONE	Transfer Done 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	Unused	RW	0x0
14	LMONO	Unused	RW	0x0
13	LEN	LoSSI enable 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	Read Enable 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	Automatically De-assert Chip Select 0 = Don't automatically de-assert chip select at the end of a DMA transfer; chip select is manually controlled by software. 1 = Automatically de-assert chip select at the end of a DMA transfer (as determined by SPIDLEN)	RW	0x0

Bits	Name	Description	Type	Reset
10	INTR	Interrupt on RXR 0 = Don't generate interrupts on RX FIFO condition. 1 = Generate interrupt while RXR = 1.	RW	0x0
9	INTD	Interrupt on Done 0 = Don't generate interrupt on transfer complete. 1 = Generate interrupt when DONE = 1.	RW	0x0
8	DMAEN	DMA Enable 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	Transfer Active 0 = Transfer not active. /CS lines are all high (assuming CSPOL = 0). RXR and DONE are 0. Writes to SPI_FIFO write data into bits 15:0 of SPIDLEN and bits 7:0 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 SPI_FIFO write data to TX FIFO. TA is cleared by a dma_frame_end pulse from the DMA controller.	RW	0x0
6	CSPOL	Chip Select Polarity 0 = Chip select lines are active low 1 = Chip select lines are active high	RW	0x0
5:4	CLEAR	FIFO Clear 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.	W1SC	0x0
3	CPOL	Clock Polarity 0 = Rest state of clock = low. 1 = Rest state of clock = high.	RW	0x0
2	CPHA	Clock Phase 0 = First SCLK transition at middle of data bit. 1 = First SCLK transition at beginning of data bit.	RW	0x0
1:0	CS	Chip Select 00 = Chip select 0 01 = Chip select 1 10 = Chip select 2 11 = Reserved	RW	0x0

FIFO Register

Description

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

Table 161. FIFO Register

Bits	Name	Description	Type	Reset
31:0	DATA	DMA Mode (DMAEN set) 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 Poll/Interrupt Mode (DMAEN clear, TA set) Writes to the register write bytes to TX FIFO. Reads from register read bytes from the RX FIFO	RW	0x00000000

CLK Register

Description

This register allows the SPI clock rate to be set.

Table 162. CLK Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	CDIV	Clock Divider SCLK = Core Clock / CDIV If CDIV is set to 0, the divisor is 65536. The divisor must be a multiple of 2. Odd numbers rounded down. The maximum SPI clock rate is of the APB clock.	RW	0x0000

DLEN Register

Description

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

Table 163. DLEN Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	LEN	Data Length 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	0x0000

LTOH Register

Description

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

Table 164. LTOH Register

Bits	Name	Description	Type	Reset
31:4	Reserved.	-	-	-
3:0	TOH	This sets the Output Hold delay in APB clocks. A value of 0 causes a 1 clock delay.	RW	0x1

DC Register

Description

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.

Table 165. DC Register

Bits	Name	Description	Type	Reset
31:24	RPANIC	DMA Read Panic Threshold. Generate the Panic signal to the RX DMA engine whenever the RX FIFO level is greater than this amount.	RW	0x30
23:16	RDREQ	DMA Read Request Threshold. 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 RX FIFO isn't empty).	RW	0x20
15:8	TPANIC	DMA Write Panic Threshold. 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	DMA Write Request Threshold. Generate a DREQ signal to the TX DMA engine whenever the TX FIFO level is less than or equal to this amount.	RW	0x20

9.6. Software Operation

9.6.1. Polled

1. Set CS, CPOL, CPHA as required and set TA = 1
2. Poll TXD writing bytes to SPI_FIFO, RXD reading bytes from SPI_FIFO until all data written
3. Poll DONE until it goes to 1
4. Set TA = 0

9.6.2. Interrupt

1. Set INTR and INTD. These can be left set over multiple operations.
2. Set CS, CPOL, CPHA as required and set TA = 1. This will immediately trigger a first interrupt with DONE = 1.
3. On interrupt:
    - If DONE is set and data to write (this means it is the first interrupt), write up to 64 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.
    - If RXR is set read 48 bytes data from SPI_FIFO and if more data to write, write up to 48 bytes to SPI_FIFO.

9.6.3. 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.

1. Enable DMA DREQs by setting the DMAEN bit and ADCS if required.
2. Program two DMA Control Blocks, one for each DMA controller.
3. DMA channel 1 Control Block should have its PERMAP set to SPI*n* TX and should be set to write 'transfer length' + 1 words to SPI_FIFO. The data should comprise:

a. 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.).

b. 'Transfer length' number in words of data to send.

4. DMA channel 2 Control Block should have its PERMAP set to SPI*n* RX and should be set to read 'transfer length' words from SPI_FIFO.
5. Point each DMA channel at its CB and set its ACTIVE bit to 1.
6. On receipt of an interrupt from DMA channel 2, the transfer is complete.

9.6.4. 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.

Chapter 10. System Timer

10.1. Overview

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`.

10.2. System Timer Registers

Table 166. System Timer Registers

Offset	Name	Description
0x00	CS	System Timer Control/Status
0x04	CLO	System Timer Counter Lower 32 bits
0x08	CHI	System Timer Counter Higher 32 bits
0x0c	C0	System Timer Compare 0
0x10	C1	System Timer Compare 1
0x14	C2	System Timer Compare 2
0x18	C3	System Timer Compare 3

CS Register

Description

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.

Table 167. CS Register

Bits	Name	Description	Type	Reset
31:4	Reserved.	-	-	-
3	M3	System Timer Match 3 0 = No Timer 3 match since last cleared. 1 = Timer 3 match detected.	W1C	0x0
2	M2	System Timer Match 2 0 = No Timer 2 match since last cleared. 1 = Timer 2 match detected.	W1C	0x0
1	M1	System Timer Match 1 0 = No Timer 1 match since last cleared. 1 = Timer 1 match detected.	W1C	0x0

Bits	Name	Description	Type	Reset
0	M0	System Timer Match 0 0 = No Timer 0 match since last cleared. 1 = Timer 0 match detected.	W1C	0x0

CLO Register

Description

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.

Table 168. CLO Register

Bits	Name	Description	Type	Reset
31:0	CNT	Lower 32-bits of the free running counter value.	RO	0x00000000

CHI Register

Description

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.

Table 169. CHI Register

Bits	Name	Description	Type	Reset
31:0	CNT	Higher 32-bits of the free running counter value.	RO	0x00000000

C0, C1, C2, C3 Registers

Description

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.

Table 170. C0, C1, C2, C3 Registers

Bits	Name	Description	Type	Reset
31:0	CMP	Compare value for match channel *n*.	RW	0x00000000

Chapter 11. UART

11.1. Overview

The BCM2711 device has six UARTs. One mini UART (UART1) and five PL011 UARTs (UART0, UART2, UART3, UART4 & UART5). This section describes the PL011 UARTs. For details of the mini UART see Section 2.2.

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)

The UARTs provide:

- Separate 32x8 transmit and 32x12 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, 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.

11.2. 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

11.3. Primary UART Inputs and Outputs

The UARTs have 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 how the various UART signals (including the mini UART) map on the General Purpose I/O (GPIO). For more details on how to select alternate functions refer to Chapter 5.

Table 171. UART Assignment on the GPIO Pin map

	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	High					TXD2	
GPIO1	High					RXD2	
GPIO2	High					CTS2	
GPIO3	High					RTS2	
GPIO4	High					TXD3	
GPIO5	High					RXD3	
GPIO6	High					CTS3	
GPIO7	High					RTS3	
GPIO8	High					TXD4	
GPIO9	High					RXD4	
GPIO10	High					CTS4	
GPIO11	High					RTS4	
GPIO12	High					TXD5	
GPIO13	High					RXD5	
GPIO14	Low	TXD0				CTS5	TXD1
GPIO15	Low	RXD0				RTS5	RXD1
GPIO16	Low				CTS0		CTS1
GPIO17	Low				RTS0		RTS1
GPIO30	Low				CTS0		CTS1
GPIO31	Low				RTS0		RTS1
GPIO32	Low				TXD0		TXD1
GPIO33	Low				RXD0		RXD1
GPIO36	High			TXD0			
GPIO37	Low			RXD0			
GPIO38	Low			RTS0			
GPIO39	Low			CTS0			
GPIO40	Low						TXD1
GPIO41	Low						RXD1

	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO42	Low						RTS1
GPIO43	Low						CTS1

11.4. UART Interrupts

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

UARTINTR, this is an OR function of the five individual masked outputs:

- UARTRXINTR
- UARTTXINTR
- 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:
  - 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.

UARTTXINTR

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 transmitter's 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.

UARTRXINTR

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.

11.5. Register View

The PL011 UARTs are mapped onto the following base addresses:

- UART0: 0x7e201000
- UART2: 0x7e201400
- UART3: 0x7e201600
- UART4: 0x7e201800
- UART5: 0x7e201a00

They have the following memory-mapped registers.

Table 172. UART Registers

Offset	Name	Description
0x00	DR	Data Register
0x04	RSRECR	
0x18	FR	Flag register
0x20	ILPR	not in use
0x24	IBRD	Integer Baud rate divisor
0x28	FBRD	Fractional Baud rate divisor
0x2c	LCRH	Line Control register
0x30	CR	Control register
0x34	IFLS	Interrupt FIFO Level Select Register
0x38	IMSC	Interrupt Mask Set Clear Register
0x3c	RIS	Raw Interrupt Status Register
0x40	MIS	Masked Interrupt Status Register
0x44	ICR	Interrupt Clear Register
0x48	DMACR	DMA Control Register
0x80	ITCR	Test Control register
0x84	ITIP	Integration test input reg
0x88	ITOP	Integration test output reg
0x8c	TDR	Test Data reg

DR Register

Description

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).

Table 173. DR Register

Bits	Name	Description	Type	Reset
31:12	Reserved.	-	-	-
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.	RO	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.	RO	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.	RO	0x0
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.	RO	0x0
7:0	DATA	Receive (read) data character. Transmit (write) data character.	RW	0x00

RSRECR Register

Description

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, before reading the error status associated with that data character from this register.

Table 174. RSRECR Register

Bits	Name	Description	Type	Reset
31:4	Reserved.	-	-	-
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

Bits	Name	Description	Type	Reset
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

Description

The UART_FR Register is the flag register.

Table 175. FR Register

Bits	Name	Description	Type	Reset
31:9	Reserved.	-	-	-
8	RI	Unsupported, write zero, read as don't care	RO	0x0
7	TXFE	Transmit FIFO empty. The meaning of this bit depends on the state of the FEN bit in the Line Control Register, UART_LCRH. If the FIFO is disabled, this bit is set when the transmit holding register is empty. 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.	RO	0x1
6	RXFF	Receive FIFO full. The meaning of this bit depends on the state of the FEN bit in the UART_LCRH Register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.	RO	0x0
5	TXFF	Transmit FIFO full. The meaning of this bit depends on the state of the FEN bit in the UART_LCRH Register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.	RO	0x0

Bits	Name	Description	Type	Reset
4	RXFE	Receive FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UART_LCRH Register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.	RO	0x0
3	BUSY	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. This bit is set as soon as the transmit FIFO becomes non-empty, regardless of whether the UART is enabled or not.	RO	0x0
2	DCD	Unsupported, write zero, read as don't care	RO	0x0
1	DSR	Unsupported, write zero, read as don't care	RO	0x0
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.	RO	0x0

ILPR Register

Description

This is the disabled IrDA register, writing to it has no effect and reading returns 0.

Table 176. ILPR Register

Bits	Name	Description	Type	Reset
31:0	Reserved.	-	-	-

IBRD Register

Description

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

Table 177. IBRD Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15:0	IBRD	The integer baud rate divisor.	RW	0x0000

FBRD Register

Description

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.

Table 178. FBRD Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-
5:0	FBRD	The fractional baud rate divisor.	RW	0x00

LCRH Register

Description

The UART_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.

Table 179. LCRH Register

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
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 180.	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 180.	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 180.	RW	0x0

Bits	Name	Description	Type	Reset
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

Table 180. UART parity bits

PEN	EPS	SPS	Parity bit (transmitted or checked)
0	x	x	Not transmitted or checked
1	1	0	Even parity
1	0	0	Odd parity
1	0	1	1
1	1	1	0

CR Register

Description

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.

Table 181. CR Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
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

Bits	Name	Description	Type	Reset
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
6:3	Reserved.	-	-	-
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	UART enable: 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

Description

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.

Table 182. IFLS Register

Bits	Name	Description	Type	Reset
31:12	Reserved.	-	-	-
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	0x2
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	0x2

IMSC Register

Description

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.

Table 183. IMSC Register

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-
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
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

Description

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.

Table 184. RIS Register

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-

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

MIS Register

Description

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.

Table 185. MIS Register

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-
10	OEMIS	Overrun error masked interrupt status. Returns the masked interrupt state of the UARTOEINTR interrupt.	RO	0x0
9	BEMIS	Break error masked interrupt status. Returns the masked interrupt state of the UARTBEINTR interrupt.	RO	0x0
8	PEMIS	Parity error masked interrupt status. Returns the masked interrupt state of the UARTPEINTR interrupt.	RO	0x0
7	FEMIS	Framing error masked interrupt status. Returns the masked interrupt state of the UARTFEINTR interrupt.	RO	0x0
6	RTMIS	Receive timeout masked interrupt status. Returns the masked interrupt state of the UARTRTINTR interrupt.	RO	0x0
5	TXMIS	Transmit masked interrupt status. Returns the masked interrupt state of the UARTTXINTR interrupt.	RO	0x0
4	RXMIS	Receive masked interrupt status. Returns the masked interrupt state of the UARTRXINTR interrupt.	RO	0x0

Bits	Name	Description	Type	Reset
3	DSRMMIS	Unsupported, write zero, read as don't care	RO	0x0
2	DCDMMIS	Unsupported, write zero, read as don't care	RO	0x0
1	CTSMMIS	nUARTCTS modem masked interrupt status. Returns the masked interrupt state of the UARTCTSINTR interrupt.	RO	0x0
0	RIMMIS	Unsupported, write zero, read as don't care	RO	0x0

ICR Register

Description

The UART_ICR Register is the interrupt clear register.

Table 186. ICR Register

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-
10	OEIC	Overrun error interrupt clear. Clears the UARTOEINTR interrupt.	WO	0x0
9	BEIC	Break error interrupt clear. Clears the UARTBEINTR interrupt.	WO	0x0
8	PEIC	Parity error interrupt clear. Clears the UARTPEINTR interrupt.	WO	0x0
7	FEIC	Framing error interrupt clear. Clears the UARTFEINTR interrupt.	WO	0x0
6	RTIC	Receive timeout interrupt clear. Clears the UARTRTINTR interrupt.	WO	0x0
5	TXIC	Transmit interrupt clear. Clears the UARTTXINTR interrupt.	WO	0x0
4	RXIC	Receive interrupt clear. Clears the UARTRXINTR interrupt.	WO	0x0
3	DSRMIC	Unsupported, write zero, read as don't care	WO	0x0
2	DCDMIC	Unsupported, write zero, read as don't care	WO	0x0
1	CTSMIC	nUARTCTS modem interrupt clear. Clears the UARTCTSINTR interrupt.	WO	0x0
0	RIMIC	Unsupported, write zero, read as don't care	WO	0x0

DMACR Register

Description

The UART_DMACR Register is the DMA control register.

Table 187. DMACR Register

Bits	Name	Description	Type	Reset
31:3	Reserved.	-	-	-
2	DMAONERR	DMA on error. If this bit is set to 1, the DMA receive request outputs are disabled when the UART error interrupt is asserted.	RW	0x0
1	TXDMAE	Transmit DMA enable. If this bit is set to 1, DMA for the transmit FIFO is enabled.	RW	0x0

Bits	Name	Description	Type	Reset
0	RXDMAE	Receive DMA enable. If this bit is set to 1, DMA for the receive FIFO is enabled.	RW	0x0

ITCR Register

Description

This is the Test Control Register UART_ITCR.

Table 188. ITCR Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	ITCR1	Test FIFO enable. When this bit it 1, a write to the Test Data Register, UART_TDR writes data into the receive FIFO, and reading from the UART_TDR 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

ITIP Register

Description

This is the Test Control Register UART_ITIP.

Table 189. ITIP Register

Bits	Name	Description	Type	Reset
31:4	Reserved.	-	-	-
3	ITIP3	Reads return the value of the nUARTCTS primary input.	RW	0x0
2:1	Reserved.	-	-	-
0	ITIP0	Reads return the value of the UARTRXD primary input.	RW	0x0

ITOP Register

Description

This is the Test Control Register UART_ITOP.

Table 190. ITOP Register

Bits	Name	Description	Type	Reset
31:12	Reserved.	-	-	-
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

Bits	Name	Description	Type	Reset
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
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	ITOP6	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.	-	-	-
3	ITOP3	Primary output. Writes specify the value to be driven on nUARTRTS.	RW	0x0
2:1	Reserved.	-	-	-
0	ITOP0	Primary output. Writes specify the value to be driven on UARTTXD.	RW	0x0

TDR Register

Description

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.

Table 191. TDR Register

Bits	Name	Description	Type	Reset
31:11	Reserved.	-	-	-
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	0x000

Chapter 12. Timer (ARM side)

12.1. Overview

The ARM Timer is ***based*** on a ARM SP804, 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.

12.2. Timer Registers

The base address for the ARM timer register is `0x7e00b000`.

Table 192. Timer Registers

Offset	Name	Description
0x400	LOAD	Load
0x404	VALUE	Value (Read-Only)
0x408	CONTROL	Control
0x40c	IRQCNTL	IRQ Clear/Ack (Write-Only)
0x410	RAWIRQ	RAW IRQ (Read-Only)
0x414	MSKIRQ	Masked IRQ (Read-Only)
0x418	RELOAD	Reload
0x41c	PREDIV	Pre-divider (Not in real 804!)
0x420	FREECNT	Free running counter (Not in real 804!)

LOAD Register

Description

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.

Table 193. Timer Load register

Bits	Name	Description	Type	Reset
31:0	LOAD	Timer load value.	RW	0x00000000

VALUE Register

Description

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.

Table 194. Timer Value register

Bits	Name	Description	Type	Reset
31:0	VALUE	Current timer value.	RO	0x00000000

CONTROL Register

Description

The standard SP804 timer control register consists of 8 bits but in the BCM2711 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 BCM2711 implementation are shown in italics.

Table 195. Timer control register

Bits	Name	Description	Type	Reset
31:24	Reserved.	-	-	-
23:16	FREEDIV	*Free running counter pre-scaler. Freq is sys_clk/(prescale+1)* ***These bits do not exist in a standard 804 timer!***	RW	0x3e
15:10	Reserved.	-	-	-
9	ENAFREE	*0 : Free running counter Disabled* *1 : Free running counter Enabled* ***This bit does not exist in a standard 804 timer!***	RW	0x0
8	DBGHALT	*0 : Timers keeps running if ARM is in debug halted mode* *1 : Timers halted if ARM is in debug halted mode* ***This bit does not exist in a standard 804 timer!***	RW	0x0
7	ENABLE	0 : Timer disabled 1 : Timer enabled	RW	0x0
6	Reserved.	-	-	-
5	IE	0 : Timer interrupt disabled 1 : Timer interrupt enabled	RW	0x1
4	Reserved.	-	-	-
3:2	DIV	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 *(Undefined in 804)*	RW	0x0
1	32BIT	0 : 16-bit counters 1 : 32-bit counter	RW	0x0
0	Reserved.	-	-	-

IRQCNTL Register

Description

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".

Table 196. Timer IRQ clear register

Bits	Name	Description	Type	Reset
31:1	Reserved.	-	-	-
0	INT	Write a 1 to clear the interrupt	WO	0x0

RAWIRQ Register

Description

The raw IRQ register is a read-only register. It shows the status of the interrupt pending bit.

Table 197. Timer Raw IRQ register

Bits	Name	Description	Type	Reset
31:1	Reserved.	-	-	-
0	INT	0 : The interrupt pending bit is clear 1 : The interrupt pending bit is set.	RO	0x0

The interrupt pending bit is set each time the value register is counted down to zero. The interrupt pending bit can not by itself generate interrupts. Interrupts can only be generated if the interrupt enable bit is set.

MSKIRQ Register

Description

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.

Table 198. Timer Masked IRQ register

Bits	Name	Description	Type	Reset
31:1	Reserved.	-	-	-
0	INT	0 : Interrupt line not asserted. 1 :Interrupt line is asserted, (the interrupt pending and the interrupt enable bit are set.)	RO	0x0

RELOAD Register

Description

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.

Table 199. Timer Reload register

Bits	Name	Description	Type	Reset
31:0	LOAD	Timer load value.	RW	0x00000000

PREDIV Register

Table 200. Timer pre-divider register

Bits	Name	Description	Type	Reset
31:10	Reserved.	-	-	-
9:0	PREDIV	Pre-divider value.	RW	0x07d

The Pre-divider register is not present in the SP804.

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:

$$timer\_clock = \frac{apb\_clock}{pre\_divider + 1}$$

The reset value of this register is 0x7D so gives a divide by 126.

FREECNT Register

Table 201. Free running counter

Bits	Name	Description	Type	Reset
31:0	FREECNT	Counter value	RO	0x00000000

The free running counter is not present in the SP804.

The free running counter is a 32-bit 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.

Chapter 13. ARM Mailboxes

13.1. Overview

There are 16 ARM Mailboxes which can be used to send messages or signals between the ARM cores. Each mailbox is a 32-bit wide value with separate write-set and write-clear registers (see Section 6.5 for more information about write-set / write-clear registers), for a total of 32 registers.

NOTE

The ARM Mailboxes described here (in the ARM_LOCAL block) are distinct from the VPU Mailboxes (in the ARMC block).

There are no differences between any of the ARM mailboxes, so it is left to the programmer to decide how to use them. Mailbox bits can be set by writing to the appropriate MBOX_SET register. Each mailbox generates an interrupt whenever any of its bits are non-zero - refer to Chapter 6 for details on how these interrupts are routed. The mailbox's value can be read from the appropriate MBOX_CLR register, and mailbox bits can be cleared by writing to the appropriate MBOX_CLR register (these last two steps would typically be performed inside the relevant ARM core's interrupt handler).

13.2. Registers

The ARM_LOCAL register base address is `0x4c0000000`. Note that, unlike other peripheral addresses in this document, this is an ARM-only address and not a legacy master address. If Low Peripheral mode is enabled this base address becomes `0xff800000`.

The write-set registers (MBOX_SET) are write-only, but the write-clear registers (MBOX_CLR) are read-write.

Table 202. ARM Mailbox registers

Offset	Name	Description
0x80	MBOX_SET00	Mailbox 00 Set Bit Register
0x84	MBOX_SET01	Mailbox 01 Set Bit Register
0x88	MBOX_SET02	Mailbox 02 Set Bit Register
0x8c	MBOX_SET03	Mailbox 03 Set Bit Register
0x90	MBOX_SET04	Mailbox 04 Set Bit Register
0x94	MBOX_SET05	Mailbox 05 Set Bit Register
0x98	MBOX_SET06	Mailbox 06 Set Bit Register
0x9c	MBOX_SET07	Mailbox 07 Set Bit Register
0xa0	MBOX_SET08	Mailbox 08 Set Bit Register
0xa4	MBOX_SET09	Mailbox 09 Set Bit Register
0xa8	MBOX_SET10	Mailbox 10 Set Bit Register
0xac	MBOX_SET11	Mailbox 11 Set Bit Register
0xb0	MBOX_SET12	Mailbox 12 Set Bit Register
0xb4	MBOX_SET13	Mailbox 13 Set Bit Register
0xb8	MBOX_SET14	Mailbox 14 Set Bit Register

Offset	Name	Description
0xbc	MBOX_SET15	Mailbox 15 Set Bit Register
0xc0	MBOX_CLR00	Mailbox 00 Clear Bit Register
0xc4	MBOX_CLR01	Mailbox 01 Clear Bit Register
0xc8	MBOX_CLR02	Mailbox 02 Clear Bit Register
0xcc	MBOX_CLR03	Mailbox 03 Clear Bit Register
0xd0	MBOX_CLR04	Mailbox 04 Clear Bit Register
0xd4	MBOX_CLR05	Mailbox 05 Clear Bit Register
0xd8	MBOX_CLR06	Mailbox 06 Clear Bit Register
0xdc	MBOX_CLR07	Mailbox 07 Clear Bit Register
0xe0	MBOX_CLR08	Mailbox 08 Clear Bit Register
0xe4	MBOX_CLR09	Mailbox 09 Clear Bit Register
0xe8	MBOX_CLR10	Mailbox 10 Clear Bit Register
0xec	MBOX_CLR11	Mailbox 11 Clear Bit Register
0xf0	MBOX_CLR12	Mailbox 12 Clear Bit Register
0xf4	MBOX_CLR13	Mailbox 13 Clear Bit Register
0xf8	MBOX_CLR14	Mailbox 14 Clear Bit Register
0xfc	MBOX_CLR15	Mailbox 15 Clear Bit Register

MBOX_SET00, MBOX_SET01, ..., MBOX_SET14, MBOX_SET15 Registers

Description

Writing a '1' to a bit position in this register causes the corresponding bit in the mailbox word to be set to 1.
There are 16 mailboxes in total, four per ARM core. Mailboxes 4C to 4C+3 'belong' to core number C.
Each mailbox may raise an interrupt to its core when any bits in the 32-bit word are set to '1'.

Table 203. MBOX_SET00, MBOX_SET01, ..., MBOX_SET14, MBOX_SET15 Registers

Bits	Name	Description	Type	Reset
31:00	MBOX_DATA	32-bit mailbox word	WO	0x00000000

MBOX_CLR00, MBOX_CLR01, ..., MBOX_CLR14, MBOX_CLR15 Registers

Description

Writing a '1' to a bit position in this register causes the corresponding bit in the mailbox word to be cleared to 0. A read returns the current state of the mailbox word.
There are 16 mailboxes in total, four per ARM core. Mailboxes 4C to 4C+3 'belong' to core number C.
Each mailbox may raise an interrupt to its core when any bits in the 32-bit word are set to '1'.

Table 204. MBOX_CLR00, MBOX_CLR01, ..., MBOX_CLR14, MBOX_CLR15 Registers

Bits	Name	Description	Type	Reset
31:00	MBOX_DATA	32-bit mailbox word	W1C	0x00000000

Raspberry Pi is a trademark of Raspberry Pi Ltd

Raspberry Pi Ltd