RP1 Peripherals

DRAFT

Colophon

Copyright © 2023 Raspberry Pi Ltd

This documentation is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International (CC BY-ND).

Portions Copyright © 2019-2023 Synopsys, Inc.

All rights reserved. Used with permission. Synopsys & DesignWare are registered trademarks of Synopsys, Inc.

Portions Copyright © 2000-2001, 2005, 2007, 2009, 2011-2012, 2016 ARM Limited.

All rights reserved. Used with permission.

Portions Copyright © 2004—2016 Cadence Design Systems, Inc.

All rights reserved. Used with permission.

build-date: 2023-11-07
build-version: b9b6f74-clean

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 of 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.

DRAFT

Table of contents

Colophon . . . 1
Legal disclaimer notice . . . 1
1. Introduction . . . 4
2. System description . . . 6
2.1. Interconnect details . . . 7
2.2. Address map . . . 7
2.3. Peripheral address map details . . . 8
2.3.1. PCIe and 40-bit to peripheral address mapping . . . 8
2.4. Atomic register access . . . 11
2.5. Clocks . . . 11
2.5.1. Crystal and PLLs . . . 11
2.5.2. Core Clock . . . 12
2.5.3. RGMII clocks . . . 12
2.5.4. Audio clocks . . . 12
2.5.5. Other Peripheral clocks . . . 13
2.5.6. Clock controls . . . 13
3. Low speed peripherals . . . 14
3.1. GPIO . . . 14
3.1.1. Function select . . . 14
3.1.2. Interrupts . . . 17
3.1.3. Pads . . . 17
3.1.4. Registers . . . 18
3.2. UART . . . 34
3.2.1. Register base addresses . . . 34
3.3. RIO . . . 34
3.3.1. Latency considerations . . . 35
3.3.2. Register base addresses . . . 35
3.4. PWM . . . 35
3.4.1. Overview . . . 36
3.4.2. Features . . . 36
3.4.3. Principle of operation . . . 36
3.4.4. Output modes . . . 36
3.4.5. List of registers . . . 38
3.5. I2C . . . 48
3.5.1. Features . . . 48
3.5.2. Register base addresses . . . 49
3.6. SPI . . . 50
3.6.1. Instance configurations . . . 50
3.6.2. Register base addresses . . . 51
3.7. I2S . . . 51
3.7.1. Instance configurations . . . 51
3.7.2. Register base addresses . . . 52
3.8. TICKS . . . 52
3.8.1. List of registers . . . 52
4. SDIO . . . 60
4.1. Configuration and feature set . . . 60
4.2. Supported speed modes . . . 60
4.3. Register base addresses . . . 61
5. USB . . . 62
5.1. Register base addresses . . . 62
6. PCI Express endpoint controller . . . 63
6.1. PCIe endpoint configuration registers . . . 63
6.2. MSIx configuration registers . . . 79
7. Ethernet . . . 81
7.1. Registers . . . 82

DRAFT

8. Displays and cameras . . . 88
8.1. DSI host . . . 88
8.2. CSI-2 host . . . 88
8.3. Registers . . . 89
9. DMA . . . 90
9.1. Register base addresses . . . 90
Appendix A: Documentation release history . . . 91

DRAFT

Chapter 1. Introduction

RP1 is a peripheral controller, designed by Raspberry Pi for use on Raspberry Pi 5. It connects to an application processor (AP) via a PCIe 2.0 x4 bus, and aggregates many digital controllers and analog PHYs for Raspberry Pi 5's external interfaces. In the case of Raspberry Pi 5, the AP is the 16nm Broadcom BCM2712.

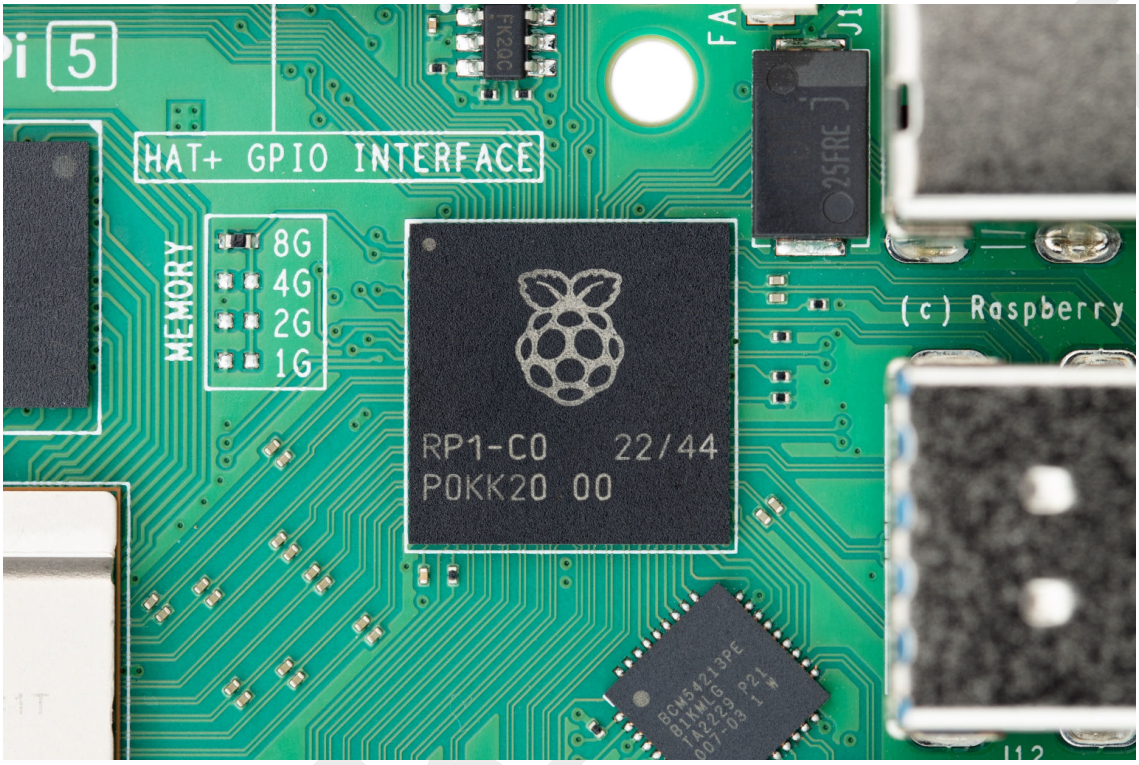


Figure 1. Raspberry Pi RP1 Southbridge

Raspberry Pi single-board computers, with the exception of Raspberry Pi Zero, have historically adopted a two-chip architecture. The chipset comprises an AP, which provides the core high-speed digital functionality and a subset of the platform interfaces; and a peripheral controller, which provides any additional required interfaces.

In the case of Raspberry Pi 1, 1+, 2, 3 and 3+, the peripheral controller provides Ethernet and downstream USB 2.0 ports, and is connected to the AP via USB 2.0. In the case of Raspberry Pi 4, it provides downstream USB 3.0 and USB 2.0 ports, and is connected to the AP via PCI Express.

On Raspberry Pi 5, RP1 provides the following key external interfaces:

- USB. Two independent XHCI controllers are each connected to a single USB 3.0 PHY, and a single USB 2.0 PHY. Together, they support more than 10Gbps of downstream USB traffic.
- MIPI camera and display. Two MIPI CSI-2 camera controllers, and two MIPI DSI display controllers, are connected to two shared 4-lane MIPI DPHY transceiver PHYs. Together, they support 8Gbps of downstream traffic, to two cameras, two displays, or one camera and one display. Each camera controller incorporates an image signal processor front-end (ISP-FE) which pre-processes incoming image data.
- Gigabit Ethernet. An integrated media access controller (MAC) drives an external Gigabit PHY over an RGMII bus.
- General-purpose I/O (GPIO). Twenty-eight GPIO pins are provided, to implement the standard Raspberry Pi 40-pin GPIO connector.
- Low-speed peripherals. GPIO alternate functions are provided that have compatibility with the feature set offered on Raspberry Pi 4 Model B.

The internal fabric allows prioritisation of real-time camera and display traffic over non-real-time USB and Ethernet traffic. QoS signalling over the PCI Express link supports dynamic prioritisation between traffic from RP1, and traffic from real-time and non-real-time bus masters within the AP.

RP1 has several additional features to maximise use-case flexibility:

- DMAC. An eight-channel DMA Controller for servicing low-speed peripherals.
- PLLs. Three integrated PLLs consisting of 2 fractional-N PLLs to support generation of independent video and audio clocks, and 1 integer PLL to generate system clocks.
- ADC. A five-input successive-approximation analogue-to-digital converter with 12-bit (ENOB of 9.5-bit) resolution at 500kSPS. 4 external inputs, one internal temperature sensor
- Shared SRAM. 64kB of general-purpose RAM that the AP or RP1 bus masters can access.
- Timebase generators. Configurable 'ticks' for pacing DMA, or for debouncing GPIO events.

Moving the bulk of the platform interfaces to RP1 simplifies the design and reduces the cost of the AP, and makes it easier to migrate to newer, more advanced process nodes.

DRAFT

Chapter 2. System description

RP1 is shown in Figure 2.

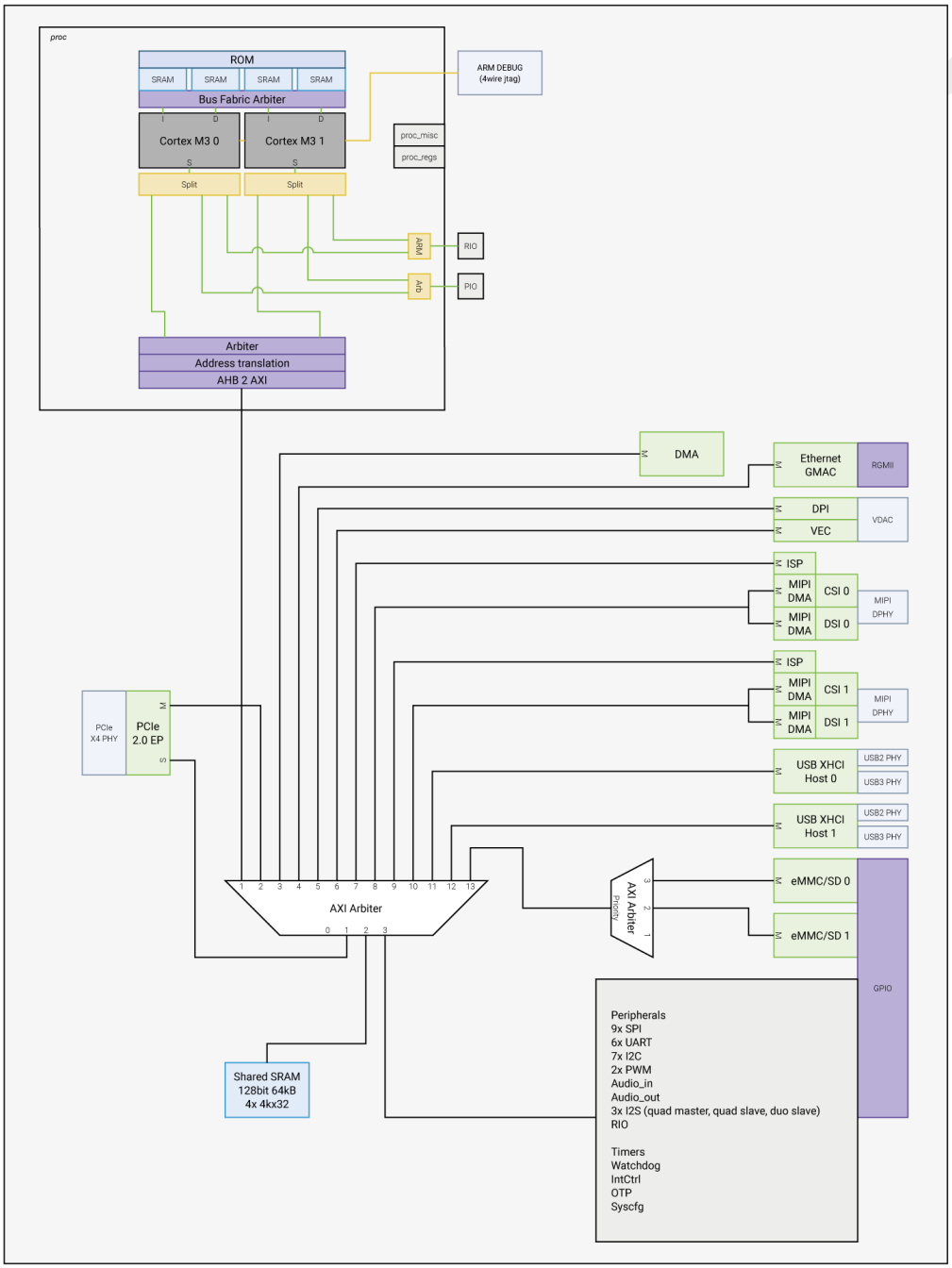


Figure 2. System diagram of RP1.

Primary connectivity to the AP is via a PCIe 2.0 X4 bus. The PCIe Endpoint (EP) controller connects to the AXI busfabric as a bus master and a bus slave, providing access from the AP to RP1, and from RP1 to AP host memory.

A local dual-core ARM Cortex M3 processor cluster is used to perform platform configuration and management. The processor cluster has access to both peripheral registers and a steerable window into AP host memory. The cluster also has per core local tightly-coupled memories and a shared boot ROM.

A 64kB shared SRAM is instantiated for general-purpose use.

All bus masters can access the shared SRAM, and the PCIe slave port (to access host memory on the AP). The PCIe EP

and local processors can also access peripheral control and configuration registers.

2.1. Interconnect details

The interconnect's maximum burst length (16 beats) and data bus width (128b) have been chosen to correspond to the PCIe endpoint's Maximum Packet Size (256B), therefore naturally-aligned bursts from bus masters are not decomposed into multiple upstream TLPs. The endpoint controller may split downstream TLPs into multiple AXI transfers as required, but if downstream bursts are naturally-aligned then decomposition is avoided.

- Certain bus masters are AXI4 compliant. For these master ports there are integrated shims that perform transparent AXI4 to AXI3 protocol conversion.
- SDIO0 uses a 64-bit data bus, so generate TLPs that are up to half the Maximum Packet Size.
- Bus masters typically have implementation-specific means of limiting the type and length of burst traffic that they can produce, independently of the endpoint configuration.
- There are no PCIe data link-layer semantics that allow for interpretation of WRAP or FIXED type AXI transactions, therefore all upstream transactions of this type are decomposed, and there is no assembly of back-to-back downstream reads or writes to the same address into a FIXED transaction.

Behaviour of the various bus masters can be monitored or constrained by shims instantiated on each bus master port - these shims allow for throttling based on number of pending transactions, ensuring AXI and PCIe address boundary requirements are met, and have statistics counters for Quality-of-Service or performance measurement.

2.2. Address map

A summary of System and Processor view of address space is in Table 1.

NOTE

The processors have a 32-bit address view whereas the PCIe Host and other bus masters have a 40-bit address view.

NOTE

Ports prefixed with `Proc` are accessible only to the management processor complex. Bus masters other than the DMA controller cannot access peripheral registers, but can access shared SRAM.

Table 1. Address Map summary

System Addr (40-bit) Base	Size	Proc Addr (32-bit) Base	Description
0x00.0000.0000	512G	n/a	PCIe Outbound direct mapped space
0x80.0000.0000	256G	n/a	PCIe Outbound ATU space
n/a	64k	0x0000.0000	Proc ROM (shared for both procs)
n/a	8k	0x1000.0000	Proc Local ISRAM (local to each proc)
n/a	8k	0x1000.2000	Proc Local DSRAM (local to each proc)
0xC0.2000.0000	256k	0x2000.0000	Shared SRAM (64k)
0xC0.4000.0000	256k	0x4000.0000	Periphs (APB0) 16x16k
0xC0.4004.0000	256k	0x4004.0000	Periphs (APB1) 16x16k

DRAFT

0xC0.4008.0000	256k	0x4008.0000	Periphs (APB2) 16x16k
0xC0.400c.0000	256k	0x400c.0000	Periphs (APB3) 16x16k
0xC0.4010.0000	256k	0x4010.0000	Periphs (APB4) 16x16k
0xC0.4014.0000	256k	0x4014.0000	Periphs (APB5) 16x16k
0xC0.4018.0000	64k	0x4018.0000	Periphs (AHB) 4x16k
0xC0.4020.0000	4M	0x4040.0000	Periphs (AXI) 4x1M
n/a	1G	0x8000.0000	Proc PCIe Outbound ATU window
n/a	256k	0xE000.0000	Proc PPB (Internal to each proc)
n/a	768k	0xE040.0000	Proc PPB (External to each proc)
n/a	64k	0xF000.0000	Proc AHB peripherals

2.3. Peripheral address map details

The Peripheral Address Map lists the peripheral decode ports on the interconnect, their bus type, and whether they have atomic accesses.

2.3.1. PCIe and 40-bit to peripheral address mapping

The addresses given in Table 2 are compatible with the processor cluster's 32-bit view of the system address map. 40-bit bus masters internal to the chip (i.e. DMA) must use the System Addr Bases in Table 1.

The AP accesses Peripherals and Shared SRAM over PCIe as offsets from the assigned base addresses in BAR1 and BAR2 respectively.

Table 2. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
sysinfo	APB	Y	0x40000000
syscfg	APB	Y	0x40008000
otp	APB	Y	0x4000c000
power	APB	Y	0x40010000
resets	APB	Y	0x40014000
clocks_main	APB	Y	0x40018000
clocks_video	APB	Y	0x4001c000
pll_sys	APB	Y	0x40020000
pll_audio	APB	Y	0x40024000
pll_video	APB	Y	0x40028000
uart0	APB	N	0x40030000
uart1	APB	N	0x40034000
uart2	APB	N	0x40038000
uart3	APB	N	0x4003c000
uart4	APB	N	0x40040000

uart5	APB	N	0x40044000
spi8	APB	N	0x4004c000
spi0	APB	N	0x40050000
spi1	APB	N	0x40054000
spi2	APB	N	0x40058000
spi3	APB	N	0x4005c000
spi4	APB	N	0x40060000
spi5	APB	N	0x40064000
spi6	APB	N	0x40068000
spi7	APB	N	0x4006c000
i2c0	APB	N	0x40070000
i2c1	APB	N	0x40074000
i2c2	APB	N	0x40078000
i2c3	APB	N	0x4007c000
i2c4	APB	N	0x40080000
i2c5	APB	N	0x40084000
i2c6	APB	N	0x40088000
audio_in	APB	Y	0x40090000
audio_out	APB	Y	0x40094000
pwm0	APB	Y	0x40098000
pwm1	APB	Y	0x4009c000
i2s0	APB	N	0x400a0000
i2s1	APB	N	0x400a4000
i2s2	APB	N	0x400a8000
timer	APB	Y	0x400ac000
sdio0_cfg	APB	Y	0x400b0000
sdio1_cfg	APB	Y	0x400b4000
busfabric_monitor	APB	Y	0x400c0000
busfabric_axishim	APB	Y	0x400c4000
adc	APB	Y	0x400c8000
io_bank0	APB	Y	0x400d0000
io_bank1	APB	Y	0x400d4000
io_bank2	APB	Y	0x400d8000
sys_rio0	APB	Y	0x400e0000
sys_rio1	APB	Y	0x400e4000
sys_rio2	APB	Y	0x400e8000
pads_bank0	APB	Y	0x400f0000

pads_bank1	APB	Y	0x400f4000
pads_bank2	APB	Y	0x400f8000
pads_eth	APB	Y	0x400fc000
eth	APB	N	0x40100000
eth_cfg	APB	Y	0x40104000
pcie	APB	Y	0x40108000
mipi0_csidma	APB	N	0x40110000
mipi0_csihost	APB	N	0x40114000
mipi0_dsidma	APB	N	0x40118000
mipi0_dsihost	APB	N	0x4011c000
mipi0_cfg	APB	Y	0x40120000
mipi0_isp	APB	N	0x40124000
mipi1_csidma	APB	N	0x40128000
mipi1_csihost	APB	N	0x4012c000
mipi1_dsidma	APB	N	0x40130000
mipi1_dsihost	APB	N	0x40134000
mipi1_cfg	APB	Y	0x40138000
mipi1_isp	APB	N	0x4013c000
video_out_cfg	APB	Y	0x40140000
video_out_vec	APB	Y	0x40144000
video_out_dpi	APB	Y	0x40148000
xosc	APB	Y	0x40150000
watchdog	APB	Y	0x40154000
dma_tick	APB	Y	0x40158000
usbhost0_cfg	APB	Y	0x40160000
usbhost1_cfg	APB	Y	0x40164000
rosc0	APB	Y	0x40168000
rosc1	APB	Y	0x4016c000
vbusctrl	APB	Y	0x40170000
ticks	APB	Y	0x40174000
pio	APB	Y	0x40178000
sdio0	AHB	N	0x40180000
sdio1	AHB	N	0x40184000
dma	AHB	N	0x40188000
usbhost0	AXI3	N	0x40200000
usbhost1	AXI3	N	0x40300000
exac	AXI3	N	0x40400000

2.4. Atomic register access

Each peripheral register block with atomic access is allocated 4kB of address space, with registers accessed using one of four methods, selected by address decode.

- Addr + 0x0000 : normal read write access
- Addr + 0x1000 : atomic XOR on write, reads have no side-effects
- Addr + 0x2000 : atomic bitmask set on write, normal read access
- Addr + 0x3000 : atomic bitmask clear on write, normal read access

The write aliasing allows individual fields of a control register to be modified without performing a read-modify-write sequence in software: instead the changes are posted to the peripheral, and performed in-situ. Without this capability, read-modify-write cycles will incur twice the PCIe link round-trip latency. Using posted writes will also reduce the requirement for interrupt service routines to use software exclusivity mechanisms like spinlocks or mutexes.

The read alias at `0x1000` will return normal read data but will not advance the state of any RF/RWF registers. This allows debug access to peek at the next word in a FIFO interface.

The four atomic access aliases occupy a total of 16kB.

2.5. Clocks

The reference clock is a 50MHz crystal input. The frequency is selected to meet PHY requirements for USB, Ethernet, PCIe, and MIPI DPHY.

There are 3 internal PLLs, notionally allocated to these functions:

- CORE: System clocks
- AUDIO: Audio clock
- VIDEO: Video output (DAC / VEC)

There are also PLLs embedded in IP interfaces which are dedicated to that interface to generate the high speed link clock:

- PCIe
- USB3
- USB2
- DSI

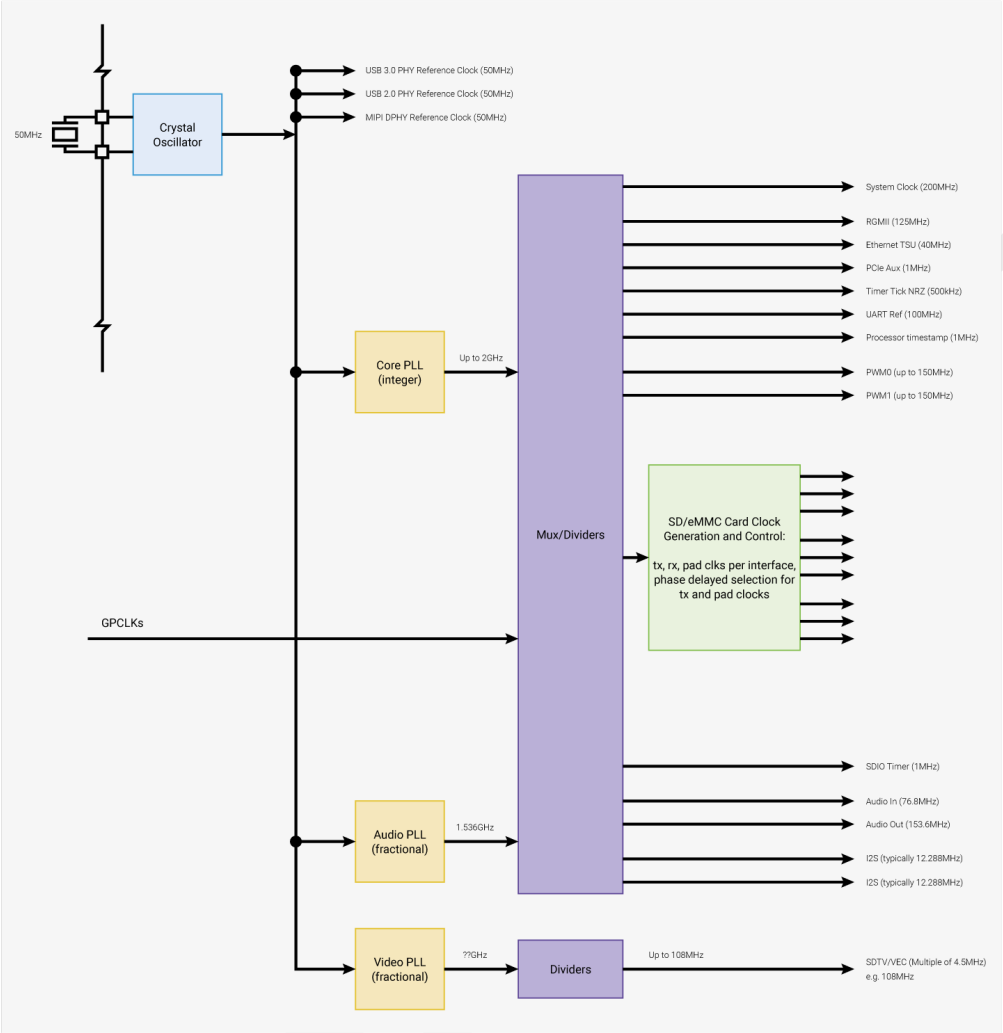
Internal clocks are generated from clock dividers using a selectable clock source.

2.5.1. Crystal and PLLs

A diagram showing Crystal, PLLs, and clock dividers is below.

DRAFT

Figure 3. Clock diagram



2.5.2. Core Clock

The Core clock (`clk_sys`) typically runs at 200MHz. The Core PLL VCO is kept at 2GHz so other fixed frequencies can be generated from this source

2.5.3. RGMII clocks

The RGMII clock is 125MHz, generated from the core PLL.

2.5.4. Audio clocks

The Audio PLL will run at a VCO frequency of 1.536GHz to generate internal audio clocks.

- AUDIO_IN - 76.8MHz, internally divided to produce mic_clk to GPIO pads.
- AUDIO_OUT - 153.6MHz
- I2S master clock - typically 12.288MHz, but varies between codec chips as $2^N × 48000$ (still an integer division from VCO freq).

2.5.5. Other Peripheral clocks

UART interfaces have an independent baud clock (`clk_uart`), typically 48MHz.

SPI master interfaces are clocked from `clk_sys`, but the SPI slave interface has the TX shift register directly clocked from the GPIO SCLK input pad. I2C interfaces are also clocked from `clk_sys`. Both SPI and I2C have internal divisors that control the interface data rate.

There are two PWM controllers with outputs driven from an independent clock (`clk_pwm`) for an up-to 150MHz reference clock.

2.5.6. Clock controls

There base address of the RIO registers is:

Table 3. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
clocks_main	APB	Y	`0x40018000`
clocks_video	APB	Y	`0x4001c000`

For a reference implementation see the RP1 Clocks Linux kernel driver.

DRAFT

Chapter 3. Low speed peripherals

3.1. GPIO

RP1 has 28 multi-functional General-Purpose Input/Output pins available to implement the standard Raspberry Pi 40-pin GPIO connector.

The pins are in a single electrical bank (VDDIO0). The GPIO bank (IO_BANK0) can be powered from 1.8V or 3.3V, but interface timings have been specified at 3.3V. Each pin can be controlled directly by software, or by a number of other functional blocks.

The bank supports the following functions:

- 5 × UART
- 6 × SPI
- 4 × I2C
- 2 × I2S - 1× Clock Producer instance, 1× Clock Consumer instance.
- RIO - Registered IO interface
- 24-bit DPI output
- 4-channel PWM output
- AUDIO_OUT - Stereo PWM audio output
- GPCLK - General-purpose clock input and output
- eMMC/SDIO bus with a 4-bit interface
- Interrupt generation from pin level or edge transitions

The functional blocks and their locations on the GPIO pins have been chosen to match user-facing functions on the 40-pin header of a Raspberry Pi 4 Model B.

For a detailed description of each of the functions, refer to the datasheet sections linked above.

NOTE

IO_BANK1 and IO_BANK2 are reserved for internal use.

3.1.1. Function select

The function allocated to each GPIO is selected by writing to the `FUNCSEL` field of the GPIO's `CTRL` register. The functions available on each IO are show in Table 4.

DRAFT

Table 4. GPIO function selection

	Function								
GPIO	a0	a1	a2	a3	a4	a5	a6	a7	a8
0	SPI0_SIO[3]	DPI_PCLK	UART1_TX	I2C0_SDA		SYS_RIO[0]	PROC_RIO[0]	PIO[0]	SPI2_CSn[0]
1	SPI0_SIO[2]	DPI_DE	UART1_RX	I2C0_SCL		SYS_RIO[1]	PROC_RIO[1]	PIO[1]	SPI2_SIO[1]
2	SPI0_CSn[3]	DPI_VSYNC	UART1_CTS	I2C1_SDA	UART0_IR_RX	SYS_RIO[2]	PROC_RIO[2]	PIO[2]	SPI2_SIO[0]
3	SPI0_CSn[2]	DPI_HSYNC	UART1_RTS	I2C1_SCL	UART0_IR_TX	SYS_RIO[3]	PROC_RIO[3]	PIO[3]	SPI2_SCLK
4	GPCLK[0]	DPI_D[0]	UART2_TX	I2C2_SDA	UART0_RI	SYS_RIO[4]	PROC_RIO[4]	PIO[4]	SPI3_CSn[0]
5	GPCLK[1]	DPI_D[1]	UART2_RX	I2C2_SCL	UART0_DTR	SYS_RIO[5]	PROC_RIO[5]	PIO[5]	SPI3_SIO[1]
6	GPCLK[2]	DPI_D[2]	UART2_CTS	I2C3_SDA	UART0_DCD	SYS_RIO[6]	PROC_RIO[6]	PIO[6]	SPI3_SIO[0]
7	SPI0_CSn[1]	DPI_D[3]	UART2_RTS	I2C3_SCL	UART0_DSR	SYS_RIO[7]	PROC_RIO[7]	PIO[7]	SPI3_SCLK
8	SPI0_CSn[0]	DPI_D[4]	UART3_TX	I2C0_SDA		SYS_RIO[8]	PROC_RIO[8]	PIO[8]	SPI4_CSn[0]
9	SPI0_SIO[1]	DPI_D[5]	UART3_RX	I2C0_SCL		SYS_RIO[9]	PROC_RIO[9]	PIO[9]	SPI4_SIO[0]
10	SPI0_SIO[0]	DPI_D[6]	UART3_CTS	I2C1_SDA		SYS_RIO[10]	PROC_RIO[10]	PIO[10]	SPI4_SIO[1]
11	SPI0_SCLK	DPI_D[7]	UART3_RTS	I2C1_SCL		SYS_RIO[11]	PROC_RIO[11]	PIO[11]	SPI4_SCLK
12	PWM0[0]	DPI_D[8]	UART4_TX	I2C2_SDA	AUDIO_OUT_L	SYS_RIO[12]	PROC_RIO[12]	PIO[12]	SPI5_CSn[0]
13	PWM0[1]	DPI_D[9]	UART4_RX	I2C2_SCL	AUDIO_OUT_R	SYS_RIO[13]	PROC_RIO[13]	PIO[13]	SPI5_SIO[1]
14	PWM0[2]	DPI_D[10]	UART4_CTS	I2C3_SDA	UART0_TX	SYS_RIO[14]	PROC_RIO[14]	PIO[14]	SPI5_SIO[0]
15	PWM0[3]	DPI_D[11]	UART4_RTS	I2C3_SCL	UART0_RX	SYS_RIO[15]	PROC_RIO[15]	PIO[15]	SPI5_SCLK
16	SPI1_CSn[2]	DPI_D[12]	MIPI0_DSI_TE		UART0_CTS	SYS_RIO[16]	PROC_RIO[16]	PIO[16]	
17	SPI1_CSn[1]	DPI_D[13]	MIPI1_DSI_TE		UART0_RTS	SYS_RIO[17]	PROC_RIO[17]	PIO[17]	
18	SPI1_CSn[0]	DPI_D[14]	I2S0_SCLK	PWM0[2]	I2S1_SCLK	SYS_RIO[18]	PROC_RIO[18]	PIO[18]	GPCLK[1]
19	SPI1_SIO[1]	DPI_D[15]	I2S0_WS	PWM0[3]	I2S1_WS	SYS_RIO[19]	PROC_RIO[19]	PIO[19]	
20	SPI1_SIO[0]	DPI_D[16]	I2S0_SDI[0]	GPCLK[0]	I2S1_SDI[0]	SYS_RIO[20]	PROC_RIO[20]	PIO[20]	
21	SPI1_SCLK	DPI_D[17]	I2S0_SDO[0]	GPCLK[1]	I2S1_SDO[0]	SYS_RIO[21]	PROC_RIO[21]	PIO[21]	

	Function								
22	SDIO0_CLK	DPI_D[18]	I2S0_SDI[1]	I2C3_SDA	I2S1_SDI[1]	SYS_RIO[22]	PROC_RIO[22]	PIO[22]	
23	SDIO0_CMD	DPI_D[19]	I2S0_SDO[1]	I2C3_SCL	I2S1_SDO[1]	SYS_RIO[23]	PROC_RIO[23]	PIO[23]	
24	SDIO0_DAT[0]	DPI_D[20]	I2S0_SDI[2]		I2S1_SDI[2]	SYS_RIO[24]	PROC_RIO[24]	PIO[24]	SPI2_CSn[1]
25	SDIO0_DAT[1]	DPI_D[21]	I2S0_SDO[2]	AUDIO_IN_CLK	I2S1_SDO[2]	SYS_RIO[25]	PROC_RIO[25]	PIO[25]	SPI3_CSn[1]
26	SDIO0_DAT[2]	DPI_D[22]	I2S0_SDI[3]	AUDIO_IN_DAT0	I2S1_SDI[3]	SYS_RIO[26]	PROC_RIO[26]	PIO[26]	SPI5_CSn[1]
27	SDIO0_DAT[3]	DPI_D[23]	I2S0_SDO[3]	AUDIO_IN_DAT1	I2S1_SDO[3]	SYS_RIO[27]	PROC_RIO[27]	PIO[27]	SPI1_CSn[1]

DRAFT

Each GPIO can have one function selected at a time. Likewise, each peripheral input (e.g. I2C3_SCL) should only be selected on one GPIO at a time. If the same peripheral input is connected to multiple GPIOs, the peripheral sees the logical OR of these GPIO inputs.

NOTE

Function selections without a named function in this list are reserved.

3.1.2. Interrupts

An interrupt can be generated for every GPIO pin in eight scenarios:

- Level High: the GPIO pin is a logical 1
- Debounced Level High: the GPIO pin has been a logical 1 for longer than the debounce time
- Level Low: the GPIO pin is a logical 0
- Debounced Level Low: the GPIO pin has been a logical 0 for longer than the debounce time
- Edge High: the GPIO has transitioned from a logical 0 to a logical 1
- Filtered Edge High: the GPIO has transitioned from a logical 0 to a logical 1 after the filter time has expired
- Edge Low: the GPIO has transitioned from a logical 1 to a logical 0
- Filtered Edge Low: the GPIO has transitioned from a logical 1 to a logical 0 after the filter time has expired

The filtered version of interrupts are controlled by a time constant register, which operates in units of IO_BANK0 ticks. See the IO_BANK0_GPIOn_CTRL.F_M register and Section 3.8.

The level interrupts are not latched. This means that if the pin is a logical 1 and the level high interrupt is active, it will become inactive as soon as the pin changes to a logical 0. The edge interrupts are stored in the GPIOn_STATUS register and cleared by writing 1 to the GPIOn_CTRL.IRQRESET register bit. All interrupt sources are ORed together and presented in a top-level enable, status, and force registers for three interrupt destinations: Proc 0, Proc 1, and PCIe (host processor). For PCIE the registers are enable (PCIE_INTE0), status (PCIE_INTS0), and force (PCIE_INTF0).

NOTE

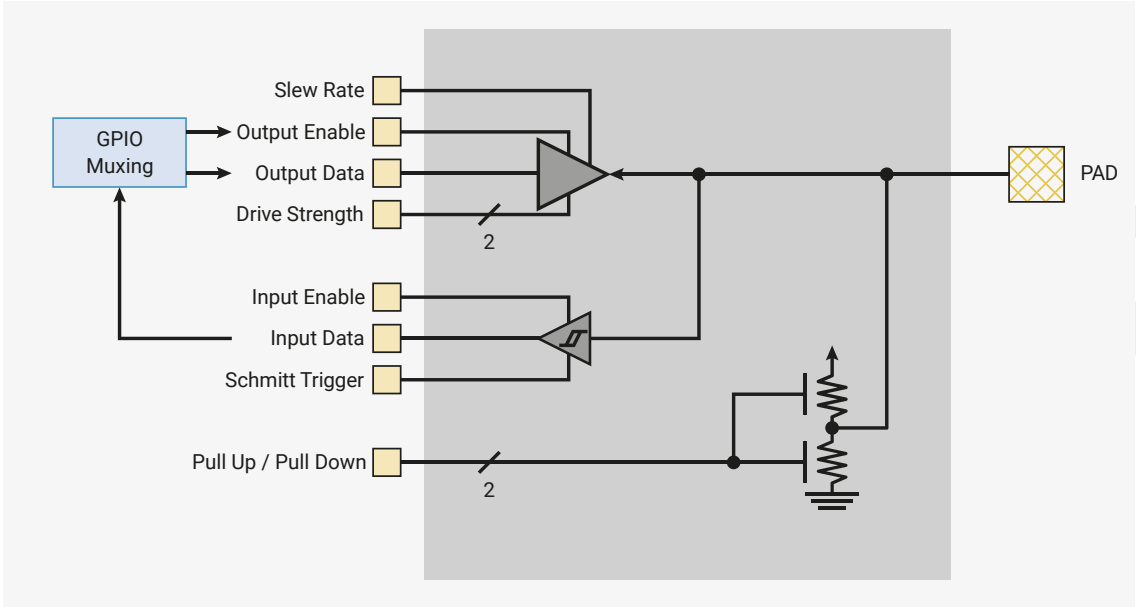
If any level-based interrupts are required, then the interrupt-to-message translation block (see PCIE MSIn_CFG section) must enable the IACK mechanism to properly sequence software through the Pending, Active, and EOI states. Interrupts may be missed by the host processor if this feature is not used.

3.1.3. Pads

Each GPIO is connected via a bidirectional CMOS pad, see Figure 4. The GPIOs offer:

- Fault-tolerant operation - very little current flows into the pin whilst it is below 3.63V and IOVDD is 0V
- Output drive strength of 2mA, 4mA, 8mA or 12mA
- Optional input Schmitt trigger hysteresis
- Optional output slew rate limiter
- Integrated pull-up, pull-down, bus-keeper or high-impedance behaviour when the output drive is disabled
- Input buffer disable to reduce current consumption when the pad is unconnected, or connected to an indeterminate signal level
- ESD rated to 4kV HBM, 500V CDM, 200V MM

Figure 4. Diagram of a single IO pad.



The pad's Output Enable, Output Data and Input Data ports are connected, via the IO mux, to the function controlling the pad. These signals can be individually manipulated (forced 1, forced 0, or inverted) by software with the IO_BANK0_GPIOn_CTRL override register fields. All other ports are controlled from the pad control register. See PADS_BANK0_GPIO0 for an example.

Both the output signal level and acceptable input signal level at the pad are determined by the bank IO supply (VDDIO0). VDDIO0 can be any nominal voltage between 1.8V and 3.3V, but to meet specification when powered at 1.8V, the pad input thresholds must be adjusted by writing a 1 to the PADS_BANK0_VOLTAGE_SELECT register. By default the pad input thresholds are valid for an VDDIO voltage between 2.5V and 3.3V. Using a voltage of 1.8V with the default input thresholds is a safe operating mode, though it will result in input thresholds that don't meet specification.

Using VDDIO voltages greater than 1.8V, with the input thresholds set for 1.8V may result in damage to the chip.

3.1.4. Registers

The base address of the GPIO registers are:

Table 5. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
io_bank0	APB	Y	0x400d0000
pads_bank0	APB	Y	0x400f0000

For a reference implementation see the RP1 GPIO Linux kernel driver.

Table 6. List of IO_BANK0 registers

Offset	Name	Info
0x000	GPIO0_STATUS	GPIO status
0x004	GPIO0_CTRL	GPIO control including function select and overrides.
0x008	GPIO1_STATUS	GPIO status
0x00c	GPIO1_CTRL	GPIO control including function select and overrides.
0x010	GPIO2_STATUS	GPIO status
0x014	GPIO2_CTRL	GPIO control including function select and overrides.
0x018	GPIO3_STATUS	GPIO status

Offset	Name	Info
0x01c	GPIO3_CTRL	GPIO control including function select and overrides.
0x020	GPIO4_STATUS	GPIO status
0x024	GPIO4_CTRL	GPIO control including function select and overrides.
0x028	GPIO5_STATUS	GPIO status
0x02c	GPIO5_CTRL	GPIO control including function select and overrides.
0x030	GPIO6_STATUS	GPIO status
0x034	GPIO6_CTRL	GPIO control including function select and overrides.
0x038	GPIO7_STATUS	GPIO status
0x03c	GPIO7_CTRL	GPIO control including function select and overrides.
0x040	GPIO8_STATUS	GPIO status
0x044	GPIO8_CTRL	GPIO control including function select and overrides.
0x048	GPIO9_STATUS	GPIO status
0x04c	GPIO9_CTRL	GPIO control including function select and overrides.
0x050	GPIO10_STATUS	GPIO status
0x054	GPIO10_CTRL	GPIO control including function select and overrides.
0x058	GPIO11_STATUS	GPIO status
0x05c	GPIO11_CTRL	GPIO control including function select and overrides.
0x060	GPIO12_STATUS	GPIO status
0x064	GPIO12_CTRL	GPIO control including function select and overrides.
0x068	GPIO13_STATUS	GPIO status
0x06c	GPIO13_CTRL	GPIO control including function select and overrides.
0x070	GPIO14_STATUS	GPIO status
0x074	GPIO14_CTRL	GPIO control including function select and overrides.
0x078	GPIO15_STATUS	GPIO status
0x07c	GPIO15_CTRL	GPIO control including function select and overrides.
0x080	GPIO16_STATUS	GPIO status
0x084	GPIO16_CTRL	GPIO control including function select and overrides.
0x088	GPIO17_STATUS	GPIO status
0x08c	GPIO17_CTRL	GPIO control including function select and overrides.
0x090	GPIO18_STATUS	GPIO status
0x094	GPIO18_CTRL	GPIO control including function select and overrides.
0x098	GPIO19_STATUS	GPIO status
0x09c	GPIO19_CTRL	GPIO control including function select and overrides.
0x0a0	GPIO20_STATUS	GPIO status
0x0a4	GPIO20_CTRL	GPIO control including function select and overrides.
0x0a8	GPIO21_STATUS	GPIO status

Offset	Name	Info
0x0ac	GPIO21_CTRL	GPIO control including function select and overrides.
0x0b0	GPIO22_STATUS	GPIO status
0x0b4	GPIO22_CTRL	GPIO control including function select and overrides.
0x0b8	GPIO23_STATUS	GPIO status
0x0bc	GPIO23_CTRL	GPIO control including function select and overrides.
0x0c0	GPIO24_STATUS	GPIO status
0x0c4	GPIO24_CTRL	GPIO control including function select and overrides.
0x0c8	GPIO25_STATUS	GPIO status
0x0cc	GPIO25_CTRL	GPIO control including function select and overrides.
0x0d0	GPIO26_STATUS	GPIO status
0x0d4	GPIO26_CTRL	GPIO control including function select and overrides.
0x0d8	GPIO27_STATUS	GPIO status
0x0dc	GPIO27_CTRL	GPIO control including function select and overrides.
0x100	INTR	Raw Interrupts
0x104	PROC0_INTE	Interrupt Enable for proc0
0x108	PROC0_INTF	Interrupt Force for proc0
0x10c	PROC0_INTS	Interrupt status after masking & forcing for proc0
0x110	PROC1_INTE	Interrupt Enable for proc1
0x114	PROC1_INTF	Interrupt Force for proc1
0x118	PROC1_INTS	Interrupt status after masking & forcing for proc1
0x11c	PCIE_INTE	Interrupt Enable for pcie
0x120	PCIE_INTF	Interrupt Force for pcie
0x124	PCIE_INTS	Interrupt status after masking & forcing for pcie

IO_BANK0: GPIO0_STATUS, GPIO1_STATUS, ..., GPIO26_STATUS, GPIO27_STATUS Registers

Offsets: 0x000, 0x008, ..., 0x0d0, 0x0d8

Description

GPIO status

Table 7. GPIO0_STATUS, GPIO1_STATUS, ..., GPIO26_STATUS, GPIO27_STATUS Registers

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
29	IRQTOPROC	interrupt to processors, after mask and override is applied	RO	0x0
28	IRQCOMBINED	interrupt to processors, after masking	RO	0x0
27	EVENT_DB_LEVEL_HIGH	Debounced input pin is high	RO	0x0
26	EVENT_DB_LEVEL_LOW	Debounced input pin is low	RO	0x0

Bits	Name	Description	Type	Reset
25	EVENT_F_EDGE_HIGH	Input pin has seen a filtered rising edge. Clear with ctrl_irqreset	RO	0x0
24	EVENT_F_EDGE_LOW	Input pin has seen a filtered falling edge. Clear with ctrl_irqreset	RO	0x0
23	EVENT_LEVEL_HIGH	Input pin is high	RO	0x0
22	EVENT_LEVEL_LOW	Input pin is Low	RO	0x0
21	EVENT_EDGE_HIGH	Input pin has seen rising edge. Clear with ctrl_irqreset	RO	0x0
20	EVENT_EDGE_LOW	Input pin has seen falling edge. Clear with ctrl_irqreset	RO	0x0
19	INTOPERI	input signal to peripheral, after filtering and override are applied, not valid if inisdirect=1	RO	0x0
18	INFILTERED	input signal from pad, after filtering is applied but before override, not valid if inisdirect=1	RO	0x0
17	INFROMPAD	input signal from pad, before filtering and override are applied	RO	0x0
16	INISDIRECT	input signal from pad, goes directly to the selected peripheral without filtering or override	RO	0x0
15:14	Reserved.	-	-	-
13	OETOPAD	output enable to pad after register overide is applied	RO	0x0
12	OEFROMPERI	output enable from selected peripheral, before register overide is applied	RO	0x0
11:10	Reserved.	-	-	-
9	OUTTOPAD	output signal to pad after register overide is applied	RO	0x0
8	OUTFROMPERI	output signal from selected peripheral, before register overide is applied	RO	0x0
7:0	Reserved.	-	-	-

IO_BANK0: GPIO0_CTRL, GPIO1_CTRL, ..., GPIO26_CTRL, GPIO27_CTRL Registers

Offsets: 0x004, 0x00c, ..., 0x0d4, 0x0dc

Description

GPIO control including function select and overrides.

Table 8. GPIO0_CTRL, GPIO1_CTRL, ..., GPIO26_CTRL, GPIO27_CTRL Registers

Bits	Name	Description	Type	Reset
31:30	IRQOVER	0x0 → don't invert the interrupt 0x1 → invert the interrupt 0x2 → drive interrupt low 0x3 → drive interrupt high	RW	0x0
29	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
28	IRQRESET	0x0 → do nothing 0x1 → reset the interrupt edge detector	SC	0x0
27	IRQMASK_DB_LEVEL_HIGH	Masks the debounced level high interrupt into the interrupt output	RW	0x0
26	IRQMASK_DB_LEVEL_LOW	Masks the debounced level low interrupt into the interrupt output	RW	0x0
25	IRQMASK_F_EDGE_HIGH	Masks the filtered edge high interrupt into the interrupt output	RW	0x0
24	IRQMASK_F_EDGE_LOW	Masks the filtered edge low interrupt into the interrupt output	RW	0x0
23	IRQMASK_LEVEL_HIGH	Masks the level high interrupt into the interrupt output	RW	0x0
22	IRQMASK_LEVEL_LOW	Masks the level low interrupt into the interrupt output	RW	0x0
21	IRQMASK_EDGE_HIGH	Masks the edge high interrupt into the interrupt output	RW	0x0
20	IRQMASK_EDGE_LOW	Masks the edge low interrupt into the interrupt output	RW	0x0
19:18	Reserved.	-	-	-
17:16	INOVER	0x0 → don't invert the peri input 0x1 → invert the peri input 0x2 → drive peri input low 0x3 → drive peri input high	RW	0x0
15:14	OEOVER	0x0 → drive output enable from peripheral signal selected by funcsel 0x1 → drive output enable from inverse of peripheral signal selected by funcsel 0x2 → disable output 0x3 → enable output	RW	0x0
13:12	OUTOVER	0x0 → drive output from peripheral signal selected by funcsel 0x1 → drive output from inverse of peripheral signal selected by funcsel 0x2 → drive output low 0x3 → drive output high	RW	0x0
11:5	F_M	Filter/debounce time constant M	RW	0x04
4:0	FUNCSEL	Function select. 31 == NULL. See GPIO function table for available functions.	RW	0x1f

IO_BANK0: INTR Register

Offset: 0x100

Description

Raw Interrupts

Table 9. INTR Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RO	0x0
26	GPIO26		RO	0x0
25	GPIO25		RO	0x0
24	GPIO24		RO	0x0
23	GPIO23		RO	0x0
22	GPIO22		RO	0x0
21	GPIO21		RO	0x0
20	GPIO20		RO	0x0
19	GPIO19		RO	0x0
18	GPIO18		RO	0x0
17	GPIO17		RO	0x0
16	GPIO16		RO	0x0
15	GPIO15		RO	0x0
14	GPIO14		RO	0x0
13	GPIO13		RO	0x0
12	GPIO12		RO	0x0
11	GPIO11		RO	0x0
10	GPIO10		RO	0x0
9	GPIO9		RO	0x0
8	GPIO8		RO	0x0
7	GPIO7		RO	0x0
6	GPIO6		RO	0x0
5	GPIO5		RO	0x0
4	GPIO4		RO	0x0
3	GPIO3		RO	0x0
2	GPIO2		RO	0x0
1	GPIO1		RO	0x0
0	GPIO0		RO	0x0

IO_BANK0: PROC0_INTE Register

Offset: 0x104

Description

Interrupt Enable for proc0

Table 10. PROC0_INTE Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
27	GPIO27		RW	0x0
26	GPIO26		RW	0x0
25	GPIO25		RW	0x0
24	GPIO24		RW	0x0
23	GPIO23		RW	0x0
22	GPIO22		RW	0x0
21	GPIO21		RW	0x0
20	GPIO20		RW	0x0
19	GPIO19		RW	0x0
18	GPIO18		RW	0x0
17	GPIO17		RW	0x0
16	GPIO16		RW	0x0
15	GPIO15		RW	0x0
14	GPIO14		RW	0x0
13	GPIO13		RW	0x0
12	GPIO12		RW	0x0
11	GPIO11		RW	0x0
10	GPIO10		RW	0x0
9	GPIO9		RW	0x0
8	GPIO8		RW	0x0
7	GPIO7		RW	0x0
6	GPIO6		RW	0x0
5	GPIO5		RW	0x0
4	GPIO4		RW	0x0
3	GPIO3		RW	0x0
2	GPIO2		RW	0x0
1	GPIO1		RW	0x0
0	GPIO0		RW	0x0

IO_BANK0: PROC0_INTF Register

Offset: 0x108

Description

Interrupt Force for proc0

Table 11. PROC0_INTF Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RW	0x0

Bits	Name	Description	Type	Reset
26	GPIO26		RW	0x0
25	GPIO25		RW	0x0
24	GPIO24		RW	0x0
23	GPIO23		RW	0x0
22	GPIO22		RW	0x0
21	GPIO21		RW	0x0
20	GPIO20		RW	0x0
19	GPIO19		RW	0x0
18	GPIO18		RW	0x0
17	GPIO17		RW	0x0
16	GPIO16		RW	0x0
15	GPIO15		RW	0x0
14	GPIO14		RW	0x0
13	GPIO13		RW	0x0
12	GPIO12		RW	0x0
11	GPIO11		RW	0x0
10	GPIO10		RW	0x0
9	GPIO9		RW	0x0
8	GPIO8		RW	0x0
7	GPIO7		RW	0x0
6	GPIO6		RW	0x0
5	GPIO5		RW	0x0
4	GPIO4		RW	0x0
3	GPIO3		RW	0x0
2	GPIO2		RW	0x0
1	GPIO1		RW	0x0
0	GPIO0		RW	0x0

IO_BANK0: PROC0_INTS Register

Offset: 0x10c

Description

Interrupt status after masking & forcing for proc0

Table 12. PROC0_INTS Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RO	0x0
26	GPIO26		RO	0x0

Bits	Name	Description	Type	Reset
25	GPIO25		RO	0x0
24	GPIO24		RO	0x0
23	GPIO23		RO	0x0
22	GPIO22		RO	0x0
21	GPIO21		RO	0x0
20	GPIO20		RO	0x0
19	GPIO19		RO	0x0
18	GPIO18		RO	0x0
17	GPIO17		RO	0x0
16	GPIO16		RO	0x0
15	GPIO15		RO	0x0
14	GPIO14		RO	0x0
13	GPIO13		RO	0x0
12	GPIO12		RO	0x0
11	GPIO11		RO	0x0
10	GPIO10		RO	0x0
9	GPIO9		RO	0x0
8	GPIO8		RO	0x0
7	GPIO7		RO	0x0
6	GPIO6		RO	0x0
5	GPIO5		RO	0x0
4	GPIO4		RO	0x0
3	GPIO3		RO	0x0
2	GPIO2		RO	0x0
1	GPIO1		RO	0x0
0	GPIO0		RO	0x0

IO_BANK0: PROC1_INTE Register

Offset: 0x110

Description

Interrupt Enable for proc1

Table 13. PROC1_INTE Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RW	0x0
26	GPIO26		RW	0x0
25	GPIO25		RW	0x0

Bits	Name	Description	Type	Reset
24	GPIO24		RW	0x0
23	GPIO23		RW	0x0
22	GPIO22		RW	0x0
21	GPIO21		RW	0x0
20	GPIO20		RW	0x0
19	GPIO19		RW	0x0
18	GPIO18		RW	0x0
17	GPIO17		RW	0x0
16	GPIO16		RW	0x0
15	GPIO15		RW	0x0
14	GPIO14		RW	0x0
13	GPIO13		RW	0x0
12	GPIO12		RW	0x0
11	GPIO11		RW	0x0
10	GPIO10		RW	0x0
9	GPIO9		RW	0x0
8	GPIO8		RW	0x0
7	GPIO7		RW	0x0
6	GPIO6		RW	0x0
5	GPIO5		RW	0x0
4	GPIO4		RW	0x0
3	GPIO3		RW	0x0
2	GPIO2		RW	0x0
1	GPIO1		RW	0x0
0	GPIO0		RW	0x0

IO_BANK0: PROC1_INTF Register

Offset: 0x114

Description

Interrupt Force for proc1

Table 14. PROC1_INTF Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RW	0x0
26	GPIO26		RW	0x0
25	GPIO25		RW	0x0
24	GPIO24		RW	0x0

Bits	Name	Description	Type	Reset
23	GPIO23		RW	0x0
22	GPIO22		RW	0x0
21	GPIO21		RW	0x0
20	GPIO20		RW	0x0
19	GPIO19		RW	0x0
18	GPIO18		RW	0x0
17	GPIO17		RW	0x0
16	GPIO16		RW	0x0
15	GPIO15		RW	0x0
14	GPIO14		RW	0x0
13	GPIO13		RW	0x0
12	GPIO12		RW	0x0
11	GPIO11		RW	0x0
10	GPIO10		RW	0x0
9	GPIO9		RW	0x0
8	GPIO8		RW	0x0
7	GPIO7		RW	0x0
6	GPIO6		RW	0x0
5	GPIO5		RW	0x0
4	GPIO4		RW	0x0
3	GPIO3		RW	0x0
2	GPIO2		RW	0x0
1	GPIO1		RW	0x0
0	GPIO0		RW	0x0

IO_BANK0: PROC1_INTS Register

Offset: 0x118

Description

Interrupt status after masking & forcing for proc1

Table 15. PROC1_INTS Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RO	0x0
26	GPIO26		RO	0x0
25	GPIO25		RO	0x0
24	GPIO24		RO	0x0
23	GPIO23		RO	0x0

Bits	Name	Description	Type	Reset
22	GPIO22		RO	0x0
21	GPIO21		RO	0x0
20	GPIO20		RO	0x0
19	GPIO19		RO	0x0
18	GPIO18		RO	0x0
17	GPIO17		RO	0x0
16	GPIO16		RO	0x0
15	GPIO15		RO	0x0
14	GPIO14		RO	0x0
13	GPIO13		RO	0x0
12	GPIO12		RO	0x0
11	GPIO11		RO	0x0
10	GPIO10		RO	0x0
9	GPIO9		RO	0x0
8	GPIO8		RO	0x0
7	GPIO7		RO	0x0
6	GPIO6		RO	0x0
5	GPIO5		RO	0x0
4	GPIO4		RO	0x0
3	GPIO3		RO	0x0
2	GPIO2		RO	0x0
1	GPIO1		RO	0x0
0	GPIO0		RO	0x0

IO_BANK0: PCIE_INTE Register

Offset: 0x11c

Description

Interrupt Enable for pcie

Table 16. PCIE_INTE Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RW	0x0
26	GPIO26		RW	0x0
25	GPIO25		RW	0x0
24	GPIO24		RW	0x0
23	GPIO23		RW	0x0
22	GPIO22		RW	0x0

Bits	Name	Description	Type	Reset
21	GPIO21		RW	0x0
20	GPIO20		RW	0x0
19	GPIO19		RW	0x0
18	GPIO18		RW	0x0
17	GPIO17		RW	0x0
16	GPIO16		RW	0x0
15	GPIO15		RW	0x0
14	GPIO14		RW	0x0
13	GPIO13		RW	0x0
12	GPIO12		RW	0x0
11	GPIO11		RW	0x0
10	GPIO10		RW	0x0
9	GPIO9		RW	0x0
8	GPIO8		RW	0x0
7	GPIO7		RW	0x0
6	GPIO6		RW	0x0
5	GPIO5		RW	0x0
4	GPIO4		RW	0x0
3	GPIO3		RW	0x0
2	GPIO2		RW	0x0
1	GPIO1		RW	0x0
0	GPIO0		RW	0x0

IO_BANK0: PCIE_INTF Register

Offset: 0x120

Description

Interrupt Force for pcie

Table 17. PCIE_INTF Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RW	0x0
26	GPIO26		RW	0x0
25	GPIO25		RW	0x0
24	GPIO24		RW	0x0
23	GPIO23		RW	0x0
22	GPIO22		RW	0x0
21	GPIO21		RW	0x0

Bits	Name	Description	Type	Reset
20	GPIO20		RW	0x0
19	GPIO19		RW	0x0
18	GPIO18		RW	0x0
17	GPIO17		RW	0x0
16	GPIO16		RW	0x0
15	GPIO15		RW	0x0
14	GPIO14		RW	0x0
13	GPIO13		RW	0x0
12	GPIO12		RW	0x0
11	GPIO11		RW	0x0
10	GPIO10		RW	0x0
9	GPIO9		RW	0x0
8	GPIO8		RW	0x0
7	GPIO7		RW	0x0
6	GPIO6		RW	0x0
5	GPIO5		RW	0x0
4	GPIO4		RW	0x0
3	GPIO3		RW	0x0
2	GPIO2		RW	0x0
1	GPIO1		RW	0x0
0	GPIO0		RW	0x0

IO_BANK0: PCIE_INTS Register

Offset: 0x124

Description

Interrupt status after masking & forcing for pcie

Table 18. PCIE_INTS Register

Bits	Name	Description	Type	Reset
31:28	Reserved.	-	-	-
27	GPIO27		RO	0x0
26	GPIO26		RO	0x0
25	GPIO25		RO	0x0
24	GPIO24		RO	0x0
23	GPIO23		RO	0x0
22	GPIO22		RO	0x0
21	GPIO21		RO	0x0
20	GPIO20		RO	0x0

Bits	Name	Description	Type	Reset
19	GPIO19		RO	0x0
18	GPIO18		RO	0x0
17	GPIO17		RO	0x0
16	GPIO16		RO	0x0
15	GPIO15		RO	0x0
14	GPIO14		RO	0x0
13	GPIO13		RO	0x0
12	GPIO12		RO	0x0
11	GPIO11		RO	0x0
10	GPIO10		RO	0x0
9	GPIO9		RO	0x0
8	GPIO8		RO	0x0
7	GPIO7		RO	0x0
6	GPIO6		RO	0x0
5	GPIO5		RO	0x0
4	GPIO4		RO	0x0
3	GPIO3		RO	0x0
2	GPIO2		RO	0x0
1	GPIO1		RO	0x0
0	GPIO0		RO	0x0

Table 19. List of PADS_BANK0 registers

Offset	Name	Info
0x00	VOLTAGE_SELECT	Voltage select. Per bank control
0x04	GPIO0	Pad control register
0x08	GPIO1	Pad control register
0x0c	GPIO2	Pad control register
0x10	GPIO3	Pad control register
0x14	GPIO4	Pad control register
0x18	GPIO5	Pad control register
0x1c	GPIO6	Pad control register
0x20	GPIO7	Pad control register
0x24	GPIO8	Pad control register
0x28	GPIO9	Pad control register
0x2c	GPIO10	Pad control register
0x30	GPIO11	Pad control register
0x34	GPIO12	Pad control register

Offset	Name	Info
0x38	GPIO13	Pad control register
0x3c	GPIO14	Pad control register
0x40	GPIO15	Pad control register
0x44	GPIO16	Pad control register
0x48	GPIO17	Pad control register
0x4c	GPIO18	Pad control register
0x50	GPIO19	Pad control register
0x54	GPIO20	Pad control register
0x58	GPIO21	Pad control register
0x5c	GPIO22	Pad control register
0x60	GPIO23	Pad control register
0x64	GPIO24	Pad control register
0x68	GPIO25	Pad control register
0x6c	GPIO26	Pad control register
0x70	GPIO27	Pad control register

PADS_BANK0: VOLTAGE_SELECT Register

Offset: 0x00

Table 20. VOLTAGE_SELECT Register

Bits	Description	Type	Reset
31:1	Reserved.	-	-
0	Voltage select. Per bank control 0x0 → Set voltage to 3.3V (DVDD >= 2V5) 0x1 → Set voltage to 1.8V (DVDD <= 1V8)	RW	0x0

PADS_BANK0: GPIO0, GPIO1, ..., GPIO26, GPIO27 Registers

Offsets: 0x04, 0x08, ..., 0x6c, 0x70

Description

Pad control register

Table 21. GPIO0, GPIO1, ..., GPIO26, GPIO27 Registers

Bits	Name	Description	Type	Reset
31:8	Reserved.	-	-	-
7	OD	Output disable. Has priority over output enable from peripherals	RW	0x1
6	IE	Input enable	RW	0x0
5:4	DRIVE	Drive strength. 0x0 → 2mA 0x1 → 4mA 0x2 → 8mA 0x3 → 12mA	RW	0x1
3	PUE	Pull up enable	RW	varies

Bits	Name	Description	Type	Reset
2	PDE	Pull down enable	RW	varies
1	SCHMITT	Enable schmitt trigger	RW	0x1
0	SLEWFAST	Slew rate control. 1 = Fast, 0 = Slow	RW	0x0

3.2. UART

RP1 has six instances of a UART peripheral, five of which are available on GPIO bank 0, based on the ARM Primecell UART (PL011) (Revision r1p5).

Each instance supports the following features:

- Separate 32 × 8 Tx and 32 × 12 Rx FIFOs
- Programmable baud rate generator, clocked by `clk_uart`
- Standard asynchronous communication bits (start, stop, parity) added on transmit and removed on receive
- Line break detection
- Programmable serial interface (5, 6, 7, or 8 bits)
- 1 or 2 stop bits
- Programmable hardware flow control

3.2.1. Register base addresses

There are six instances of the UART peripheral.

Table 22. Peripheral address map

Block	Bus Type	Atomic Access	Address
uart0	APB	N	`0x40030000`
uart1	APB	N	`0x40034000`
uart2	APB	N	`0x40038000`
uart3	APB	N	`0x4003c000`
uart4	APB	N	`0x40040000`
uart5	APB	N	`0x40044000`

For a reference implementation see the PL1011 Linux kernel driver.

3.3. RIO

The Registered IO (RIO) interface allows the host processor to manipulate GPIOs. There are four registers:

- `RIO_OUT` controls the GPIO output drive
- `RIO_OE` controls the GPIO output drive enable
- `RIO_NOSYNC_IN` samples the GPIO inputs directly
- `RIO_SYNC_IN` samples the GPIO inputs, each synchronised with a 2-stage synchroniser to clk_sys

These registers have aliases at the SET, CLEAR, and XOR offsets where writes to these addresses are atomically

committed.

3.3.1. Latency considerations

The PCIe link between RP1 and the host processor inserts an unavoidable amount of latency, typically 1µs at the design link width and speed. This is of chief concern for applications that rely on rapid but timely write sequences, e.g. bit-bashed protocols, or applications that need to respond quickly to the state change of a pin.

3.3.1.1. Read latency

PCIe reads consist of a request and a response, therefore incur at least double the link latency. To avoid round-tripping twice in a poll-type loop, it is recommended to insert a write barrier after the last write that toggles a pin, then issue the read. This ensures that a write has sufficient time to change the actual output state of the pin, and a read will sample the point after the state change.

NOTE

Other bus master read activity to peripherals sharing the same APB splitter as RIO0, namely the ADC, may affect the speed at which RIO operations happen. Avoid polling the ADC's status register and instead use DMA or FIFO-based operation if simultaneous use of both is required.

3.3.1.2. Write latency

Writes across PCIe are naturally pipelined, as they are all Posted transactions. Pipelining writes using stores without memory barriers and using CPU-local busy-wait loops to space them will largely hide the effect of latency, but precautions should be taken to avoid reordering of processor store instructions through an architecture-specific mechanism, e.g. MMU page flags. For AARCH64, the recommended mapping is `Device_nGnRE`.

3.3.1.3. Link power saving (ASPM)

If PCIe ASPM is enabled, then the respective L0s or bidirectional L1 wake latency will be added to the first read or write to be presented to the Root Complex. This will add delays of approximately 2µs for an L0s wake and 5µs for an L1 wake. If infrequent access is likely, for example if the delay inside a GPIO polling loop is 10-100µs, then disable ASPM to keep the link in L0.

3.3.2. Register base addresses

There base address of the RIO registers is:

Table 23. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
sys_rio0	APB	Y	`0x400e0000`

For a reference implementation see the RP1 GPIO Linux kernel driver.

3.4. PWM

3.4.1. Overview

The PWM peripheral is a flexible waveform generator with a variety of operational modes. RP1 has two instances of the PWM peripheral, one of which is available on GPIO bank 0.

3.4.2. Features

- Four independent output channels with separate duty-cycle and range registers
- 32-bit counter widths
- Seven output generation modes
- Optional per-channel output inversion
- Channels can be selectively bound to a common range register for synchronous operation
- Optional duty-cycle data FIFO with DMA support
- Channels can be selectively bound to the FIFO output and fed duty-cycle data in round-robin sequence
- Optional sigma-delta noise shaping engine with integral dither to support high-precision DC outputs

3.4.3. Principle of operation

Each PWM output channel generates an output waveform that is a function of its `MODE`, `RANGE`, and `DUTY` inputs. When enabled, the channel's internal counter is preloaded with either 0 or `RANGE`, and then counts up or down depending on the selected mode. When the internal count overflows, either at zero or when the count is equal to the value in the range register, the channel signals a duty-cycle strobe to pop the next value. This strobe has different effects depending on the overall block setup.

The duty-cycle value presented to the PWM channel can optionally pass through a sigma-delta noise-shaping engine. This treats input duty-cycle values as 16-bit signed numbers and quantises them to a smaller width. It then filters the resulting quantisation error with a second-order loop that significantly reduces the noise power at low frequencies, at the expense of more noise at higher frequencies. The noise-shaping engine has optional in-loop dither to suppress idle tones, and a configurable static bias that can convert input signed quantities to unsigned output quantities. The noise-shaping algorithm advances on each assertion of the duty-cycle strobe from the respective PWM channel.

The source of a channel's duty-cycle is selectable between a common FIFO interface or a register. The `DUTY` register can be freely updated by host software, the value is synchronised and latched on strobe assertion.

The asynchronous FIFO has a single 32-bit input port and selectable output striping. The FIFO input is striped according to the channels that are enabled and using the FIFO. Data written to the DUTY_FIFO register is forwarded round-robin to each channel's FIFO segment. The FIFO's pop signal is asserted based on a logical OR of a masked version of each individual channel's pop signal - see the `CHANx_CTRL.FIFO_POP_MASK` and `CHANx_CTRL.BIND` registers.

The FIFO has a DMA handshake connection that allows the DMAC to pump data into FIFO_DUTY.

The block has an overall settings update register that can synchronously enable, disable, start and stop channels glitchlessly. A top-level interrupt is provided that can trigger on FIFO state, DMA request assertion, or each channel's overflow strobe.

The PWM circuitry is clocked from `clk_pwm`, and the register interface is clocked from `clk_sys`.

3.4.4. Output modes

PWM output generators are configurable for each channel in its `CHANx_CTRL.MODE` register.

3.4.4.1. PWM_MODE_TRAILING_EDGE

The internal count starts at 0, and increments on each cycle until it reaches RANGE. It then reloads at 0 and signals an overflow. The output is 1 when the internal count is less than the DUTY input and 0 when the internal count is greater than or equal to the DUTY input.

3.4.4.2. PWM_MODE_LEADING_EDGE

The internal count starts at RANGE, and decrements on each cycle until it reaches 0. It then reloads at RANGE and signals an overflow. The output is 1 when the internal count is less than the DUTY input and 0 when the internal count is greater than or equal to the DUTY input. This produces the inverse of the PWM_MODE_TRAILING_EDGE waveform.

3.4.4.3. PWM_MODE_DOUBLE_EDGE

The internal counter starts at RANGE, and decrements on each cycle until it reaches 0. It then starts incrementing each cycle until it reaches RANGE and signals an overflow. The output is 1 when the internal count is less than the DUTY input and 0 when the internal count is greater than or equal to the DUTY input. This mode produces a centre-aligned (also known as phase-correct) pulse.

3.4.4.4. PWM_MODE_PDM

The internal counter starts at 0 and increments on each cycle until it reaches RANGE. It then reloads at 0 and signals an overflow. The output is produced by a comparison operating on the output of an error accumulator, which is initialised to 0 and updated as follows:

```
If Accumulator + Duty >= Range:
   Accumulator = Accumulator + Duty - Range
   Output = 1
Else:
   Accumulator = Accumulator + Duty
   Output = 0
Endif
```

This produces a pulse-density output train whose proportion of 1 bits is equal to the ratio between DUTY and RANGE.

3.4.4.5. PWM_MODE_PPM

The internal counter starts at 0 and increments on each cycle until it reaches RANGE. It then reloads at 0 and signals an overflow. When the internal count is equal to DUTY the output is 1, and is 0 otherwise. This generates pulse-position modulation.

3.4.4.6. PWM_MODE_SERIALISER_MSB

The internal counter starts at 0 and increments on each cycle until it reaches RANGE. It then reloads at 0 and signals an overflow. The DUTY input is treated as a shift register. The selected bit in DUTY is `31 - (COUNT % 32)`.

3.4.4.7. PWM_MODE_SERIALISER_LSB

The internal counter starts at 0 and increments on each cycle until it reaches `RANGE`. It then reloads at 0 and signals an overflow. The `DUTY` input is treated as a shift register. The selected bit in `DUTY` is `COUNT % 32`.

3.4.5. List of registers

The PWM registers start at:

Table 24. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
pwm0	APB	Y	`0x40098000`
pwm1	APB	Y	`0x4009c000`

For a reference implementation see the RP1 PWM Linux kernel driver.

Table 25. List of PWM registers

Offset	Name	Info
0x00	GLOBAL_CTRL	Global control bits
0x04	FIFO_CTRL	FIFO thresholding and status
0x08	COMMON_RANGE	
0x0c	COMMON_DUTY	
0x10	DUTY_FIFO	
0x14	CHAN0_CTRL	Channel 0 control register
0x18	CHAN0_RANGE	
0x1c	CHAN0_PHASE	
0x20	CHAN0_DUTY	
0x24	CHAN1_CTRL	Channel 1 control register
0x28	CHAN1_RANGE	
0x2c	CHAN1_PHASE	
0x30	CHAN1_DUTY	
0x34	CHAN2_CTRL	Channel 2 control register
0x38	CHAN2_RANGE	
0x3c	CHAN2_PHASE	
0x40	CHAN2_DUTY	
0x44	CHAN3_CTRL	Channel 3 control register
0x48	CHAN3_RANGE	
0x4c	CHAN3_PHASE	
0x50	CHAN3_DUTY	
0x54	INTR	Raw Interrupts
0x58	INTE	Interrupt Enable
0x5c	INTF	Interrupt Force
0x60	INTS	Interrupt status after masking & forcing

PWM: GLOBAL_CTRL Register

Offset: 0x00

Description

Global control bits

Table 26. GLOBAL_CTRL Register

Bits	Name	Description	Type	Reset
31	SET_UPDATE	To prevent mis-sampling of multi-bit bus signals in the PWM clock domain, this bit should be used to trigger a settings update. This ensures that all PWM channel settings update on the same PWM clock cycle. Write 1 to trigger a settings update to the block. Self clears to 0. This bit affects the chan*_en bits, chan*_phase, chan*_ctrl and common_range registers. Writes to the *_duty and *_range registers have an integral update strobe and writes take effect on the next counter overflow of the respective PWM channel.	SC	0x0
30:4	Reserved.	-	-	-
3	CHAN3_EN		RW	0x0
2	CHAN2_EN		RW	0x0
1	CHAN1_EN		RW	0x0
0	CHAN0_EN	1 - Enable the respective PWM channel in the mode set by the chanN_ctrl registers 0 - Channel disabled	RW	0x0

PWM: FIFO_CTRL Register

Offset: 0x04

Description

FIFO thresholding and status

Table 27. FIFO_CTRL Register

Bits	Name	Description	Type	Reset
31	DREQ_EN	1: Generate DMA request signals to the DMA controller 0: Don't generate request signals - the dreq_active interrupt is unaffected.	RW	0x0
30:21	Reserved.	-	-	-
20:16	DWELL_TIME	Delay in number of bus cycles before successive DREQs are generated. Used to account for system bus latency in write data arriving at the FIFO.	RW	0x02
15:11	THRESHOLD	Threshold for the comparator. DREQ is asserted when level <= threshold.	RW	0x00
10:7	Reserved.	-	-	-
6	FLUSH_DONE	FIFO flush completed in the PWM clock domain	RO	0x0
5	FLUSH	Assert to flush FIFO	RW	0x0

Bits	Name	Description	Type	Reset
4:0	LEVEL	Number of available words in the FIFO	RO	0x00

PWM: COMMON_RANGE Register

Offset: 0x08

Table 28. COMMON_RANGE Register

Bits	Description	Type	Reset
31:0	Counter range register for channels that are set to use channel binding	RW	0x00000000

PWM: COMMON_DUTY Register

Offset: 0x0c

Table 29. COMMON_DUTY Register

Bits	Description	Type	Reset
31:0	Counter compare register for channels that are set to use channel binding and are not set to use the common FIFO	RW	0x00000000

PWM: DUTY_FIFO Register

Offset: 0x10

Table 30. DUTY_FIFO Register

Bits	Description	Type	Reset
31:0	32-bit interface to a 128-bit backed duty cycle FIFO In round-robin fashion, 32-bit writes to this address are sequentially packed as 32*n-bit words that are pushed into the duty cycle FIFO. N varies as per the number of enabled channels set to use the FIFO. A distributor checks which channels are enabled and using the FIFO, and writes the 32-bit words accordingly.	WF	0x00000000

PWM: CHAN0_CTRL Register

Offset: 0x14

Description

Channel 0 control register

Table 31. CHAN0_CTRL Register

Bits	Name	Description	Type	Reset
31:16	SDM_BIAS	Unsigned offset to be added to the output PWM code generated by the sigma-delta modulator.	RW	0x0000
15:12	SDM_BITWIDTH	Quantise the 16-bit input to a (sdm_bitwidth+1)-bit output. 0 → 1-bit output.	RW	0x0
11:9	Reserved.	-	-	-
8	FIFO_POP_MASK	0 - Counter overflow events do not generate FIFO pop events 1 - Counter overflow events generate FIFO pop events	RW	0x1
7	DITHER	1 - When SDM mode is used, add a 1-bit LSB dither inside the noise shaping loop to suppress idle tones 0 - No dither applied	RW	0x0

Bits	Name	Description	Type	Reset
6	SDM	1 - Use sigma-delta noise shaping modulator. In conjunction with sdm_bitwidth, treat the duty cycle as a 16-bit signed truncation of the 32-bit duty cycle value and quantise to (sdm_bitwidth+1)-bits. The resulting quantisation noise is filtered using a 2nd-order loop. 0 - Bypass modulator	RW	0x0
5	USEFIFO	1 - Use the duty_fifo. Note: setting bind=0 and usefifo=1 will lead to unpredictable operation 0 - Use the duty cycle register common_duty/chan0_duty	RW	0x0
4	BIND	1 - Bind Channel 0 to the common_range and common_duty/duty_fifo registers 0 - Channel 0 uses chan0_range and chan0_duty	RW	0x0
3	INVERT	1 - Invert the output bit	RW	0x0
2:0	MODE	PWM generation mode 0x0 → Generates 0 0x1 → Trailing-edge mark-space PWM modulation 0x2 → Phase-correct mark-space PWM modulation 0x3 → Pulse-density encoded output 0x4 → MSB Serialiser output. 0x5 → Pulse position modulated output - a single high-pulse is transmitted per cycle. 0x6 → Leading-edge mark-space PWM modulation 0x7 → LSB Serialiser output.	RW	0x0

PWM: CHAN0_RANGE Register

Offset: 0x18

Table 32. CHAN0_RANGE Register

Bits	Description	Type	Reset
31:0	Channel 0 counter range	RW	0x00000000

PWM: CHAN0_PHASE Register

Offset: 0x1c

Table 33. CHAN0_PHASE Register

Bits	Description	Type	Reset
31:0	Channel 0 counter phase offset register This register preloads the internal counter such that phase offsets between channels can be introduced. Do not set higher than the respective range register.	RW	0x00000000

PWM: CHAN0_DUTY Register

Offset: 0x20

Table 34. CHAN0_DUTY Register

Bits	Description	Type	Reset
31:0	Channel 0 counter compare register	RW	0x00000000

PWM: CHAN1_CTRL Register

Offset: 0x24

Description

Channel 1 control register

Table 35. CHAN1_CTRL Register

Bits	Name	Description	Type	Reset
31:16	SDM_BIAS	Unsigned offset to be added to the output PWM code generated by the sigma-delta modulator.	RW	0x0000
15:12	SDM_BITWIDTH	Quantise the 16-bit input to a (sdm_bitwidth+1)-bit output. 0 → 1-bit output.	RW	0x0
11:9	Reserved.	-	-	-
8	FIFO_POP_MASK	0 - Counter overflow events do not generate FIFO pop events 1 - Counter overflow events generate FIFO pop events	RW	0x1
7	DITHER	1 - When SDM mode is used, add a 1-bit LSB dither inside the noise shaping loop to suppress idle tones 0 - No dither applied	RW	0x0
6	SDM	1 - Use sigma-delta noise shaping modulator. In conjunction with sdm_bitwidth, treat the duty cycle as a 16-bit signed truncation of the 32-bit duty cycle value and quantise to (sdm_bitwidth+1)-bits. The resulting quantisation noise is filtered using a 2nd-order loop. 0 - Bypass modulator	RW	0x0
5	USEFIFO	1 - Use the duty_fifo. Note: setting bind=0 and usefifo=1 will lead to unpredictable operation 0 - Use the duty cycle register common_duty/chan1_duty	RW	0x0
4	BIND	1 - Bind Channel 1 to the common_range and common_duty/duty_fifo registers 0 - Channel 1 uses chan1_range and chan1_duty	RW	0x0
3	INVERT	1 - Invert the output bit	RW	0x0
2:0	MODE	PWM generation mode 0x0 → Generates 0 0x1 → Trailing-edge mark-space PWM modulation 0x2 → Phase-correct mark-space PWM modulation 0x3 → Pulse-density encoded output 0x4 → MSB Serialiser output. 0x5 → Pulse position modulated output - a single high-pulse is transmitted per cycle. 0x6 → Leading-edge mark-space PWM modulation 0x7 → LSB Serialiser output.	RW	0x0

PWM: CHAN1_RANGE Register

Offset: 0x28

Table 36. CHAN1_RANGE Register

Bits	Description	Type	Reset
31:0	Channel 1 counter range	RW	0x00000000

PWM: CHAN1_PHASE Register

Offset: 0x2c

Table 37. CHAN1_PHASE Register

Bits	Description	Type	Reset
31:0	Channel 1 counter phase offset register This register preloads the internal counter such that phase offsets between channels can be introduced. Do not set higher than the respective range register.	RW	0x00000000

PWM: CHAN1_DUTY Register

Offset: 0x30

Table 38. CHAN1_DUTY Register

Bits	Description	Type	Reset
31:0	Channel 1 counter compare register	RW	0x00000000

PWM: CHAN2_CTRL Register

Offset: 0x34

Description

Channel 2 control register

Table 39. CHAN2_CTRL Register

Bits	Name	Description	Type	Reset
31:16	SDM_BIAS	Unsigned offset to be added to the output PWM code generated by the sigma-delta modulator.	RW	0x0000
15:12	SDM_BITWIDTH	Quantise the 16-bit input to a (sdm_bitwidth+1)-bit output. 0 → 1-bit output.	RW	0x0
11:9	Reserved.	-	-	-
8	FIFO_POP_MASK	0 - Counter overflow events do not generate FIFO pop events 1 - Counter overflow events generate FIFO pop events	RW	0x1
7	DITHER	1 - When SDM mode is used, add a 1-bit LSB dither inside the noise shaping loop to suppress idle tones 0 - No dither applied	RW	0x0
6	SDM	1 - Use sigma-delta noise shaping modulator. In conjunction with sdm_bitwidth, treat the duty cycle as a 16-bit signed truncation of the 32-bit duty cycle value and quantise to (sdm_bitwidth+1)-bits. The resulting quantisation noise is filtered using a 2nd-order loop. 0 - Bypass modulator	RW	0x0

Bits	Name	Description	Type	Reset
5	USEFIFO	1 - Use the duty_fifo. Note: setting bind=0 and usefifo=1 will lead to unpredictable operation 0 - Use the duty cycle register common_duty/chan2_duty	RW	0x0
4	BIND	1 - Bind Channel 2 to the common_range and common_duty/duty_fifo registers 0 - Channel 2 uses chan2_range and chan2_duty	RW	0x0
3	INVERT	1 - Invert the output bit	RW	0x0
2:0	MODE	PWM generation mode 0x0 → Generates 0 0x1 → Trailing-edge mark-space PWM modulation 0x2 → Phase-correct mark-space PWM modulation 0x3 → Pulse-density encoded output 0x4 → MSB Serialiser output. 0x5 → Pulse position modulated output - a single high-pulse is transmitted per cycle. 0x6 → Leading-edge mark-space PWM modulation 0x7 → LSB Serialiser output.	RW	0x0

PWM: CHAN2_RANGE Register

Offset: 0x38

Table 40. CHAN2_RANGE Register

Bits	Description	Type	Reset
31:0	Channel 2 counter range	RW	0x00000000

PWM: CHAN2_PHASE Register

Offset: 0x3c

Table 41. CHAN2_PHASE Register

Bits	Description	Type	Reset
31:0	Channel 2 counter phase offset register This register preloads the internal counter such that phase offsets between channels can be introduced. Do not set higher than the respective range register.	RW	0x00000000

PWM: CHAN2_DUTY Register

Offset: 0x40

Table 42. CHAN2_DUTY Register

Bits	Description	Type	Reset
31:0	Channel 2 counter compare register	RW	0x00000000

PWM: CHAN3_CTRL Register

Offset: 0x44

Description

Channel 3 control register

Table 43. CHAN3_CTRL Register

Bits	Name	Description	Type	Reset
31:16	SDM_BIAS	Unsigned offset to be added to the output PWM code generated by the sigma-delta modulator.	RW	0x0000
15:12	SDM_BITWIDTH	Quantise the 16-bit input to a (sdm_bitwidth+1)-bit output. 0 → 1-bit output.	RW	0x0
11:9	Reserved.	-	-	-
8	FIFO_POP_MASK	0 - Counter overflow events do not generate FIFO pop events 1 - Counter overflow events generate FIFO pop events	RW	0x1
7	DITHER	1 - When SDM mode is used, add a 1-bit LSB dither inside the noise shaping loop to suppress idle tones 0 - No dither applied	RW	0x0
6	SDM	1 - Use sigma-delta noise shaping modulator. In conjunction with sdm_bitwidth, treat the duty cycle as a 16-bit signed truncation of the 32-bit duty cycle value and quantise to (sdm_bitwidth+1)-bits. The resulting quantisation noise is filtered using a 2nd-order loop. 0 - Bypass modulator	RW	0x0
5	USEFIFO	1 - Use the duty_fifo. Note: setting bind=0 and usefifo=1 will lead to unpredictable operation 0 - Use the duty cycle register common_duty/chan3_duty	RW	0x0
4	BIND	1 - Bind Channel 3 to the common_range and common_duty/duty_fifo registers 0 - Channel 3 uses chan3_range and chan3_duty	RW	0x0
3	INVERT	1 - Invert the output bit	RW	0x0
2:0	MODE	PWM generation mode 0x0 → Generates 0 0x1 → Trailing-edge mark-space PWM modulation 0x2 → Phase-correct mark-space PWM modulation 0x3 → Pulse-density encoded output 0x4 → MSB Serialiser output. 0x5 → Pulse position modulated output - a single high-pulse is transmitted per cycle. 0x6 → Leading-edge mark-space PWM modulation 0x7 → LSB Serialiser output.	RW	0x0

PWM: CHAN3_RANGE Register

Offset: 0x48

Table 44. CHAN3_RANGE Register

Bits	Description	Type	Reset
31:0	Channel 3 counter range	RW	0x00000000

PWM: CHAN3_PHASE Register

Offset: 0x4c

Table 45. CHAN3_PHASE Register

Bits	Description	Type	Reset
31:0	Channel 3 counter phase offset register This register preloads the internal counter such that phase offsets between channels can be introduced. Do not set higher than the respective range register.	RW	0x00000000

PWM: CHAN3_DUTY Register

Offset: 0x50

Table 46. CHAN3_DUTY Register

Bits	Description	Type	Reset
31:0	Channel 3 counter compare register	RW	0x00000000

PWM: INTR Register

Offset: 0x54

Description

Raw Interrupts

Table 47. INTR Register

Bits	Name	Description	Type	Reset
31:9	Reserved.	-	-	-
8	CHAN3_RELOAD		WC	0x0
7	CHAN2_RELOAD		WC	0x0
6	CHAN1_RELOAD		WC	0x0
5	CHAN0_RELOAD		WC	0x0
4	DREQ_ACTIVE		RO	0x0
3	FIFO_FULL		RO	0x0
2	FIFO_EMPTY		RO	0x0
1	FIFO_OVERFLOW		WC	0x0
0	FIFO_UNDERFLOW		WC	0x0

PWM: INTE Register

Offset: 0x58

Description

Interrupt Enable

Table 48. INTE Register

Bits	Name	Description	Type	Reset
31:9	Reserved.	-	-	-
8	CHAN3_RELOAD		RW	0x0
7	CHAN2_RELOAD		RW	0x0
6	CHAN1_RELOAD		RW	0x0
5	CHAN0_RELOAD		RW	0x0
4	DREQ_ACTIVE		RW	0x0
3	FIFO_FULL		RW	0x0

Bits	Name	Description	Type	Reset
2	FIFO_EMPTY		RW	0x0
1	FIFO_OVERFLOW		RW	0x0
0	FIFO_UNDERFLOW		RW	0x0

PWM: INTF Register

Offset: 0x5c

Description

Interrupt Force

Table 49. INTF Register

Bits	Name	Description	Type	Reset
31:9	Reserved.	-	-	-
8	CHAN3_RELOAD		RW	0x0
7	CHAN2_RELOAD		RW	0x0
6	CHAN1_RELOAD		RW	0x0
5	CHAN0_RELOAD		RW	0x0
4	DREQ_ACTIVE		RW	0x0
3	FIFO_FULL		RW	0x0
2	FIFO_EMPTY		RW	0x0
1	FIFO_OVERFLOW		RW	0x0
0	FIFO_UNDERFLOW		RW	0x0

PWM: INTS Register

Offset: 0x60

Description

Interrupt status after masking & forcing

Table 50. INTS Register

Bits	Name	Description	Type	Reset
31:9	Reserved.	-	-	-
8	CHAN3_RELOAD		RO	0x0
7	CHAN2_RELOAD		RO	0x0
6	CHAN1_RELOAD		RO	0x0
5	CHAN0_RELOAD		RO	0x0
4	DREQ_ACTIVE		RO	0x0
3	FIFO_FULL		RO	0x0
2	FIFO_EMPTY		RO	0x0
1	FIFO_OVERFLOW		RO	0x0
0	FIFO_UNDERFLOW		RO	0x0

3.5. I2C

The I2C bus is a two-wire serial interface, consisting of a serial data line SDA and a serial clock SCL. These wires carry information between the devices connected to the bus. Each device is recognised by a unique address and can operate as either a transmitter or receiver, depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

NOTE

The I2C block must be programmed to operate in master or slave mode only. Operating as a master and slave simultaneously is not supported.

The I2C block can operate in these modes:

- standard mode (with data rates from 0 to 100kbps)
- fast mode (with data rates less than or equal to 400kbps)
- fast mode plus (with data rates less than or equal to 1000kbps)

These modes are not supported:

- High-speed mode (with data rates less than or equal to 3.4Mbps)
- Ultra-Fast Speed Mode (with data rates less than or equal to 5Mbps)

RP1 has seven identical instances of an I2C controller, four of which are available on GPIO bank 0. The external pins of each controller are connected to GPIO pins as defined in the GPIO muxing table. The muxing options give some IO flexibility.

NOTE

Only six of the seven I2C controllers are connected to DMA.

3.5.1. Features

Each I2C controller is based on a configuration of the Synopsys DW_apb_i2c (v2.02) IP. The following features are supported:

- Master or Slave (Default to Master mode)
- Standard mode, Fast mode or Fast mode plus
- Default slave address 0x055
- 32-element transmit buffer
- 32-element receive buffer
- Can be driven from DMA (instances 0-5 only)
- Can generate interrupts

With each controller configured as follows (each instance is fully independent):

- 32-bit APB access
- Supports Standard mode, Fast mode or Fast mode plus (not High speed)
- Default slave address of 0x055
- Master or Slave mode

DRAFT

- Master by default (Slave mode disabled at reset)
- 7-bit addressing supported in master mode
- 32 entry transmit buffer
- 32 entry receive buffer
- Allows restart conditions when a master (can be disabled for legacy device support)
- Configurable timing to adjust TsuDAT/ThDAT
- General calls responded to on reset
- Interface to DMA
- Single interrupt output
- Configurable timing to adjust clock frequency
- Spike suppression (default 7 clk_sys cycles)
- Can NACK after data received by Slave
- Hold transfer when TX FIFO empty
- Hold bus until space available in RX FIFO
- Restart detect interrupt in Slave mode
- Optional blocking Master commands (not enabled by default)
- Support SMBus mode

3.5.1.1. Standard

The I2C controller was designed for I2C Bus specification, version 6.0, dated April 2014.

3.5.1.2. Clocking

All clocks in the I2C controller are connected to `clk_sys`, including `ic_clk` which is mentioned in later sections. The I2C clock is generated by dividing down this clock, controlled by registers inside the block.

3.5.1.3. IOs

Each controller must connect its clock `SCL` and data `SDA` to one pair of GPIOs. The I2C standard requires that drivers drive a signal low, or when not driven the signal will be pulled high. This applies to SCL and SDA. The GPIO pads should be configured for:

- pull-up enable
- slew rate limited
- Schmitt trigger enabled.

3.5.2. Register base addresses

There are seven instances of the I2C peripheral.

Table 51. Peripheral address map

Block	Bus Type	Atomic Access	Address
i2c0	APB	N	`0x40070000`

i2c1	APB	N	0x40074000
i2c2	APB	N	0x40078000
i2c3	APB	N	0x4007c000
i2c4	APB	N	0x40080000
i2c5	APB	N	0x40084000
i2c6	APB	N	0x40088000

For a reference implementation see the Designware I2C Linux kernel driver.

3.6. SPI

RP1 has nine Synchronous Serial Interface (SSI) controllers, six of which are available on GPIO bank 0. Each SSI controller is based on a configuration of the Synopsys DW_apb_ssi IP (v4.02a).

3.6.1. Instance configurations

Instances are statically configured as either master or slave peripherals, with variable numbers of connected I/O and chip-select lines. The master configuration can operate in standard SPI, dual-SPI or quad-SPI modes, depending on the number of available I/O lines. The slave configuration only operates in standard SPI mode.

DDR operation is not supported.

Instance ID	Master/Slave	Chip-select count	Max I/O width
SPI0	M	4	Quad
SPI1	M	3	Dual
SPI2	M	2	Dual
SPI3	M	2	Dual
SPI4	S	1	Single
SPI5	M	2	Dual
SPI6	M	3	Dual
SPI7	S	1	Single
SPI8	M	2	Dual

3.6.1.1. IO connections

On master instances:

- `SSI_CLK` Connected to output clock *spiX_sclk*
- `SS_[Y]_N` Chip-select output connected to pad output _spiX_ss_n[y]_
- `TXD[3:0]` and `RXD[3:0]` Transmit/receive data is connected to pad signals *spiX_sio[3:0]*

On slave instances:

- `SCLK_IN` Connected to input clock *spiX_sclk*

- `SS_IN_N` Connected to chip select input *spiX_ss_n[0]*
- `TXD` Connected to pad output *spiX_sio[0]*
- `RXD` Connected to pad input *spiX_sio[1]*

Master clock connections are as follows:

- *pclk* and *sclk* are driven from `clk_sys`

Slave clocks on `SCLK_IN` are synchronised to `clk_sys`.

3.6.2. Register base addresses

There are nine instances of the SSI peripheral.

Table 52. Peripheral address map

Block	Bus Type	Atomic Access	Address
spi8	APB	N	`0x4004c000`
spi0	APB	N	`0x40050000`
spi1	APB	N	`0x40054000`
spi2	APB	N	`0x40058000`
spi3	APB	N	`0x4005c000`
spi4	APB	N	`0x40060000`
spi5	APB	N	`0x40064000`
spi6	APB	N	`0x40068000`
spi7	APB	N	`0x4006c000`

For a reference implementation see the Designware SPI/SSI Linux kernel driver.

3.7. I2S

RP1 has three instances of the Synopsys Designware I2S peripheral, revision 1.11a, two of which are available on GPIO bank 0. Each I2S instance can operate in a bidirectional mode with a configurable number of channel pairs.

3.7.1. Instance configurations

- I2S0 is a clock-producer (master) with up to 4 bidirectional channels
- I2S1 is a clock-consumer (slave) with up to 4 bidirectional channels
- I2S2 is a clock-consumer (slave) with up to 2 bidirectional channels

I2S0 and I2S1 occupy the same set of GPIO pins in the mux map table. This is required as the clock direction, input or output, is statically configured. Users should select the I2S instance according to their attached codec.

- Each channel's I2S data receive pin is connected to `sdi[n]`
- Each channel's I2S data transmit pin is connected to `sdo[n]`
- Each instance consumes two DMA handshake ports on the DMAC - one for all RX channels and one for all TX channels
- Tx and Rx data FIFOs have been configured for a separate and overall depth of 16

- Each instance has a single top-level interrupt output for CPU-driven operation
- Maximum audio channel data resolution is 32 bits, i.e. `I2S_RX_WORDSIZE_n=32` and `I2S_TX_WORDSIZE_n=32`

3.7.2. Register base addresses

There are three instances of the I2S peripheral.

Table 53. Peripheral address map

Block	Bus Type	Atomic Access	Address
i2s0	APB	N	`0x400a0000`
i2s1	APB	N	`0x400a4000`
i2s2	APB	N	`0x400a8000`

For a reference implementation see the Synopsys I2S Linux kernel driver.

3.8. TICKS

Tick generators are low-frequency timing sources that provide events for several other subsystems.

The ticks are independently controllable and use `clk_ref` (nominally `XOSC` frequency) as the clock source. The tick output is a 9-bit integer division of this clock, and the divisor can be modified at runtime.

`TICK_DMA0` and `TICK_DMA1` are routed to dummy DREQ generators in the `DMA_TICK` block to generate paced transfer requests for DMA. `TICK_IOBANK0` provides the timebase for debounce and filtering features of the GPIO interrupt generators.

NOTE

Other tick instances are reserved for internal use.

3.8.1. List of registers

Table 54. List of TICKS registers

Offset	Name	Info
0x00	TIMER_CTRL	Controls the tick generator
0x04	TIMER_CYCLES	
0x08	TIMER_COUNT	
0x0c	WATCHDOG_CTRL	Controls the tick generator
0x10	WATCHDOG_CYCLES	
0x14	WATCHDOG_COUNT	
0x18	PROC_CTRL	Controls the tick generator
0x1c	PROC_CYCLES	
0x20	PROC_COUNT	
0x24	DMA0_CTRL	Controls the tick generator
0x28	DMA0_CYCLES	
0x2c	DMA0_COUNT	
0x30	DMA1_CTRL	Controls the tick generator

DRAFT

Offset	Name	Info
0x34	DMA1_CYCLES	
0x38	DMA1_COUNT	
0x3c	IO_BANK0_CTRL	Controls the tick generator
0x40	IO_BANK0_CYCLES	
0x44	IO_BANK0_COUNT	
0x48	IO_BANK1_CTRL	Controls the tick generator
0x4c	IO_BANK1_CYCLES	
0x50	IO_BANK1_COUNT	
0x54	IO_BANK2_CTRL	Controls the tick generator
0x58	IO_BANK2_CYCLES	
0x5c	IO_BANK2_COUNT	

TICKS: TIMER_CTRL Register

Offset: 0x00

Description

Controls the tick generator

Table 55. TIMER_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: TIMER_CYCLES Register

Offset: 0x04

Table 56. TIMER_CYCLES Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Total number of clk_tick cycles before the next tick.	RW	0x000

TICKS: TIMER_COUNT Register

Offset: 0x08

Table 57. TIMER_COUNT Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: WATCHDOG_CTRL Register

Offset: 0x0c

Description

Controls the tick generator

Table 58. WATCHDOG_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: WATCHDOG_CYCLES Register

Offset: 0x10

Table 59. WATCHDOG_CYCLES Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Total number of clk_tick cycles before the next tick.	RW	0x000

TICKS: WATCHDOG_COUNT Register

Offset: 0x14

Table 60. WATCHDOG_COUNT Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: PROC_CTRL Register

Offset: 0x18

Description

Controls the tick generator

Table 61. PROC_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: PROC_CYCLES Register

Offset: 0x1c

Table 62. PROC_CYCLES Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Total number of clk_tick cycles before the next tick.	RW	0x000

TICKS: PROC_COUNT Register

Offset: 0x20

Table 63. PROC_COUNT Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: DMA0_CTRL Register

Offset: 0x24

Description

Controls the tick generator

Table 64. DMA0_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: DMA0_CYCLES Register

Offset: 0x28

Table 65. DMA0_CYCLES Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Total number of clk_tick cycles before the next tick.	RW	0x000

TICKS: DMA0_COUNT Register

Offset: 0x2c

Table 66. DMA0_COUNT Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: DMA1_CTRL Register

Offset: 0x30

Description

Controls the tick generator

Table 67. DMA1_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: DMA1_CYCLES Register

Offset: 0x34

Table 68. DMA1_CYCLES Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Total number of clk_tick cycles before the next tick.	RW	0x000

TICKS: DMA1_COUNT Register

Offset: 0x38

Table 69. DMA1_COUNT Register

Bits	Description	Type	Reset
31:9	Reserved.	-	-
8:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: IO_BANK0_CTRL Register

Offset: 0x3c

Description

Controls the tick generator

Table 70. IO_BANK0_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: IO_BANK0_CYCLES Register

Offset: 0x40

Table 71. IO_BANK0_CYCLES Register

Bits	Description	Type	Reset
31:28	Reserved.	-	-
27:0	Total number of clk_tick cycles before the next tick.	RW	0x0000000

TICKS: IO_BANK0_COUNT Register

Offset: 0x44

Table 72. IO_BANK0_COUNT Register

Bits	Description	Type	Reset
31:28	Reserved.	-	-
27:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: IO_BANK1_CTRL Register

Offset: 0x48

Description

Controls the tick generator

Table 73. IO_BANK1_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: IO_BANK1_CYCLES Register

Offset: 0x4c

Table 74. IO_BANK1_CYCLES Register

Bits	Description	Type	Reset
31:28	Reserved.	-	-
27:0	Total number of clk_tick cycles before the next tick.	RW	0x0000000

TICKS: IO_BANK1_COUNT Register

Offset: 0x50

Table 75. IO_BANK1_COUNT Register

Bits	Description	Type	Reset
31:28	Reserved.	-	-
27:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

TICKS: IO_BANK2_CTRL Register

Offset: 0x54

Description

Controls the tick generator

Table 76. IO_BANK2_CTRL Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	RUNNING	Is the tick generator running?	RO	-
0	ENABLE	start / stop tick generation	RW	0x0

TICKS: IO_BANK2_CYCLES Register

Offset: 0x58

Table 77. IO_BANK2_CYCLES Register

Bits	Description	Type	Reset
31:28	Reserved.	-	-
27:0	Total number of clk_tick cycles before the next tick.	RW	0x0000000

TICKS: IO_BANK2_COUNT Register

Offset: 0x5c

Table 78. IO_BANK2_COUNT Register

Bits	Description	Type	Reset
31:28	Reserved.	-	-
27:0	Count down timer: the remaining number clk_tick cycles before the next tick is generated.	RO	-

3.8.1.1. DMA_TICK

This register block allows for fine-tuning of the tick-to-DREQ interfaces.

- Whether a dma_finish strobe from the DMAC stops tick generation
- Whether dma_req or both dma_req and dma_single are asserted on each tick
- Optional idle/dwell cycle insertion

Table 79. List of DMA_TICK registers

Offset	Name	Info
0x0	TICK0_EN	These bits self-clear to 0 if the tick generator is stopped by a dma_finish or abort.
0x4	TICK0_CTRL	
0x8	TICK1_EN	These bits self-clear to 0 if the tick generator is stopped by a dma_finish or abort.
0xc	TICK1_CTRL	

DMA_TICK: TICK0_EN Register

Offset: 0x0

Description

These bits self-clear to 0 if the tick generator is stopped by a dma_finish or abort.

Table 80. TICK0_EN Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	SINGLE	1 - If enable=1, generate dma_single requests as well as burst requests	RWF	0x0
0	REQ	1 - Enable generation of handshakes on tick0 output.	RWF	0x0

DMA_TICK: TICK0_CTRL Register

Offset: 0x4

Table 81. TICK0_CTRL Register

Bits	Name	Description	Type	Reset
31:13	Reserved.	-	-	-
12	DREQ	Current state of the dreq input (debug only, asserted for 1 cycle)	RO	0x0
11:9	Reserved.	-	-	-
8:4	DWELL	Handshake state machine idle dwell period in bus-clk cycles	RW	0x00
3:2	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
1	DISABLE	Write-1 to the SET alias to force-disable the handshake - NB: will race with any DMAC activity. Use with caution.	SAC	0x0
0	FINISH_CLEAR	1 - clear the enable and single registers when the DMAC asserts dma_finish	RW	0x0

DMA_TICK: TICK1_EN Register

Offset: 0x8

Description

These bits self-clear to 0 if the tick generator is stopped by a dma_finish or abort.

Table 82. TICK1_EN Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1	SINGLE	1 - If enable=1, generate dma_single requests as well as burst requests	RWF	0x0
0	REQ	1 - Enable generation of handshakes on tick1 output	RWF	0x0

DMA_TICK: TICK1_CTRL Register

Offset: 0xc

Table 83. TICK1_CTRL Register

Bits	Name	Description	Type	Reset
31:13	Reserved.	-	-	-
12	DREQ	Current state of the dreq input (debug only, asserted for 1 cycle)	RO	0x0
11:9	Reserved.	-	-	-
8:4	DWELL	Handshake state machine idle dwell period in bus-clk cycles	RW	0x00
3:2	Reserved.	-	-	-
1	DISABLE	Write-1 to the SET alias to force-disable the handshake - NB: will race with any DMAC activity. Use with caution.	SAC	0x0
0	FINISH_CLEAR	1 - clear the enable and single registers when the DMAC asserts dma_finish	RW	0x0

Chapter 4. SDIO

The SDIO controller is a Synopsys MSHC peripheral v1.70a. There are two separate, identical instances which each support SDIO v4.2 and eMMC v5.1. One instance is available on GPIO bank 0.

The peripheral is compliant with the SD Host Controller specification v4.20. Users should refer to the SD Association's Physical Specification and Host Controller Specification for the programmer's model.

4.1. Configuration and feature set

The SDIO controller has been configured with:

- AXI master port, AHB configuration port
- Operation up to 50MHz DDR (UHS-I DDR50)
- 4-bit bus interface width
- Integrated descriptor DMA engine supporting SDMA/ADMA2/ADMA3
- 16-phase clock-to-data skew tuning engine
- Mode 1 retuning (software-initiated)

These features are not supported:

- Command queueing
- Dedicated card-detect pin
- UHS-II interface
- eMMC Boot Protocol

Tuning support is provided by a custom DLL IP that uses the VCO output from PLL_SYS at 1GHz to provide 10 or 20 clock phases. The 16-way phase select from the controller tuning engine is mapped on to the available phases by a linear compression or expansion, depending on the selected mode.

4.2. Supported speed modes

SD 3.3V Legacy Speeds:

- DS25 (25MHz SDR)
- HS50 (50MHz SDR)

UHS-I 1.8V speed grades:

- SDR12/SDR25 (25/50MHz SDR)
- DDR50 (50MHz DDR)

eMMC 1.8V speeds:

- High speed SDR (50Mhz SDR)
- High speed DDR (50MHz DDR)

DRAFT

NOTE

The eMMC standard specifies 52MHz maximum for high-speed modes, but the controller has a maximum frequency of 50MHz.

NOTE

The SDIO clock generator can be configured to output higher card clocks, but operation in a pseudo-SDR104 mode is not guaranteed.

4.3. Register base addresses

There are two instances of the MSHC peripheral.

Table 84. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
sdio0_ip	AHB	N	0x40180000
sdio1_ip	AHB	N	0x40184000

DRAFT

Chapter 5. USB

The USB Host subsystem is based on Synopsys IP dwc_usb3, v3.30b. There are two identical USB3.0 xHCI Host Controllers conforming to the Extensible Host Controller Interface Specification v1.2. Each controller has two downstream ports, implemented with one USB2.0 PHY and one USB3.0 PHY. The USB 3.0 ports are backward-compatible with USB 2.0, so up to four USB 2.0 or two USB 2.0 and two USB 3.0 connections may be supported.

The controllers are configured with two USB2.0 HS/FS/LS Bus Instances and a dedicated SuperSpeed Bus Instance, so every downstream port has independent and uncontended bandwidth. The controllers each support 64 device slots and up to 64 periodic endpoints. The controllers have been configured for high system latency tolerance; SuperSpeed and High Speed FIFO sizes have been increased from their defaults to eight and three packets respectively, and TRB caches are sized accordingly. Four interrupt vectors are provided per controller.

The controllers do not support:

- Battery Charging (ACA) functionality
- xHCI Debug Port Capability
- PHY VBUS sensing
- Device-mode operation.

5.1. Register base addresses

There are two instances of the USB xHCI peripheral.

Table 85. Peripheral address map

Block	Bus Type	Atomic Access	Address
usbhost0_cfg	APB	Y	0x40160000
usbhost1_cfg	APB	Y	0x40164000
usbhost0	AXI3	N	0x40200000
usbhost1	AXI3	N	0x40300000

DRAFT

Chapter 6. PCI Express endpoint controller

The internal features of RP1 communicate with the AP via a PCI Express link. PCIe is a symmetric serial protocol using an aggregate of individual lanes, which operate in tandem to form a link. RP1 supports up to a Gen 2.0 4-lane link configuration.

Communication over the PCIe link is handled by a Synopsys Designware PCIe Endpoint Controller (v5.30a). This controller handles inbound (downstream) requests from Root Complexes, and forwards outbound (upstream) requests from internal bus masters. The maximum unidirectional link bandwidth is 14.7Gbit/s, and close to full-duplex bidirectional bandwidth can be achieved.

The PCIe EP controller has been configured with:

- Gen 2 four-lane PHY with automatic or manual lane flip and reversal
- Data-link layer MTU of 256 bytes
- Single Function, single Virtual Channel
- Three Base Address Registers (32-bit non-prefetchable)
- Integrated MSIx capability
- Advanced Error Reporting capability
- Tolerance of Read Completion TLPs sized up to the MTU size
- ASPM L0s+L1 with Reference Clock Removal
- AXI3 bridge interface (40-bit address bus, 128-bit data bus)
- Four configurable inbound address translation windows (4kB to 4GB)
- Four configurable outbound address translation windows (4kB to 4GB)
- Maximum 32 pending inbound non-posted transactions
- Maximum 32 pending outbound non-posted transactions

Other features of the PCIe EP block include:

- A priority-forwarding mechanism that allows bus masters to signal elevated QoS requests on ARQOS/AWQOS, or instead via configurable heuristics
- Priority requests are transmitted as a Vendor Defined Message to the RC
- Configurable per-master Traffic Classes allowing relaxation of PCIe data ordering rules for masters with independent traffic streams
- A configuration interface that allows control over most PCI space register defaults - e.g. to shrink the configured link width, selectively disable ASPM, resize BARs
- An interrupt vector configuration block that allows for reliable generation of MSIx messages from level-sensitive interrupt sources
- Power-management controls for management firmware to coordinate entry and exit of low-power chip states

6.1. PCIe endpoint configuration registers

These registers allow management firmware to handle link-up/link-down events, control powersave features, monitor link state, and for host software to configure MSIx vectors.

DRAFT

Table 86. List of PCIE_CFG registers

Offset	Name	Info
0x000	DBI	address fields for dbi access
0x004	CONTROL	miscellaneous control bits
0x008	MSIX_CFG_0	msix configuration
0x00c	MSIX_CFG_1	msix configuration
0x010	MSIX_CFG_2	msix configuration
0x014	MSIX_CFG_3	msix configuration
0x018	MSIX_CFG_4	msix configuration
0x01c	MSIX_CFG_5	msix configuration
0x020	MSIX_CFG_6	msix configuration
0x024	MSIX_CFG_7	msix configuration
0x028	MSIX_CFG_8	msix configuration
0x02c	MSIX_CFG_9	msix configuration
0x030	MSIX_CFG_10	msix configuration
0x034	MSIX_CFG_11	msix configuration
0x038	MSIX_CFG_12	msix configuration
0x03c	MSIX_CFG_13	msix configuration
0x040	MSIX_CFG_14	msix configuration
0x044	MSIX_CFG_15	msix configuration
0x048	MSIX_CFG_16	msix configuration
0x04c	MSIX_CFG_17	msix configuration
0x050	MSIX_CFG_18	msix configuration
0x054	MSIX_CFG_19	msix configuration
0x058	MSIX_CFG_20	msix configuration
0x05c	MSIX_CFG_21	msix configuration
0x060	MSIX_CFG_22	msix configuration
0x064	MSIX_CFG_23	msix configuration
0x068	MSIX_CFG_24	msix configuration
0x06c	MSIX_CFG_25	msix configuration
0x070	MSIX_CFG_26	msix configuration
0x074	MSIX_CFG_27	msix configuration
0x078	MSIX_CFG_28	msix configuration
0x07c	MSIX_CFG_29	msix configuration
0x080	MSIX_CFG_30	msix configuration
0x084	MSIX_CFG_31	msix configuration
0x088	MSIX_CFG_32	msix configuration
0x08c	MSIX_CFG_33	msix configuration

Offset	Name	Info
0x090	MSIX_CFG_34	msix configuration
0x094	MSIX_CFG_35	msix configuration
0x098	MSIX_CFG_36	msix configuration
0x09c	MSIX_CFG_37	msix configuration
0x0a0	MSIX_CFG_38	msix configuration
0x0a4	MSIX_CFG_39	msix configuration
0x0a8	MSIX_CFG_40	msix configuration
0x0ac	MSIX_CFG_41	msix configuration
0x0b0	MSIX_CFG_42	msix configuration
0x0b4	MSIX_CFG_43	msix configuration
0x0b8	MSIX_CFG_44	msix configuration
0x0bc	MSIX_CFG_45	msix configuration
0x0c0	MSIX_CFG_46	msix configuration
0x0c4	MSIX_CFG_47	msix configuration
0x0c8	MSIX_CFG_48	msix configuration
0x0cc	MSIX_CFG_49	msix configuration
0x0d0	MSIX_CFG_50	msix configuration
0x0d4	MSIX_CFG_51	msix configuration
0x0d8	MSIX_CFG_52	msix configuration
0x0dc	MSIX_CFG_53	msix configuration
0x0e0	MSIX_CFG_54	msix configuration
0x0e4	MSIX_CFG_55	msix configuration
0x0e8	MSIX_CFG_56	msix configuration
0x0ec	MSIX_CFG_57	msix configuration
0x0f0	MSIX_CFG_58	msix configuration
0x0f4	MSIX_CFG_59	msix configuration
0x0f8	MSIX_CFG_60	msix configuration
0x0fc	MSIX_CFG_61	msix configuration
0x100	MSIX_CFG_62	msix configuration
0x104	MSIX_CFG_63	msix configuration
0x108	INTSTATL	Raw interrupt status [31:0]
0x10c	INTSTATH	Raw interrupt status [63:32]
0x110	PHY_TEST	address fields for pcie phy_test access
0x114	PHY_PARAM_CTRL0	phy parameter control0
0x118	PHY_PARAM_CTRL1	phy parameter control1
0x11c	PHY_CR_ACC_CTRL	phy control-register access - control

Offset	Name	Info
0x120	PHY_CR_ACC_DATA	phy control-register access - read/write data
0x124	LTSSM_STATE_NEW	new ltssm state
0x12c	LTSSM_STATE_FILTER_0	ltssm_state filter0
0x130	LTSSM_STATE_FILTER_1	ltssm_state filter1
0x134	LTSSM_STATE_FILTER_2	ltssm_state filter2
0x138	LTSSM_STATE_FILTER_3	ltssm_state filter3
0x140	AXI_LOOKUP_0	
0x144	AXI_LOOKUP_1	
0x148	AXI_LOOKUP_2	
0x14c	AXI_LOOKUP_3	
0x150	AXI_LOOKUP_4	
0x154	AXI_LOOKUP_5	
0x158	AXI_LOOKUP_6	
0x15c	AXI_LOOKUP_7	
0x160	AXI_LOOKUP_8	
0x164	AXI_LOOKUP_9	
0x168	AXI_LOOKUP_10	
0x16c	AXI_LOOKUP_11	
0x170	AXI_LOOKUP_12	
0x174	AXI_LOOKUP_13	
0x178	AXI_LOOKUP_14	
0x17c	AXI_LOOKUP_15	
0x180	VDM_CONFIG_REG0	vdm config_reg0
0x184	VDM_CONFIG_REG1	vdm config_reg1
0x188	VDM_PANIC_REG	vdm panic_reg
0x18c	VDM_TEST	vdm test requesters
0x190	VDM_HEADER	vendor defined message header
0x194	PM_CONTROL	power management control
0x198	PM_STATUS	power management status
0x19c	MONITOR0	internal signals for debug
0x1a0	MONITOR1	internal signals for debug
0x1a4	MONITOR2	internal signals for debug
0x1a8	INTR	Raw Interrupts
0x1ac	INTE	Interrupt Enable
0x1b0	INTF	Interrupt Force
0x1b4	INTS	Interrupt status after masking & forcing

PCIE_CFG: DBI Register

Offset: 0x000

Description

address fields for dbi access

Table 87. DBI Register

Bits	Name	Description	Type	Reset
31:14	Reserved.	-	-	-
13:5	ATU_REGION	iatu access: region number,dir	RW	0x000
4:2	FUNC_NUM	cdm access: function number	RW	0x0
1:0	ASEL	b0:cs2, b1:cdm/elbi select; 00=normal/misc; 01=shadow; 10=elbi; 11=iatu/dma	RW	0x0

PCIE_CFG: CONTROL Register

Offset: 0x004

Description

miscellaneous control bits

Table 88. CONTROL Register

Bits	Name	Description	Type	Reset
31:10	Reserved.	-	-	-
9	MEM_PD	Power down all PCIe memories	RW	0x0
8	CPERSTN_ASSERT		RW	0x0
7	CPERSTN_DISABLE	gate cperstn source	RW	0x0
6:4	CLK2FC_SEL	select a clock to be sent to the frequency counter 0x7 → frequency 250MHz 0x6 → frequency 250MHz 0x5 → frequency 125MHz 0x4 → frequency 100MHz 0x3 → frequency 125/250MHz 0x2 → frequency 125MHz 0x1 → frequency variable 0x0 → NONE	RW	0x0
3	SET_HOT_RST_APP_LTSSM_ENABLE	enables link training/establishment following hot reset request. Self clearing.	SC	0x0
2	APP_LTSSM_ENABLE	enable link training/establishment	RW	0x1
1	APP_DBI_RO_WR_DISABLE	dbi read-only write disable; disables writing to misc_control_1:dbi_ro_wr_en field	RW	0x0
0	APP_REQ_RETRY_EN	defer upstream pcie config requests; requests completed with a retry status	RW	0x0

PCIE_CFG: MSIX_CFG_0, MSIX_CFG_1, ..., MSIX_CFG_62, MSIX_CFG_63 Registers

Offsets: 0x008, 0x00c, ..., 0x100, 0x104

Description

msix configuration

Table 89. MSIX_CFG_0, MSIX_CFG_1, ..., MSIX_CFG_62, MSIX_CFG_63 Registers

Bits	Name	Description	Type	Reset
31:19	Reserved.	-	-	-
18:16	FUNC	pcie function	RW	0x0
15	Reserved.	-	-	-
14:12	TC	pcie traffic class	RW	0x0
11:4	Reserved.	-	-	-
3	IACK_EN	enable iack functionality	RW	0x0
2	IACK	Interrupt acknowledge. Writing a 1 clears the interrupt mask that was automatically set when the interrupt was generated. Self clearing bit	SC	0x0
1	TEST	ORed with interrupt source for test purposes	RW	0x0
0	ENABLE	interrupt enable	RW	0x0

PCIE_CFG: INTSTATL Register

Offset: 0x108

Description

Raw interrupt status [31:0]

Table 90. INTSTATL Register

Bits	Name	Description	Type	Reset
31:0	INTSTAT	Raw interrupt status	RO	0x00000000

PCIE_CFG: INTSTATH Register

Offset: 0x10c

Description

Raw interrupt status [63:32]

Table 91. INTSTATH Register

Bits	Name	Description	Type	Reset
31:0	INTSTAT	Raw interrupt status	RO	0x00000000

PCIE_CFG: PHY_TEST Register

Offset: 0x110

Description

address fields for pcie phy_test access

Table 92. PHY_TEST Register

Bits	Name	Description	Type	Reset
31:3	Reserved.	-	-	-
2	BURNIN	All circuits activator	RW	0x0
1	BYPASS	All Circuits Power-Down but Leave Reference Clocks Active	RW	0x0

DRAFT

Bits	Name	Description	Type	Reset
0	POWERDOWN	All Circuits Power-Down Control in the PHY for IDDQ testing	RW	0x0

PCIE_CFG: PHY_PARAM_CTRL0 Register

Offset: 0x114

Description

phy parameter control0

Table 93. PHY_PARAM_CTRL0 Register

Bits	Name	Description	Type	Reset
31:27	PHY_TX3_TERM_OFFSET	tx3 termination offset	RW	0x00
26:24	PHY_RX3_EQ	rx3 equalizer	RW	0x2
23:19	PHY_TX2_TERM_OFFSET	tx2 termination offset	RW	0x00
18:16	PHY_RX2_EQ	rx2 equalizer	RW	0x2
15:11	PHY_TX1_TERM_OFFSET	tx1 termination offset	RW	0x00
10:8	PHY_RX1_EQ	rx1 equalizer	RW	0x2
7:3	PHY_TX0_TERM_OFFSET	tx0 termination offset	RW	0x00
2:0	PHY_RX0_EQ	rx0 equalizer	RW	0x2

PCIE_CFG: PHY_PARAM_CTRL1 Register

Offset: 0x118

Description

phy parameter control1

Table 94. PHY_PARAM_CTRL1 Register

Bits	Name	Description	Type	Reset
31:25	PCS_TX_SWING_LOW	tx amplitude - low swing mode	RW	0x73
24:18	PCS_TX_SWING_FULL	tx amplitude - full swing mode	RW	0x73
17:12	PCS_TX_DEEMPH_GEN1	gen2 tx deemphasis at 6.0db	RW	0x18
11:6	PCS_TX_DEEMPH_GEN2_6DB	gen2 tx deemphasis at 6.0db	RW	0x21
5:0	PCS_TX_DEEMPH_GEN2_3P5DB	gen2 tx deemphasis at 3.5db	RW	0x18

PCIE_CFG: PHY_CR_ACC_CTRL Register

Offset: 0x11c

Description

phy control-register access - control

Table 95. PHY_CR_ACC_CTRL Register

Bits	Name	Description	Type	Reset
31:20	Reserved.	-	-	-
19	PHY_PIPE_RST	gated into phy pcs/pipe reset - advisable to assert during phy control-register writes	RW	0x0
18	BUSY	interface busy	RO	0x0
17	READ	register read command - self-clearing	SC	0x0
16	WRITE	register write command - self clearing	SC	0x0
15:0	ADDR	register address	RW	0x0000

PCIE_CFG: PHY_CR_ACC_DATA Register

Offset: 0x120

Description

phy control-register access - read/write data

Table 96. PHY_CR_ACC_DATA Register

Bits	Name	Description	Type	Reset
31:16	RDATA	register read data	RO	0x0000
15:0	WDATA	register write data	RW	0x0000

PCIE_CFG: LTSSM_STATE_NEW Register

Offset: 0x124

Description

new ltssm state

Table 97. LTSSM_STATE_NEW Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-
5:0	STATE	new ltssm_state	RF	0x00

PCIE_CFG: LTSSM_STATE_FILTER_0 Register

Offset: 0x12c

Description

ltssm_state filter0

Table 98. LTSSM_STATE_FILTER_0 Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-
5:0	VAL	ltssm state filter - filter out state changes to this state.	RW	0x3f

PCIE_CFG: LTSSM_STATE_FILTER_1 Register

Offset: 0x130

Description

ltssm_state filter1

Table 99. LTSSM_STATE_FILTER_1 Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
5:0	VAL	ltssm state filter - filter out state changes to this state.	RW	0x3f

PCIE_CFG: LTSSM_STATE_FILTER_2 Register

Offset: 0x134

Description

ltssm_state filter2

Table 100. LTSSM_STATE_FILTER_2 Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-
5:0	VAL	ltssm state filter - filter out state changes to this state.	RW	0x3f

PCIE_CFG: LTSSM_STATE_FILTER_3 Register

Offset: 0x138

Description

ltssm_state filter3

Table 101. LTSSM_STATE_FILTER_3 Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-
5:0	VAL	ltssm state filter - filter out state changes to this state.	RW	0x3f

PCIE_CFG: AXI_LOOKUP_0 Register

Offset: 0x140

Table 102. AXI_LOOKUP_0 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_1 Register

Offset: 0x144

Table 103. AXI_LOOKUP_1 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_2 Register

Offset: 0x148

Table 104. AXI_LOOKUP_2 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_3 Register

Offset: 0x14c

Table 105. AXI_LOOKUP_3 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_4 Register

Offset: 0x150

Table 106. AXI_LOOKUP_4 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_5 Register

Offset: 0x154

Table 107. AXI_LOOKUP_5 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_6 Register

Offset: 0x158

Table 108. AXI_LOOKUP_6 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_7 Register

Offset: 0x15c

Table 109. AXI_LOOKUP_7 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_8 Register

Offset: 0x160

Table 110. AXI_LOOKUP_8 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_9 Register

Offset: 0x164

Table 111. AXI_LOOKUP_9 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_10 Register

Offset: 0x168

Table 112. AXI_LOOKUP_10 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_11 Register

Offset: 0x16c

Table 113. AXI_LOOKUP_11 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_12 Register

Offset: 0x170

Table 114. AXI_LOOKUP_12 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_13 Register

Offset: 0x174

Table 115. AXI_LOOKUP_13 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_14 Register

Offset: 0x178

Table 116. AXI_LOOKUP_14 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: AXI_LOOKUP_15 Register

Offset: 0x17c

Table 117. AXI_LOOKUP_15 Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6:4	VDM_TC	vdm traffic class	RW	0x0
3	Reserved.	-	-	-
2:0	TLP_TC	tlp traffic class	RW	0x0

PCIE_CFG: VDM_CONFIG_REG0 Register

Offset: 0x180

Description

vdm config_reg0

Table 118. VDM_CONFIG_REG0 Register

Bits	Name	Description	Type	Reset
31:0	PAYLOAD	message payload	RW	0x00000000

PCIE_CFG: VDM_CONFIG_REG1 Register

Offset: 0x184

Description

vdm config_reg1

Table 119. VDM_CONFIG_REG1 Register

Bits	Name	Description	Type	Reset
31:1	Reserved.	-	-	-
0	REQ	message request - write a 1 to generate config message - self clearing	SC	0x0

PCIE_CFG: VDM_PANIC_REG Register

Offset: 0x188

Description

vdm panic_reg

Table 120. VDM_PANIC_REG Register

Bits	Name	Description	Type	Reset
31:8	PAYLOAD	message payload [31:8]	RW	0x000000
7:0	Reserved.	-	-	-

PCIE_CFG: VDM_TEST Register

Offset: 0x18c

Description

vdm test requesters

Table 121. VDM_TEST Register

Bits	Name	Description	Type	Reset
31:16	Reserved.	-	-	-
15	REQ3_PANIC	req3 panic bit: a state change triggers vdm message	RW	0x0
14:12	REQ3_TC	req3 traffic class	RW	0x0
11	REQ2_PANIC	req2 panic bit: a state change triggers vdm message	RW	0x0
10:8	REQ2_TC	req2 traffic class	RW	0x0
7	REQ1_PANIC	req1 panic bit: a state change triggers vdm message	RW	0x0
6:4	REQ1_TC	req1 traffic class	RW	0x0
3	REQ0_PANIC	req0 panic bit: a state change triggers vdm message	RW	0x0
2:0	REQ0_TC	req0 traffic class	RW	0x0

PCIE_CFG: VDM_HEADER Register

Offset: 0x190

Description

vendor defined message header

DRAFT

Table 122. VDM_HEADER Register

Bits	Name	Description	Type	Reset
31	Reserved.	-	-	-
30:23	CODE	code	RW	0x7f
22:15	TAG	tag	RW	0x00
14:12	FUNC_NUM	function number	RW	0x0
11:10	ATTR	attribute	RW	0x0
9	TD	tlp digest	RW	0x0
8	EP	error poisoned	RW	0x0
7:5	TC	traffic class	RW	0x0
4:0	TYPE	type; fmt hardwired to 3'b001; default: Msg	RW	0x10

PCIE_CFG: PM_CONTROL Register

Offset: 0x194

Description

power management control

Table 123. PM_CONTROL Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6	APP_CLK_REQ_N	refclk required - set if refclk can be removed	RW	0x0
5	APPS_PM_XMT_PME	request L1/L2 PM exit - self clearing PME support must be enabled (sends PME message)	SC	0x0
4	APP_REQ_ENTR_L1	request L1-ASPM entry - self clearing	SC	0x0
3	APP_READY_ENTR_L23	ready to enter L2/L3 state	RW	0x1
2	APP_REQ_EXIT_L1	request L1 exit	RW	0x0
1	APP_XFER_PENDING	indicates app has transfers pending, and so inhibits entry to L1-ASPM note: assertion does not prevent entry to L1-PM, but does cause immediate exit	RW	0x1
0	APP_CLK_PM_EN	enable clock PM feature	RW	0x0

PCIE_CFG: PM_STATUS Register

Offset: 0x198

Description

power management status

Table 124. PM_STATUS Register

Bits	Name	Description	Type	Reset
31:22	Reserved.	-	-	-
21	PM_XTLH_BLOCK_TLP	indicates app must stop generating new outgoing requests TLPs due to PM state	RO	0x0

Bits	Name	Description	Type	Reset
20:18	PM_CURNT_STATE	for debug: current link state 0 : L0 1 : L0s 2 : L1 3 : L23RDY, aux power 4 : L23RDY, no aux power	RO	0x0
17:13	PM_MASTER_STATE	curent state of PM master controller	RO	0x00
12:8	PM_SLAVE_STATE	curent state of PM slave controller	RO	0x00
7:5	PM_DSTATE	current d-state	RO	0x0
4	PM_LINKST_L2_EXIT	PM exiting L2 state	RO	0x0
3	PM_LINKST_IN_L2	PM in L2 state	RO	0x0
2	PM_LINKST_IN_L1	PM in L1 state	RO	0x0
1	PM_L1_ENTRY_STARTED	L1 entry process is on progress	RO	0x0
0	PM_LINKST_IN_L0S	PM in L0s state	RO	0x0

PCIE_CFG: MONITOR0 Register

Offset: 0x19c

Description

internal signals for debug

Table 125. MONITOR0 Register

Bits	Name	Description	Type	Reset
31:0	CXPL_DEBUG_INFO_L	cxpl_debug_info[31:0]	RO	0x00000000

PCIE_CFG: MONITOR1 Register

Offset: 0x1a0

Description

internal signals for debug

Table 126. MONITOR1 Register

Bits	Name	Description	Type	Reset
31:0	CXPL_DEBUG_INFO_H	cxpl_debug_info[63:32]	RO	0x00000000

PCIE_CFG: MONITOR2 Register

Offset: 0x1a4

Description

internal signals for debug

Table 127. MONITOR2 Register

Bits	Name	Description	Type	Reset
31:24	Reserved.	-	-	-

Bits	Name	Description	Type	Reset
23	RADM_XFER_PENDING	radm_xfer_pending	RO	0x0
22	BRDG_DBI_XFER_PENDING	brdg_dbi_xfer_pending	RO	0x0
21	BRDG_SLV_XFER_PENDING	brdg_slv_xfer_pending	RO	0x0
20	RDLH_LINK_UP	data link layer up	RO	0x0
19	SMLH_LINK_UP	phy link up	RO	0x0
18	LINK_RESET_REQ	state of link_reset_req signal, synchronised	RO	0x0
17	PERSTN	state of pcie perstn signal, synchronised	RO	0x0
16	CORE_ALIVE	state of core_alive signal, synchronised	RO	0x0
15:0	CXPL_DEBUG_INFO_EI	cxpl_debug_info_ei[15:0]	RO	0x0000

PCIE_CFG: INTR Register

Offset: 0x1a8

Description

Raw Interrupts

Table 128. INTR Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6	RDLH_LINK_UP		WC	0x0
5	SMLH_LINK_UP		WC	0x0
4	RADM_PM_TURNOFF		WC	0x0
3	LTSSM_STATE_FIFO_NOT_EMPTY		RO	0x0
2	LINK_RESET_REQ		WC	0x0
1	PERSTN		WC	0x0
0	CORE_ALIVE		WC	0x0

PCIE_CFG: INTE Register

Offset: 0x1ac

Description

Interrupt Enable

Table 129. INTE Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6	RDLH_LINK_UP		RW	0x0
5	SMLH_LINK_UP		RW	0x0
4	RADM_PM_TURNOFF		RW	0x0
3	LTSSM_STATE_FIFO_NOT_EMPTY		RW	0x0

Bits	Name	Description	Type	Reset
2	LINK_RESET_REQ		RW	0x0
1	PERSTN		RW	0x0
0	CORE_ALIVE		RW	0x0

PCIE_CFG: INTF Register

Offset: 0x1b0

Description

Interrupt Force

Table 130. INTF Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6	RDLH_LINK_UP		RW	0x0
5	SMLH_LINK_UP		RW	0x0
4	RADM_PM_TURNOFF		RW	0x0
3	LTSSM_STATE_FIFO_NOT_EMPTY		RW	0x0
2	LINK_RESET_REQ		RW	0x0
1	PERSTN		RW	0x0
0	CORE_ALIVE		RW	0x0

PCIE_CFG: INTS Register

Offset: 0x1b4

Description

Interrupt status after masking & forcing

Table 131. INTS Register

Bits	Name	Description	Type	Reset
31:7	Reserved.	-	-	-
6	RDLH_LINK_UP		RO	0x0
5	SMLH_LINK_UP		RO	0x0
4	RADM_PM_TURNOFF		RO	0x0
3	LTSSM_STATE_FIFO_NOT_EMPTY		RO	0x0
2	LINK_RESET_REQ		RO	0x0
1	PERSTN		RO	0x0
0	CORE_ALIVE		RO	0x0

6.2. MSIx configuration registers

Interrupts transmitted across a PCIe link are serialised as Message-Signalled Interrupts. These are single write operations that encode the interrupt vector in the write data, and target a special address in the RC that translates this data into an indexed interrupt vector on the AP. MSIs are only signalled on an assertion of an interrupt, i.e. the messages follow rising-edge triggered semantics.

MSIx is an extension of the MSI standard, which allows for per-vector configuration of destination address and write data. Up to 1024 vectors are supported per function; RP1 has 61 vectors connected to interrupt sources.

Most top-level interrupts in RP1 assert a physical line continuously for as long as the logical OR of all their interrupt sources is high. This leads to a possible race condition that would lead to a 'stuck' interrupt:

- The AP receives a MSI for a peripheral
- The AP reads the active interrupt register
- It does some processing as a result of those active interrupts
- The peripheral then asserts another active interrupt for a different reason
- The interrupt-acknowledge write from the AP only clears the first set of interrupts that it read
- The interrupt wire never deasserts, so no further MSI messages are generated.

This will then cause host software to miss any further interrupt events. Polling in a loop until a read of the set of active interrupts is 0 would guarantee that future interrupt activity will generate an MSI, but is wasteful.

To avoid this race condition and the inefficient workaround, each MSIx vector has an optional acknowledge function controlled by `MSIn_CFG.IACK_EN` that masks the interrupt line when initially asserted, and unmasks the interrupt line when the respective `MSIn_CFG.IACK` register bit is written with a 1. If a peripheral interrupt is still asserted at the time the IACK register is written, a new MSIx write is generated.

NOTE

The only true edge-level interrupts in RP1 are the set of vectors assigned to `USBHOST0` and `USBHOST1`.

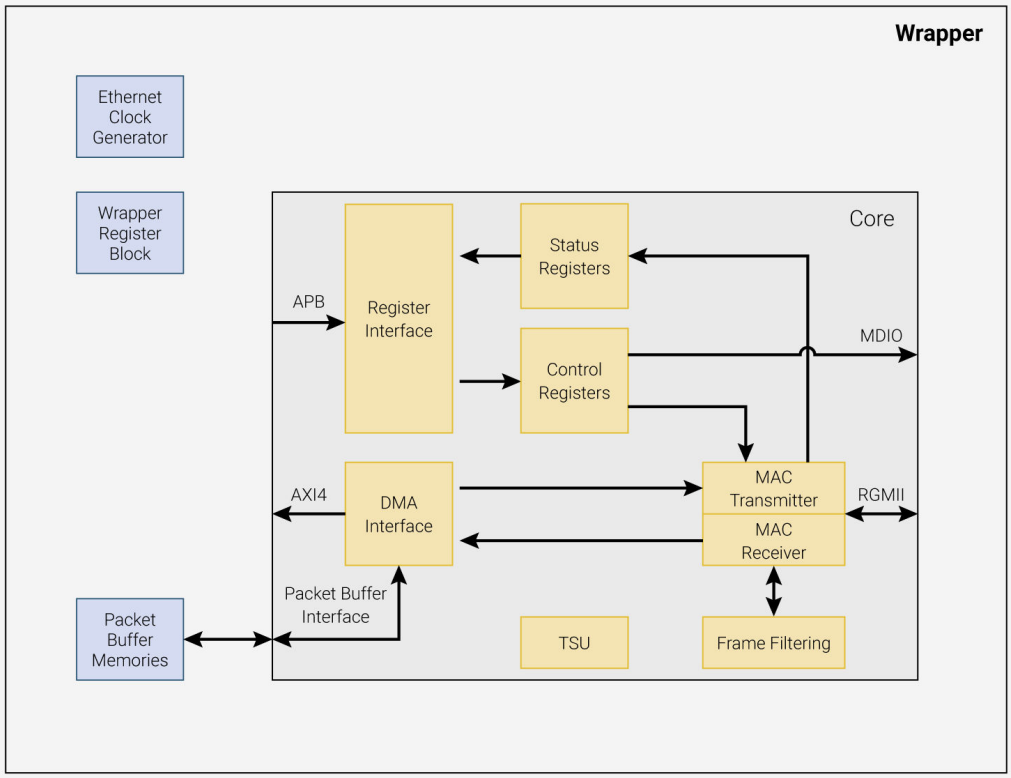
DRAFT

Chapter 7. Ethernet

The ethernet subsystem is implemented using the Cadence Gigabit Ethernet MAC (GEM_GXL 1p09) IP. The GEM_GXL configuration is:

- 10/100/1000 Mbps Ethernet MAC compatible with the IEEE 802.3 standard.
- RGMII interface.
- Half or full duplex.
- 64-bit MAC data bus width
- SRAM based packet buffer DMA 64-bit address, 128-bit data.
- AXI4 bus master interface.
- Four maskable address filters (source or destination).
- Support for jumbo frames up to 16383 bytes.
- Internal loopback.
- DMA full store and forward mode (partial store and forward not supported).
- IEEE 1588 time stamp unit.

Figure 5. Ethernet controller configuration



The core is well documented in "Cadence Gigabit Ethernet MAC with DMA, 1588, AVB and PCS (GEM_GXL) User Guide Rev 15".

7.1. Registers

The base address of the Ethernet peripheral is:

Table 132. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
eth	APB	N	0x40100000
eth_cfg	APB	Y	0x40104000

Table 133. List of ETH_CFG registers

Offset	Name	Info
0x00	CONTROL	General Ethernet control register
0x04	STATUS	General Ethernet status register
0x08	TSU_TIMER_CNT0	
0x0c	TSU_TIMER_CNT1	
0x10	TSU_TIMER_CNT2	
0x14	CLKGEN	Clock control, can be changed on-the-fly
0x18	CLK2FC	
0x1c	INTR	Raw Interrupts
0x20	INTE	Interrupt Enable
0x24	INTF	Interrupt Force
0x28	INTS	Interrupt status after masking & forcing

ETH_CFG: CONTROL Register

Offset: 0x00

Description

General Ethernet control register

Table 134. CONTROL Register

Bits	Name	Description	Type	Reset
31:5	Reserved.	-	-	-
4	MEM_PD	Memory power down	RW	0x0
3	BUSERR_EN	Enable MAC bus errors to pass through to the fabric. The MAC normally generates bus errors for any unmapped address, which causes the debugger to generate lots of bus errors. Therefore default is that MAC bus errors are off	RW	0x0
2:1	TSU_INC_CTRL	tsu increment control - directly drives gem_tsu_inc_ctrl[1:0]	RW	0x0
0	TSU_MS	tsu master/slave - directly drives gem_tsu_ms pin	RW	0x0

ETH_CFG: STATUS Register

Offset: 0x04

Description

General Ethernet status register

Table 135. STATUS Register

Bits	Name	Description	Type	Reset
31:6	Reserved.	-	-	-
5	AWLEN_ILLEGAL	Illegal AXI write address transaction - larger than 16 beats. Will need block reset, and maybe system reset	RO	0x0
4	ARLEN_ILLEGAL	Illegal AXI read address transaction - larger than 16 beats. Will need block reset, and maybe system reset	RO	0x0
3	RGMII_DUPLEX	rgmii duplex mode	RO	0x0
2:1	RGMII_SPEED	rgmii speed - 0=10Mb; 1=100Mb; 2=1Gb	RO	0x2
0	RGMII_LINK_STATUS	rgmii link status	RO	0x0

ETH_CFG: TSU_TIMER_CNT0 Register

Offset: 0x08

Table 136. TSU_TIMER_CNT0 Register

Bits	Name	Description	Type	Reset
31:0	CNT0	tsu timer count value [31:0]	RO	0x00000000

ETH_CFG: TSU_TIMER_CNT1 Register

Offset: 0x0c

Table 137. TSU_TIMER_CNT1 Register

Bits	Name	Description	Type	Reset
31:0	CNT1	tsu timer count value [63:32]	RO	0x00000000

ETH_CFG: TSU_TIMER_CNT2 Register

Offset: 0x10

Table 138. TSU_TIMER_CNT2 Register

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
29:0	CNT2	tsu timer count value [93:64]	RO	0x00000000

ETH_CFG: CLKGEN Register

Offset: 0x14

Description

Clock control, can be changed on-the-fly

Table 139. CLKGEN Register

Bits	Name	Description	Type	Reset
31:10	Reserved.	-	-	-
9	TXCLKDELEN	Adds delay to the rgmii_tx_clk	RW	0x0
8	DC50	Enables duty cycle correction for odd divisors	RW	0x0
7	ENABLE	Starts and stops the clock generator cleanly	RW	0x1
6	KILL	Asynchronously kills the clock generator	RW	0x0
5:4	SPEED_FROM_MAC		RO	0x0

Bits	Name	Description	Type	Reset
3	SPEED_OVERRIDE_EN	Use speed we specify here instead of speed from mac speed - 0=10M; 1=100M (default); 2=1000M	RW	0x0
2	Reserved.	-	-	-
1:0	SPEED_OVERRIDE		RW	0x0

ETH_CFG: CLK2FC Register

Offset: 0x18

Table 140. CLK2FC Register

Bits	Name	Description	Type	Reset
31:2	Reserved.	-	-	-
1:0	SEL	0x0 → NONE 0x1 → rgmii_tx_clk 0x2 → rgmii_rx_clk	RW	0x0

ETH_CFG: INTR Register

Offset: 0x1c

Description

Raw Interrupts

Table 141. INTR Register

Bits	Name	Description	Type	Reset
31:13	Reserved.	-	-	-
12	IEEE1588_TSU_TIMER_CMP_VAL	TSU timer comparison valid. Asserted high when upper 70 bits of TSU timer count value are equal to programmed comparison value.	WC	0x0
11	IEEE1588_SOF_RX	Asserted when the SFD is detected on a receive frame	WC	0x0
10	IEEE1588_SYNC_FRAME_RX	Asserted if PTP sync frame is detected on receive.	WC	0x0
9	IEEE1588_DELAY_REQ_RX	Asserted if PTP delay request frame is detected on receive.	WC	0x0
8	IEEE1588_PDELAY_REQ_RX	Asserted if PTP peer delay request frame is detected on receive.	WC	0x0
7	IEEE1588_PDELAY_RESP_RX	Asserted if PTP peer delay response frame is detected on receive.	WC	0x0
6	IEEE1588_SOF_TX	Asserted when the SFD is detected on a transmit frame, deasserted at end of frame	WC	0x0
5	IEEE1588_SYNC_FRAME_TX	Asserted if PTP sync frame is detected on transmit.	WC	0x0
4	IEEE1588_DELAY_REQ_TX	Asserted if PTP delay request frame is detected on transmit.	WC	0x0
3	IEEE1588_PDELAY_REQ_TX	Asserted if PTP peer delay request frame is detected on receive.	WC	0x0

Bits	Name	Description	Type	Reset
2	IEEE1588_PDELAY_RESP_TX	Asserted if PTP peer delay response frame is detected on transmit.	WC	0x0
1	WOL	Wake-on-LAN interrupt	WC	0x0
0	ETHERNET	Top-level IP interrupt	RO	0x0

ETH_CFG: INTE Register

Offset: 0x20

Description

Interrupt Enable

Table 142. INTE Register

Bits	Name	Description	Type	Reset
31:13	Reserved.	-	-	-
12	IEEE1588_TSU_TIMER_CMP_VAL	TSU timer comparison valid. Asserted high when upper 70 bits of TSU timer count value are equal to programmed comparison value.	RW	0x0
11	IEEE1588_SOF_RX	Asserted when the SFD is detected on a receive frame	RW	0x0
10	IEEE1588_SYNC_FRAME_RX	Asserted if PTP sync frame is detected on receive.	RW	0x0
9	IEEE1588_DELAY_REQ_RX	Asserted if PTP delay request frame is detected on receive.	RW	0x0
8	IEEE1588_PDELAY_REQ_RX	Asserted if PTP peer delay request frame is detected on receive.	RW	0x0
7	IEEE1588_PDELAY_RESP_RX	Asserted if PTP peer delay response frame is detected on receive.	RW	0x0
6	IEEE1588_SOF_TX	Asserted when the SFD is detected on a transmit frame, deasserted at end of frame	RW	0x0
5	IEEE1588_SYNC_FRAME_TX	Asserted if PTP sync frame is detected on transmit.	RW	0x0
4	IEEE1588_DELAY_REQ_TX	Asserted if PTP delay request frame is detected on transmit.	RW	0x0
3	IEEE1588_PDELAY_REQ_TX	Asserted if PTP peer delay request frame is detected on receive.	RW	0x0
2	IEEE1588_PDELAY_RESP_TX	Asserted if PTP peer delay response frame is detected on transmit.	RW	0x0
1	WOL	Wake-on-LAN interrupt	RW	0x0
0	ETHERNET	Top-level IP interrupt	RW	0x0

ETH_CFG: INTF Register

Offset: 0x24

Description

Interrupt Force

Table 143. INTF Register

Bits	Name	Description	Type	Reset
31:13	Reserved.	-	-	-
12	IEEE1588_TSU_TIMER_CMP_VAL	TSU timer comparison valid. Asserted high when upper 70 bits of TSU timer count value are equal to programmed comparison value.	RW	0x0
11	IEEE1588_SOF_RX	Asserted when the SFD is detected on a receive frame	RW	0x0
10	IEEE1588_SYNC_FRAME_RX	Asserted if PTP sync frame is detected on receive.	RW	0x0
9	IEEE1588_DELAY_REQ_RX	Asserted if PTP delay request frame is detected on receive.	RW	0x0
8	IEEE1588_PDELAY_REQ_RX	Asserted if PTP peer delay request frame is detected on receive.	RW	0x0
7	IEEE1588_PDELAY_RESP_RX	Asserted if PTP peer delay response frame is detected on receive.	RW	0x0
6	IEEE1588_SOF_TX	Asserted when the SFD is detected on a transmit frame, deasserted at end of frame	RW	0x0
5	IEEE1588_SYNC_FRAME_TX	Asserted if PTP sync frame is detected on transmit.	RW	0x0
4	IEEE1588_DELAY_REQ_TX	Asserted if PTP delay request frame is detected on transmit.	RW	0x0
3	IEEE1588_PDELAY_REQ_TX	Asserted if PTP peer delay request frame is detected on receive.	RW	0x0
2	IEEE1588_PDELAY_RESP_TX	Asserted if PTP peer delay response frame is detected on transmit.	RW	0x0
1	WOL	Wake-on-LAN interrupt	RW	0x0
0	ETHERNET	Top-level IP interrupt	RW	0x0

ETH_CFG: INTS Register

Offset: 0x28

Description

Interrupt status after masking & forcing

Table 144. INTS Register

Bits	Name	Description	Type	Reset
31:13	Reserved.	-	-	-
12	IEEE1588_TSU_TIMER_CMP_VAL	TSU timer comparison valid. Asserted high when upper 70 bits of TSU timer count value are equal to programmed comparison value.	RO	0x0
11	IEEE1588_SOF_RX	Asserted when the SFD is detected on a receive frame	RO	0x0
10	IEEE1588_SYNC_FRAME_RX	Asserted if PTP sync frame is detected on receive.	RO	0x0
9	IEEE1588_DELAY_REQ_RX	Asserted if PTP delay request frame is detected on receive.	RO	0x0
8	IEEE1588_PDELAY_REQ_RX	Asserted if PTP peer delay request frame is detected on receive.	RO	0x0

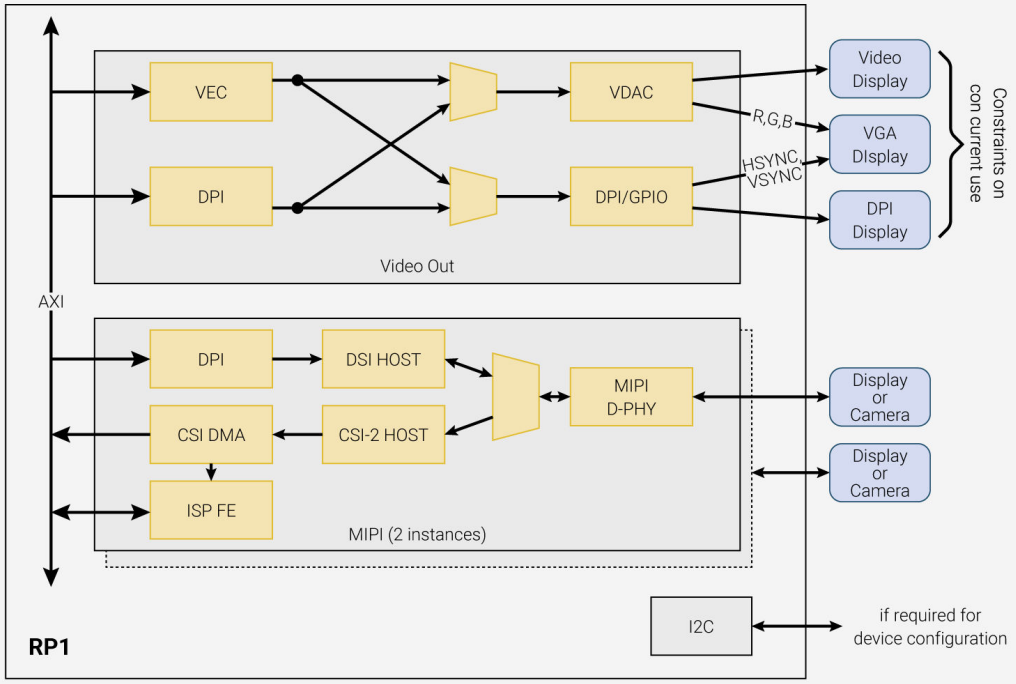
Bits	Name	Description	Type	Reset
7	IEEE1588_PDELAY_RESP_RX	Asserted if PTP peer delay response frame is detected on receive.	RO	0x0
6	IEEE1588_SOF_TX	Asserted when the SFD is detected on a transmit frame, deasserted at end of frame	RO	0x0
5	IEEE1588_SYNC_FRAME_TX	Asserted if PTP sync frame is detected on transmit.	RO	0x0
4	IEEE1588_DELAY_REQ_TX	Asserted if PTP delay request frame is detected on transmit.	RO	0x0
3	IEEE1588_PDELAY_REQ_TX	Asserted if PTP peer delay request frame is detected on receive.	RO	0x0
2	IEEE1588_PDELAY_RESP_TX	Asserted if PTP peer delay response frame is detected on transmit.	RO	0x0
1	WOL	Wake-on-LAN interrupt	RO	0x0
0	ETHERNET	Top-level IP interrupt	RO	0x0

DRAFT

Chapter 8. Displays and cameras

Figure 6 presents an overview of display and camera support in RP1.

Figure 6. Video Out and MIPI interfaces in RP1



The Video Encoder (VEC), together with the built-in Video DAC, can generate composite or Y/C output for a variety of TV standards (interlaced or progressive).

A DPI block can output (progressive, separated sync) video in parallel through GPIO pins, with programmable format and timings. By routing RGB through the Triple Video DAC, it is possible to support VGA output. Contention for shared resources may restrict the concurrent use of VEC and DPI.

RP1 also has two independent four-lane MIPI D-PHY interfaces, each of which can support either a DSI display or a CSI-2 camera. It includes facilities to stream RGB video out, handle incoming CSI-2 imagery and metadata, and perform some basic image processing and statistics.

8.1. DSI host

The DSI host interface supports a pixel interface (eDPI) which allows pixel data to be sent for command mode or video mode displays.

8.2. CSI-2 host

The CSI-2 host interface has a data only interface (IDI). CSI controller outputs a clock at link byte clock (line rate/8) called `clk_csi2_byte`. Maximum frequency will be 1500MHz/8 = 187.5MHz.

8.3. Registers

The base address of the MIPI peripherals are:

Table 145. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
mipi0_csidma	APB	N	0x40110000
mipi0_csihost	APB	N	0x40114000
mipi0_dsidma	APB	N	0x40118000
mipi0_dsihost	APB	N	0x4011c000
mipi0_cfg	APB	Y	0x40120000
mipi0_isp	APB	N	0x40124000
mipi1_csidma	APB	N	0x40128000
mipi1_csihost	APB	N	0x4012c000
mipi1_dsidma	APB	N	0x40130000
mipi1_dsihost	APB	N	0x40134000
mipi1_cfg	APB	Y	0x40138000
mipi1_isp	APB	N	0x4013c000

DRAFT

Chapter 9. DMA

An eight-channel Synopsys AXI DMAC based on version 1.02a-lp02 is instantiated. The purpose of this DMA controller is to service low-bandwidth APB peripherals and interface with their flow control handshakes to pace DMA transfers across PCIe. A secondary intent is that DMA traffic can be more efficiently bursted across the PCIe using this controller, as each TLP carries a fixed overhead that becomes large when small (e.g. single 32-bit) transfers are used.

The AXI DMAC has eight almost-identical channels. Channels 1 and 2 have double-sized internal FIFOs with 8-beat storage, channels 3-8 have 4-beat storage. The DMA channels all arbitrate internally with configurable priority, and conduct reads and writes over a single 128-bit AXI master port. An AXI slave port is used for access to control and status registers. The DMAC is configured to allow Channel Aborts, does not permit Channel Locking, and has a maximum transfer block size of `2^18 - 1` elements.

The AXI DMAC will only issue as many outstanding AXI transfers as there is space in the channel's internal FIFO, so read bandwidth is heavily dependent on PCIe link round-trip time. Typical per-channel read bandwidth is expected to be 500-600Mbs, and write bandwidth 2Gbps. The vast majority of the target peripherals operate with single-digit megabits per second of throughput.

The DMAC is configured with a separate core clock, running at 100MHz. Clock-crossing between the clk_sys domain and clk_dma domain is done internally using SNPS synchroniser modules. The DMAC is configured with CSLP (top-level automatic clock gating) to gate off large sections of the core clock when idle and enabled.

Peripherals that implement a DMA flow control handshake are connected to the DMAC according to the handshake table below. Almost all peripherals are connected, with the exception of I2C6. There are also two "tick" peripherals for periodic triggering of a handshake input, to allow for time-based pacing of arbitrary transfers.

9.1. Register base addresses

There is one instance of the DMA peripheral.

Table 146. Peripheral Address Map

Block	Bus Type	Atomic Access	Address
dma	AHB	N	`0x40188000`

For a reference implementation see the Synopsys DesignWare AXI DMA Linux kernel driver.

DRAFT

Appendix A: Documentation release history

Table 147. Documentation release history

Release	Date	Description
1.0	06 Oct 2023	First draft release
1.1	07 Nov 2023	Final draft

DRAFT

Raspberry Pi is a trademark of Raspberry Pi Ltd