Rockchip RK3066 and RK3188 clock controller and PMU

Created from the constants, clock definitions and PMU accesses found in the kernel sources released to the public under the GPL. It therefore might contain errors or unknown values.

Clock controller

External clocks that have to/can be supplied are

CLOCK	Description
XIN24M	24MHz XTAL
SUSPEND	32kHz suspend clock
EXT_HSADC	
RMII_CLKIN	Only referenced as FIXME in clock_data.c

Register-Map

Register	Address	Description
PLL_CON_0	0x20000000	APLL_CON_0 PLL_NR and PLL_NO values
PLL_CON_1	0x20000004	APLL_CON_1 PLL_NF value
PLL_CON_2	0x20000008	APLL_CON_2 PLL_BWADJ value
PLL_CON_3	0x2000000C	APLL_CON_3 PLL settings
PLL_CON_4	0x20000010	DPLL_CON_0 PLL_NR and PLL_NO values
PLL_CON_5	0x20000014	DPLL_CON_1 PLL_NF value
PLL_CON_6	0x20000018	DPLL_CON_2 PLL_BWADJ value
PLL_CON_7	0x2000001C	DPLL_CON_3 PLL settings
PLL_CON_8	0x20000020	CPLL_CON_0 PLL_NR and PLL_NO values
PLL_CON_9	0x20000024	CPLL_CON_1 PLL_NF value
PLL_CON_10	0x20000028	CPLL_CON_2 PLL_BWADJ value
PLL_CON_11	0x2000002C	CPLL_CON_3 PLL settings

PLL_CON_12	0x20000030	GPLL_CON_0 PLL_NR and PLL_NO values
PLL_CON_13	0x20000034	GPLL_CON_1 PLL_NF value
PLL_CON_14	0x20000038	GPLL_CON_2 PLL_BWADJ value
PLL_CON_15	0x2000003C	GPLL_CON_3 PLL settings
MODE_CON	0x20000040	PLL operating modes (normal, slow)
CLKSEL_CON_0	0x20000044	Clock multiplexer and divider settings
…	…	…
CLKSEL_CON_34	0x200000CC	Clock multiplexer and divider settings
CLKGATE_CON_0	0x200000D0	Clock gate settings
…	…	…
CLKGATE_CON_9	0x200000F4	Clock gate settings
GLB_SRST_FST	0x20000100	
GLB_SRST_SND	0x20000104	
SOFTRST_CON_0	0x20000110	
…	…	
SOFTRST_CON_8	0x20000130	
MISC_CON	0x20000134	
GLB_CNT_TH	0x20000140	

PLL configuration

Rockchip code indicates that before the PLL values can be safely changed, it has to be put into slow mode, to be resumed into the normal mode after the settle delay has elapsed.

The PLL rate is calculated as

f_out = (f_in * nf) / (nr * no)

Additional constraints found in the code are: [nb=bwadj+1;0:11;nb=nf/2, FIXME: correct?] and bwadj is always set as (nf >> 1)

The reset-delay (rst_dly) in µs of the PLL is calulated as

delay = ((nr * 500) / 24 + 1).

The lock status of the PLLs can be read from GRF_SOC_STATUS0[8:5] for

[GPLL, CPLL, APLL, DPLL].

PLL operation modes

According to the RK28xx manual (ARM9), which has a smilar clock controller the operating modes set in MODE_CON mean the following:

- Normal mode: Normal operating mode
- Slow mode: Low frequency clock (24MHz) without PLL
- Deep Slow mode : More Low frequency clock (32.768KHz) without PLL

  It's currently unknown if the RK3xxx PLL-modes mean the same.

Changing PLL-rates

To change the rate of a PLL the Rockchip code does put it into SLOW_MODE (in CRU_MODE_CON), then powers it down, via xPLL_CON_3. After are 6 dsb() calls after which the new xPLL_CON_0 and xPLL_CON_1 values are written followed by a rk30_clock_udelay(1) after which the powerdown gets lifted (via xPLL_CON_3).

Then it is necessary to wait for the pll to lock, before leaving the SLOW_MODE again.

In contrast on the rk3188+ and all rk30xx the change is done slightly different. The pll is put in SLOW-mode again but after this it is put into its RESET state. Then the pllcon0,1,2 changes are written to the registers, followed by an udelay(5). After the pll is moved out of its reset state followed by dynamic reset_delay calculated from the new pll values. After this follows the already known waiting for the pll to lock and its move out of the slow mode.

The difference is probably the time spent in delay functions. While the commonly used function has delays of 5+varying number of usec the non-plus-rk3188 variant can do the same in 1usec + a little delay created by 6 dsb() calls, making it slightly faster.

According to the older RK28xx manual, to change a PLL rate, it can be either put into BYPASS (xPLL_CON_3) or SLOW (MODE_CON) mode, where SLOW mode is preferred.

Changing APLL-rates

The APLL supplies among others the CPU, so more care is necessary when changing its rate. MUX_CORE should be reparented to the GPLL to keep supplying the ARMCLK, after which the APLL rate can be changed. After the APLL locks, the MUX_CORE can again be reparented to the APLL.

xPLL_CON_0 configures the nr and no values

xPLL_CON_0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:14]	-
PLL_NR	[13:8]	FIXME: one of the PLL values
UNKNOWN	[7:5]	-
PLL_NO	[5:0]	FIXME: one of the PLL values ([3:0] on the rk3188+)

xPLL_CON_1 configures the nf value

xPLL_CON_1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
PLL_NF	[15:0]	FIXME: one of the PLL values ([13:0] on the so called rk3188plus)

xPLL_CON_2 configures the BWADJ value

xPLL_CON_2	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
PLL_BWADJ	[12:0]	FIXME: one of the PLL values, set in the code as (nf >> 1)

xPLL_CON_3 configures reset and power settings

xPLL_CON_3	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:6]	-
RESET	[5]	Put pll in reset state, set to 0 to resume
UNKNOWN	[4:3]	-
STANDBY	[2]	Put pll in standby mode
POWER_DOWN	[1]	Power down the PLL (also known as PWR_DN)
BYPASS	[0]	Bypass the PLL, FIXME: does this use XIN24M directly then?

MODE_CON configures basic PLL operating modes

In the upstream kernel code, slow-mode combined with a powerdown of the pll is used when the target rate is 24MHz (the rate of XIN24M). This suggests that slow-mode bypasses the pll completely. What is the difference to the BYPASS-bit in the PLL_CON_3 ?

MODE_CON	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:14]	-
GPLL_MODE	[13:12]	Operating mode of GPLL 00 = slow, 01 = normal, 10 = deep slow, 11 = reserved
UNKNOWN	[11:10]	-
CPLL_MODE	[9:8]	Operating mode of CPLL 00 = slow, 01 = normal, 10 = deep slow, 11 = reserved
UNKNOWN	[7:6]	-
DPLL_MODE	[5:4]	Operating mode of DPLL 00 = slow, 01 = normal, 10 = deep slow, 11 = reserved
UNKNOWN	[3:2]	-
APLL_MODE	[1:0]	Operating mode of APLL 00 = slow, 01 = normal, 10 = deep slow, 11 = reserved

RK2928 Clock divider and multiplexer configuration

Warning: RK2928 uses a slightly different type of PLL, so the PLL description above does not apply.

CLKSEL_CON_0 and 1 configure the dividers and muxes of core system clocks

Some of these settings are tightly bound to the APLL and/or GPLL rates and should be readjusted when the PLL rates are changed.

CLKSEL_CON_0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:14]	-
MUX_ACLK_CPU	[13]	Select source for core clock 0 = APLL, 1 = GPLL
DIV_ACLK_CPU	[12:8]	Divide MUX_ACLK_CPU by (DIV_ACLK_CPU + 1)
MUX_CORE	[7]	Select source for core clock 0 = APLL, 1 = GATE_GPLL_CPU (also known as cpu_gpll_path)
UNKNOWN	[6:5]	-
DIV_CORE	[4:0]	Divide MUX_CORE by (DIV_CORE + 1). This is the CPU clock.

CLKSEL_CON_1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:14]	
DIV_PCLK_CPU	[13:12]	Divide GATE_ACLK_CPU by (2 ^ DIV_PCLK_CPU)
UNKNOWN	[11:10]	-
DIV_HCLK_CPU	[9:8]	Divide GATE_ACLK_CPU by (2 ^ DIV_HCLK_CPU)
DIV_ACLK_CORE	[4]	Divide DIV_CORE by (DIV_ACLK_CORE + 1)
DIV_CORE_PERI	[3:0]	Divide DIV_CORE by (DIV_CORE_PERI + 1)

CLKSEL_CON_10 configures peripheral ACLK, PCLK, HCLK core rates

CLKSEL_CON_10	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_ACLK_PERI	[15]	Select PLL source for ACLK_PERI 0 = GPLL, 1 = CPLL
UNKNOWN	[14]	-
DIV_PCLK_PERI	[13:12]	Divide GATE_ACLK_PERI by (2 ^ DIV_PCLK_PERI)
UNKNOWN	[11:10]	-
DIV_HCLK_PERI	[9:8]	Divide GATE_ACLK_PERI by (2 ^ DIV_HCLK_PERI)
UNKNOWN	[7:5]	-
DIV_ACLK_PERI	[4:0]	Divide MUX_ACLK_PERI by (DIV_ACLK_PERI + 1)

RK3066 Clock divider and multiplexer configuration

CLKSEL_CON_0 and 1 configure the dividers and muxes of core system clocks

Some of these settings are tightly bound to the APLL and/or GPLL rates and should be readjusted when the PLL rates are changed.

CLKSEL_CON_0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:9]	-
MUX_CORE	[8]	Select source for core clock 0 = APLL, 1 = GATE_GPLL_CPU (also known as cpu_gpll_path)
DIV_CORE_PERI	[7:6]	Divide DIV_CORE by (2 ^ (DIV_CORE_PERI + 1))
UNKNOWN	[5]	-
DIV_CORE	[4:0]	Divide MUX_CORE by (DIV_CORE + 1). This is the CPU clock.

CLKSEL_CON_1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DIV_HCLK_AHB2APB	[15:14]	Divide GATE_HCLK_AHB2APB by (2 ^ DIV_HCLK_AHB2APB)
DIV_PCLK_CPU	[13:12]	Divide GATE_ACLK_CPU by (2 ^ DIV_PCLK_CPU)
UNKNOWN	[11:10]	-
DIV_HCLK_CPU	[9:8]	Divide GATE_ACLK_CPU by (2 ^ DIV_HCLK_CPU)
UNKNOWN	[7:6]	-
DIV_ACLK_CPU	[3:0]	Divide DIV_CORE by (DIV_ACLK_CORE + 1) for DIV_ACLK_CORE values < 4, for everything greater divide DIV_CORE by 8 (write 4 into DIV_ACLK_CORE to set)

CLKSEL_CON_10 configures peripheral ACLK, PCLK, HCLK core rates

CLKSEL_CON_10	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_ACLK_PERI	[15]	Select PLL source for ACLK_PERI 0 = GPLL, 1 = CPLL
UNKNOWN	[14]	-
DIV_PCLK_PERI	[13:12]	Divide GATE_ACLK_PERI by (2 ^ DIV_PCLK_PERI)
UNKNOWN	[11:10]	-
DIV_HCLK_PERI	[9:8]	Divide GATE_ACLK_PERI by (2 ^ DIV_HCLK_PERI)
UNKNOWN	[7:5]	-
DIV_ACLK_PERI	[4:0]	Divide MUX_ACLK_PERI by (DIV_ACLK_PERI + 1)

[FIXME]

RK3188 Clock divider and multiplexer configuration

CLKSEL_CON_0 and 1 configure the dividers and muxes of core system clocks

Some of these settings are tightly bound to the APLL and/or GPLL rates and should be readjusted when the PLL rates are changed.

CLKSEL_CON_0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:14]	-
DIV_CORE	[13:9]	Divide MUX_CORE by (DIV_CORE + 1). This is the CPU clock.
MUX_CORE	[8]	Select source for core clock 0 = APLL, 1 = GATE_GPLL_CPU (also known as cpu_gpll_path)
DIV_CORE_PERI	[7:6]	Divide DIV_CORE by (2 ^ (DIV_CORE_PERIPH + 1))
MUX_ACLK_CPU	[5]	Select source for ACLK_CPU clock 0 = APLL, 1 = GPLL (also known as DIV_CPU)
DIV_ACLK_CPU	[4:0]	Divide MUX_ACLK_CPU by (DIV_CPU + 1) (also known as DIV_CPU)

CLKSEL_CON_1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DIV_HCLK_AHB2APB	[15:14]	Divide GATE_HCLK_AHB2APB by (2 ^ DIV_AHB2APB)
DIV_PCLK_CPU	[13:12]	Divide GATE_ACLK_CPU by (2 ^ DIV_PCLK_CPU)
UNKNOWN	[11:10]	-
DIV_HCLK_CPU	[9:8]	Divide GATE_ACLK_CPU by (2 ^ DIV_HCLK_CPU)
UNKNOWN	[7:6]	-
DIV_ACLK_CORE	[5:3]	Divide DIV_CORE by (DIV_ACLK_CORE + 1) for DIV_ACLK_CORE values < 4, for everything greater divide DIV_CORE by 8 (write 4 into DIV_ACLK_CORE to set)
UNKNOWN	[2:0]	-

CLKSEL_CON_2 configures the I2S clock pll-source

CLKSEL_CON_2	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_I2S_PLL	[15]	Select source for DIV_I2S and DIV_SPDIF 0 = GPLL, 1 = CPLL
UNKNOWN	[14:0]	-

CLKSEL_CON_3 configures I2S clocks

CLKSEL_CON_3	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:10]	-
MUX_I2S	[9:8]	Select source for I2S 00 = GATE_DIV_I2S, 01 = GATE_FRAC_I2S, 10 = CLK12M, 11

		= reserved
UNKNOWN	[7:5]	-
DIV_I2S	[4:0]	Divide MUX_I2S_PLL by (DIV_I2S + 1)

CLKSEL_CON_4 configures something

CLKSEL_CON_4	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:0]	-

CLKSEL_CON_5 configures SPDIF clocks

CLKSEL_CON_5	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:10]	-
MUX_SPDIF	[9:8]	Select source for SPDIF 00 = GATE_DIV_SPDIF, 01 = GATE_FRAC_SPDIF, 10 = CLK12M, 11 = reserved
UNKNOWN	[7:5]	-
DIV_SPDIF	[4:0]	Divide MUX_I2S_PLL by (DIV_SPDIF + 1)

CLKSEL_CON_6 configures something

CLKSEL_CON_6	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:0]	-

CLKSEL_I2S_FRAC (CLKSEL_CON_7) configures the fraction-divider of i2s0

I2S_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_I2S0_SRC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.
DENOMINATOR	[15:0]	

CLKSEL_CON_8 configures something

CLKSEL_CON_8	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:0]	-

CLKSEL_SPDIF_FRAC (CLKSEL_CON_9) configures the fraction-divider of spdif

SPDIF_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_SPDIF_SRC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.

DENOMINATOR	[15:0]	

CLKSEL_CON_10 configures peripheral ACLK, PCLK, HCLK core rates

CLKSEL_CON_10	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_ACLK_PERI	[15]	Select PLL source for ACLK_PERI 0 = CPLL, 1 = GPLL
UNKNOWN	[14]	-
DIV_PCLK_PERI	[13:12]	Divide GATE_ACLK_PERI by (2 ^ DIV_PCLK_PERI)
UNKNOWN	[11:10]	-
DIV_HCLK_PERI	[9:8]	Divide GATE_ACLK_PERI by (2 ^ DIV_HCLK_PERI)
UNKNOWN	[7:5]	-
DIV_ACLK_PERI	[4:0]	Divide MUX_ACLK_PERI by (DIV_ACLK_PERI + 1)

CLKSEL_CON_11 configures MMC0 and HSICPHY dividers

CLKSEL_CON_11	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:14]	-
DIV_HSICPHY	[13:8]	Divide MUX_HSICPHY by FIXME
UNKNOWN	[7:6]	-
DIV_MMC0	[5:0]	Divide HCLK_PERI by (DIV_MMC0 + 1) [with DIV_MMC0 + 1 only 1, 2, 4, …, 2*n] (also known as DIV_SDMMC)

CLKSEL_CON_12 configures MMC1, eMMC dividers and the UART pll-source

CLKSEL_CON_12	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_UART_PLL	[15]	Select source for DIV_UARTx 0 = GPLL, 1 = CPLL
UNKNOWN	[14]	-
DIV_MMC2	[13:8]	Divide HCLK_PERI by (DIV_MMC2 + 1) [with DIV_MMC2 + 1 only 1, 2, 4, …, 2*n] (also known as DIV_EMMC)
UNKNOWN	[7:6]	-
DIV_MMC1	[5:0]	Divide HCLK_PERI by (DIV_MMC1 + 1) [with DIV_MMC1 + 1 only 1, 2, 4, …, 2*n] (also known as DIV_SDIO)

UART0 (CLKSEL_CON_13) configures the mux and divider of uart0

UART0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing

UNKNOWN	[15:10]	-
MUX_UART0	[9:8]	Select source for UART0 0 = GATE_UART0, 1 = GATE_UART0_FRAC, 10 = XIN24M, 11 = reserved
UNKNOWN	[7]	-
DIV_UART0	[6:0]	Divide MUX_UART_PLL by (DIV_UART0 + 1)

UART1 (CLKSEL_CON_14) configures the mux and divider of uart1

UART1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:10]	-
MUX_UART1	[9:8]	Select source for UART1 0 = GATE_UART1, 1 = GATE_UART1_FRAC, 10 = XIN24M, 11 = reserved
UNKNOWN	[7]	-
DIV_UART1	[6:0]	Divide MUX_UART_PLL by (DIV_UART1 + 1)

UART2 (CLKSEL_CON_15) configures the mux and divider of uart2

UART2	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:10]	-
MUX_UART2	[9:8]	Select source for UART2 0 = GATE_UART2, 1 = GATE_UART2_FRAC, 10 = XIN24M, 11 = reserved
UNKNOWN	[7]	-
DIV_UART2	[6:0]	Divide MUX_UART_PLL by (DIV_UART2 + 1)

UART3 (CLKSEL_CON_16) configures the mux and divider of uart3

UART3	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:10]	-
MUX_UART3	[9:8]	Select source for UART3 0 = GATE_UART3, 1 = GATE_UART3_FRAC, 10 = XIN24M, 11 = reserved
UNKNOWN	[7]	-
DIV_UART3	[6:0]	Divide MUX_UART_PLL by (DIV_UART3 + 1)

UART0_FRAC (CLKSEL_CON_17) configures the fraction-divider of uart0

UART0_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_UART0_SRC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.

DENOMINATOR	[15:0]	

UART1_FRAC (CLKSEL_CON_18) configures the fraction-divider of uart1

UART1_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_UART1_SRC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.
DENOMINATOR	[15:0]	

UART2_FRAC (CLKSEL_CON_19) configures the fraction-divider of uart2

UART2_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_UART2_SRC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.
DENOMINATOR	[15:0]	

UART3_FRAC (CLKSEL_CON_20) configures the fraction-divider of uart3

UART3_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_UART3_SRC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.
DENOMINATOR	[15:0]	

MAC (CLKSEL_CON_21) configures the MAC clock

MAC	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:13]	-
DIV_MAC	[12:8]	Divide MUX_MAC by (DIV_MAC_PLL + 1)
UNKNOWN	[7:5]	-
MUX_SCLK_MAC	[4]	0 = GATE_DIV_MAC, 1 = FIXME: rmii_clkin
UNKNOWN	[3:1]	-
MUX_MAC	[0]	Select source for DIV_MAC 0 = GPLL, 1 = DPLL

HSADC (CLKSEL_CON_22) configures HSADC sources and dividers

HSADC	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DIV_HSADC	[15:8]	Divide MUX_HSADC_PLL by (DIV_HSADC + 1)

INV_HSADC	[7]	„Invert" MUX_HSADC (currently implemented as MUX)
UNKNOWN	[6]	-
MUX_HSADC	[5]	Select source for hsadc clock 0 = GATE_HSADC, 1 = GATE_HSADC_FRAC, 2 = EXT_HSADC
UNKNOWN	[4:1]	
MUX_HSADC_PLL	[0]	Select source-pll for hsadc clock 0 = GPLL, 1 = CPLL

HSADC_FRAC (CLKSEL_CON_23) configures the hsadc-fraction-divider

HSADC_FRAC	Bit	Description
NUMERATOR	[31:16]	Multiply GATE_HSADC with NUMERATOR and divide through DENOMINATOR. DENOMINATOR/NUMERATOR must be < 20.
DENOMINATOR	[15:0]	

SARADC_DIV (CLKSEL_CON_24) configures SARADC divider

SARADC_DIV	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DIV_SARADC	[15:8]	Divide XIN24M by (DIV_SARADC + 1)
UNKNOWN	[14:0]	-

SPI_SRC_DIV (CLKSEL_CON_25) configures the SPI source dividers

SPI_SRC_DIV	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15]	-
DIV_SPI1	[14:8]	Divide PCLK_PERI by (DIV_SPI1 + 1)
UNKNOWN	[7]	-
DIV_SPI0	[6:0]	Divide PCLK_PERI by (DIV_SPI0 + 1)

CLKSEL_CON_26 configures the DDR clock source and divider

DDR_CLK	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:9]	-
MUX_DDR	[8]	Select source for ddr clk 0 = DPLL, 1 = GATE_GPLL_DDR (or ddr_gpll_path)
UNKNOWN	[7:2]	-
DIV_DDR	[1:0]	Divide MUX_DDR by (2^DIV_DDR)

DCLK_LCDC0 (CLKSEL_CON_27) configures LCDC0 source and divider

DCLK_LCDC0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DIV_DCLK_LCDC0	[15:8]	Divide MUX_DCLK_LCDC0 by (DIV_DCLK_LCDC0 + 1)
UNKNOWN	[7:1]	-

MUX_DCLK_LCDC0	[0]	Select source for dclk_lcdc0 0 = CPLL, 1 = GPLL

DCLK_LCDC1 (CLKSEL_CON_28) configures LCDC1 source and divider

DCLK_LCDC1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DIV_DCLK_LCDC1	[15:8]	Divide MUX_DCLK_LCDC1 by (DIV_DCLK_LCDC1 + 1)
UNKNOWN	[7:1]	-
MUX_DCLK_LCDC1	[0]	Select source for dclk_lcdc1 0 = CPLL, 1 = GPLL

CIF_OUT (CLKSEL_CON_29) configures the Camera Interface clock output

CIF_OUT	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:8]	-
MUX_CIF_OUT	[7]	Select source for cif_out 0 = GATE_DIV_CIF_OUT, 1 = XIN24M
UNKNOWN	[6]	-
DIV_CIF_OUT	[5:1]	Divide MUX_CIF_OUT_PLL by (DIV_CIF_OUT + 1)
MUX_CIF_OUT_PLL	[0]	Select source for cif_out_pll 0 = CPLL, 1 = GPLL

UNKNOWN (CLKSEL_CON_30) configures the CIF clock output and HSICPHY

UNKNOWN	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:9]	-
MUX_CIF_IN	[8]	Select source for cif_in 0 = FIXME: pclkin_cif, 1 = FIXME: inv_cif
UNKNOWN	[7:2]	-
MUX_HSICPHY	[1:0]	Select source for hsicphy 0 = GATE_OTGPHY0, 1 = GATE_OTGPHY1, 2 = GPLL, 3 = CPLL

ACLK_LCDCx_PRE (CLKSEL_CON_31) configures „something"

ACLK_LCDCx	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_ACLK_LCDC1_PRE	[15]	Select source for ACLK_LCDC1_PRE 0 = CPLL, 1 = GPLL
UNKNOWN	[14:13]	-
DIV_ACLK_LCDC1_PRE	[12:8]	Divide MUX_ACLK_LCDC0_PRE by (DIV_ACLK_LCDC0_PRE + 1)
MUX_ACLK_LCDC0_P	[7]	Select source for ACLK_LCDC0_PRE

RE		0 = CPLL, 1 = GPLL
UNKNOWN	[6:5]	-
DIV_ACLK_LCDC0_PRE	[4:0]	Divide MUX_ACLK_LCDC0_PRE by (DIV_ACLK_LCDC0_PRE + 1)

ACLK_VxPU (CLKSEL_CON_32) configures „something"

ACLK_VxPU	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
MUX_ACLK_VDPU	[15]	Select source for aclk_vdpu 0 = CPLL, 1 = GPLL
UNKNOWN	[14:13]	-
DIV_ACLK_VDPU	[12:8]	Divide MUX_ACLK_VDPU by (DIV_ACLK_VDPU + 1)
MUX_ACLK_VEPU	[7]	Select source for aclk_vepu 0 = CPLL, 1 = GPLL
UNKNOWN	[6:5]	-
DIV_ACLK_VEPU	[4:0]	Divide MUX_ACLK_VEPU by (DIV_ACLK_VEPU + 1)

UNKNOWN (CLKSEL_CON_33) configures „something"

UNKNOWN	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:0]	-

ACLK_GPU (CLKSEL_CON_34) configures „something"

ACLK_GPU	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
UNKNOWN	[15:8]	-
MUX_ACLK_GPU	[7]	Select source for aclk_gpu 0 = CPLL, 1 = GPLL
UNKNOWN	[6:5]	-
DIV_ACLK_GPU	[4:0]	Divide MUX_ACLK_GPU by (DIV_ACLK_GPU + 1)

Clock gate configuration

CLKGATE_CON_0 configures „something"

CLKGATE_CON_0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
TESTCLK	[15]	unknown
FRAC_SPDIF	[14]	Enable DIV_SPDIF_FRAC
DIV_SPDIF	[13]	Enable DIV_SPDIF
RESERVED	[12]	-
RESERVED	[11]	-

FRAC_I2S	[10]	Enable DIV_I2S0_FRAC
DIV_I2S	[9]	Enable DIV_I2S0
RESERVED	[8]	-
ACLK_CORE	[7]	Enable DIV_ACLK_CORE
ATCLK_CPU	[6]	Enable PCLK_CPU into FIXME
PCLK_CPU	[5]	Enable DIV_PCLK_CPU
HCLK_CPU	[4]	Enable DIV_HCLK_CPU
ACLK_CPU	[3]	Enable DIV_ACLK_CPU
DDR	[2]	Enable DIV_DDR
CPU_GPLL	[1]	Enable GPLL into MUX_CORE
CORE_PERI	[0]	Enable DIV_CORE_PERIPH

CLKGATE_CON_1 configures „something"

CLKGATE_CON_1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
FRAC_UART3	[15]	Enable FRAC_UART3
DIV_UART3	[14]	Enable DIV_UART3
FRAC_UART2	[13]	Enable FRAC_UART2
DIV_UART2	[12]	Enable DIV_UART2
FRAC_UART1	[11]	Enable FRAC_UART1
DIV_UART1	[10]	Enable DIV_UART1
FRAC_UART0	[9]	Enable FRAC_UART0
DIV_UART0	[8]	Enable DIV_UART0
GPLL_DDR	[7]	Enable GPLL into MUX_DDR
OTGPHY1	[6]	Enable XIN24M into the 2nd OTG controller (FIXME: a sub-clock is called xxx_480m, what is happening there? Same for OTGPHY0)
OTGPHY0	[5]	Enable XIN24M into the 1st OTG controller
ACLK_LCDC1_SRC	[4]	Enable DIV_ACLK_LCDC1_PRE
JTAG	[3]	Unknown
TIMER3	[2]	Enable XIN24M into the Timer-3 block
TIMER1	[1]	Enable XIN24M into the Timer-1 block
TIMER0	[0]	Enable XIN24M into the Timer-0 block

CLKGATE_CON_2 configures „something"

CLKGATE_CON_2	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
RESERVED	[15]	-
DIV_MMC2	[14]	Enable DIV_MMC2 into the 3rd MMC controller (eMMC)

DIV_MMC1	[13]	Enable DIV_MMC1 into the 2nd MMC controller (also called SDIO_SRC) (CIU clock)
MAC_LBTEST	[12]	Enable MAC_REF as „mii_tx“
DIV_MMC0	[11]	Enable DIV_MMC0 into the 1st MMC controller (also called MMC0_SRC) (CIU clock)
DIV_SPI1	[10]	Enable DIV_SPI1 into the 2nd SPI controller
DIV_SPI0	[9]	Enable DIV_SPI0 into the 1st SPI controller
DIV_SARADC	[8]	Enable DIV_SARADC into the SARADC controller
FRAC_HSADC	[7]	Enable FRAC_HSADC
DIV_HSADC	[6]	Enable DIV_HSADC
DIV_MAC	[5]	Enable DIV_MAC
SMC	[4]	Enable HCLK_PERI into the Static memory controller
PCLK_PERI	[3]	Enable DIV_PCLK_PERI
HCLK_PERI	[2]	Enable DIV_HCLK_PERI
ACLK_PERI	[1]	Enable DIV_ACLK_PERI
PERI_SRC	[0]	Enable ACLK_PERI into FIXME

CLKGATE_CON_3 configures „something“

CLKGATE_CON_3	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
ACLK_GPU_SRC	[15]	FIXME: unknown
TIMER6	[14]	Enable XIN24M into the Timer-6 block
RESERVED	[13]	-
HCLK_VDPU	[12]	Enable ACLK_VDPU into the Video Decoder (FIXME: correct?, special recalc function doing parent_rate / 4)
ACLK_VDPU	[11]	Enable DIV_ACLK_VDPU into the Video Decoder
HCLK_VEPU	[10]	Enable ACLK_VEPU into the Video Encoder (FIXME: correct?, special recalc function doing parent_rate / 4)
ACLK_VEPU	[9]	Enable DIV_ACLK_VEPU into the Video Encoder
TIMER5	[8]	Enable XIN24M into the Timer-5 block
DIV_CIF_OUT	[7]	Enable DIV_CIF_OUT
HSICPHY	[6]	FIXME: unknown
TIMER4	[5]	Enable XIN24M into the Timer-4 block
TIMER2	[4]	Enable XIN24M into the Timer-2 block
PCLKIN_CIF	[3]	FIXME: seems to be an external clock that gets fed to the SoC
DCLK_LCDC1_SRC	[2]	Enable DIV_DCLK_LCDC1
DCLK_LCDC0_SRC	[1]	Enable DIV_DCLK_LCDC0
ACLK_LCDC0_SRC	[0]	Enable DIV_ACLK_LCDC0_PRE

CLKGATE_CON_4 configures „something“

CLKGATE_CON_4	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
HCLK_IMEM0	[15]	Enable HCLK_CPU into FIXME
HCLK_IMEM1	[14]	Enable HCLK_CPU into FIXME
RESERVED	[13]	-
ACLK_INTMEM	[12]	Enable ACLK_CPU into the on-chip SRAM
RESERVED	[11]	-
ACLK_STRC_SYS	[10]	Enable ACLK_CPU into FIXME
HCLK_AHB2APB	[9]	Enable HCLK_CPU into DIV_HCLK_AHB2APB
HCLK_CPUBUS	[8]	Enable HCLK_CPU into FIXME
HCLK_PERI_EMEM	[7]	Enable HCLK_PERI into FIXME
HCLK_PERI_AHB_ARBI	[6]	Enable HCLK_PERI into FIXME
HCLK_PERI_USB	[5]	Enable HCLK_PERI, used by HCLK_OTGx
ACLK_PERI_NIU	[4]	Enable ACLK_PERI (also known as ACLK_PEI_NIU)
ACLK_PERI_AXI_MATRIX	[3]	Enable ACLK_PERI into FIXME
ACLK_CPU_PERI	[2]	Enable ACLK_PERI into FIXME
PCLK_PERI_AXI_MATRIX	[1]	Enable PCLK_PERI into FIXME
HCLK_PERI_AXI_MATRIX	[0]	Enable HCLK_PERI into FIXME

CLKGATE_CON_5 configures „something"

CLKGATE_CON_5	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
RESERVED	[15]	-
RESERVED	[14]	-
HCLK_OTG0	[13]	Enable HCLK_PERI_USB into the 1st OTG controller
HCLK_MMC2	[12]	Enable HCLK_PERI into the EMMC controller (BIU clock)
HCLK_MMC1	[11]	Enable HCLK_PERI into the 2nd MMC controller (also called HCLK_SDIO) (BIU clock)
HCLK_MMC0	[10]	Enable HCLK_PERI into the 1st MMC controller (also called HCLK_SDMMC) (BIU clock)
HCLK_NAND	[9]	Enable HCLK_PERI into the NAND controller
ACLK_SMC	[8]	Enable ACLK_PERI into the Static Memory Controller
PCLK_DDRUPCTL	[7]	Enable PCLK_CPU into FIXME
HCLK_ROM	[6]	Enable HCLK_CPU into FIXME
PCLK_PMU	[5]	Enable PCLK_CPU into the PMU
PCLK_GRF	[4]	Enable PCLK_CPU into „General Register Files", aka PINMUX/PINCONF, DMA and general multi-purpose registers.

PCLK_TZPC	[3]	Enable PCLK_CPU into Trustzone Protection Controller
PCLK_EFUSE	[2]	Enable PCLK_CPU into the eFuse
ACLK_DMAC2	[1]	Enable ACLK_PERI into the 2nd DMA controller
ACLK_DMAC1	[0]	Enable ACLK_CPU into the 1st DMA controller

CLKGATE_CON_6 configures „something"

CLKGATE_CON_6	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
RESERVED	[15]	-
RESERVED	[14]	-
ACLK_VIO0	[13]	Enable DIV_ACLK_LCDC0 into FIXME (also known as aclk-lcdc0-pre)
HCLK_VIO_BUS	[12]	Enable HCLK_CPU into FIXME
ACLK_RGA	[11]	Enable ACLK_VIO1 into Raster Graphic Accelerator
HCLK_RGA	[10]	Enable HCLK_CPU into Raster Graphic Accelerator
HCLK_IPP	[9]	Enable HCLK_CPU into the Image Postprocessor
ACLK_IPP	[8]	Enable ACLK_VIO0 into the Image Postprocessor
RESERVED	[7]	-
RESERVED	[6]	-
ACLK_CIF	[5]	Enable ACLK_VIO0 into the Camera interface
HCLK_CIF	[4]	Enable HCLK_CPU into the Camera interface
ACLK_LCDC1	[3]	Enable ACLK_VIO1 into the 2nd LCD controller
HCLK_LCDC1	[2]	Enable HCLK_CPU into the 2nd LCD controller
HCLK_LCDC0	[1]	Enable HCLK_CPU into the 1st LCD controller
ACLK_LCDC0	[0]	Enable ACLK_VIO0 into the 1st LCD controller

CLKGATE_CON_7 configures „something"

CLKGATE_CON_7	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
PCLK_WDT	[15]	Enable PCLK_PERI into the Watchdog controller
PCLK_SARADC	[14]	Enable PCLK_PERI into the SARADC controller
PCLK_SPI1	[13]	Enable PCLK_PERI into the 2dn SPI controller
PCLK_SPI0	[12]	Enable PCLK_PERI into the 1st SPI controller
PCLK_PWM23	[11]	Enable PCLK_PERI into the PWM controller
PCLK_PWM01	[10]	Enable PCLK_CPU into the PWM controller
PCLK_TIMER2	[9]	Enable PCLK_CPU into the 3rd timer
RESERVED	[8]	-
PCLK_TIMER0	[7]	Enable PCLK_CPU into the 1st timer

HCLK_PIDF	[6]	Enable HCLK_PERI into the PID-Filter
HCLK_HSADC	[5]	Enable HCLK_PERI into the HSADC controller
HCLK_HSIC	[4]	Enable HCLK_PERI into the HSIC controller
HCLK_OTG1	[3]	Enable HCLK_PERI_USB into the 2nd OTG controller
HCLK_I2S	[2]	Enable HCLK_CPU into I2S controller
HCLK_SPDIF	[1]	Enable HCLK_CPU into SPDIF controller
HCLK_EMAC	[0]	Enable HCLK_PERI into the EMAC controller

CLKGATE_CON_8 configures „something"

CLKGATE_CON_8	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
RESERVED	[15]	-
RESERVED	[14]	-
ACLK_GPS	[13]	Enable ACLK_PERI into the GPS controller
PCLK_GPIO3	[12]	Enable PCLK_PERI into the 4th GPIO controller
PCLK_GPIO2	[11]	Enable PCLK_CPU into the 3rd GPIO controller
PCLK_GPIO1	[10]	Enable PCLK_CPU into the 2nd GPIO controller
PCLK_GPIO0	[9]	Enable PCLK_CPU into the 1st GPIO controller
PCLK_I2C4	[8]	Enable PCLK_PERI into the 5th I2C controller
PCLK_I2C3	[7]	Enable PCLK_PERI into the 4th I2C controller
PCLK_I2C2	[6]	Enable PCLK_PERI into the 3rd I2C controller
PCLK_I2C1	[5]	Enable PCLK_CPU into the 2nd I2C controller
PCLK_I2C0	[4]	Enable PCLK_CPU into the 1st I2C controller
PCLK_UART3	[3]	Enable PCLK_PERI into the 4th serial controller
PCLK_UART2	[2]	Enable PCLK_PERI into the 3rd serial controller
PCLK_UART1	[1]	Enable DIV_AHB2APB into the 2nd serial controller
PCLK_UART0	[0]	Enable DIV_AHB2APB into the 1st serial controller

CLKGATE_CON_9 configures „something"

CLKGATE_CON_9	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
RESERVED	[15]	-
RESERVED	[14]	-
RESERVED	[13]	-
RESERVED	[12]	-
RESERVED	[11]	-
RESERVED	[10]	-
RESERVED	[9]	-

RESERVED	[8]	-
ACLK_GPU	[7]	Enable DIV_ACLK_GPU
PCLK_PUBL	[6]	Enable PCLK_CPU into the DDR
ACLK_VIO1	[5]	Enable DIV_ACLK_LCDC1 into FIXME (also known as aclk-lcdc1-pre)
CORE_L2C	[4]	Enable DIV_CORE into the L2Cache
ATCLK	[3]	Enable PCLK_CPU into FIXME
CLK_TRACE	[2]	FIXME: unknown
PCLK_DBG	[1]	Enable PCLK_CPU into FIXME
CORE_DBG	[0]	Enable DIV_CORE into FIXME

Softreset control registers

Registers controlling the reset state of different SoC peripherals

SOFTRST_CON_0	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
L2C	[15]	
STRC_SYS_AXI	[14]	
CORE1_WDT	[13]	
CORE0_WDT	[12]	
CORE3_DBG	[11]	
CORE2_DBG	[10]	
CORE1_DBG	[9]	
CORE0_DBG	[8]	
MCORE_DBG	[7]	
CORE3	[6]	
CORE2	[5]	
CORE1	[4]	
CORE0	[3]	
MCORE	[2]	
PTM_CORE3	[1]	
PTM_CORE2	[0]	

SOFTRST_CON_1	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
TIMER6	[15]	
EFUSE_APB	[14]	
TIMER3	[13]	

TIMER1	[12]	
TIMER0	[11]	
SPDIR	[10]	
TIMER5	[9]	
I2S	[8]	
TIMER4	[7]	
ROM	[6]	
INTMEM	[5]	
DMA1	[4]	
AHB2APB	[3]	
RESERVED	[2]	-
CPUSYS_AHB	[1]	
TIMER2	[0]	

SOFTRST_CON_2	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
I2C4	[15]	
I2C3	[14]	
I2C2	[13]	
I2C1	[12]	
I2C0	[11]	
UART3	[10]	
UART2	[9]	
UART1	[8]	
UART0	[7]	
RESERVED	[6]	-
PTM3_ATB	[5]	
PTM3	[4]	
GPIO3	[3]	
GPIO2	[2]	
GPIO1	[1]	
GPIO0	[0]	

SOFTRST_CON_3	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
USB_PERI	[15]	

EMEM_PERI	[14]	
CPU_PERI	[13]	
PERIPH_NIU	[12]	
PERIPHSYS_APB	[11]	
PERIPHSYS_AHB	[10]	
PERIPHSYS_AXI	[9]	
PMU	[8]	
GRF	[7]	
PMU_ATP	[6]	
TPIU_ATB	[5]	
DAP_SYS	[4]	
DAP	[3]	
DAP_PO	[2]	
PWM1	[1]	
PWM0	[0]	

SOFTRST_CON_4	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
DDRMSCH	[15]	
TIMER_APB	[14]	
PIDFILTER	[13]	
HSADC	[12]	
HSICPHY	[11]	
OTGC1	[10]	
USBPHY1	[9]	
USBOTG1	[8]	
OTGC0	[7]	
USBPHY0	[6]	
USBOTG0	[5]	
NANDC	[4]	
GPS	[3]	
MAC	[2]	
SMC	[1]	
DMA2	[0]	

SOFTRST_CON_5	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing

CORE3_WDT	[15]	
CORE2_WDT	[14]	
DDRPHY_CTL	[13]	
PTM2	[12]	
DDRCTL_APB	[11]	
DDRCTL	[10]	
DDRPHY_APB	[9]	
DDRPHY	[8]	
SARADC	[7]	
WDT	[6]	
SPI1	[5]	
SPI0	[4]	
MMC2	[3]	
MMC1	[2]	
MMC0	[1]	
TZPC	[0]	

SOFTRST_CON_6	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
PTM2_ATB	[15]	With comment „SOFT_RST_6RES15“
CIF	[14]	
RGA_AHB	[13]	
RGA_AXI	[12]	
IPP_AHB	[11]	
IPP_AXI	[10]	
LCDC1_DCLK	[9]	
LCDC1_AHB	[8]	
LCDC1_AXI	[7]	
LCDC0_DCLK	[6]	
LCDC0_AHB	[5]	
LCDC0_AXI	[4]	
VIO_BUS_AHB	[3]	
VIO0_AXI	[2]	
RESERVED	[1]	-
RESERVED	[0]	-

SOFTRST_CON_	Bit	Description

7		
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
CTI4_APB	[15]	
TFUN_APB	[14]	
TFUN_ATP	[13]	
CTI3_APB	[12]	
CTI3	[11]	
GPU_NIU_AXI	[10]	
GPU_BRIDGE_AXI	[9]	
GPU_CORE	[8]	
CTI2_APB	[7]	
CTI2	[6]	
HSIC_AHB	[5]	
VCODEC_NIU_AXI	[4]	
CPU_VCODEC	[3]	
VIO1_AXI	[2]	
VCODEC_AHB	[1]	
VCODEC_AXI	[0]	

SOFTRST_CON_8	Bit	Description
HIWORD-MASK	[31:16]	Enable corresponding bit [15:0] for writing
TS	[15]	
CTM	[14]	
PTM1_ATB	[13]	
PTM1	[12]	
PTM0_ATB	[11]	
PTM0	[10]	
PTM_CORE1	[9]	
PTM_CORE0	[8]	
CTI1_APB	[7]	
CTI1	[6]	
CTI0_APB	[5]	
CTI0	[4]	
DBG_APB	[3]	
CORE_DBG	[2]	
TRACE	[1]	
TPIU_APB	[0]	

Unknown registers

GLB_SRST_FST	Bit	Description
FIXME		Not used in the upstream code at all

GLB_SRST_SND	Bit	Description
FIXME		During system reset, 0xeca8 gets written to it

MISC_CON	Bit	Description
FIXME		rk_pm_soc_prepare writes 0x07000000 into it

GLB_CNT_TH	Bit	Description
FIXME		Not used in the upstream code at all

System peripheral supplies

Peripheral	Supplied by clock
SMP_TWD	GATE_CORE_PERIPH

Power management unit (PMU)

Handling the power domains

The current handling is, to change the relevant bit in PMU_PWRDN_CON and busy wait until the corresponding bit in PMU_PWRDN_ST reflects that new value.

When starting the power domain of one of the cpu core, the core will execute directly after its start the code that is starting at position 0x0 of the sram.

Idle requests to a power domain

It is possible to put components handled by the PMU into an idle mode or get them out of them. The current code does this like the following:

change the relevant bit in PMU_MISC_CON1

busy-wait until the relevant ACK-bit in PMU_PWRDN_ST reflects that new value

busy-wait until the relevant IDLE-bit in PMU_PWRDN_ST reflects that new value

System reset

The current handling is:

- setting boot mode and boot flags in SYS_REG0 and SYS_REG1
- disabling the „remap" by setting bit 12 of GRF_SOC_CON0 to 0 (write 1 << 12+16 | 0 << 12)
- setting all PLLs to slow mode
- writing 0xeca8 into CRU_GLB_SRST_SND

Register-Map

Register	Address	Description
PMU_WAKEUP_CFG0	0x20004000	
PMU_WAKEUP_CFG1	0x20004004	
PMU_PWRDN_CON	0x20004008	
PMU_PWRDN_ST	0x2000400C	
PMU_INT_CON	0x20004010	
PMU_INT_ST	0x20004014	
PMU_MISC_CON	0x20004018	
PMU_OSC_CNT	0x2000401C	
PMU_PLL_CNT	0x20004020	
PMU_PMU_CNT	0x20004024	
PMU_DDRIO_PWRDN_CNT	0x20004028	
PMU_WAKEUP_RST_CLR_CNT	0x2000402C	
PMU_SCU_PWRDN_CNT	0x20004030	
PMU_SCU_PWRUP_CNT	0x20004034	
PMU_MISC_CON1	0x20004038	

PMU_GPIO0_CON	0x2000403C	Unused in current kernels
PMU_SYS_REG0	0x20004040	
PMU_SYS_REG1	0x20004044	
PMU_SYS_REG2	0x20004048	
PMU_SYS_REG3	0x2000404C	

PMU_PWRDN_CON controls the power state of core power domains

PMU_PWRDN_CON	Bit	Description
RESERVED	[31:11]	-
PD_CS	[10]	Also known as PD_DBG
PD_GPU	[9]	
PD_VIDEO	[8]	
PD_VIO	[7]	
PD_PERI	[6]	
PD_CPU	[5]	
PD_SCU	[4]	Set to 1 to power down the SCU
PD_A9_3	[3]	Set to 1 to power down the 4th cpu core
PD_A9_2	[2]	Set to 1 to power down the 3rd cpu core
PD_A9_1	[1]	Set to 1 to power down the 2nd cpu core
PD_A9_0	[0]	Set to 1 to power down the 1st cpu core

PMU_PWRDN_ST shows the real power- and idle-states of core power domains

The bits get set to their respective value after the powerstate was sucessfully changed.

PMU_PWRDN_ST	Bit	Description
ACK_IDLE_CPU	[31]	Beginning of ack_mask
ACK_IDLE_PERI	[30]	
ACK_IDLE_GPU	[29]	
ACK_IDLE_VIDEO	[28]	
ACK_IDLE_VIO	[27]	
IDLE_CPU	[26]	Beginning of idle_mask
IDLE_PERI	[25]	
IDLE_GPU	[24]	
IDLE_VIDEO	[23]	

IDLE_VIO	[22]	
UNKNOWN	[21:19]	
ACK_IDLE_CORE	[18]	
ACK_IDLE_DMA	[17]	
UNKNOWN	[16]	
IDLE_CORE	[15]	
IDLE_DMA	[14]	
UNKNOWN	[13:11]	
PD_CS	[10]	Also known as PD_DBG
PD_GPU	[9]	
PD_VIDEO	[8]	
PD_VIO	[7]	
PD_PERI	[6]	
PD_CPU	[5]	
PD_SCU	[4]	Power state of the SCU 0 = running, 1 = powered down
PD_A9_3	[3]	Power state of the 4th cpu core 0 = running, 1 = powered down
PD_A9_2	[2]	Power state of the 3rd cpu core 0 = running, 1 = powered down
PD_A9_1	[1]	Power state of the 2nd cpu core 0 = running, 1 = powered down
PD_A9_0	[0]	Power state of the 1st cpu core 0 = running, 1 = powered down

PMU_MISC_CON1

PMU_MISC_CON1	Bit	Description
UNKNOWN	[31:17]	-
IDLE_REQ_DMA	[16]	
UNKNOWN	[15]	-
IDLE_REQ_CORE	[14]	
UNKNOWN	[13:6]	-
IDLE_REQ_VIO	[5]	
IDLE_REQ_VIDEO	[4]	
IDLE_REQ_GPU	[3]	
IDLE_REQ_PERI	[2]	
IDLE_REQ_CPU	[1]	
UNKNOWN	[0]	-

PMU_SYS_REG0

PMU_SYS_REG0	Bit	Description
REBOOT_FLAG	[31:8]	0x5242C3 = Loader reboot, 0xC35242 = Kernel reboot
REBOOT_TYPE	[7:0]	0 = normal, 1 = loader, 2 = maskrom, 3 = recover, 4 = norecover, 5 = secondos, 6 = wipedata, 7 = wipeall, 8 = checkimg, 9 = fastboot Most likely this register is just preserved over a reboot, so that the bootloader or next kernel can read it. According to the code, REG0 is mainly for the loader.

PMU_SYS_REG1

PMU_SYS_REG1	Bit	Description
BOOT_MODE	[x:0]	0 = normal, 1 = factory2, 2 = recovery, 3 = charge, 4 = power_test, 5 = offmode_charging, 6 = reboot, 7 = panic, 8 = watchdog Most likely this register is just preserved over a reboot, so that the bootloader or next kernel can read it. According to the code, REG0 is mainly for the kernel.

PMU_SYS_REG2

PMU_SYS_REG2	Bit	Description
UNKNOWN	[31:0]	-

PMU_SYS_REG3

PMU_SYS_REG3	Bit	Description
UNKNOWN	[31:0]	-

PMU_SYS_REG0

CLKSEL_CON_34	Bit	Description