CJC6822A

A Cortex-M0+ based MCU,designed for USB Headphone Appliances

Revision History

Version	Author	Date	Note
V1.0	Loyal	2021-07-15	Draft vision
V1.1	Loyal	2022-02-15	Mistake
V1.2	Loyal	2022-06-14	Add Power-on sequence

1. Overview

CJC6822 is a Cortex-M0+ based MCU, designed for USB headphone appliances.
It integrates one 32-bit RISC CPU with 16KB SRAM, USB, UART, IIC, audio codec, GPIO, TIMER, WDT, PWM, SPI, IIS, SPDIF, SARADC, PLL, LDO etc.
CJC6822 can boot from external flash through SPI interface. After powered on, the program is read from external flash into internal SRAM for execution.
The CJC6822 can run up to 48MHz, and it is designed with special care to minimize the power consumption while allowing for the flexibility to reach for high performance. It includes the clock gating for individual IP, and CJC6822 can be further operated under different power-saving modes: Normal, Idle, Standby, Power-down, different mode have different clock and power strategy.

1.1 Features

- **Cortex-M0+ Like**
- **LDO**
- ◆ Built-in LDO for wide operating voltage range:3.3V/1.8V
- **Memory**
- ◆ Support program memory up to 16KB
- ◆ RAM:16KB SRAM
- **In-system programming & In-Circuit programming by USB/UART**
- **Clock control**
- ◆ Programmable system clock source
- ◆ 4-6MHz internal RC-oscillator(1% accuracy at 25℃)
- ◆ 12MHz clock from USB oscillator
- ◆ Support external crystal oscillator
- ◆ 10KHz internal low-power RC-oscillator for watchdog and idle wake-up
- **USB Compliance**
- ◆ USB Spec.V2.0 high speed/full speed mode compatible
- ◆ USB Audio Class V1.0/V2.0 compatible
- ◆ USB Human Interface Device V1.1 compatible
- ◆ Support USB suspend/resume/reset function
- ◆ Support control, interrupt, bulk and isochronous data transfer
- **Audio codec**
- ◆ Default sample rate:192K/176.4K/96K/88.2K/48K/44.1K
- ◆ Support bit length:16/20/24/32bit
- **I/O port**
- ◆ Up to 8 general purpose I/O(GPIO)
- **TIMER**
- ◆ 3 internal timers
- ◆ Internal or external clock source selection

- ◆ Interrupt can be issued upon overflow and time-up
- ◆ Each timer has two match registers
- ◆ Supports the incrementing and decrementing models

- **Watchdog Timer**

- ◆ During the timeout, the outputs are one or a combination of the following signals
  -----System reset
  -----System interrupt
  -----External interrupt
- ◆ 32-bit down counter
- ◆ Internal or external clock source selection
- ◆ A variable time-out period of reset
- ◆ Access protection

- **PWM**

- ◆ One 16-bit timers PWM channel
- ◆ Programmable duty control of output waveform
- ◆ Auto reload mode or one-shot pulse mode
- ◆ Capture and compare function

- **UART**

- ◆ Programmable baud rates ,Baud rate up to 921.6K
- ◆ Support 38KHz house IR transceiver

- **SPI**

- ◆ One specified SPI interface as AHB device for boot loader and APB device for write back
- ◆ speed up to 40MHz

- **I2C**

- ◆ compatible with Philips IIC standard

- **I2S/SPDIF**

- ◆ support the Sony/Philips Digital Interface Format(SPDIF) transmitter
- ◆ support master/slave mode and 16/24/32bit data width

- **SARADC**

- ◆ 1 channel analog input
- ◆ 8bit SARADC, 4bit accuracy is guaranteed

- **Brown out reset**

- ◆ Programmable 3 threshold levels: 2.7V/2.4V/2.0V(default 2.0V)
- ◆ Optional BOD interrupt or reset

- **Operating temperature:-20~+85 Degree**
- **Package: QFN48 6*6**

1.2 System diagram

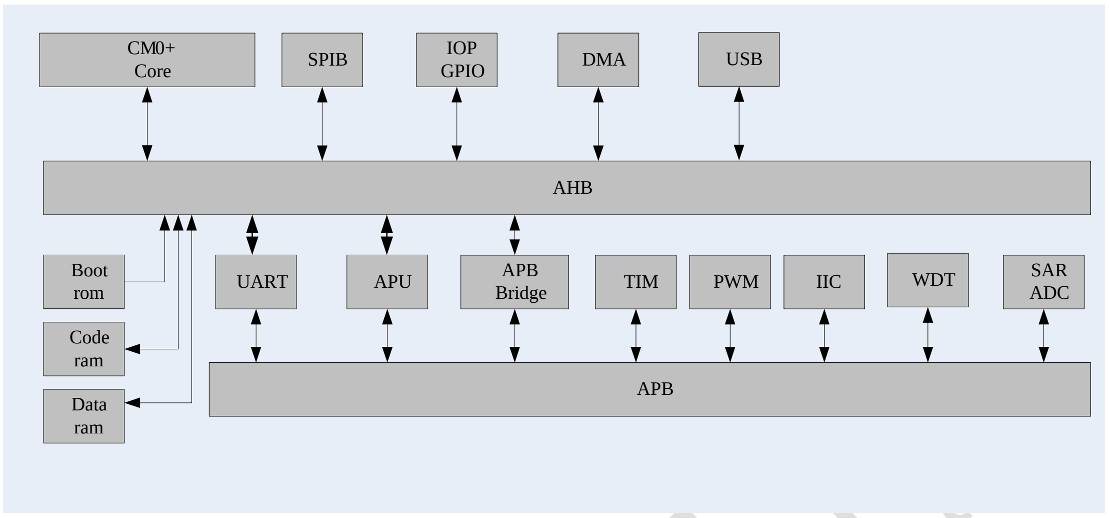
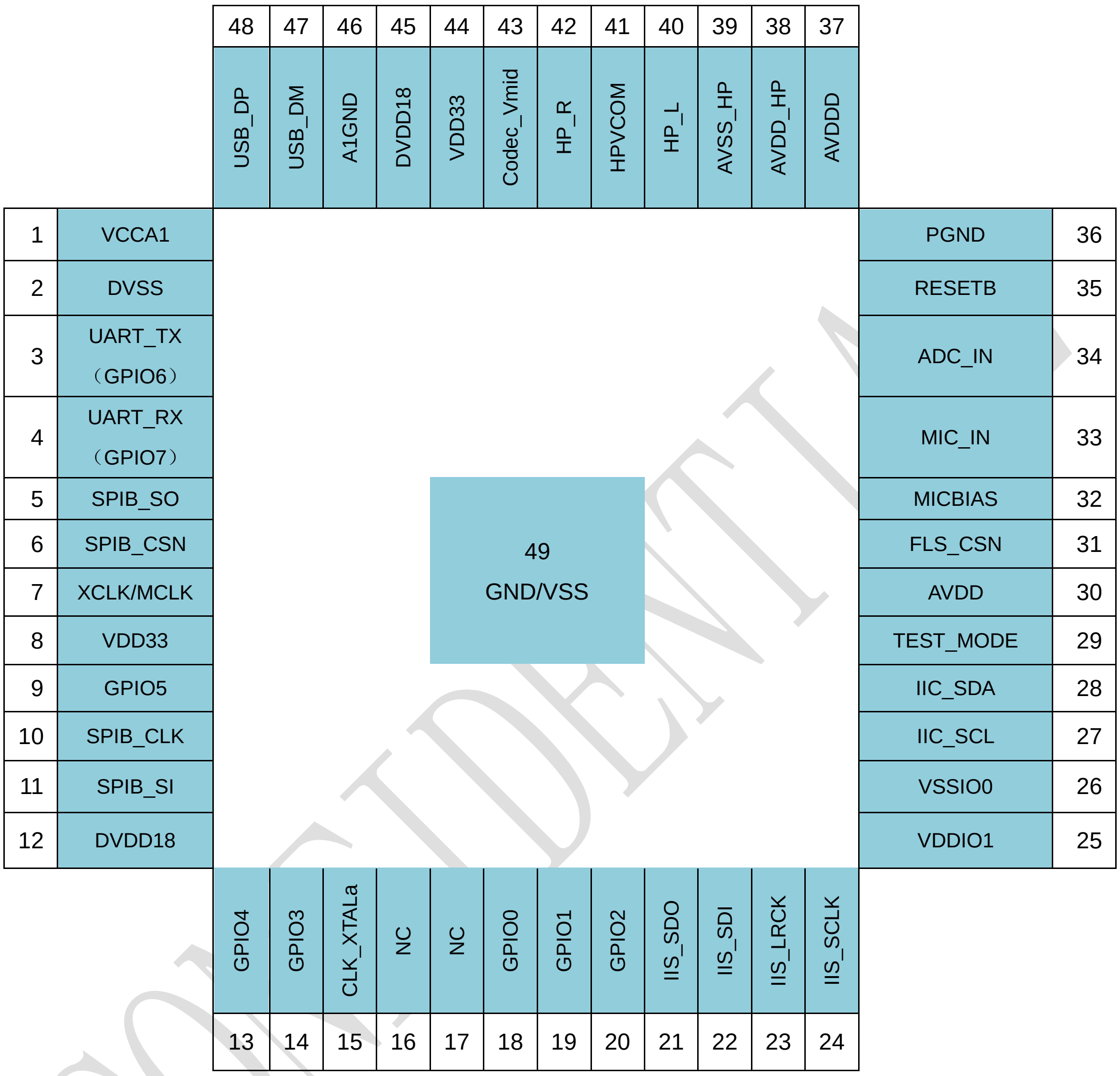


Figure 1. CJC6822 chip block diagram

2 .PIN diagram

2.1 QFN48 PACKAGE



3 .PIN description

QFN48 PACKAGE PIN DESCRIPTION

48 Pin No.	PIN_NAME	IO_TYPE	COMMENT
DIGITAL PIN			
3	UART_TX	IO	UART transmitter(GPIO6)
4	UART_RX	IO	UART transmitter(GPIO7)
5	SPIB_SO	IO	SPI data out for boot flash
6	SPIB_CSN	IO	SPI CSN
9	GPIO5	IO	GPIO5
10	SPIO_CLK	IO	SPI CLOCK for boot flash
11	SPIB_SI	IO	SPI data in for boot flash
13	GPIO4	IO	GPIO4
14	GPIO3	IO	GPIO3
16	SWDIO	IO	NC
17	SWCLK	IO	NC
18	GPIO0	IO	GPIO0(PWM OUT)
19	GPIO1	IO	GPIO1
20	GPIO2	IO	GPIO2
21	IIS_SDO	IO	IIS DATA OUTPUT
22	IIS_SDI	IO	IIS DATA INPUT
23	IIS_LRCK	IO	IIS SAMPLING RATE CLOCK IN/OUT
24	IIS_SCLK	IO	IIS DATA CLOCK IN/OUT
27	IIC_SCL	IO	IIC DATA CLOCK IN/OUT
28	IIC_SDA	IO	IIC DATA IN/OUT
31	FLS_CSN	IO	SPI CHIP SELECT FOR BOOT FLASH
7	XCLK/MCLK	IO	IIS MCLK
29	TEST MODE		GND
30			
ANALOG PIN			
48	USB_DP	IO	USB DP
32	MIC_BIAS	IO	Mic Bias
33	MIC_IN	IO	Mic input
34	ADC_IN	IO	ADC input
35	RESETB	IO	RESET
40	HP_L	IO	left HP
41	HP_VCOM	IO	HP Voltage Reference
42	HP_R	IO	right HP
43	Codec_Vmid	IO	Codec Voltage Reference
47	USB_DM	IO	USB DM
48	USB_DP	IO	USB DP
POWER			
1	VCCA1	POWER	Power for USB digital

2	DVSS	GROUND	Ground for Digital
8	VDD33	POWER	Power for IO
12	DVDD18	POWER	Power for Core
25	VDDIO1	POWER	Power for IO
26	VSSIO0	GROUND	Ground for IO
30	BOOT_MODE	POWER	Power for Core
37	AVDD	POWER	Power for HP
38	AVDD_HP	POWER	Power for HP
39	AVSS_HP	POWER	Ground for HP
44	FLS_VDD33	POWER	power for Flash
45	DVDD18	POWER	Power for Core
46	A1GND	GROUND	Ground for USB
49	GND/VSS	GROUND	Ground for Digital
RESET			
36	GND	IO	Ground for IO
CLK			
15	CLK_XTALa	IO	12MHz OSC INPUT(default NC)

4 .ELECTRICAL CHARACTERISTICS

Test Conditions

AVDD = AVDD_HP = VDD33 = VCCA2 = VCCA1 = VDD_IO1 = 3.3V, All module is power on, TA = 25℃(unless otherwise noted).

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Line Input to ADC						
Input Signal Level (0dB)	VINLINE			1AVD D		VPP
Signal to Noise Ratio	SNR	A-weighted, 0dB gain @ fs = 48kHz		86		dB
Dynamic Range	DR	A-weighted, -60dB full scale input		86		dB
Total Harmonic Distortion +Noise Ratio	THD+N	input, 0dB gain		-71		dB
Power Supply Rejection Ratio	PSRR	1kHz, 50mVpp		-33		dB

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
LINE OUT FROM DAC						
Input Signal Level (0dB)	VINLINE			1AVD D		VPP
Signal to Noise Ratio	SNR	A-weighted, 0dB gain @ fs = 48kHz		95		dB
Dynamic Range	DR	A-weighted, -60dB full scale input		95		dB
Total Harmonic Distortion +Noise Ratio	THD+N	input, 0dB gain		-68		dB
Power Supply Rejection Ratio	PSRR	1kHz, 50mVpp		-33		dB

5 .Function description

5.1 CJC6822 address map

Figure 3 shows CJC6822 address map.

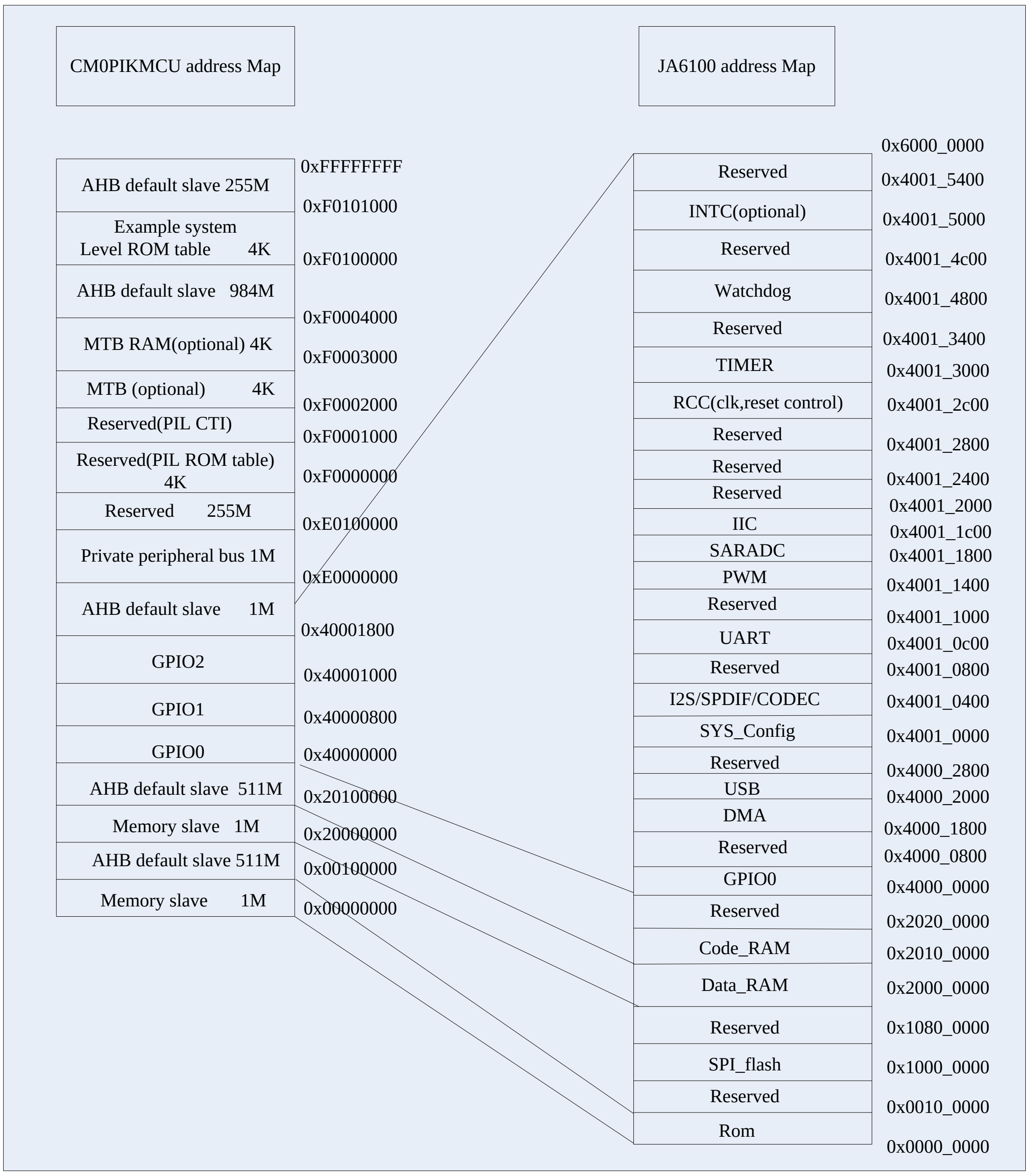


Figure 3 CJC6822 address map

5.2 BUS interface unit

CJC6822 chip integrate 2 AHB bus and 1 APB (AMBA protocol compatible). CPU core operates as AHB master in one AHB bus, and DMA controller operates as AHB master on other AHB bus. One AHB2APB Bridge is used for peripherals configuration.(see Figure1).

5.3 ROM

CJC6822 integrate 1KB boot ROM . When ISP is available, CPU Boots from internal boot ROM, Receives program code from UART bus and Stores in external flash. If normal mode is enabled, CPU Boots from internal boot ROM, Fetch program code from external SPI flash and Stores in internal SRAM, then, re-mapping memory configuration, boots from internal SRAM.

5.4 SRAM

The embedded high-speed SRAM is designed for both program code and scratchpad RAM. CJC6822 integrates one 16KB SRAM as the system program memory, 16KB SRAM as the data memory.

5.5 PLL and clock generation

PLL module generates system and block level clock from the 12MHz USB-oscillator or an external 12MHz crystal. CJC6822 chip contains two clock domains: one is system PLL clock source domain and another is Audio processor clock source domain. System PLL clock source domain includes CPU clock, AHB clock, APB clock based on the 12MHz clock source; Audio processor clock source domain offer clock source for audio processor according to the audio sampling rate, if sampling rate is 8KHz, 16KHz, 32KHz, 48KHz and so on, the APU clock is 24.576MHz and if sampling rate is 22KHz, 44.1KHz and so on, the APU clock is 22.5792MHz. The APU PLL output can be configured via registers.
CJC6822 chip has one internal low-power oscillator to generate 10KHz output. Figure 4 is CJC6822 clock diagram.

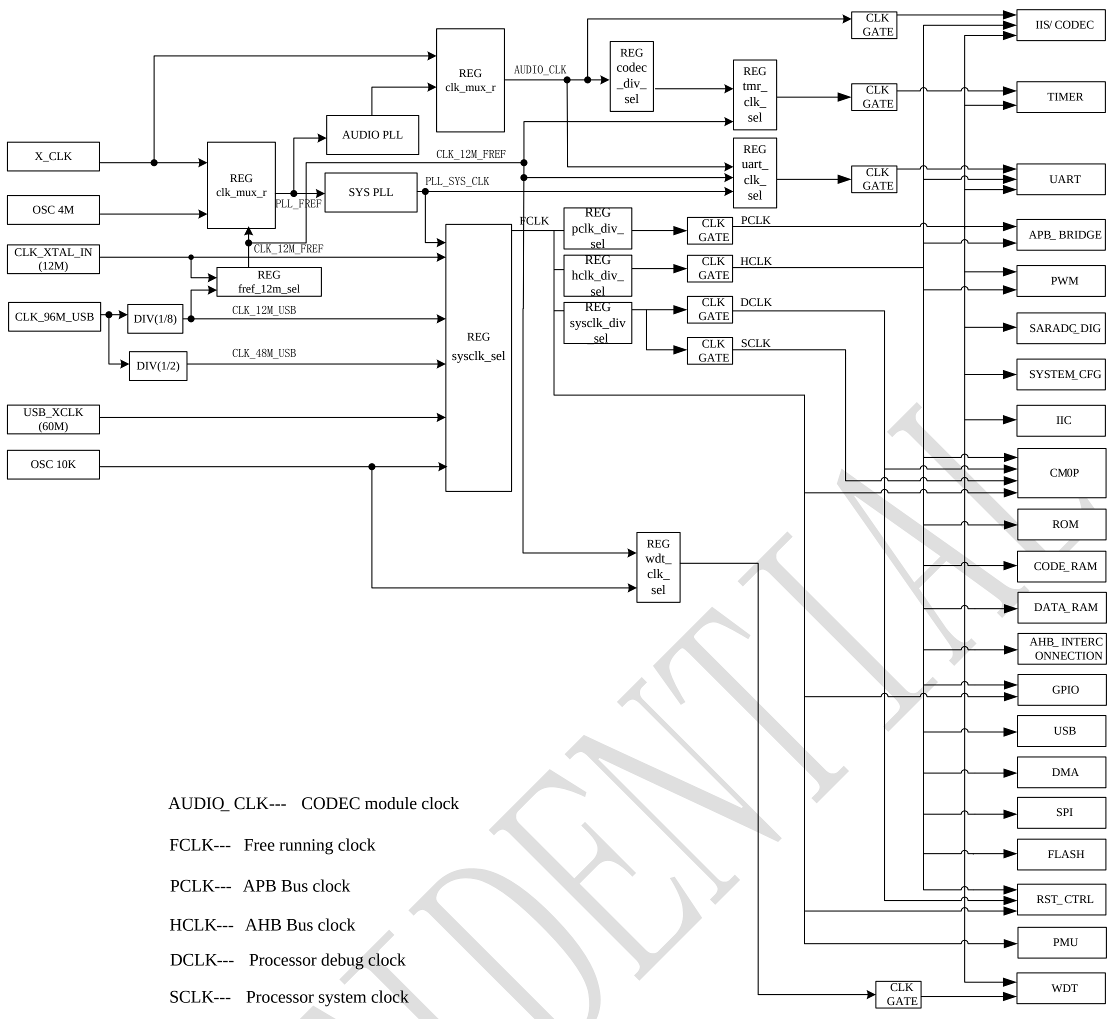


Figure 4 CJC6822 clock diagram

Table 2 system control register list (BaseAddr = 0x4001_0000).

Addr.	Name	Type	Default	Bit	Default
0x00	R0	R/W	0x0	[31:16]	Reserved
				[15:14]	mux_ctrl_gpio7: 2‘h1:GPIO7, 2‘h0：UART_RXD
				[13:12]	mux_ctrl_gpio6: 2‘h1: GPIO6, 2‘h2：UART_TXD
				[11:10]	mux_ctrl_gpio5: 2‘h1:GPIO5
				[9:8]	mux_ctrl_gpio4: 2‘h1:GPIO4
				[7:6]	mux_ctrl_gpio3: 2‘h1:GPIO3
				[5:4]	mux_ctrl_gpio2: 2‘h1:GPIO2
				[3:2]	mux_ctrl_gpio1: 2‘h1：GPIO1
				[1:0]	mux_ctrl_gpio0: 2‘h1：GPIO0, 2‘h2：PWM output
0x04	R1	R/W	0x0	[31:1]	Reserved
				[0]	sys_remap : 1:enable & reset 1: Address 32’h0000_0000 remap to code_ram 0: Address 32’h0000_0000 remap to ROM
0x08	R2	R/W	0x0	[31:9]	Reserved
				[8]	PLL 12MHz reference clock source select:

					clk_xtal_in: 12MHz clock from IO PAD CLK_XTAL clk_12m_usb: 12MHz clock from USB_oscillator fref_12m_sel: 1,clk_12m_fref = clk_xtal_in; 0,clk_12m_fref = clk_12m_usb

				[7:4]	A clock mux for PLL reference clock select, audio clock source select and X_CLK PAD clock out source select.
					clk_4m: 4MHz clock from RC_oscillator
					clk_12m_fref: 12MHz clock for reference
					clk_i: clock input from IO PAD X_CLK
					pll_audio_clk_i: audio clock from PLL
					pll_sys_clk: system clock from PLL
					osc_10k: 10kHz clock from internal oscillator
					clk_o: clock output from IO PAD X_CLK
					
					clk_mux:
					b0000: pll_fref = clk_4m;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = 1'h0;
					b0001: pll_fref = clk_12m_fref;
					pll_audio_clk_w = clk_i;
					clk_o = 1'h0;
					b0010: pll_fref = clk_i;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = 1'h0;
					b1000: pll_fref = clk_12m_fref;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = pll_sys_clk;
					b1001: pll_fref = clk_12m_fref;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = pll_audio_clk_i;
					b1010: pll_fref = clk_12m_fref;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = osc_10k;
					b1011: pll_fref = clk_12m_fref;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = clk_12m_fref;
					other: pll_fref = clk_12m_fref;
					pll_audio_clk_w = pll_audio_clk_i;
					clk_o = 1'h0;

				[3]	pll_bypass 1: Bypass PLL, sys_clk/audio_clk = fref_clk
				[2:1]	lvr_in :

					config the threshold voltage for low voltage reset b00 : 2.0V b01 : 2.4V b10 : 2.7V b11 : 3V
				[0]	lvr_en : enable low voltage reset
0x0C	R3	R/W	0x0	[31:4]	Reserved
				[3]	Sys_pll_rstn: Sys clock PLL reset 1:reset 0: work
				[2]	Audio_pll_rstn: audio pll reset 1: reset 0: work
				[1]	Sys_pll_pdn: Sys clock PLL power down 1:power down 0: work
				[0]	Audio_pll_pdn: audio pll power down 1:power down 0: work
0x10	R4	R/W	0x0	[31:24]	Reserved
				[23:22]	codec_div_sel: pll_audio_clk_w divider 0: pll_audio_clk_div = pll_audio_clk_w/2 1: pll_audio_clk_div = pll_audio_clk_w/4 2: pll_audio_clk_div = pll_audio_clk_w/8 Others: pll_audio_clk_div = pll_audio_clk_w/2
				[21:16]	tmr_clk_sel [17:16] == 'h0: tmr1_clk_w=clk_12m_fref [17:16] == 'h1: tmr1_clk_w=pll_audio_clk_div [19:18] == 'h0: tmr2_clk_w=clk_12m_fref [19:18] == 'h1: tmr2_clk_w=pll_audio_clk_div [21:20] == 'h0: tmr3_clk_w=clk_12m_fref [21:20] == 'h1: tmr3_clk_w=pll_audio_clk_div
				[15:14]	uart_clk_sel: uart clock select 0 fref 12MHz 1 pll 48MHz 2 audio clk Other fref 12MHz
				[13:12]	Wdt_clk_sel: watchdog extclk select 0 :pll fref 12MHz 1 :10K osc Other: pll fref 12MHz
				[11:9]	pclk_div_sel: ratio of pclk 0: 1/1 1:1/2 2:1/4 3:1/8 4:1/16 other:reserved

				[8:6]	hclk_div_sel:ratio of hclk/core_clk 0:1/1 1:1/2 2:1/4 3:1/8 4:1/16 other:reserved
				[5:3]	sysclk_div_sel:Ratio of core_clk/system clock 0:1/1 1:1/2 2:1/4 3:1/8 4:1/16 other:reserved
				[2:0]	sysclk_sel 000: pll_sys_clk 001:clk_12m_usb 010: usb_cnt[0],usb_96m_clk/2 011:usb_xclk 100:osc_10k 101:clk_xtal_in Other: reserved
0x14	R5	R/W	0x24031270	[31]	Reserved
				[30]	pll_dther_en: 1:enable 0: disable
				[29:23]	Pll_mul_num, integral part of the multiple of PLL_VCO, enabled when pll_n_sel = 2'b10.
				[22:21]	pll_n_sel, ratio selector of pll 00: audio clk 24.576MHz 01: audio clk 22.5792MHz 10: manual ratio 11: reserved
				[20:5]	pll_sin16_in, fractional part of the multiple of PLL_VCO, enabled when pll_n_sel = 2'b10. pll_sin16_in[15:14] are sign bits, pll_sin16_in[13:0] are effective bits. pll_sin16_in[15:14] = 2'b00: pll_sin16_in[13:0]/(2^14) pll_sin16_in[15:14] = 2'b11: -(~pll_sin16_in[13:0] + 1'b1)/(2^14) others: reserved For example: pll_n_sel = 2’b10, pll_mul_num = 7’b0010011, pll_sin16_in = 16’b1111_0100_0011_1010, pll_div_num = 5’b10100, pll_fref = 12MHz. Fpll_vco = pll_fref * (pll_mul_num + pll_sin16_in/2^14) = 12 * (19 - 0.18396) = 225.79248MHz Fpll_clk_out = Fpll_vco/(pll_div_num/2) = 225.79248/10 = 22.579248MHz

				[4:0]	pll_div_num, div_num of PLL_VCO clock. pll_audio_clk = pll_vco_clk/ (pll_div_num/2).
0x18	R6	R/W	0x0	[31:5]	Reserved
				[4:3]	Pll_icp_trim:pll charge pump charge current control 00:default
				[2:1]	Pll_vco_trim, trim the frequency of pll_vco clock 2’b00: default, 24.576MHz 2’b01: 22.5792MHz,
				[0]	Pll_vco_test_en: pll vco test enable 1:enable 0: disable

5.6 DMA

DMA is designed to enhance the system performance and reduce the processor-interrupt generation. The system efficiency is improved by employing the high-speed data transfers between the system and the device. The DMA controller provides up to 16 configurable channels (Figure 5) for memory-to-memory, memory-to-peripheral, peripheral-to-peripheral, and peripheral-to-memory transfers with a shared buffer. Figure 5 shows the DMA controller module block diagram.

DMA consists of 5 main blocks: AHB master interfaces, AHB slave interface, FIFO buffer, and DMA core. AHB master interface transfer data between the system and the DMA FIFO, system can configure the DMA controller through AHB slave interface, FIFO buffer provides the buffer between the source and the destination, and DMA core is configurable up to an 16-channel DMA engine, both source and destination are on AHB Bus, Each channel can be assigned with a group priority level, and the same group priority is serviced in the round-robin fashion.

DMA controller uses the 4-group priority and the round-robin scheme to select which channel to serve. Arbitration is based on the priority level of the channels. If the channels have the same priority level, the arbitration will then be based on the round robin scheme. Each channel has a 2-bit priority value associated with it. A value of 3 indicates the highest priority level and a value of 0 indicates the lowest priority.

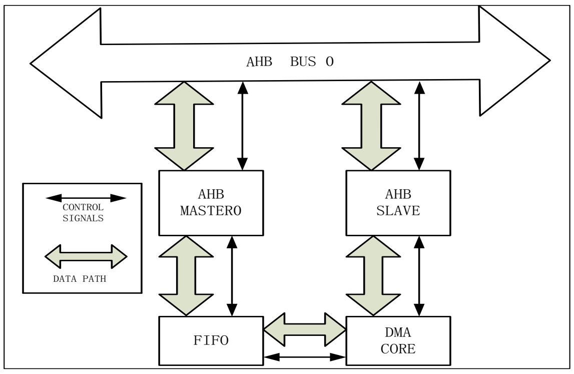


Figure 5 DMA controller module block diagram

DMA controller has one type work mode: hardware handshake mode.
Hardware handshake mode: when the channel wins the arbitration, the DMA controller will wait for the external DMA request to be asserted before starting the DMA transfer. Each time the DMA request is asserted, the controller transfers units equal to SRC_BURST_SIZE. When SRC_BURST_SIZE transfer is completed, the DMA controller asserts the acknowledge and then re-arbitrates among all DMA requests. After detecting the assertion of acknowledge, the external device should de-assert the DMA request to let the DMA controller de-assert acknowledge. After TOT_SIZE transfers have been done, the DMA controller asserts TC[0] (bit 0 of Terminal Count Status Register (TC)), dma_tc[0] and both dmaint_tc and dmaint interrupts (if not masked).
During the transfer, if the source or destination slave returns an ERROR response, the DMA will set the ERR bit and terminate the DMA transfer at once.
During the transfer, if the software sets the abort bit, after finishing SRC_BURST_SIZE transfers or TOT_SIZE transfers, the DMA controller will set the ABT bit and terminate the DMA transfer at once.

Figure 6 show the DMA hardware handshake mode protocol.

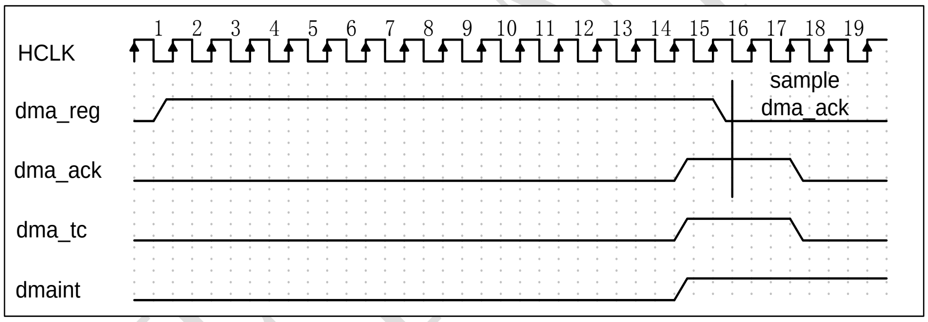


Figure 6 DMA hardware handshake mode protocol

Table 3 DMA control register list (BaseAddr = 0x4000_1800)

Name	Addr	Width	Access	Description
GLOBAL registers				
INT	+0	8	RO	Interrupt status register
INT_TC	+4	8	RO	Interrupt for terminal count status register
INT_TC_CLR	+8	8	WO	Interrupt for terminal count clear register
INT_ERR/ABT	+c	32	RO	Interrupt for Error/Abrot status register
INT_ERR/ABT_CLR	+10	32	WO	Interrupt for Error/Abrot clear register
TC	+14	8	RO	Terminal count status register
ERR/ABT	+18	32	RO	Error/Abrot status register
CH_EN	+1c	8	RO	Channel enable status register
CH_BUSY	+20	8	RO	Channel busy register status register
CSR	+24	8	RW	Main configuration status register
SYNC	+28	8	RW	Sync register
DMAC_REVITION	+30	32	RO	DMAC revition register

DMAC_REATURE	+34	32	RO	DMAC feature register
CHANNELn regsiters				
Cn_CSR	+100+20*(n-1)	32	RW	Channel O control register
Cn_CFG	+104+20*(n-1)	32	RW	Channel O configuration register
Cn_Srcaddr	+108+20*(n-1)	32	RW	Channel O source register
Cn_Dstaddr	+10c+20*(n-1)	32	RW	Channel O destination register
Cn_LLP	+110+20*(n-1)	32	RW	Channel O linked list pointer register
Cn_SIZE	+104+20*(n-1)	32	RW	Channel O transfer size register

Interrupt status register offset:0x00 default:0x0000_0000

[31:6]	reserved	
[5:0]	int_s	The result of (int_abt | int_err | int_tc),from chennal 0 to 5

INT_TC Register offset:0x04 default:0x0000_0000

[31:6]	reserved	
[5:0]	int_tc	Status of the DMA terminal count interrupts after masking,from chennal 0 to 5 0: Channel has no pending interrupt. 1: Channel has a pending interrupt.

INT_TC_CLR Register offset:0x08 default:0x0000_0000

[31:6]	reserved	
[5:0]	tc_clr	Write 1 to clear the INT_TC and TC status,from chennal 0 to 5

INT_ERR/INT_ABT Register offset:0x0C default:0x0000_0000

[31:22]	reserved	
[21:16]	int_abt	Status of the DMA abort interrupts after masking,from chennal 0 to 5 0: Channel has no pending interrupt. 1: Channel has a pending interrupt.
[15:6]	reserved	
[5:0]	int_err	Status of the DMA error interrupts after masking,from chennal 0 to 5 0: Channel has no pending interrupt. 1: Channel has a pending interrupt.

ERR_CLR/ABT_CLR Register offset:0x10 default:0x0000_0000

[31:22]	reserved	
[21:16]	abt_clr	Write 1 to clear the INT_ABT and ABT status,from chennal 0 to 5
[15:6]	reserved	

[5:0]	err_clr	Write 1 to clear the INT_ERR and ERR status,from chennal 0 to 5

TC Register offset:0x14 default:0x0000_0000

[31:6]	reserved	
[5:0]	tc	Status of the DMA terminal count,from chennal 0 to 5 0: Channel has no terminal count status. 1: Channel has a terminal count status.

ERR/ABT Register offset:0x18 default:0x0000_0000

[31:22]	reserved	
[21:16]	int_abt	Status of the DMA abort interrupts after masking,from chennal 0 to 5 0: Channel has no pending interrupt. 1: Channel has a pending interrupt.
[15:6]	reserved	
[5:0]	int_err	Status of the DMA error interrupts after masking,from chennal 0 to 5 0: Channel has no pending interrupt. 1: Channel has a pending interrupt.

CH_EN status Register offset:0x1C default:0x0000_0000

[31:8]	reserved	
[5:0]	ch_en	Status of the channel CH_EN bit of C0_CSR to C5_CSR register 0: CH_EN = 0 1: CH_EN = 1

CH_BUSY Register offset:0x20 default:0x0000_0000

[31:6]	reserved	
[5:0]	ch_busy	Status of the channel BUSY bit of C0_CFG to C5_CFG register 0: BUSY = 0 1: BUSY = 1

CSR(Configuration Status Register) offset:0x24 default:0x0000_0000

[31:3]	reserved	

[2]	m1end	AHB Master 1 endian configuration: 0 = Little-endian 1 = Big-endian
[1]	m0end	AHB Master 0 endian configuration: 0 = Little-endian 1 = Big-endian
[0]	dmacen	DMA controller enable 0 = Disable 1 = Enable

SYNC Register offset:0x28 default:0x0000_0000

[31:6]	reserved	
[5:0]	sync	DMA synchronization logic enable for channel 0~5 request: 0: Disable 1: Enable

DMAC Feature Register offset:0x34 default:0x0000_6103

[31:16]	reserved	
[15:12]	DMA_MAX_CHNO_N	DMA maximum channel number, N can be configured from 1 to 8
[11]	reserved	
[10]	DMA_HAVE_BRIDGE_value	1: DMA has built in a simple bridge.0: DMA has not built in a simple bridge.
[9]	DMA_HAVE_AHB1_value	1: DMA has AHB 0 and AHB 1.0: DMA only has AHB 0
[8]	DMA_HAVE_LINKLIST_value	1: DMA supports link list.0: DMA does not support link list
[7:4]	reserved	
[3:0]	DMA_FF_ADD_WIDTH	FIFO ram address width

Channel Control Register (Cn_CSR) default:0x0000_1200

[31]	int_tc_msk	Terminal count status mask for current transaction: 0: When terminal count happens, TC status register will be set (default). 1: When terminal count happens, TC status register will not be set.

[30:27]	reserved	
[26:24]	dma_ff_th	DMA FIFO threshold value: 000: Threshold value = 1 001: Threshold value = 2 010: Threshold value = 4 011: Threshold value = 8 100: Threshold value = 16 101~111: Threshold value = 1 When DMA FIFO space ≥ DMA_FF_TH, then DMA controller will start to transfer the data from the source to FIFO. When the number of valid data in the DMA FIFO is greater than DMA_FF_TH, then the DMA controller will start to pop out data from FIFO to the destination. Notice that, DMA_FF_TH can not be larger than 1/2 DMA FIFO size.
[23:22]	chpri	Channel priority level: 3: Highest priority 2: 2nd high priority 1: 3rd high priority 0: Lowest priority (Default)
[21]	prot3	PROT: Protection information for cacheability 0: Not cacheable (Default) 1: Cacheable
[20]	prot2	PROT: Protection information for cacheability 0: Not cacheable (Default) 1: Cacheable
[19]	prot1	PROT: Protection information for cacheability 0: Not cacheable (Default) 1: Cacheable
[18:16]	src_sz	Source burst size selection 000: Burst size = 1 (default) 001: Burst size = 4 010: Burst size = 8 011: Burst size = 16 100: Burst size = 32 101: Burst size = 64 110: Burst size = 128 111: Burst size = 256
[15]	chabt	Transaction abort Writing 1 to this bit will cause the DMA to stop the

		current transfer, then set the chabt[n] bit of Error/Abort Status Register and assert dmaint interrupt if INT_ABT_MST = 0.
[14]	Reserved	
[13:11]	swidth	Source transfer width The hardware automatically packs and unpacks the data as required. 000: Transfer width is 8 bits. 001: Transfer width is 16 bits. 010: Transfer width is 32 bits (Default). Others: Reserved
[10:8]	dwidth	Destination transfer width The hardware automatically packs and unpacks the data as required. 000: Transfer width is 8 bits. 001: Transfer width is 16 bits. 010: Transfer width is 32 bits (Default). Others: Reserved
[7]	mode	0: Normal mode (Default) 1: Hardware handshake mode
[6:5]	sad_ctl[1：0]	Source address control 00: Increment source address (Default) 01: Decrement source address 10: Fixed source address 11: Reserved
[4:3]	dad_ctl[1：0]	Destination Address Control 00: Increment destination address (Default) 01: Decrement destination address 10: Fixed destination address 11: Reserved
[2]	src_sel	0: AHB Master 0 is the source (Default) 1: AHB Master 1 is the source
[1]	dst_sel	0: AHB Master 0 is the destination (Default) 1: AHB Master 1 is the destination
[0]	ch_en	Channel Enable 0: Disable (Default) 1: Enable

Channel Configuration Register (Cn_CFG)   default:0x0000_2087

[31:20]	reserved	
[19:16]	chllp_cnt	Chain transfer counter This counter is reset to 0 when CH_EN changes from 0

		to 1.
[15:14]	reserved	
[13]	dst_he	Destination Hardware Handshake Mode enable: 0: Disable 1: Enable When you disable the destination hardware handshake, DMA will start transfer data without waiting the destination request. This bit is only valid when DMAC is in the Hardware Handshake Mode
[12:9]	dst_rs	Destination DMA request select: It specifies which dma_req as the destination req, and is used only when DMA Hardware Handshake Mode is enabled.
[8]	chbusy	1:The DMA channel is busy.
[7]	src_he	Source Hardware Handshake Mode enable: 0: Disable 1: Enable
[6:3]	src_rs	Source DMA request select: It specifies which dma_req as the source req, and is used only when DMA Hardware Handshake Mode is enabled.
[2]	int_abt_msk	Channel abort interrupt mask 0: No mask interrupt 1: Mask interrupt (Default)
[1]	int_err_msk	Channel error interrupt mask 0: No mask interrupt 1: Mask interrupt (Default)
[0]	int_tc1_msk	Channel terminal count interrupt mask 0: No mask interrupt 1: Mask interrupt (Default)

Channel Source Address Register (Cn_SrcAddr)   default:0x0000_0000

[31:0]	chsad	Source starting address Note: When the DMA transaction is done, its value changes to the DMA source ending address.

Channel Destination Address Register (Cn_DstAddr)   default:0x0000_0000

[31:0]	chdad	Destination starting address Note: When the DMA transaction is done, its value changes to the DMA destination ending address.

Linked List Descriptor Pointer (Cn_LLP)   default:0x0000_0000

[31:2]	chllp[31:2]	Linked list descriptor pointer address
[1]	reserved	
[0]	chllp[0]	Master for loading the next LLP:

		0: Load the next LLP from the AHB Master 0 (Default) 1: Load the next LLP from the AHB Master 1

Transfer Size Register (Cn_SIZE) default:0x0000_0000

[31:22]	reserved	

[21:0]	chtsz[21:0]	TOT_SIZE: Total transfer size
		The transfer unit depends on the source width. For example:
		SRC_WIDTH = 000, unit: 8-bit
		SRC_WIDTH = 001, unit: 16-bit
		SRC_WIDTH = 010, unit: 32-bit
		SRC_WIDTH = 011, unit: 64-bit

Table 4 DMA channel distribution

			Description
DMA	Channel_0	UART TX	DMA req/ack for UART
	Channel_1	RX	
	Channel_2	I2S/SPDIF ADC_RX	DMA req/ack for codec
	Channel_3	/CODEC DAC_TX	
			DMA req/ack for I2S/SPDIF
	Channel_4	IIS_RX	
	Channel_5	IIS_TX	
	Channel_6	USB TX_1	DMA req/ack for USB
	Channel_7	TX_2	
	Channel_8	TX_3	
	Channel_9	TX_4	
	Channel_10	RX_1	
	Channel_11	RX_2	
	Channel_12	RX_3	
	Channel_13	RX_4	
	Channel_14	Reserved	
	Channel_15		

Table 4 shows the DMA channel distribution, 14 channels DMA channel are used. UART_TX and UART_RX will occupy two dedicated DMA channel, ADC_RX, DAC_TX, IIS_RX and IIS_TX will occupy four dedicated DMA channel, USB will occupy eight dedicated DMA channel.

5.7 Interrupt controller

Interrupt controller module has NVIC mode to communicate with CPU. It supports 32 NVIC priority level interrupt inputs. Provides 0(max.)~7(min.) configurable priority levels for each NVIC interrupt input.

Table 5 CJC6822 NVIC interrupt input distribution

		Signal	
NVIC ID	IRQ_0	Sys_gpio0_int	Interrupt for GPIO
	IRQ_1	Sys_tm0_int	Interrupt for timer0
	IRQ_2	Sys_tm1_int	Interrupt for timer1
	IRQ_3	Sys_wdt_int	Interrupt for watchdog
	IRQ_4	Sys_uart_int	Interrupt for uart
	IRQ_5	Sys_spi_int	Interrupt for spi
	IRQ_6	Sys_iic_int	Interrupt for iic
	IRQ_7	Sys_dmac_int	Interrupt for dmac
	IRQ_8	Sys_saradc_int	Interrupt for saradc
	IRQ_9	Sys_usb_mc_nint	Interrupt for usb
	IRQ_10	Sys_usb_sof_int	Interrupt for usb frame sync pulse(disabled)
	IRQ_11	Sys_dma_nint_usb_w	Interrupt for usb_dma
	IRQ_12		
	IRQ_13		
	IRQ_14		
	IRQ_15		
	IRQ_16		
	IRQ_17		
	IRQ_18		
	IRQ_19		
	IRQ_20		
	IRQ_21		
	IRQ_22		
	IRQ_23		
	IRQ_24		
	IRQ_25		
	IRQ_26		
	IRQ_27		
	IRQ_28		
	IRQ_29		
	IRQ_30		
	IRQ_31		
	IRQ_32		

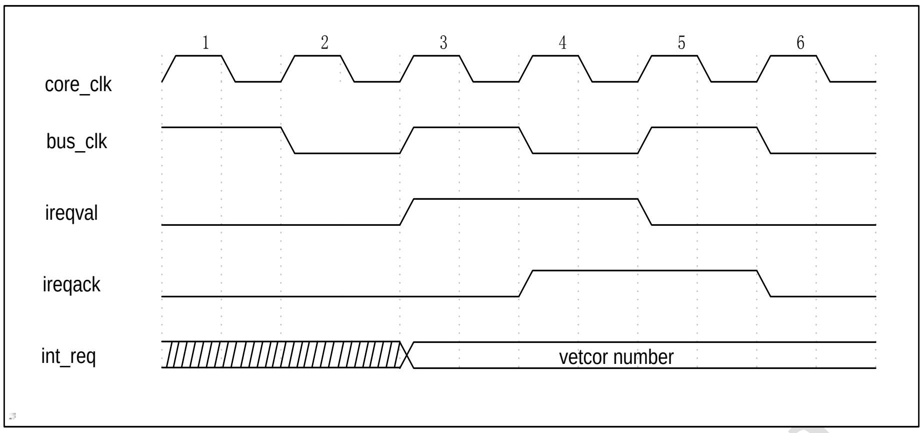


Figure 7 the interrupt REQ/ACK timing sequence diagram.

5.8 GPIO

GPIO controller is an AHB bus device communicates with CM0+ core. Each GPIO can be programmed as an input or output. It is used to input/output data from the system and device.

This GPIO can also be an interrupt input.

The GPIO provides up to 8 programmable I/O ports and each port can be independently programmed.

Table 6 summary of general purpose I/O registers (BaseAddr = 0x4000_0000)

Addr.	Name	Type	Default	Description
+0x000	GPIODATA	R/W	0x0	Reads the value of the GPIOIN pins, or sets the value driven onto GPIOUT pins.
+0x400	GPIODIR	R/W	0x0	GPIO direction register 0:Input 1:Output
+0x410	GPIOIE	R/W	0x0	GPIO interrupt enable register 0:Pin interrupt is disabled 1:Pin interrupt is enabled

When GPIOIE enable, the change of GPIO input can arose interrupt. Interrupt time continues one hclk clock.

5.9 SARADC

SARADC is accessible by CM0+ core via APB bus. This peripheral is used sampling the external sensor, voltage signal, transfer them to digital data by SARADC block, update the status register and interrupt signal, exchange data with CJC6822 processor.

The SARADC supports four external analog input signal come from sensor, mechanism key etc, the sample time is about 400Hz and sample sequence is one by one, the SARADC transfer result is store in internal register. After finishing one round, interrupt signal will generate, processor will respond this interrupt and enter into ISR.

SARADC unit is a 3.3V power supply analog module, co-work with decimation filter to implement the analog-to-digital transfer. ADC control include module timing generation, register control, interrupt generation and APB bus wrapper.Table 8 show this module register list.

Table 7 SARADC module register list (BaseAddr = 0x4001_1800).

Offset	Type	Name	Bit	Description	Default
0X00	R/W	ADC_CTRL	[31:18]	TRIG_DELAY: delay from the end of one sample to the start of the next sample. One ADC sample cycle include 0: 1 saradc_clk delay N:N+1 saradc_clk delay So, if TRIG_DELAY is set to N, then one ADC sample cycle includes (N+16) sarade_clk cycles	0x0
			[17:14]	Reserved	
			[13:8]	PCLK_DIV: SARADC clock is 4*(pclk_div+1) division of hclk. (Make sure the division of hclk is less than the highest frequency of SARADC workable clock)	
			[7:2]	Reserved	
			[1]	INTP_EN: channel interrupt enable 1: enable 0:disable	
			[0]	SARADC_EN: channel enable 1:enable 0:disable	
0X04	R/W	DAT_STATUS	[31:1]	Reserved	0x0
			[0]	ADC_READY: ADC is ready 1:ready 0: not ready Write 1 to this bit will clear the according bits. And at the same time, the interrupt and status will be cleared.	
0X08	R/W	DAT_RESULT	[31:8]	Reserved	0x0
			[7:0]	SARADC result	

5.10 CODEC/IIS/SPDIF

CJC6822 audio processor can be accessed via AHB bus or APB bus. CM0+ configures audio codec register by Ahb2apb Bridge and DAM translates data with codec by AHB bus. CODEC receives data from IIS/SPDIF interface and DMA fifo data and sends to DAC module. ADC module disposes analog data (3bit dsm data) and sends to IIS/SPDIF interface output and DMA FIFO output. Figure 8 shows the block diagram. IIS interface and SPDIF interface are selected by configure FFMT register and they don’t occur at one time.

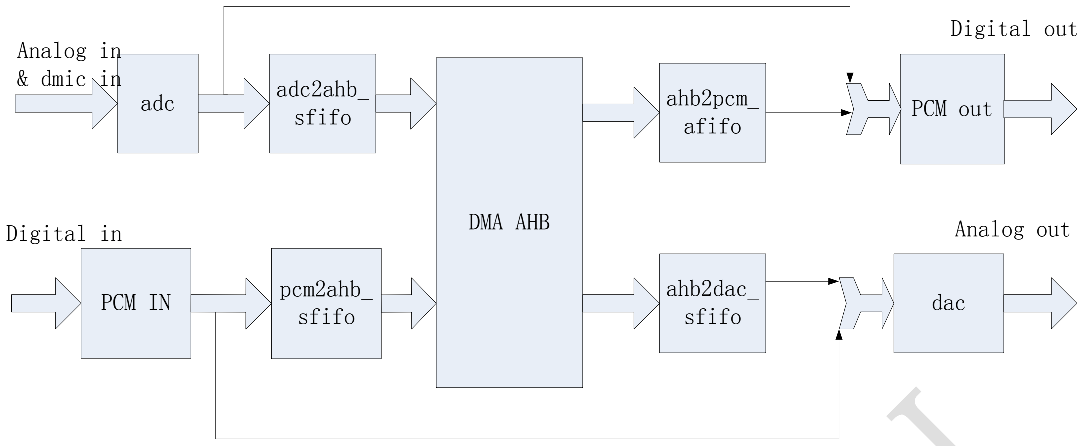


Figure 8 APU block diagram

Table 8 APU module register list (BaseAddr=0x4001_0400)

Register	Bit	Name	Description	Default Value
0x00 Input volume	[31]	dacdiv2	DAC 6dB attenuate enable 0 = disabled (0dB) 1= -6dB enabled	0x40008f57
	[30]	dacmu	Digital soft mute 1=mute 0= no mute (signal active)	
	[29:28]	deemph	De-emphasis control 11 = 48KHz sample rate 10 = 44.1KHz sample rate 01 = 32KHz sample rate 00 = No De-emphasis	
	[27]	adcdiv2	ADC 6dB attenuate enable 0 = disable(0dB) 1 = -6dB enabled	
	[26]		Reserved	
	[25]	hpor	ADC channel store dc offset when high-pass filter disabled 1 = store offset 0 = clear offset	
	[24]		***Reserved***	
	[23]		***Reserved***	
	[22]	ldcm (rdcm)	0 = adc polarity not inverted 1 = adc polarity invert	
	[21:20]	adcpol	00 = dmic polarity not inverted 01 = dmic1 polarity invert 10 = dmic2 polarity invert 11 = dmic1 & dmic2 polarity inverted	
	[19:18]		***Reserved***	

	[17]	adchpd	Adchpd_ana determine high-pass filter behavior 1’b0 = ADC hpf on 1’b1 = ADC hpf off	
	[16]		***Reserved***	
	[15:14]		***Reserved***	
	[13:12]	lmicboost	Microphone Gain Boost 00 = Boost off (bypassed) 01 = 13dB boost 10 = 20dB boost 11 = 29dB boost	
	[11]	codec_mute_pga	Input analogue mute. Default 1.	
	[10:8]	maxgain	Set Maximum Gain of PGA 111 = +12dB 110 = +6dB ….(-6dB steps) 001=-24dB 000=-30dB	
	[7]	livu	Volume update 1=Update gain 0=Store LINVOL in intermediate latch(no gain change)	
	[6]	linmute	Left channel analog input mute. 1 = Enable Mute 0 = Disable Mute Note: LIVU must be set to un-mute.	
	[5]	lzcen	Left channel zero cross detector. 1 = change gain on zero cross only 0 = change gain immediately	
	[4:0]	linvol	Left channel input volume control 11111 = +12dB 11110 = +10.5dB … 10111=0dB ..1.5dB steps down to 00000 = -34.5dB	
0x04 LOUT1 and ROUT1 volume control	[31]		***Reserved***	0x3cf93cf9
	[30]		***Reserved***	
	[29:23]		***Reserved***	
	[22:16]		***Reserved***	
	[15]	en_hpc	Line out 1 gain control enable: 0: enabled (at the same time, o1vu must enable)	

			1: disabled	
	[14]	o1vu	Line out 1 gain enable: 1: enabled 0: disabled	
	[13:7]	rout1vol	Line out 1 right channel volume gain 1111111 = +6dB …(80 steps) 0110000 = -67dB 0111111 to 0000000 = Analogue mute	
	[6:0]	lout1vol	Line out 1 left channel volume gain 1111111 = +6dB …(80 steps) 0110000 = -67dB 0111111 to 0000000 = Analogue mute	
0x08 ADC & DAC control *When this register is writed, adc/dac digital part will reset.	[31]	Pcm_mode_set	Pcm mode 1=spdif 0=IIS	0x1000a
	[30]	adc2ahb	24bit USB data input 1= IIS data from USB 0= IIS data from AHB bus or adc	
	[29]	ahb2dac	24bit USB data input 1= dac data from USB 0= dac data from AHB bus or IIS	
	[28:27]		***Reserved***	
	[26:24]	sr_adc	ADC sample rate control 3'b000 = 8K 3'b001 = 8.0182K 3'b010 = 12K 3'b110 = 11.025K 3'b011 = 16K	
	[23:19]		***Reserved***	
	[18:17]	bcm_dac	DAC BCLK frequency 00=BCM function disabled 01=MCLK/4 10=MCLK/8 11=MCLK/16	
	[16]		***Reserved***	
	[15:12]	sr_dac	DAC sample rate control MCLK = 24.576M MCLK=22.5792M 4'b0011 = 8k 4'b1011=8.0182k 4'b0100 = 12k 4'b1100=11.025k 4'b0101 = 16k 4'b1101=22.05k 4'b1110 = 24k 4'b1000=44.1k 4'b0110 = 32k 4'b1111=88.2k	

			4’b0000 = 48k 4’b1010=176.4k 4’b0111 = 96k Default = 88.2k 4’b0010=192k Default = 96k	
	[11]	ahb2pcm	0 = iis data from ahb	
	[10]	pcm2ahb	0= iis data to ahb	
	[9]	adc2pcm	IIS/SPDIF data source 1 = adc data 0 = fifo data	
	[8]	pcm2dac	DAC data source 1 = IIS/SPDIF data 0 = fifo data	
	[7]	bclkinv	BCLK invert bit(for master and slave modes) 0 = BCLK not inverted 1 = BCLK inverted	
	[6]	ms	Master/Slave mode control 1 = Enable Master mode 0 = Enable slave mode	
	[5]	lrswap	Left/Right channel swap 1 = swap left and right DAC data in audio interface 0 = output left and right data as normal	
	[4]	lrp	Right, left and i2s modes – LRCLK polarity 1 = invert LRCLK polarity 0 = normal LRCLK polarity	
	[3:2]	wl	Audio Data Word Length 11 = 32 bits 10 = 24 bits 01 = 20 bits 00 = 16 bits	
	[1:0]	format	Audio Data Format Select 11 = DSP Mode 10 = I2S Format 01 = Left justified 00 = reserved(do not use this setting)	
0x0c DAC volume	[31:20] [11:8]		***Reserved***	0xffff
	[19:12]	rdacvol	Right DAC Digital volume control similar to LDACVOL	
	[7:0]	ldacvol	Left DAC Digital Volume Control 0000 0000 = Digital mute 0000 0001 = -127dB 0000 0010 = -126.5dB … 0.5dB steps up to	

			1111 1111 = 0dB	
0x10 Additional control	[31]	vroi	VREF to analogue output resistance 0:1.5K 1:40K	0x2dc3
	[30:29]	dmonomix	DAC mono mix 00:stereo 01:mono(L+R)/2into DACL,’0’into DACR 10:mono(L+R)/2into DACR,’0’into DACL 11: mono(L+R)/2into DACL and DACR	
	[28]	dacinv	DAC phase invert 0:non-inverted 1:inverted	
	[27]	toen	Timeout enable 0:timeout disable 1:timeout enable	
	[26]	hpflren	Adchpd_dmic and hpflren_dmic together determine high-pass filter behaviour	
	[25]	reg_tri	Tristates ADCDATA and switches LRC and BCLK to inputs 0=adcdat is an output,lrc and bclk are inputs or outputs 1=adcdat is tristated, lrc and bclk are inputs	
	[24]		***Reserved***	
	[23:22]	codec_mic	Mic signal to dac mixer volume 00=-6dB 01=-9dB 10=-12dB 11=-15dB	
	[21]	dacosr	DAC oversample rate select 1=64x(lowest power) 0=128x(best SNR)	
	[20:9]		***Reserved***	
	[8]	lavu	ADC volume update 0 = store adcvol in intermediate latch(no gain change) 1 = update left and right channel gains	
	[7:0]	ladcvol	ADC digital volume control 00000000=digital mute 00000001=-97dB 00000010=-96.5dB …0.5dB steps up to 11111111=+30dB	
0x14 Power management	[31]	codec_rstn_scf	DAC module reset of SCF 1=reset 0=work	0x80000000
	[30]	codec_en_vmid	VMID enable 0=power down 1=power up	
	[29]	codec_en_ibias	VREF	

			0=power down 1=power up	
	[28]	codec_en_micb	Micbias buffer 0=power down 1=power up	
	[27]	codec_en_llinein	Left linein buffer 0=power down 1=power up	
	[26]		reserved	
	[25]	codec_en_vref	HP buffer 0=power down 1=power up	
	[24]	codec_mic2o	Mic to dac mixer enable 0=disable 1=enable	
	[23:16]		***Reserved***	
	[19]	adc_rx_fifo_clr	write 1 to clear adc rx fifo, including address and data. Write only.	
	[18]	dac_tx_fifo_clr	write 1 to clear dac tx fifo, including address and data. Write only.	
	[17]	iis_rx_fifo_clr	write 1 to clear iis rx fifo, including address and data. Write only.	
	[16]	iis_tx_fifo_clr	write 1 to clear iis tx fifo, including address and data. Write only.	
	[15]	codec_en_lpga	1:PGA enable , default 0	
	[14]	codec_en_rpga	1:PGA enable , default 0	
	[13]	codec_en_rhp	R_Channel HP enable 0=disable 1=enable	
	[12]	codec_en_lhp	L_Channel HP enable 0=disable 1=enable	
	[11]	codec_en_hp_vmid	LOUT/ROUT COMMON GROUND Enable (HPCOM) . Enables HPCOM on then LOUT/ ROUT can capless connect to headphone 0: Disable HPCOM drive 1: Enable HPCOM drive	
	[10:9]		reserved	
	[8]	digenb	Master clock disable 0=master clock enabled 1=master clock disabled	
	[7]	codec_en_radc	ana_en,codec_en_adc drive to top	
	[6]	iis_clr_reg	I2S reset 0=reset 1=work	
	[5]	adc_clr_reg	ADC reset 0=reset 1=work	
	[4]	dac_clr_reg	DAC reset 0=reset 1=work	

	[3]		***Reserved***	
	[2]	dacr_en	DAC right enable 0=power down 1=power up	
	[1]	dacl_en	DAC left enable 0=power down 1=power up	
	[0]	codec_en_ladc	ana_en,codec_en_adc drive to top	
0x18 Fifo depth control	[31]		***Reserved***	0x3333
	[30:25]	codec_iplus	0 = normal ibias current 1 = adding ibias current codec_iplus[5] HP ibias current codec_iplus[4] R-dac channel ibias current codec_iplus[3] L-dac channel ibias current codec_iplus[2] adc ibias current(else op1st) codec_iplus[1] adc op1st ibias current codec_iplus[0] pga ibias current	
	[24:16]		***Reserved***	
	[15:12]	iis_tx_trig	iis tx fifo dma request threshhold, when the number of empty units is larger or equal than this number, send dma request	
	[11:8]	iis_rx_trig	iis rx fifo dma request threshhold, when the number of occupied units is larger or equal than this (number + 1) , send dma request	
	[7:4]	dac_tx_trig	dac tx fifo dma request threshhold, when the number of empty unit is larger or equal than this number, send dma request	
	[3:0]	adc_rx_trig	adc rx fifo dma request threshhold, when the number of occupied unit is larger or equal than this (number + 1) , send dma request	
0x1c Test mode	[31:8]		***Reserved***	0x00
	[7]	bcm_gate_en	Enable BCLK clock gating	
	[6:0]		***Reserved***	
0x20	[31:0]	adc_rx_fifo	DMA read adc_rx_fifo，read only	0x00
0x24	[31:0]	dac_tx_fifo	DMA write dac_tx_fifo，write only	0x00
0x28	[31:0]	iis_rx_fifo	DMA read iis_rx_fifo，read only	0x00
0x2c	[31:0]	iis_tx_fifo	DMA write iis_tx_fifo，write only	0x00

* When read any other address from codec, read data is {24’h0, iis_rx_fifo_empty, iis_rx_fifo_full, iis_tx_fifo_empty, iis_tx_fifo_full, adc_rx_fifo_empty, adc_rx_fifo_full,

dac_tx_fifo_empty, dac_tx_fifo_full}.

5.2.1 IIS interface

The IIS interface supports ADCDAT output, DMA FIFO output, DMA FIFO input and DACDAT input. It supports several data format such as IIS, Left_justified, DSP, Right_justified, and it supports 16bit, 20bit, 24bit, 32bit word length. Figure9 to figure 12 show the timing sequence example for different format.

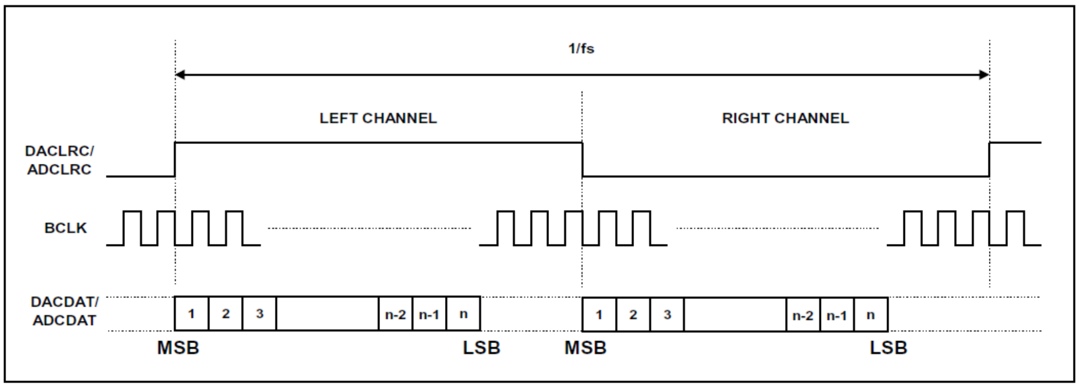


Figure 9 Left Justified Audio Interface (assuming n-bit word length)

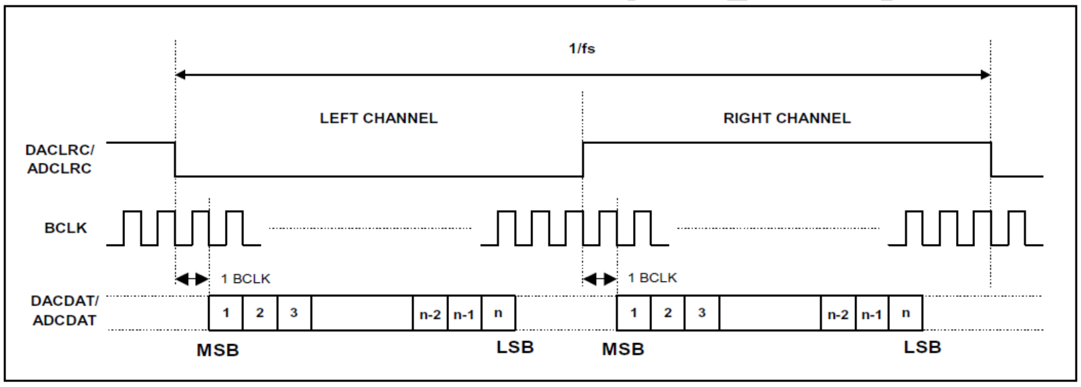


Figure 10 I2S Justified Audio Interface (assuming n-bit word length)

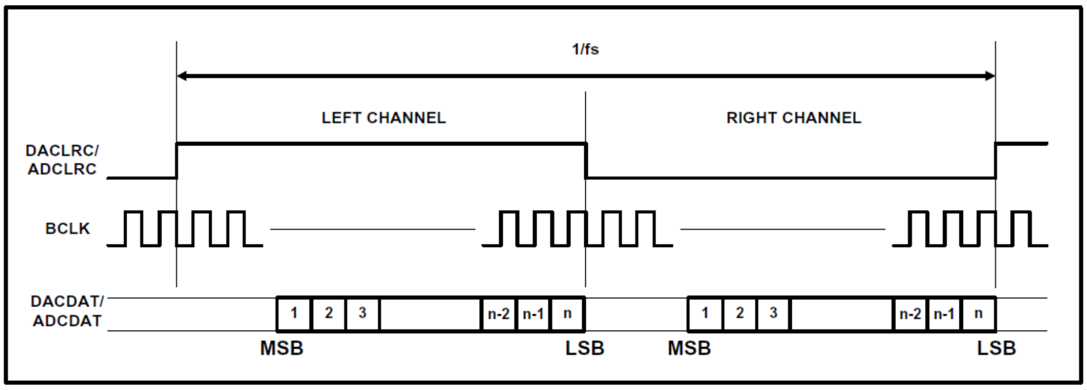


Figure 11 Right Justified Audio Interface (assuming n-bit word length)

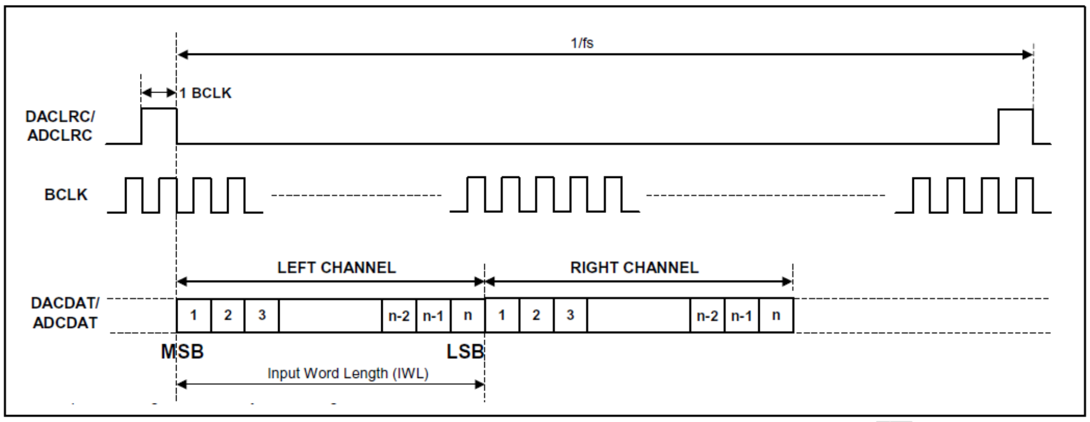


Figure 12 DSP Audio Interface (assuming n-bit word length)

5.11 SPDIF interface

The SPDIF is a point-to-point protocol for serial transmission of digital audio through a single transmission line. It provides two channels for audio data, a method for communicating control information and some error detection capabilities. The control information is transmitted as one bit per sample and accumulates in a block structure. The data is bi-phase encoded, which enables the receiver to extract a clock from the data. Coding violations, defined as preambles, are used to identify sample and block boundaries.

The SPDIF format is designed to transmit audio data. Each sample of audio data is packetized into a 32-bit sub-frame (see Figure 13) that includes additional information such as parity, validity, and user-definable bits. A frame is composed of two sub-frames, a block consists of 192 frames (see Figure 14). The first sub-frame normally starts with preamble X. However the preamble changes to preamble Z once every 192 frmaes. This defines the block structure used to organize the channel status information. The second sub-frame always starts with preamble Y. Figure 15 shows preamble X, preamble Y and preamble Z.

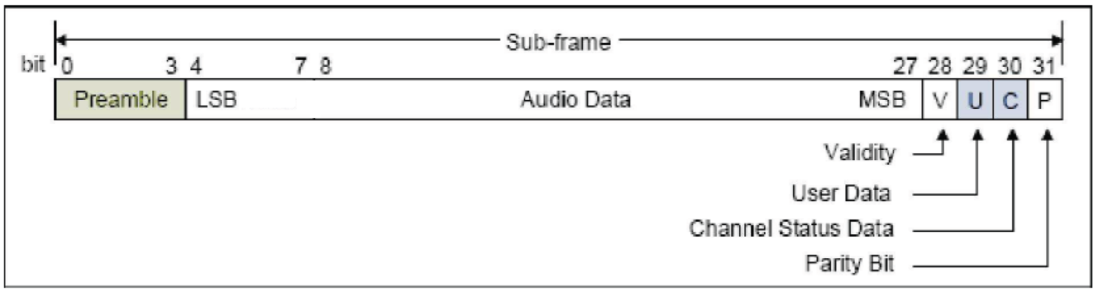


Figure 13 sub-frame structure

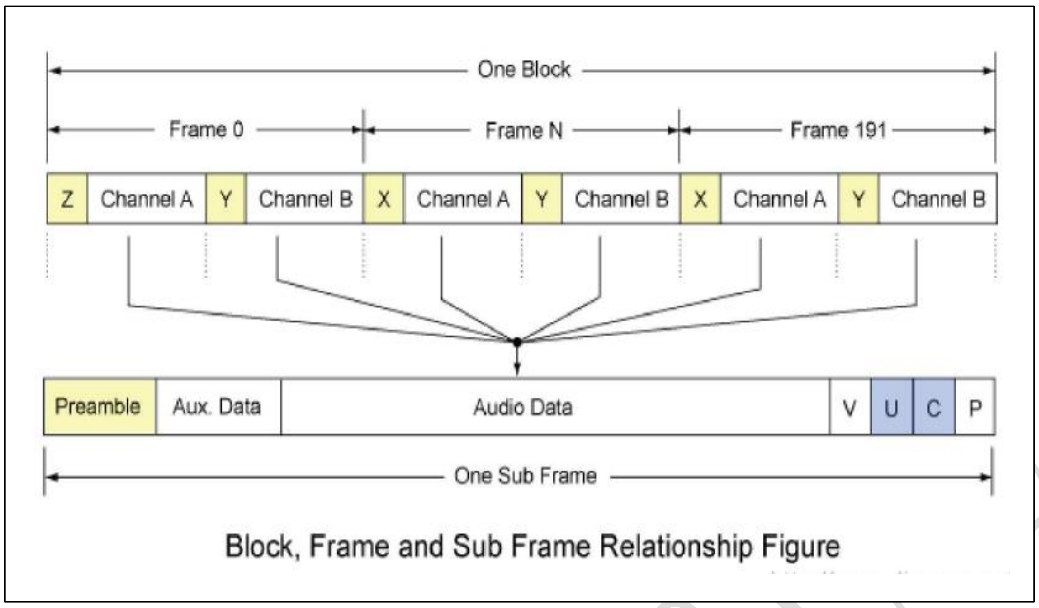


Figure 14 SPDIF block structure

	Biphase pattern Prev.'s last cell=0	Biphase pattern Prev.'s last cell=0	Channel
x	11100010	00011101	Ch.A(left)
y	11100100	00011011	Ch.B(right)
z	11101000	00010111	Ch.A(Block start)

Figure 15 Preamble configuration

The preamble X, Y, Z is not the same with the data encode. The data is bi-phase encode (see Figure 16). Coding violation, defined as preambles, are used to identify sample and block boundaries.

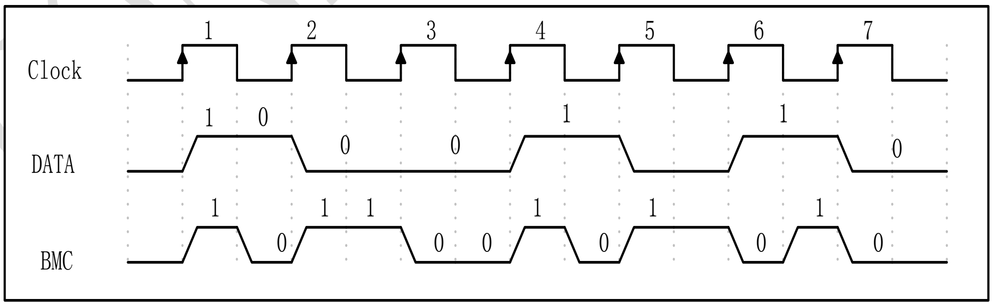
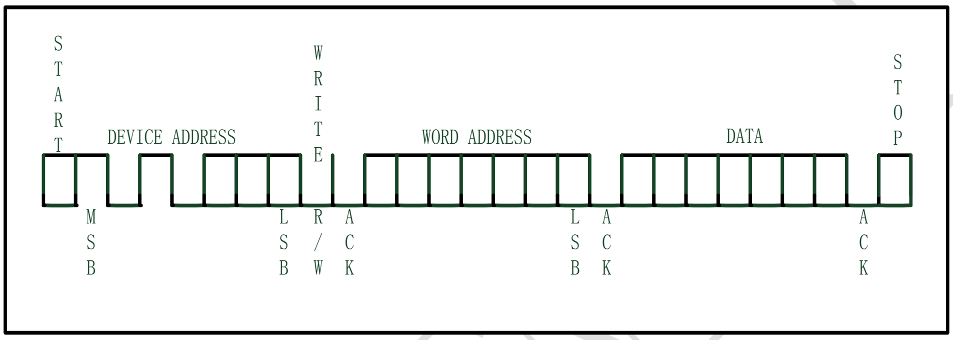


Figure 16 Bi-phase encode

5.12 IIC

IIC bus interface controller is an APB device, it allows the host processor to serve as a master or slave in the IIC bus. Data are transmitted to and received from the IIC bus via a buffered interface.

It Supports the stand and fast modes by programming the clock division register, Supports the 7-bit, 10-bit, and general-call addressing modes. It has glitch suppression capability through the debounce circuit. The salve address is Programmable, It supports the master-transmit, master-receive, slave-transmit, and slave-receive modes, and supports the multi-master mode also.



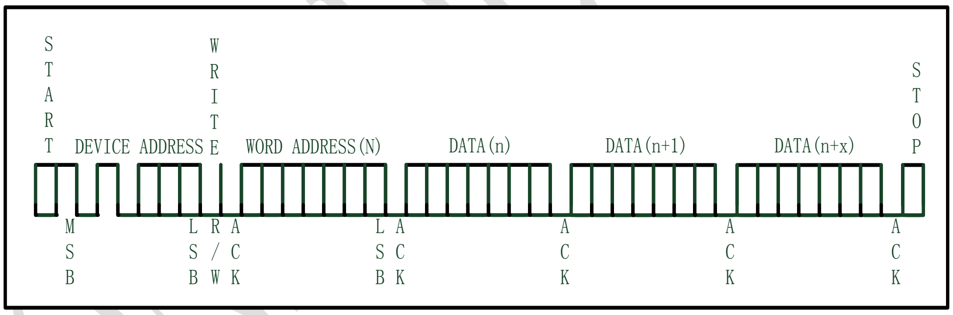
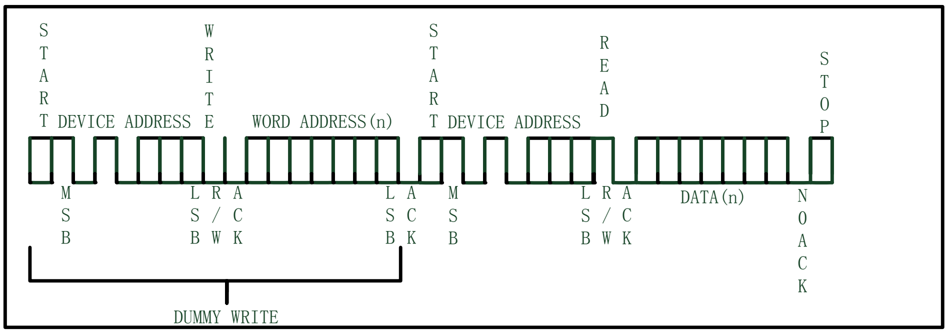


Figure 17 IIC single write and IIC burst wirte



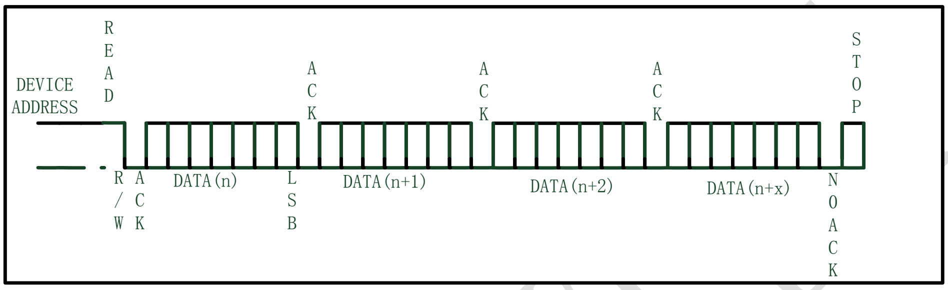


Figure 18 IIC single read and IIC burst read

Table 9 IIC controller module register list (BaseAddr=0x4001_1c00)

Oddset Address	**Type**	**Description**	**Reset Value**
0X00	R/W	IIC Control Register(CR)	0X0000_0000
0X04	R/RC	IIC Statsus Register(SR)	0X0000_0000
0X08	R/W	IIC Clock Division Register(CDR)	0X0000_0000
0X0C	R/W	IIC Data Register(DR)	0X0000_0000
0X10	R/W	IIC Slave Address Register(SAR)	0X0000_0000
0X14	R/W	IIC Setup/Hold Time and Glitch Suppression Setting Register(TGSR)	0X0000_0401
0X18	R	IIC Bus Monitor Register(BMR)	
0X30	R	IIC Revision Register	0X0000_0003

I2C Control Register(CR) offset:0x00 default:0x0000_0000

[31:18]	reserved		
[17]	Test_bit	R/W	Special test mode; it must be set to 0.
[16]	SDA_LOW	R/W	If set, the SDAout is tied to 0.
[15]	SCL_LOW	R/W	If set, the SCLout is tied to 0.
[14]	STARTI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller detects a start

			condition happening on the I2C bus.
[13]	ALI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller loses arbitration in the master mode.
[12]	SAMI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller detects a slave address that matches the SAR register or a general call address (when GC_EN is set).
[11]	STOPI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller detects a stop condition happening on the I2C bus.
[10]	BERRI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller detects non-ACK responses from the slave device after one byte of data has been sent in the master mode.
[9]	DRI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller DR register has received one data byte from the I2C bus.
[8]	DTI_EN	R/W	If set, this bit enables I2C controller to interrupt the host processor when I2C controller DR register has transmitted one data byte onto the I2C bus.
[7]	TB_EN	R/W	When Transfer Byte Enable (TB_EN) is set, I2C controller is ready to receive or transmit one byte. Otherwise, I2C controller will insert the wait state by pulling SCLout low in the I2C bus.
[6]	ACK/NACK	R/W	The acknowledge signal sent by I2C controller when I2C controller is in master-receive or slave-receive mode 0: ACK 1: NACK
[5]	STOP	R/W	I2C controller initiates a stop condition after

			transferring the next data byte on the I2C bus when I2C is in the master mode.
[4]	START	R/W	I2C controller initiates a start condition when I2C bus is idle, or initiates a repeated start condition after transferring the next data byte on the I2C bus in the master mode.
[3]	GC_EN	R/W	Enable I2C controller to respond to a general call message as a slave
[2]	SCL_EN	R/W	Enable I2C controller clock output for the master mode operation
[1]	I2C_EN	R/W	Enable the I2C bus interface controller
[0]	I2C_RST	R/W	Reset the I2C controller This bit will be automatically cleared after two PCLK clocks.

I2C Status Register (SR) offset:0x04 default:0x0000_0000

[31:12]	reserved		
[11]	START	RC	Set when I2C controller detects a start condition on the I2C bus
[10]	AL	RC	Set when I2C controller loses arbitration when operating in master mode.
[9]	GC	RC	Set when I2C controller receives a slave address that matches the general call address, when I2C controller is operating in the slave mode
[8]	SAM	RC	Set when I2C controller receives a slave address that matches the address in the slave register (SAR) when I2C controller is operating in the slave mode
[7]	STOP	RC	Set when I2C controller detects a stop condition in the I2C bus.
[6]	BERR	RC	Set when I2C controller detects non-ACK responses from the slave device after one byte of data has been transmitted when I2C controller is operating in the master mode
[5]	DR	RC	Set when the data register (DR) received one new data byte from the I2C bus
[4]	DT	RC	Set when the data register (DR) transmitted one data byte to the I2C bus
[3]	BB	R	Set when the I2C bus is busy, but the I2C controller is

			not involved in the transaction
[2]	I2CB	R	Set when the I2C controller is busy, i.e. during the time period between the START and STOP
[1]	ACK	R	Set when the I2C controller receives or sends non-acknowledgements
[0]	RW	R	Set when the I2C controller serves in a master-receive or slave-transmit mode.

* RC means read and clear

I2C Clock Division Register (CDR) offset:0x08 default:0x0000_0000

[31:18]	reserved		
[17:0]	COUNT	R/W	Counter value used to generate an I2C clock (SCLout) from the internal bus clock PCLK. The relation between PCLK and I2C bus clock (SCLout) is shown in the following equation, where GSR is TGSR[12:10] SCLout = PCLK/(2*(COUNT + 2) + GSR)

I2C Data Register (DR) offset:0x0C default:0x0000_0000

[31:8]	reserved		
[7:0]	DR	R/W	Buffer for I2C bus data transmission and receptio n, I2C data write or read window.

I2C Slave Address Register (SAR) offset:0x10 default:0x0000_0000

[31]	EN10	R/W	10-bit addressing mode enable bit
[30:10]	reserved		
[9:7]	SAR	R/W	The most significant 3-bit address to which the I2C

			controller responds when I2C operates in 10-bit addressing slave mode (EN10 =1). When EN10 = 0, the I2C controller ignores these three bits.
[6:0]	SAR	R/W	The 7-bit address to which the I2C controller responds when the I2C operates in the 7-bit addressing slave mode (EN10 =0) or the least significant 7-bit address to which the I2C controller responds when the I2C operates in the 10-bit addressing slave mode.

I2C Set/Hold Time & Glitch Suppression Setting Register (TGSR) offset:0x14 default:0x0000_0401

[31:13]	reserved		
[12:10]	GSR	R/W	These bits define the values of PCLK clock period when the I2C Bus Interface has built-in glitch suppression logic. Glitch is suppressed according to “GSR * PCLK” clock period.
[9:0]	TSR	R/W	These bits define the delay values of PCLK clock cycles that the data or acknowledgement will be driven into the I2C SDA bus after I2C SCL bus goes low. The actual delay value is GSR+TSR+4. Figure- 3 shows the relationship. Note: TSR cannot be set to zero.

I2C Bus Monitor Register (BMR) offset:0x18

[31:2]	reserved		
[1]	SCLin	R	This bit continuously reflects the value of the SCLin pin.
[0]	SDAin	R	This bit continuously reflects the value of the SDAin pin.

5.13 UART

CJC6822 UART can be accessed via AHB bus or APB bus. CM0+ configures uart register and transfers data with uart by Ahb2apb Bridge , DAM translates data with uart by AHB bus.

The system assigns two dedicated DMA channel to the UART_TX and UART_RX data transfer. UART have a programmable interrupt to the system.
UART controller is a serial communication element that implements the most common infrared communication protocols. It also support IRDA1.3 SIR protocol which is used in household electrical device IR transmitter and receiver (38KHz).
UART support two work mode: UART mode , SIR mode.
The UART mode is default enabled after power up or system reset. This mode uses a wired interface for serial communication with a remote device or a modem. It can operate in a full-duplex mode, data transmission and reception can take place simultaneously. It works as a regular serial asynchronous communication controller that converts the parallel data received from the CPU or the DMA controller into serial data. It also converts the serial data received on the serial input terminal into parallel data. The format of the serial data stream is shown in figure 19. A data character contains 5 to 8 data bits. It is preceded by a start bit and is followed by an optional parity bit and a stop bit. Data is transferred in little-endian order (Least significant bit first). The clock for both transmit and receive channels is provided by an internal baud generator that divides the pre-scaled clock by any divisor value from 1 to 216 - 1. The output clock frequency of the baud generator must be programmed to be sixteen times the baud rate value. The baud generator input clock is derived from io_irda_uclk clock through a programmable prescaler. Both the communications format and baud rate must be programmed properly before operation.

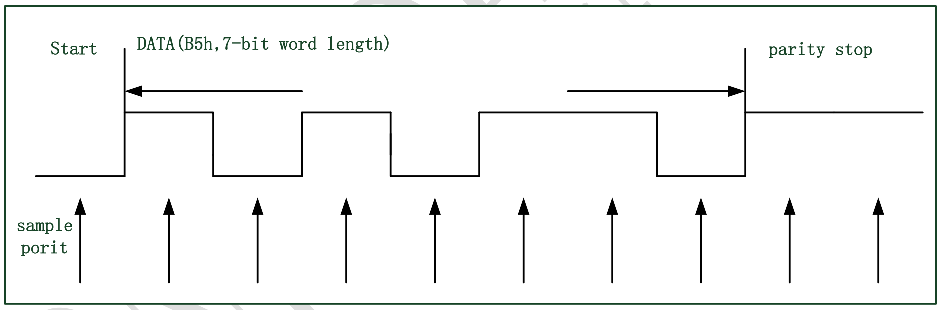


Figure 19 UART data representation and sampling

SIR (serial IR) mode supports bi-directional data communication with a remote device using the infrared radiation as the transmission medium. IrDA 1.3 SIR allows serial communication at baud rates of up to 115.2 kbps. The format of the serial data is similar to the UART data format. Each data word is sent serially beginning with a zero value start bit, followed by 8 data bits, and ending with one stop bit with a binary value of one. Sending a single infrared pulse signals a zero. A one is signaled by not sending any pulse. The width of each pulse can be either 1.6 μs or 3/16 of a single bit time. (1.6 μs equals 3/16 of a bit time at 115.2 kbps). This way, each word begins with a pulse for the start bit. The device operation in the IrDA SIR mode is similar to the operation in UART mode. The main differences are that, those data transfer operations are normally performed in half-duplex fashion.
Each data byte starts with a start bit (0), 1 byte of data, and then ends with at least a stop bit (1). Each serial data bit is encoded before transmission and decoded after reception. A 1 is decoded with no IR pulse and a 0 is decoded by sending 3/16ths of one bit time IR

pulse. Similarly, the received serial pulse is decoded as a 0 and the absence of an IR pulse is decoded as a 1, Please refer to Figure 20.

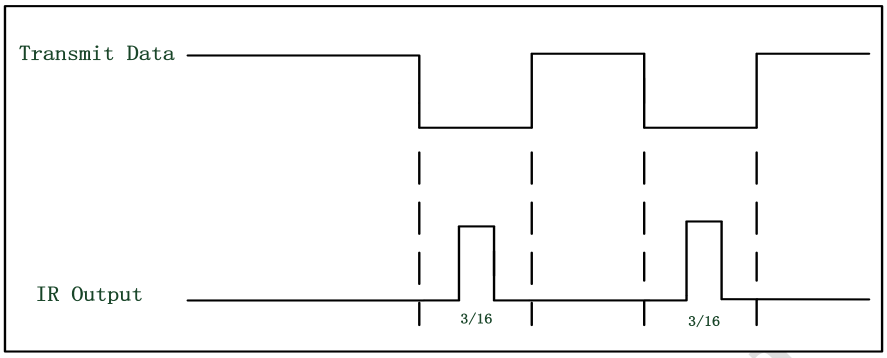


Figure 20 SIR encoding

Table 10 Uart module register list (BaseAddr=0x4001_0c00)

Offset	Type	Width	Name	Description	Reset Value
UART/Infrared SIR Mode					
+0X00	R	8	RBR	Receiver buffer register	0x00
	W	8	THR	Transmitter holding register	0x00
+0X04	R/W	4	IER	Interrupt enable register	0x00
+0X08	R	8	IIR	Interrupt identification register	0x01
	W		FCR	FIFO control register	0x00
+0X0C	R/W	8	LCR	Line control register	0x00
+0X10	R/W	7	MCR	Modem control register	0x00
+0X14	R	8	LSR	Line status register	0x00
	W		TST	Testing register	0x60
+0X18	R	8	MSR	Modem stutas register	0x00
+0X1C	R/W	8	SPR	Scratch pad register	0x00
Registers accessible whRen DLAB = 1					
+0X00	R/W	8	DLL	Baud rate divisor latch least significant byte	0x01
+0X04	R/W	8	DLM	Baud rate divisor latch most significant byte	0x00
+0X08	R/W	5	PSR	Prescaler register	0x01

Receiver Register offset:0x00 default:0x00

[7:0]	RBR	R	Receive Data Port

Transmitter Holding Register offset:0x00 default:0x00

[7:0]	THR	W	Transmit Data Port

Baud Rate Divisor Latch LSB offset:0x00 default:0x01

[7:0]	DLL	R/W	Baud Rate Divisor Latch Least Significant Byte

* Accessible when DLAB = 1

Baud Rate Divisor Latch MSB offset:0x04 default:0x00

[7:0]	DLM	R/W	Baud Rate Divisor Latch Most Significant Byte

* Accessible when DLAB =1 offset:0x08 default:0x01

[7:4]	reserved		
[3]	MODEM Status	R/W	This bit enables the modem status interrupt when set to logic 1.
[2]	Receiver Line Status	R/W	This bit enables the Receiver Line Status Interrupt when set to logic 1.
[1]	THR Empty	R/W	This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.
[0]	Receiver Data Available	R/W	This bit enables the Received Data Available Interrupt (and character reception timeout interrupts in the FIFO mode) when set to logic 1.

Interrupt Identification Register

[7:6]	FIFO mode enable	R	These two bits are set when FCR[0] is set as 1.
[5]	reserved		
[4]	Tx FIFO full	R	This bit is set as 1 when TX FIFO is full.
[3]	FIFO mode only	R	In the 16450 mode, this bit is 0. In the FIFO mode, this bit is set along with bit 2 when a timeout interrupt is pending.
[2:1]	Interrupt Identification Code	R	These bits identify the highest priority interrupt that is pending.
[0]	Interrupt Pending	R	This bit can be used in a prioritized interrupt environment to indicate whether an interrupt is pending. 0: An interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. 1: No interrupt is pending.

FIFO Control Register offset:0x08 default:0x00

[7:6]	RXFIFO_TRGL	W	Used to set the trigger level of the RX FIFO interrupt.
[5:4]	TXFIFO_TRGL	W	Used to set the trigger level of the TX FIFO interrupt.
[3]	DMA Mode	W	This bit selects the UART DMA mode. The DMA mode affects the way in inwhich the DMA signaling outputs pins (irda_nrxrdy and irda_ntxrdy) behave.
[2]	TX FIFO Reset	W	Setting this bit to logic 1 clears all bytes in the TX FIFO and resets its counter logic to 0. The shift register is not cleared, so any reception active will continue. This bit will automatically return to zero.
[1]	RX FIFO Reset	W	Setting this bit to logic 1 clears all bytes in the Rx FIFO and resets its counter logic to 0. The shift register is not cleared, so any reception active will continue. Setting this bit also clears the Status FIFO. This bit will automatically return to zero.
[0]	FIFO Enable	W	Set this bit to logic 1 enables both the transmit and receive FIFOs (And Status FIFO). Changing this bit automatically resets both FIFOs. In a FIR mode, the device driver should always set this bit as 1.

Prescaler Register offset:0x08 default:0x01

[7:5]	reserved		
[4:0]	PSR	R/W	Prescaler Value

* Accessible when DLAB =1

Line Control Register offset:0x0C default:00

[7]	DLAB	R/W	Divisor Latch Access Bit (DLAB) This bit must be set in order to access the DLL, DLM and PSR registers which program the division constants for the baud rate divider and the prescaler.
[6]	Set Break	R/W	This bit causes a break condition to be transmitted to the receiving UART. When it is set to logic 1, the serial output (io_irda_sout) is forced to the Spacing (Logic 0) state. The break is disabled by setting bit 6 to 0. The Break Control bit acts only on io_irda_sout and has no effect on the transmitter logic, so if several characters are stored in the transmit FIFO, they will be removed from this FIFO and passed sequentially to the Transmitter Shift Register which serializes them, even if Set Break is set. This fact can be useful to establish the break time making use of the THR Empty and Transmitter Empty flags of the LSR. Firmware can follow the sequence below to assure no erroneous or extraneous characters will be transmitted because of the break: Set break when transmitter is idle (LSR bit 6). Write a character with any value to THR. Wait for the transmitter to become idle (LSR bit 6), and clear break when normal transmission has to be restored.
[5]	Stick Parity	R/W	When bits 3, 4 and 5 are logic 1, the Parity bit is transmitted and checked as 0. If bits 3 and 5 are 1 and bit 4 is 0, then the Parity bit is transmitted and checked as 1. If bit 5 is 0, Stick Parity is disabled.
[4]	Even Parity	R/W	This bit is the Even Parity Select bit. When bit 3 is 1 and bit 4 is 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit.
[3]	Parity Enable	R/W	This bit is the Parity Enable bit. When this bit is a 1, a Parity bit is generated
			(Transmit data) or checked (Receive data) between the last data word
			bit and

			Stop bit of the serial data. When bit 3 is 1 and bit 4 is a 1, an even number of 1s is transmitted or checked.
[2]	Stop Bits	R/W	This bit selects the number of stop bits to be transmitted. If cleared, only one stop bit will be transmitted. If set, two stop bits (1.5 with 5-bit data) will be transmitted before the start bit of the next character. The receiver always checks only one stop bit.
[1]	WL1	R/W	This bit along with WL0 defines the word length of the data being transmitted and received.
[0]	WL0	R/W	This bit along with WL1 defines the word length of the data being transmitted and received.

Modem Control Register offset:0x10 default:0x00

[7]	reserved		
[6]	Out3	R/W	This bit controls the general purpose output io_irda_nout3 （General-purpose output 3，active low）. A 1 in this bit makes io_irda_nout3 output a 0. When this bit is cleared, io_irda_nout3 outputs a 1.
[5]	DMAmode2	R/W	This bit selects the UART/SIR DMA mode.The DMA mode2 affects the way in which the DMA signaling output pins (irda_nrxrdy and irda_ntxrdy) behave. irda_nrxrdy ：The UART/SIR mode has new received data to be transferred to the memory, low active Mode 1: In the FIFO mode (FCR[0] = 1) when DMAmode2 = 0, and FCR[3] = 1 and the trigger level or the timeout has been reached, the irda_nrxrdy pin will go low active. Once it is activated, the irda_nrxrdy will go inactive (High) when there are no more characters in the FIFO or holding register. Mode 2: In the FIFO mode (FCR[0] = 1) when DMAmode2 = 1 (FCR[3] is “don’t care”), and there is at least 1 character in the RX FIFO or receive the holding register, the irda_nrxrdy will be low active. This signal will go inactive (High) when irda_rx_ack is sampled high. irda_ntxrdy ：The UART/SIR mode is ready to receive the characters from the memory to be sent, low active. Mode 1: In the FIFO mode (FCR[0] = 1) when DMAmode2 = 0 and FCR[3] = 1 and there are no characters in the TX FIFO, the irda_ntxrdy pin will go low active. This pin will become inactive when the TX FIFO is completely full.

			Mode 2: In the FIFO mode (FCR [0] = 1) when DMAmode2 = 1 (FCR[3] is "don't care") and the number of characters in the TX FIFO is smaller than the TX FIFO trigger level, the irda_ntxrdy pin will go active (Low). This signal will go inactive (High) when irda_tx_ack is sampled high. irda_rx_ack：This signal is used to de-assert irda_nrxrdy in the DMA mode 2 irda_tx_ack：This signal is used to de-assert irda_ntxrdy in the DMA mode 2.
[4]	Loop	R/W	Loop back mode control bit. Loop back mode is intended to test the UART or SIR communication.
[3]	Out2	R/W	This bit controls the general purpose output io_irda_nout2（General-purpose output 2，active low）. A 1 in this bit makes io_irda_nout2 output a 0. When this bit is cleared, io_irda_nout2 outputs a 1.
[2]	Out1	R/W	This bit controls the general purpose output io_irda_nout1（General-purpose output 1，active low）. A 1 in this bit makes io_irda_nout1 output a 0. When this bit is cleared, io_irda_nout1 outputs a 1.
[1]	RTS (Request to Send)	R/W	This bit controls the "request to send" output （io_irda_nrts，active low）. A 1 in this bit makes io_irda_nrts output a 0. When this bit is cleared, io_irda_nrts outputs a 1. io_irda_nrts ：Request To Send This signal is controlled by a register's bit. When low, this signal informs the modem or data set that the UART is ready to exchange data. This output signal can be set to an active low by programming bit 1 of the Modem Control Register. A system reset operation sets this signal to be inactive (High). The loop mode operation holds this signal in its inactive state.
[0]	DTR (Data Terminal Ready)	R/W	This bit controls the "data terminal ready" active low output io_irda_ndtr. A 1 in this bit makes io_irda_ndtr output a 0. When this bit is cleared, io_irda_ndtr outputs a 1. io_irda_ndtr：Data Terminal Ready
			This signal is controlled by a register's bit. When low, this signal informs the modem or data set that the UART is ready to establish a communication link. This output signal can be set to an active low by programming bit 0 of the Modem Control Register to a high level. A system reset operation sets this signal to its inactive (High) state. The loop mode operation holds this signal in its inactive state.

Line Status Register offset:0x14 default:0x60

[7]	FIFO Data Error	R	If the FIFO is disabled (16450 mode), this bit is always zero. If the FIFO is active, this bit will be set as soon as any data character in the receiver’s FIFO has parity or framing error or the break indication active. This bit is cleared when the CPU reads the LSR and the rest of the data in the receiver’s FIFO do not have any of these three associated flags on.
[6]	Transmitter Empty	R	This bit is 1 when both the THR (Or TX FIFO) and the TSR (Transmitter Shift Register) are empty. Reading this bit as 1 means that no transmission is currently taking place in the io_irda_sout output pin, and that the transmission line is idle. As soon as new data is written in the THR, this bit will be cleared.
[5]	THR Empty	R	This bit indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable bit (IER [1]) is set high.
[4]	Break Interrupt	R	This bit is set to 1 if the receiver’s line input io_irda_sin was held at zero for a complete character time. That is to say, the positions corresponding to the start bit, the data, the parity bit (if any) and the (first) stop bit were all detected as zeroes.
[3]	Framing Error	R	This bit indicates that the received character did not have a valid stop bit (i.e., a 0 was detected in the (first) stop bit position instead of a 1). This bit is queued in the receiver’s FIFO in the same way as the Parity Error bit.
			When a framing error is detected, the receiver tries to resynchronize: if the next sample is again a zero, it will be taken as the beginning of a possible new start bit. This bit is cleared as soon as the LSR is read.
[2]	Parity Error	R	When this bit is set, it indicates that the parity of the received character is wrong
[1]	Overrun Error	R	When this bit is set, a character has been completely assembled in the Receiver Shift Register without having free space to put it in the receiver’s FIFO or holding register.
[0]	Data Ready	R	This bit is set if one or more characters have been received and are waiting in the receiver’s FIFO for the user to read them. It is cleared to a logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO.

Testing Register offset:0x14 default:0x00

[7:5]	reserved		
[4]	TEST_CRC_ERR	W	When set, UART generates incorrect CRC during FIR transmission.
[3]	TEST_PHY_ERR	W	When set, UART generates incorrect 4PPM encoding chips during FIR transmission.
[2]	TEST_BAUDGEN	W	This bit is used to improve baud rate generator toggle rate.
[1]	TEST_FRM_ERR	W	When set, UART generates a logic 0 STOP bit during UART transmission.
[0]	TEST_PAR_ERR	W	When set, UART generates incorrect parity during UART transmission.

Modem Status Register offset:0x18 default:0x00

[7]	DCD	R	Data Carrier Detect (DCD), which is the complement of the io_irda_ndcd input. io_irda_ndcd：Data Carrier Detect This signal is used to provide the flags and an interrupt. When low, this signal indicates that the data carrier has been detected by the modem or data set. This signal is a modem status input whose conditions can be tested by the CPU reading bit 7 of the Modem Status Register. Bit 7 is the complement of the io_irda_ndcd signal. Bit 3 of the Modem Status Register indicates whether the io_irda_ndcd input has changed the state since the previous reading of the Modem Status Register. io_irda_ndcd has no effect on the receiver.
[6]	RI	R	Ring Indicator (RI), which is the complement of the io_irda_nri input. io_irda_nri：Ring Indicator This signal is used to provide the flags and an interrupt. When low, this signal indicates that a telephone ringing signal has been received by the modem or dataset. This signal is a modem status input whose conditions can be tested by the CPU reading bit 6 of the Modem Status Register. Bit 6 is the complement of the io_irda_nri signal. Bit 2 of the Modem Status Register indicates whether the io_irda_nri input signal has changed from a low to a high state since the previous reading of the Modem Status Register.
[5]	DSR	R	Data Set Ready (DSR), which is the complement of the io_irda_ndsr input. io_irda_ndsr：Data Set Ready This signal is used to provide the flags and an interrupt. When low, this signal indicates that a modem or data set is ready to establish the communication link with the UART. This signal is a modem status input whose condition can be tested by the CPU reading bit 5 of the Modem Status Register. Bit 5 is the complement of the io_irda_ndsr signal. Bit 1 of the Modem Status Register indicates whether the io_irda_ndsr input has changed the state since the previous reading of the Modem Status Register.
[4]	CTS	R	Clear To Send (CTS), which is the complement of the io_irda_ncts input.

			io_irda_ncts：Clear To Send This signal is used to provide the flags and an interrupt. When low, this signal indicates that a modem or data set is ready to exchange data. The io_irda_ncts signal is a modem status input whose conditions can be tested by the CPU reading bit 4 of the Modem Status Register. Bit 4 is the complement of the io_irda_ncts signal. Bit 0 of the Modem Status Register indicates whether the io_irda_ncts input has changed the state since the previous reading of the Modem Status Register. io_irda_ncts has no effect on the transmitter.
[3]	Delta DCD	R	The delta-DCD flag. If set, it means that the io_irda_ndcd input has changed since the last time the microprocessor read this bit. io_irda_ndcd input. io_irda_ndcd：Data Carrier Detect This signal is used to provide the flags and an interrupt. When low, this signal indicates that the data carrier has been detected by the modem or data set. This signal is a modem status input whose conditions can be tested by the CPU reading bit 7 of the Modem Status Register. Bit 7 is the complement of the io_irda_ndcd signal. Bit 3 of the Modem Status Register indicates whether the io_irda_ndcd input has changed the state since the previous reading of the Modem Status Register. io_irda_ndcd has no effect on the receiver.
[2]	Trailing edge R1	R	This bit is set when a trailing edge is detected in the io_irda_nri input pin; that is to say, when io_irda_nri changes from 0 to 1. io_irda_nri：Ring Indicator This signal is used to provide the flags and an interrupt. When low, this signal indicates that a telephone ringing signal has been received by the modem or dataset. This signal is a modem status input whose conditions can be tested by the CPU reading bit 6 of the Modem Status Register. Bit 6 is the complement of the io_irda_nri signal. Bit 2 of the Modem Status Register indicates whether the io_irda_nri input signal has changed from a low to a high state since the previous reading of the Modem Status Register.
[1]	Delta DSR	R	If set, it means that the io_irda_ndsr input has changed since the last time the microprocessor read this bit. io_irda_ndsr：Data Set Ready This signal is used to provide the flags and an interrupt. When low, this signal indicates that a modem or data set is ready to establish the communication link with the UART. This signal is a modem status input whose condition can be tested by the CPU reading bit 5 of the Modem Status Register. Bit 5 is the complement of the io_irda_ndsr signal. Bit 1 of the Modem Status Register indicates whether the io_irda_ndsr input has changed the state since the previous reading of the Modem Status Register.
[0]	Delta CTS	R	If set, it means that the io_irda_ncts input has changed since the last

			time the microprocessor read this bit. io_irda_ncts：Clear To Send This signal is used to provide the flags and an interrupt. When low, this signal indicates that a modem or data set is ready to exchange data. The io_irda_ncts signal is a modem status input whose conditions can be tested by the CPU reading bit 4 of the Modem Status Register. Bit 4 is the complement of the io_irda_ncts signal. Bit 0 of the Modem Status Register indicates whether the io_irda_ncts input has changed the state since the previous reading of the Modem Status Register. io_irda_ncts has no effect on the transmitter.

Scratch Pad Register offset:0x1C default:0x00

[7:]	User Data	R/W	This 8-bit read/write register has no effect on the operation of the Serial Port. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

Prescaler dot Register offset:0x20 default:0x00

[7:5]	reserved		
[4:0]	PSRDOT	R/W	Prescaler dot Value

Feature Register offset:0x68 default:0x00

[7:5]	reserved		
[4]	IrDA_INSIDE	R	1: uart controller contains IrDA function 0: uart controller is a pure UART
[3:0]	FIFO_DEPTH	R	4’b0001: TX/RX FIFOs are 16-byte deep 4’b0010: TX/RX FIFOs are 32-byte deep 4’b0100: TX/RX FIFOs are 64-byte deep 4’b1000: TX/RX FIFOs are 128-byte deep

5.14 PWM

CJC6822 integrates one channel PWM as APB device. The PWM output signals are based on the pwm_clk pin and must be a minimum of 2 clock cycles wide. Various configurations can be programmed to adjust the period and the waveform of the output signals. Figure 21 shows the block diagram of PWM.

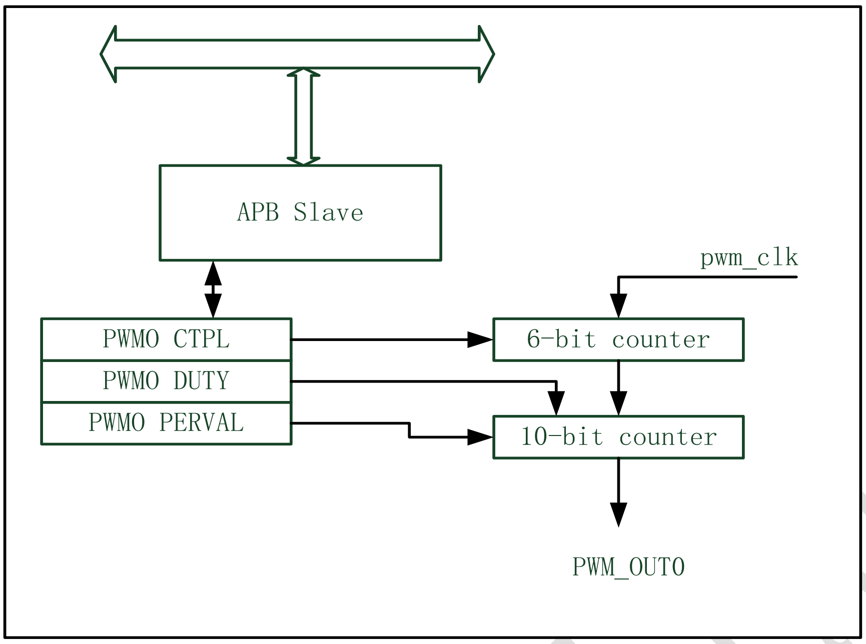


Figure 21 Block diagram of PWM

Table 11 PWM module register list (Baseaddr = 0x4001_1400)

Addr	Type	Name	Bit	Description	Default
0x00	R/W	CTRL	[31:6]	Reserved	0x0
			[5:0]	PRESCALE Determines the frequency of the PWM module clock PSCLK_PWM = pwm_clk/(CTRL+1)	
0x04	R/W	DUTY	[31:11]	Reserved	0x0
			[10]	FDCYCLE PWM full duty cycle 0=PWM_OUT0 duty cycle is determined by DCYCLE field 1=PWM_OUT0 is set high and does not toggle	
			[9:0]	DCYCLE PWM duty cycle Duty cycle of PWM_OUT.	
0x08	R/W	PERVAL	[31:10]	Reserved	0x0
			[9:0]	PERVAL PWM period control The number of PSCLK_PWM cycle that comprise one PWM_OUT cycle.	

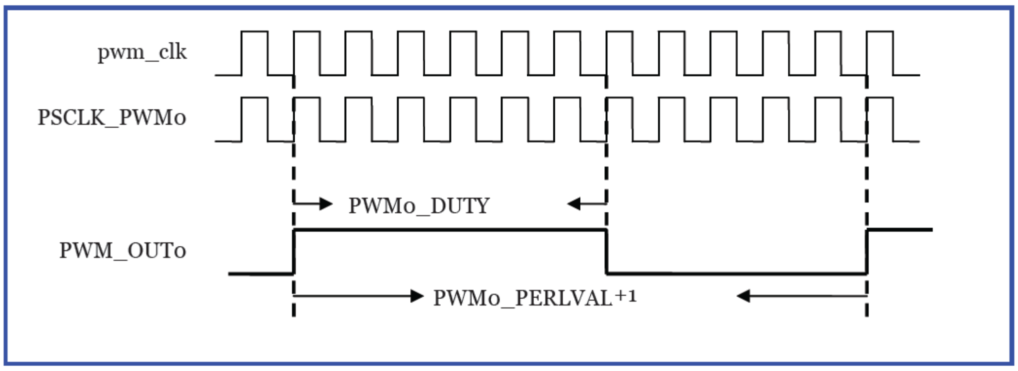


Figure 22 timing diagram of PWM

5.15 TIMER

Timer module is an APB device, it provides three independent sets of 32-bit sub-timers, and the first sub-timer is the default sub-timer. Each sub-timer can use either internal system clock (PCLK) or external clock (EXTCLK) to increase or decrease the counting. Two match registers are provided for each sub-timer. Whenever the value of the match registers equals to any one of the sub-timers, the timer interrupt is triggered immediately. The issuance of the timer interrupt can be decided by the register setting when an overflow occurs. CJC6822 assigns 3 interrupt for timer.

Table 12 Timer register list (BaseAddr = 0x4001_3000)

Offset	Type	Width	Name	Description	Reset
0x00	R/W	32	Tm1control	Timer1 control	0x--

0x04	R/W	32	Tm1load	Timer1 auto reload value	0x--
0x08	R/W	32	Tm1match1	Timer1 match value	0x--
0x0c	R/W	32	Tm1match2	Timer1 match value	0x--
0x10	R/W	32	Tm2control	Timer2 control	0x--
0x14	R/W	32	Tm2load	Timer2 auto reload value	0x--
0x18	R/W	32	Tm2match1	Timer2 match value	0x--
0x1c	R/W	32	Tm2match2	Timer2 match value	0x--
0x20	R/W	32	Tm3control	Timer3 control	0x--
0x24	R/W	32	Tm3load	Timer3 auto reload value	0x--
0x28	R/W	32	Tm3match1	Timer3 match value	0x--
0x2c	R/W	32	Tm3match2	Timer3 match value	0x--
0x30	R/W	12	Tmcr	Timer1,Timer2,Timer3 control register	0x0
0x34	R/W	9	Intrstate	Interrupt State of timer	0x0
0x38	R/W	9	Intrmask	Interrupt Mask of timer	0x0
0x3c	R	32	Tmrevision	Timer revision number	0x--

Tm1Counter, Tm2Counter,
Tm3Counter offset:0x00/0x10/0x20 default:0x0000_0000

[31:0]	TMCounter	R/W	the counter registers of Timer1, Timer2, and Timer3

Tm1Load, Tm2Load, Tm3Load offset:0x04/0x14/0x24 default:0x0000_0000

[31:0]	Tmload	R/W	Tm1Load, Tm2Load, and Tm3Load are the auto-reload registers for Timer1, Timer2, and Timer3, respectively.

Tm1Match1, Tm2Match1,
Tm3Match1 offset:0x08/0x18/0x28 default:0x0000_0000

[31:0]	TmMatch1	R/W	Tm1Match1, Tm2Match1, and Tm3Match1 are the match registers of Timer1, Timer2, and Timer3, respectively. When the values of counter(1 ~ 3) equal the value of Tm(1 ~ 3)Match1 and the Tm(1 ~ 3)Enable bit is set, then the tm(1 ~ 3)_intr will be triggered.

Tm1Match2, Tm2Match2,
Tm3Match2 offset:0x0C/0x1C/0c2C default:0x0000_0000

[31:0]	TmMatch2	R/W	Tm1Match2, Tm2Match2, and Tm3Match2 are the match registers of Timer1, Timer2, and Timer3, respectively. When the values of counter(1 ~ 3) equal the value of Tm(1 ~ 3)Match2 and the Tm(1 ~ 3)Enable bit is set, then the tm(1 ~ 3)_intr will be triggered.

TmCR offset:0x30 default:0x000

[11]	Tm3UpDown	R/W	Timer3 up or down count 0: Down count 1: Up count
[10]	Tm2UpDown	R/W	Timer2 up or down count 0: Down count 1: Up count
[9]	Tm1UpDown	R/W	Timer1 up or down count 0: Down count 1: Up count
[8]	Tm3OFEnable	R/W	Timer3 overflow interrupt enable bit 0: Disable 1: Enable
[7]	Tm3Clock	R/W	Timer3 clock source 0: PCLK 1: EXT3CLK
[6]	Tm3Enable	R/W	Timer3 enable bit 0: Disable 1: Enable
[5]	Tm2OFEnable	R/W	Timer2 overflow interrupt enable bit 0: Disable 1: Enable
[4]	Tm2Clock	R/W	Timer2 clock source 0: PCLK 1: EXT3CLK
[3]	Tm2Enable	R/W	Timer2 enable bit 0: Disable 1: Enable
[2]	Tm1OFEnable	R/W	Timer1 overflow interrupt enable bit 0: Disable 1: Enable
[1]	Tm1Clock	R/W	Timer1 clock source 0: PCLK 1: EXT3CLK
[0]	Tm1Enable	R/W	Timer1 enable bit 0: Disable 1: Enable

IntrState offset:0x34 default:0x000

[8]	Tm3Overflow	R/W	Tm3Overflow interrupt 0: No effect 1: Tmr3 counter overflow
[7]	Tm3Match2	R/W	Tm3Match2 interrupt 0: No effect 1: Tmr3 counter value equals to the value in theTm3Match2 register.
[6]	Tm3Match1	R/W	Tm3Match1 interrupt 0: No effect 1: Tmr3 counter value equals to the value in theTm3Match1 register.
[5]	Tm2Overflow	R/W	Tm2Overflow interrupt 0: No effect 1: Tmr2 counter overflow
[4]	Tm2Match2	R/W	Tm2Match2 interrupt 0: No effect 1: Tmr2 counter value equals to the value in theTm3Match2 register.
[3]	Tm2Match1	R/W	Tm2Match1 interrupt 0: No effect 1: Tmr2 counter value equals to the value in theTm3Match1 register.
[2]	Tm3Overflow	R/W	Tm1Overflow interrupt 0: No effect 1: Tmr1 counter overflow
[1]	Tm3Match2	R/W	Tm1Match2 interrupt 0: No effect 1: Tmr1 counter value equals to the value in theTm3Match2 register.
[0]	Tm3Match1	R/W	Tm1Match1 interrupt 0: No effect 1: Tmr1 counter value equals to the value in theTm3Match1 register.

IntrMask offset:0x38 default:0x000

[8]	MTm3Overflow	R/W	Mask Tm3Overflow interrupt 0: No effect 1: Tmr3Overflow in IntrState will be masked.
[7]	MTm3Match2	R/W	Mask Tm3Match2 interrupt 0: No effect 1: Tmr3Match2 in IntrState will be masked.

[6]	MTm3Match1	R/W	Mask Tm3Match1 interrupt 0: No effect 1: Tmr3Match1 in IntrState will be masked.
[5]	MTm2Overflow	R/W	Mask Tm2Overflow interrupt 0: No effect 1: Tmr2Overflow in IntrState will be masked.
[4]	MTm2Match2	R/W	Mask Tm2Match2 interrupt 0: No effect 1: Tmr2Match2 in IntrState will be masked.
[3]	MTm2Match1	R/W	Mask Tm2Match1 interrupt 0: No effect 1: Tmr2Match1 in IntrState will be masked.
[2]	MTm1Overflow	R/W	Mask Tm1Overflow interrupt 0: No effect 1: Tmr1Overflow in IntrState will be masked.
[1]	MTm1Match2	R/W	Mask Tm1Match2 interrupt 0: No effect 1: Tmr1Match2 in IntrState will be masked.
[0]	MTm1Match1	R/W	Mask Tm1Match1 interrupt 0: No effect 1: Tmr1Match1 in IntrState will be masked.

TmRevision offset:0x3C default:0x000

[31:0]	TmRevision	R	The revision number of ATFTMR010

Tm1Prescaler offset:0x40 default:0x000

[31:8]	reserved		
[7:0]	Tm1Prescaler	R/W	intialized number of the down counter Tm1counter_pre

Tm2Prescaler offset:0x44 default:0x000

[31:8]	reserved		
[7:0]	Tm2Prescaler	R/W	intialized number of the down counter Tm2counter_pre

Tm3Prescaler offset:0x48 default:0x000

[31:8]	reserved		
[7:0]	Tm3Prescaler	R/W	intialized number of the down counter Tm3counter_pre

5.16 Watchdog

Watchdog module is an APB bus device. It is used to prevent the system from the infinite loop if the software gets trapped in the deadlock. In the normal operation, the user restarts the WDT at the regular intervals before the counter counts down to 0. If the counter does reach 0, the WDT generates one or a combination of the signals, system reset, system interrupt, or external interrupt to reset the system, interrupt the system, or interrupt an external device correspondingly.

Table 13 Watchdog module register list (BaseAddr = 0x4001_4800)

Offset	Type	Width	Name	Description	Reset Value
0x00	R	32	WdControl	The WatchDog timer counter register	0x3EF1480
0x04	R/W	16	WdLosd	The WatchDog timer auto reload register The auto_reload register is set to 0x3EF1480 as the default.	0x3EF1480
0x08	W	16	WdRestart	The WatchDog timer counter register If writing oX5AB9 to this register,the The WatchDog timer will automatically reload the Wdload to Wdcounter and restart the counting.	0x0000
0x0c	R/W	5	WdCR	The WatchDog timer counter register	0x0
0x10	R	1	WdStatus	The WatchDog timer status register This bit is set when the counter reaches 0. 0:Does not reach 0.	0x0
0x14	W	1	WdClear	The WatchDog timer is cleraed Writing 1 or 0 to this register will clear the WdStatus.	0x0
0x18	R/W	8	WdIntrlen	The WatchDog timer interrupt length This register controls the length of wd_st, wd_intr,and wd_ext.the default value is 0XFF	0xEF
0x1c	R	32	WdRevision	The revition number of ATFWDT010	0x--

WdCounter offset:0x00 default:0x3EF1480

[31:0]	WdCounter	R	The WdCounter contains the current counter value. When reset, the WdCounter register is set to 0x3EF1480. After the programmer writes 0x5AB9 to the WdRestart, the value of the WdLoad will be loaded into the WdCounter. The WdCounter starts to

			decrease once the WdCR[0], the WatchDog timer enable bit, is set. If the WatchDog
			timer is disabled, the WdCounter will hold the value. If the WdCR [4] is set, the
			register is driven by an external clock, and the WdCounter will decrease in the EXTCLK frequency. This register is read only.

WdLoad offset:0x04 default:0x3EF1480

[31:0]	WdLoad	R/W	The WdLoad contains the value which will be loaded into the WdCounter. When reset or restarted, the value of the WdLoad will be automatically loaded into the WdCounter register. The reset value of the WdLoad is 0x3EF1480.

WdRestart offset:0x08 default:0x0000

[15:0]	WdRestart	W	The WdRestart is used to avoid the unexpected counting. If the programmer writes 0x5AB9 to this register, the WatchDog timer counter will load the WdLoad into the WdCounter register and the WatchDog timer counter will restart to decrease. After finishing the write cycle, the WdRestart will automatically be reset to 0.

WdCR offset:0x0C default:0x00

[4]	WdClock	R/W	The WatchDog timer clock source bit 0: PCLK 1: EXTCLK
[3]	WdExt	R/W	The WatchDog timer external signal enable bit 0: Disable 1: Enable
[2]	WdIntr	R/W	The WatchDog timer system interrupt enable bit 0: Disable 1: Enable
[1]	WdRst	R/W	The WatchDog timer system reset enable bit 0: Disable 1: Enable
[0]	WdEnable	R/W	The WatchDog timer enable bit 0: Disable 1: Enable

WdStatus offset:0x10 default:0x0

[0]	WdStatus	R	The WdStatus register records if the WatchDog timer reaches 0 or not. It is read only.

WdClear offset:0x14 default:0x00

[0]	WdClear	W	When writing 1 to this register, the WdStatus will be cleared.

WdIntrlen offset:0x18 default:0xFF

[7:0]	WdIntrlen	R/W	The WdIntrlen register decides the duration of the assertion of

			wd_rst, wd_intr, and wd_ext signals. The default value is 0xFF, which means that the default assertion duration of wd_rst, wd_intr, and wd_ext is 256 clock cycles.

WdRevision offset:0x1C default:0x0000_0000

[31:0]	WdRevision	R	The revision number of ATFWDT010

5.17 SPI

CJC6822 integrates 1 SPI interfaces. SPI is a kind of synchronous serial port interface that allows the host processor to serve as a master or a slave. It can connect to various devices by using serial protocol. It supports several kind of synchronous serial port such as the Synchronous Serial Port (SSP) from Texas Instruments, the Serial Peripheral Interface (SPI) from Motorola, MICROWIRE from National Semiconductor, I2S from Philips, AC-link from Intel, and SPDIF. And the serial data formats may range from 4 bits to 32 bits in length.

SPI module interface inlcude AHB bus interface, SPI external interface, TX/RX FIFO signal and interrupt signal.

Table 14 SPI module register list (BaseAddr = 0x1000_0000)

Offset	Type	Width	Name	Description	Reset
+0x00	R/W	32	Apb setting register	Apb setting register	0x0002_0004
+0x04	W	32	Pio register	Pio register	0x0000_0004
+0x04	R	32	Pio register	Pio register	0x0000_0004
+0x08	R/W	32	Control register	Control register	0x0000_0000
+0x0c	R/W	32	Spi status register	Spi status register	0x0000_0000
+0x10	R/W	32	Interupt ctrl register	Interupt ctrl register	0x0000_0000
+0x14	R/W	32	Interupt status register	Interupt status register	0x0000_0000
+0x18	R/W	32	Transmit ctrl register	Transmit ctrl register	0x0000_0000
+0x1c	R/W	32	transmit data register	transmit data register	x0000_0000
+0x20	R/W	32	Ahb setting register	Ahb setting register	0x0002_0001
+0x38	R/W	32	Fifo information register	Fifo information register	0x0000_0101

Apb setting register offset:0x00 default:0x0002_0004

[31:27]	reserved		
[26]	master	R/W	0:spi master 1:spi slave
[25:24]	spi_mode	R/W	cpol/cpha: 0/0, 0/1, 1/0, 1/1
[23:20]	reserved		
[19:16]	cs_period	R/W	Time between each transmitting(cs high), for spi master mode
[15:8]	clk_baud2	R/W	clk divide vector 2
[7:0]	clk_baud1	R/W	clk divide vector 1, Fsclk=

			Fhclk/(2*(clk_baud2*clk_baud1 + clk_baud1))

Pio register offset:0x04 default:0x0000_0004

[31:4]	reserved		
[3]	spi_pio_enable	W	1: cs_o/slk_o/mdata_o driven by following bits enable
[2]	spi_pio_cs_o	W	cs_o is driven by this bit when spi_pio_enable is high
[1]	spi_pio_sclk_o	W	sclk_o is driven by this bit when spi_pio_enable is high
[0]	spi_pio_mdata_o	W	mdata_o is driven by this bit when spi_pio_enable is high

Pio register offset:0x04 default:0x0000_0004

[31:5]	reserved		
[4]	spi_pio_enable	R	1: cs_o/slk_o/mdata_o driven by following bits enable 0: disabled
[3]	spi_pio_cs_i	R	spi_cs_i monitor
[2]	spi_pio_sclk_in	R	spi_sclk_in monitor
[1]	spi_pio_sdata_i	R	spi_sdata_i monitor
[0]	spi_pio_mdata_i	R	spi_mdata_i monitor

Control register offset:0x08 default:0x0000_0000

[31:23]	reserved		
[22:18]	txf_threshold	R/W	spi tx fifo int trigering threshold
[17:15]	reserved		
[14:10]	rxf_threshold	R/W	spi rx fifo int trigering threshold
[9:3]	reserved		
[2]	txf_clear	W	spi tx fifo pointer clear
[1]	rxf_clear	W	spi rx fifo pointer clear
[0]	spi_reset	W	spi software reset
[31:23]	reserved		
[22:18]	txf_ventrs	R	valid number of words to be transmited in tx fifo
[17]	txf_full	R	tx fifo full
[16]	txf_empty	R	tx fifo empty
[15]	reserved		
[14:10]	rxf_ventrs	R	valid number of words have been recived in rx fifo
[9]	rxf_full	R	rx fifo full
[8]	rxf_empty	R	rx fifo empty
[7:1]	reserved		
[0]	spi_busy	R	spi busy flag

Interupt ctrl register offset:0x10 default:0x0000_0000

[31:6]	reserved		
[5]	spi_conf_int_en	R/W	1:spi conflict interupt enable

[4]	spi_trans_end_int_en	R/W	1:spi transmitting end interupt enable
[3]	spi_txf_thres_int_en	R/W	1:spi tx fifo threshold interupt enable , txf_ventrs <= txf_threshold
[2]	spi_rxf_thres_int_en	R/W	1:spi rx fifo threshold interupt enable , rxf_ventrs >= rxf_threshold
[1]	spi_txf_under_run_int_en	R/W	1:spi tx fifo under run interupt enable
[0]	spi_rxf_over_run_int_en	R/W	1:spi rx fifo over run interupt enable

Interupt status register offset:0x14 default:0x0000_0000

[31:6]	reserved		
[5]	spi_conf_int_r	R/W	Read: spi conflict interupt status , Write 1 to clear this bit
[4]	spi_trans_end_int_r	R/W	Read: spi transmitting end interupt status, Write 1 to clear this bit
[3]	txf_thres_int_r	R/W	Read: spi tx fifo threshold interupt status, Write 1 to clear this bit
[2]	rxf_thres_int_r	R/W	Read: spi rx fifo threshold interupt status, Write 1 to clear this bit
[1]	txf_under_run_int_r	R/W	Read: spi tx fifo under run interupt status, Write 1 to clear this bit
[0]	rxf_over_run_int_r	R/W	Read: spi rx fifo over run interupt status, Write 1 to clear this bit

Transmit ctrl register offset:0x18 default:0x0000 0000

[31]	spi_enable	R/W	1: spi enable
[30:28]	trans_mode	R/W	000: SPI_APB_IDEL; 001: SPI_APB_W_ONLY; 010: SPI_APB_R_ONLY; 011: SPI_APB_R_A_W 100: SPI_APB_DUMMY; 101: SPI_APB_END1; 110: SPI_APB_END2 111: SPI_APB_END3
[27:26]	reserved		
[25:16]	txf_data_byt_num	R/W	trans length of one transmitting
[15]	reserved		
[14:12]	dummy_data_byt_num	R/W	dummy trans length of one transmitting
[11:10]	reserved		
[9:0]	rxf_data_byt_num		recive length of one transmitting

transmit data register offset:0x1C default:0x0000_0000

[31:0]	txf_data	W	tx fifo data writing window

transmit data register offset:0x1C default:0x0000_0000

[31:0]	rxf_data_out	R	rx fifo data reading window

Ahb setting register offset:0x20 default:0x0002_0001

[31:26]	reserved		
[25:24]	spi_mode	R/W	cpol/cpha: 0/0, 0/1, 1/0, 1/1
[23:20]	reserved		
[19:16]	cs_period	R/W	Time between each transmitting(cs high), for spi master mode
[15:8]	clk_baud2	R/W	clk divide vector 2
[7:0]	clk_baud1	R/W	clk divide vector 1, Fsclk= Fhclk/(2*(clk_baud2*clk_baud1 + clk_baud1))

Fifo information register offset:0x38 default:0x0000_0101

[31:10]	reserved		
[9:8]	txf_depth_info	R	tx fifo depth
[7:2]	reserved		
[1:0]	rxf_depth_info	R	rx fifo depth

5.18 USB controller

USB controller is an AHB device, the main function is to implement the data transfer between CJC6822 system and external USB master device or USB slave device.

USB controller module support USB OTG, it can work as a host to access the external USB device, it also can work as USB device being accessed by external USB master such as PC.

USB controller is compliant with USB specification revision 2.0, it is Compliant with On-The-Go supplement to USB 2.0 specification revision 1.0,it Supports UTMI+ level 2 compliant transceiver and compliant with EHCI(Enhanced Host Controller Interface Specification for USB) 1.0,it support OTG SRP(OTG Session Request Protocol) and HNP(OTG Host Negotiation Protocol) .it Supports point-to-point communications with one HS/FS/LS device, endpoint in this module is can be hardware configured as HS/FS device. Both host and device support isochronous, interrupt, control, bulk transfers. it support DMA access to internal FIFO, and support suspend mode, remote wake-up and resume.

USB controller is mainly composed of a UTM synchronization, packet encode/decode, RAM controller endpoint control and CPU interface, as shown in Figure 23.

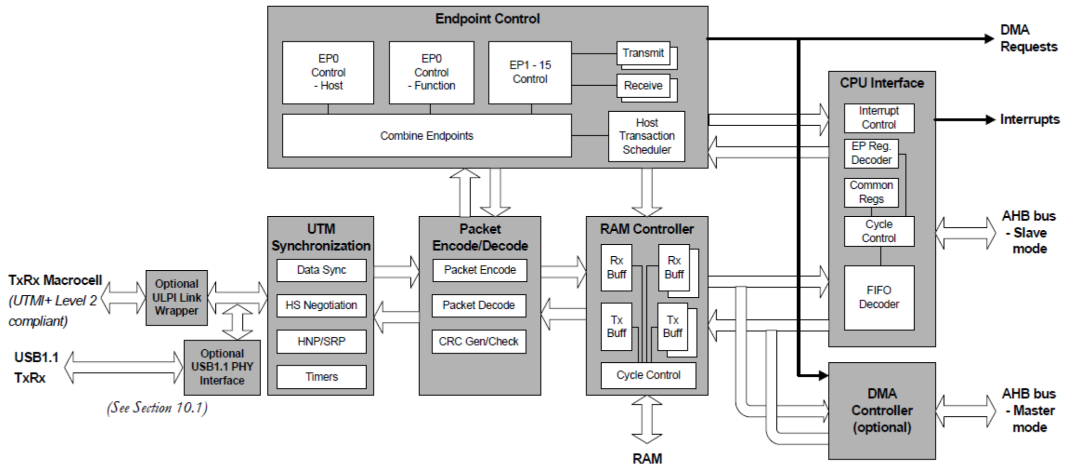


Figure 23 USB controller module block diagram

The MUSBHDRC register map is split into the following sections:

Common USB registers (00h–0Fh) – These registers provide control and status for the complete core.

Endpoint Control/Status registers (10h–1Fh, indexed) – These registers provide control and status for the endpoints. The

registers mapped into this section depend on whether the core is in Peripheral mode (DevCtl.D2=0) or in Host mode

(DevCtl.D2=1) and on the value of the Index register.

FIFOs (20h–5Fh) – This address range provides access to the endpoint FIFOs.

Additional Control and Configuration registers (60h–7Fh) – These registers provide additional device status and control.

Non-Indexed Endpoint Control/Status registers (100h and above) – The registers available at 10h–1Fh, accessible

independently of the setting of the Index register. 100h–10Fh EP0 registers; 110h–11Fh EP1 registers; 120h–12Fh EP2; et seq.

DMA Control Registers (200h and above) – These registers only appear if the design is synthesized to include optional DMA controller.

RqPktCount Registers (302h – 31Eh) – These registers are used in Host mode in conjunction with AutoReq.

The resulting Memory Map is illustrated in the diagram on the following page.

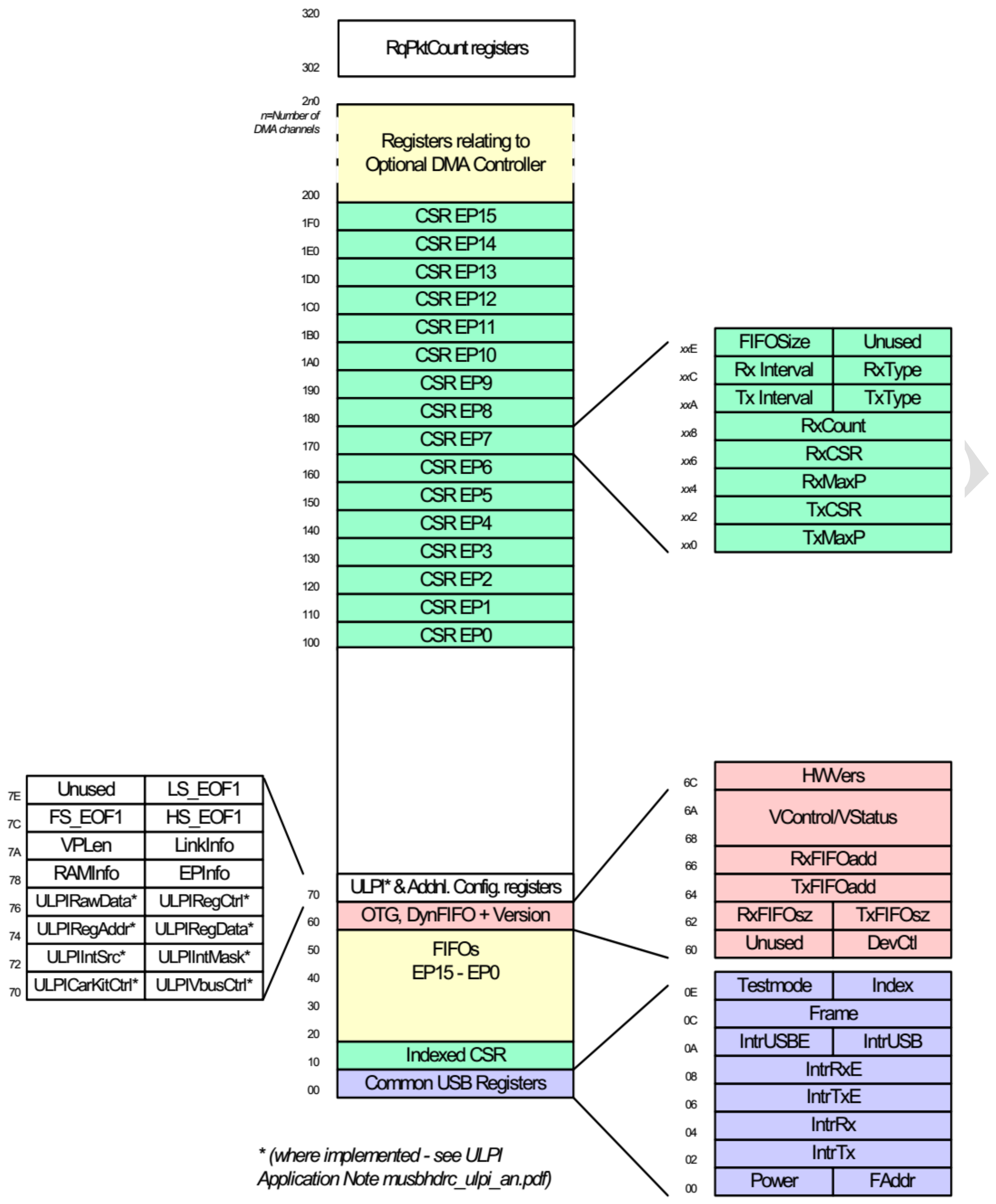


Figure 24 USB memory map

Table 15 USB register list(BaseAddr=0x4000_2000)

USB driver controller REGISTER MAP: Common USB registers				
ADDR	NAME	TYPE	DESCRIPTION	DEFAULT
00	FAddr		Function address register	8'h00
01	Power		Power management register	8’h20
02,03	IntrTx		Interrupt register for endpoint 0 plus Tx Endpoints 1 to 15	16’h0000
04,05	IntrRx		Interrupt register for Rx Endpoints 1 to 15	16’h0000
06,07	IntrTxE		Interrupt enable register for IntrTx	16’hFFFF
08,09	IntrRxE		Interrupt enable register for IntrRx	16’hFFFE

0A	IntrUSB		Interrupt register for common USB interrupts	8'h00
0B	IntrUSBE		Interrupt enable register for IntrUSB	8'h06
0C,0D	Frame		Frame number	16'h0000
0E	Index		Index register for selecting the endpoint status and control registers	4'b0000
0F	Testmode		Enables the USB 2.0 test modes	8'h00
10,11	TxMaxP		Maximum packet size for peripheral Tx endpoint. (Index register set to select Endpoints 1 – 15 only)	16'h0000
12,13	CSR0		Control Status register for Endpoint 0. (Index register set to select Endpoint 0)	8'h00
	Tx CSR		Control Status register for peripheral Tx endpoint. (Index register set to select Endpoints 1 – 15)	
14,15	RxMaxP		Maximum packet size for peripheral Rx endpoint. (Index register set to select Endpoints 1 – 15 only)	16'h0000
16,17	RxCSR		Control Status register for peripheral Rx endpoint. (Index register set to select Endpoints 1 – 15)	16'h0000
18,19	Count0		Control Status register for Endpoint 0. (Index register set to select Endpoint 0)	7'b0000000
	RxCount		Control Status register for peripheral Tx endpoint. (Index register set to select Endpoints 1 – 15)	
1A,1B	-		Reserved. Value returned affected by use in Host mode	
1C,1E	-		Unused, always return 0	
1F	ConfigData		Returns details of core configuration. (Index register set to select Endpoint 0.)	
	FIFOSize		Returns the configured size of the selected Rx FIFO and Tx FIFOs (Endpoints 1 – 15 only).	
10,11	TxMaxP		Maximum packet size for host Tx endpoint. (Index register set to select Endpoints 1 – 15 only)	16'h0000
12,13	CSR0		Control Status register for Endpoint 0. (Index register set to select Endpoint 0)	16'h0000
	Tx CSR		Control Status register for host Tx endpoint. (Index register set to select Endpoints 1 – 15)	16'h0000
14,15	RxMaxP		Maximum packet size for host Rx endpoint. (Index register set to select Endpoints 1 – 15 only)	16'h0000
16,17	RxCSR		Control Status register for host Rx endpoint. (Index register set to select Endpoints 1 – 15)	16'h0000
18,19	Count0		Number of received bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)	13'b0000000000000
	RxCount		Number of bytes in host Rx endpoint FIFO. (Index register set to select Endpoints 1 – 15)	

1A	TxType		Sets the transaction protocol and peripheral endpoint number for the host Tx endpoint. (Index register set to select Endpoints 1 – 15 only)		8’h00
1B	NAKLimit0		Sets the NAK response timeout on endpoint 0. (Index register set to select Endpoint 0)		8’b00000000
	TxInterval		Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Tx endpoint. (Index register set to select Endpoints 1 – 15 only)		
1C	RxType		Sets the transaction protocol and peripheral endpoint number for the host Rx endpoint. (Index register set to select Endpoints 1 – 15 only)		8’h00
1D	RxInterval		Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Rx endpoint. (Index register set to select Endpoints 1 – 15 only)		8’b00000000
1E	-		*Unused,* always return 0		
1F	ConfigData		Returns details of core configuration. (Index register set to select Endpoint 0).		
	FIFOSize		Returns the configured size of the selected Rx FIFO and Tx FIFOs (Endpoints 1 – 15 only).		
USB driver controller REGISTER MAP: FIFOs					
20-5F	FIFOx		FIFOs for endpoints 0-15		
60	DevCtl		OTG device control register		8’h80
61	-		Unused		
62	TxFIFOsz		Tx Endpoint FIFO size	Only used if Dynamic FIFO sizing option is selected otherwise return 0.	
63	RxFIFOsz		Rx Endpoint FIFO size		
64,65	TxFIFOadd		Tx Endpoint FIFO address		
66,67	RxFIFOadd		Rx Endpoint FIFO address		
68-6B	VControl/VStatus		UTMI+PHY Vendor registers(unused)		
6C,6D	HWVers		Hardware Version number register		
6E,6F	-		Unused		
70-77	-		ULPI Registers, only implemented where ULPI Link Wrapper is used.		
78	EPInfo		Information about numbers of Tx and Rx endpoints.		
79	RAMInfo		Information about the width of the RAM and the number of DMA channels.		
7A	LinkInfo		Information about delays to be applied		8'h5C
7B	VPLen		Duration of the VBus pulsing charge		8'h3C

7C	HS_EOF1		Time buffer available on High-Speed transactions	8'h80
7D	FS_EOF1		Time buffer available on Full-Speed transactions	8'h77
7E	LS_EOF1		Time buffer available on Low-Speed transactions	8'h72
7F	-		Unused	
USB driver controller REGISTER MAP: RqPktCount Registers (302h – 31Eh)				
300+2*n	RqPktCount		Number of requested packets for receive endpoint n (endpoints 1-15 only)	16'h0000

FADDR

Bit	Name	from CPU	from USB	Function
[7]	-	r	-	Unused, always returns 0.
[6:0]	Func Addr	rw	r	The function address.

POWER

Bit	Name	from CPU	from USB	Function
[7]	ISO Update	rw	r	When set by the CPU, the MUSBHDRC will wait for an SOF token from the time TxPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent.
[6]	Soft Conn	rw	r	If Soft Connect/Disconnect feature is enabled, then the USB D+/D- lines are enabled when this bit is set by the CPU and tri-stated when this bit is cleared by the CPU.
[5]	HS Enab	rw	r	When set by the CPU, the MUSBHDRC will negotiate for High-speed mode when the device is reset by the hub. If not set, the device will only operate in Full-speed mode.
[4]	HS Mode	r	rw	When set, this read-only bit indicates High-speed mode successfully negotiated during USB reset. In Peripheral Mode, becomes valid when USB reset completes (as indicated by USB reset interrupt). In Host Mode, becomes valid when Reset bit is cleared. Remains valid for the duration of the session.
[3]	Reset	r	rw	This bit is set when Reset signaling is present on the bus.
[2]	Resume	rw	r	Set by the CPU to generate Resume signaling when the function is in Suspend mode. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling. In Host mode, this bit is also automatically set when Resume signaling from the target is detected while the MUSBHDRC is

				suspended.
[1]	Suspend Mode	r	rw	In Host mode, this bit is set by the CPU to enter Suspend mode. In Peripheral mode, this bit is set on entry into Suspend mode. It is cleared when the CPU reads the interrupt register, or sets the Resume bit above.
[0]	Enable SuspendM	rw	r	Set by the CPU to enable the SUSPENDM signal.

INTRTX

Bit	Name	from CPU	from USB	Function
[15]	EP15 Tx	r	set	Tx Endpoint 15 interrupt.
[14]	EP14 Tx	r	set	Tx Endpoint 14 interrupt.
[13]	EP13 Tx	r	set	Tx Endpoint 13 interrupt.
[12]	EP12 Tx	r	set	Tx Endpoint 12 interrupt.
[11]	EP11 Tx	r	set	Tx Endpoint 11 interrupt.
[10]	EP10 Tx	r	set	Tx Endpoint 10 interrupt.
[9]	EP9 Tx	r	set	Tx Endpoint 9 interrupt
[8]	EP8 Tx	r	set	Tx Endpoint 8 interrupt
[7]	EP7 Tx	r	set	Tx Endpoint 7 interrupt
[6]	EP6 Tx	r	set	Tx Endpoint 6 interrupt
[5]	EP5 Tx	r	set	Tx Endpoint 5 interrupt
[4]	EP4 Tx	r	set	Tx Endpoint 4 interrupt
[3]	EP3 Tx	r	set	Tx Endpoint 3 interrupt
[2]	EP2 Tx	r	set	Tx Endpoint 2 interrupt
[1]	EP1 Tx	r	set	Tx Endpoint 1 interrupt
[0]	EP0 Tx	r	set	Tx Endpoint 0 interrupt

INTRRX

Bit	Name	from CPU	from USB	Function
[15]	EP15 Rx	r	set	Rx Endpoint 15 interrupt.
[14]	EP14 Rx	r	set	Rx Endpoint 14 interrupt.
[13]	EP13 Rx	r	set	Rx Endpoint 13 interrupt.
[12]	EP12 Rx	r	set	Rx Endpoint 12 interrupt.
[11]	EP11 Rx	r	set	Rx Endpoint 11 interrupt.
[10]	EP10 Rx	r	set	Rx Endpoint 10 interrupt.
[9]	EP9 Rx	r	set	Rx Endpoint 9 interrupt
[8]	EP8 Rx	r	set	Rx Endpoint 8 interrupt
[7]	EP7 Rx	r	set	Rx Endpoint 7 interrupt
[6]	EP6 Rx	r	set	Rx Endpoint 6 interrupt
[5]	EP5 Rx	r	set	Rx Endpoint 5 interrupt
[4]	EP4 Rx	r	set	Rx Endpoint 4 interrupt

[3]	EP3 Rx	r	set	Rx Endpoint 3 interrupt
[2]	EP2 Rx	r	set	Rx Endpoint 2 interrupt
[1]	EP1 Rx	r	set	Rx Endpoint 1 interrupt
[0]	–			Unused, always returns 0

INTRTXE

Bit	Name	from CPU	from USB	Function
[15]	EP15 Tx en	rw	r	Tx Endpoint 15 interrupt enable bits.
[14]	EP14 Tx en	rw	r	Tx Endpoint 14 interrupt enable bits .
[13]	EP13 Tx en	rw	r	Tx Endpoint 13 interrupt enable bits.
[12]	EP12 Tx en	rw	r	Tx Endpoint 12 interrupt enable bits.
[11]	EP11 Tx en	rw	r	Tx Endpoint 11 interrupt enable bits.
[10]	EP10 Tx en	rw	r	Tx Endpoint 10 interrupt enable bits.
[9]	EP9 Tx en	rw	r	Tx Endpoint 9 interrupt enable bits
[8]	EP8 Tx en	rw	r	Tx Endpoint 8 interrupt enable bits
[7]	EP7 Tx en	rw	r	Tx Endpoint 7 interrupt enable bits
[6]	EP6 Tx en	rw	r	Tx Endpoint 6 interrupt enable bits
[5]	EP5 Tx en	rw	r	Tx Endpoint 5 interrupt enable bits
[4]	EP4 Tx en	rw	r	Tx Endpoint 4 interrupt enable bits
[3]	EP3 Tx en	rw	r	Tx Endpoint 3 interrupt enable bits
[2]	EP2 Tx en	rw	r	Tx Endpoint 2 interrupt enable bits
[1]	EP1 Tx en	rw	r	Tx Endpoint 1 interrupt enable bits
[0]	EP0 Tx en	rw	r	Tx Endpoint 0 interrupt enable bits

INTRRXE

Bit	Name	from CPU	from USB	Function
[15]	EP15 Rx en	rw	r	Rx Endpoint 15 interrupt enable bits.
[14]	EP14 Rx en	rw	r	Rx Endpoint 14 interrupt enable bits .
[13]	EP13 Rx en	rw	r	Rx Endpoint 13 interrupt enable bits.
[12]	EP12 Rx en	rw	r	Rx Endpoint 12 interrupt enable bits.
[11]	EP11 Rx en	rw	r	Rx Endpoint 11 interrupt enable bits.
[10]	EP10 Rx en	rw	r	Rx Endpoint 10 interrupt enable bits.
[9]	EP9 Rx en	rw	r	Rx Endpoint 9 interrupt enable bits
[8]	EP8 Rx en	rw	r	Rx Endpoint 8 interrupt enable bits
[7]	EP7 Rx en	rw	r	Rx Endpoint 7 interrupt enable bits
[6]	EP6 Rx en	rw	r	Rx Endpoint 6 interrupt enable bits
[5]	EP5 Rx en	rw	r	Rx Endpoint 5 interrupt enable bits
[4]	EP4 Rx en	rw	r	Rx Endpoint 4 interrupt enable bits
[3]	EP3 Rx en	rw	r	Rx Endpoint 3 interrupt enable bits
[2]	EP2 Rx en	rw	r	Rx Endpoint 2 interrupt enable bits
[1]	EP1 Rx en	rw	r	Rx Endpoint 1 interrupt enable bits
[0]	–			Unused, always returns 0

INTRUSB

Bit	Name	from CPU	from USB	Function
[7]	VBus Error	r	set	Set when VBus drops below the VBus Valid threshold during a session. Only valid when MUSBHDRC is 'A' device.
[6]	Sess Req	r	set	Set when Session Request signaling has been detected. Only valid when MUSBHDRC is 'A' device.
[5]	Discon	r	set	Set in Host mode when a device disconnect is detected. Set in Peripheral mode when a session ends. Valid at all transaction speeds.
[4]	Conn	r	set	Set when a device connection is detected. Only valid in Host mode. Valid at all transaction speeds.
[3]	SOF	r	set	Set when a new frame starts.
[2]	Reset	r	set	Set in Peripheral mode when Reset signaling is detected on the bus.
	Babble	r	set	Set in Host mode when babble is detected. Note: Only active after first SOF has been sent.
[1]	Resume	r	set	Set when Resume signaling is detected on the bus while the MUSBHDRC is in Suspend mode.
[0]	Suspend	r	set	Set when Suspend signaling is detected on the bus. Only valid in Peripheral mode.

INTRUSBE

Bit	Name	from CPU	from USB	Function
[7]	VBus Error en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[6]	Sess Req en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[5]	Discon en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[4]	Conn en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[3]	SOF en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[2]	Reset en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
	Babble en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[1]	Resume en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.
[0]	Suspend en	r	set	interrupt enable bits for each of the interrupts in IntrUSB.

FRAME

Bit	Name	from CPU	from USB	Function
[15:11]		r	w	[15:11]=’b00000
[10:0]		r	w	holds the last received frame number

INDEX

Bit	Name	from CPU	from USB	Function
[3:0]		rw	r	4-bit register that determines which endpoint control/status registers are accessed at addresses 10h to 19h.

TESTMODE

Bit	Name	from CPU	from USB	Function
[7]	Force_Host	rw	r	The CPU sets this bit to instruct the core to enter Host mode when the Session bit is set, regardless of whether it is connected to any peripheral. The state of the CID input, HostDisconnect and LineState signals are ignored. The core will then remain in Host mode until the Session bit is cleared, even if a device is disconnected, and if the Force_Host bit remains set, will re-enter Host mode the next time the Session bit is set. While in this mode, the status of the HOSTDISCON signal from the PHY may be read from bit 7 of the DevCtl register. The operating speed is determined from the Force_HS and Force_FS bits as follows: Force_H S Force_FS Operating Speed 0 0 Low Speed 0 1 Full Speed 1 0 High Speed 1 1 Undefined
[6]	FIFO_Access	set	r	The CPU sets this bit to transfer the packet in the Endpoint 0 Tx FIFO to the Endpoint 0 Rx FIFO. It is cleared automatically.
[5]	Force_FS	rw	r	The CPU sets this bit either in conjunction with bit 7 above or to force the MUSBHDRC into Full speed mode when it receives a USB reset.
[4]	Force_HS	rw	r	The CPU sets this bit either in conjunction with bit 7 above or to force the MUSBHDRC into High speed mode when it receives a USB reset.
[3]	Test_Packet	rw	r	(High-speed mode) The CPU sets this bit to

				enter the Test_Packet test mode. In this mode, the MUSBHDRC repetitively transmits on the bus a 53-byte test packet
[2]	Test_K	rw	r	(High-speed mode) The CPU sets this bit to enter the Test_K test mode. In this mode, the MUSBHDRC transmits a continuous K on the bus.
[1]	Test_J	rw	r	(High-speed mode) The CPU sets this bit to enter the Test_J test mode. In this mode, the MUSBHDRC transmits a continuous J on the bus.
[0]	Test_SE0_NAK	r	r	(High-speed mode) The CPU sets this bit to enter the Test_SE0_NAK test mode. In this mode, the MUSBHDRC remains in High-speed mode but responds to any valid IN token with a NAK.

DEVCTL

Bit	Name	from CPU	from USB	Function
[7]	B-Device	r	rw	This Read-only bit indicates whether the MUSBHDRC is operating as the ‘A’ device or the ‘B’ device. 0 ⇒ ‘A’ device; 1 ⇒ ‘B’ device. Only valid while a session is in progress.
[6]	FSDev	r	rw	This Read-only bit is set when a full-speed or high-speed device has been detected being connected to the port. (High-speed devices are distinguished from full-speed by checking for high-speed chirps when the device is reset.) Only valid in Host mode.
[5]	LSDev	r	rw	This Read-only bit is set when a low-speed device has been detected being connected to the port. Only valid in Host mode.
[4:3]	VBus[1:0]	r	rw	These Read-only bits encode the current VBus level as follows: D4 D3 Meaning 0 0: Below SessionEnd 0 1: Above SessionEnd, below AValid 1 0: Above AValid, below VBusValid 1 1: Above VBusValid
[2]	Host Mode	r	rw	This Read-only bit is set when the MUSBHDRC is acting as a Host.
[1]	Host Req	rw	rw	When set, the MUSBHDRC will initiate the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed.

[0]	Session	r	rw	When operating as an ‘A’ device, this bit is set or cleared by the CPU to start or end a session. When operating as a ‘B’ device, this bit is set/cleared by the MUSBHDRC when a session starts/ends. It is also set by the CPU to initiate the Session Request Protocol, or cleared by the CPU when in Suspend mode to perform a software disconnect.

CSR0 in Peripheral mode:

Bit	Name	from CPU	from USB	Function
[15:9]	–			Unused. Return 0 when read.
[8]	FlushFIFO	set	r	The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TxPktRdy/RxPktRdy bit (below) is cleared.
[7]	ServicedSetupEnd	set	r	The CPU writes a 1 to this bit to clear the SetupEnd bit. It is cleared automatically.
[6]	ServicedRxPktRdy	set	r	The CPU writes a 1 to this bit to clear the RxPktRdy bit. It is cleared automatically.
[5]	SendStall	set	r	The CPU writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.
[4]	SetupEnd	r	set	This bit will be set when a control transaction ends before the DataEnd bit has been set. An interrupt will be generated and the FIFO flushed at this time. The bit is cleared by the CPU writing a 1 to the ServicedSetupEnd bit.
[3]	DataEnd	set	r	The CPU sets this bit: 1. When setting TxPktRdy for the last data packet. 2. When clearing RxPktRdy after unloading the last data packet. 3. When setting TxPktRdy for a zero length data packet. It is cleared automatically.
[2]	SentStall	r/clear	set	This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.
[1]	TxPktRdy	r/set	r	The CPU sets this bit after loading a data

				packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.
[0]	RxPktRdy	r	set	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU clears this bit by setting the ServicedRxPktRdy bit.

CSR0 in Host mode:

Bit	Name	from CPU	from USB	Function
[15:12]	–			Unused. Return 0 when read.
[11]	Dis Ping	rw	r	The CPU writes a 1 to this bit to instruct the core not to issue PING tokens in data and status phases of a high-speed Control transfer (for use with devices that do not respond to PING).
[10:9]	–			Unused. Return 0 when read.
[8]	FlushFIFO	set	r	The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TxPktRdy/RxPktRdy bit (below) is cleared. Note: FlushFIFO should only be used when TxPktRdy/RxPktRdy is set. At other times, it may cause data to be corrupted.
[7]	NAK Timeout	r/clear	set	This bit will be set when Endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the NAKLimit0 register. The CPU should clear this bit to allow the endpoint to continue.
[6]	StatusPkt	rw	r	The CPU sets this bit at the same time as the TxPktRdy or ReqPkt bit is set, to perform a status stage transaction. Setting this bit ensures that the data toggle is set to 1 so that a DATA1 packet is used for the Status Stage transaction.
[5]	ReqPkt	rw	rw	The CPU sets this bit to request an IN transaction. It is cleared when RxPktRdy is set.
[4]	Error	r/clear	set	This bit will be set when three attempts have been made to perform a transaction with no response from the peripheral. The CPU should clear this bit. An interrupt is generated when this bit is set.
[3]	SetupPkt	r/clear	rw	The CPU sets this bit, at the same time as the TxPktRdy bit is set, to send a SETUP token

				instead of an OUT token for the transaction.
[2]	RxStall	r/clear	set	This bit is set when a STALL handshake is received. The CPU should clear this bit.
[1]	TxPktRdy	r/set	clear	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.
[0]	RxPktRdy	r/clear	rw	This bit is set when a data packet has been received. An interrupt is generated (if enabled) when this bit is set. The CPU should clear this bit when the packet has been read from the FIFO. bit.

COUNT0

Bit	Name	from CPU	from USB	Function
[7:0]		r	w	7-bit read-only register that indicates the number of received data bytes in the Endpoint 0 FIFO.

CONFIGDATA

Bit	Name	from CPU	from USB	Function
[7]	MPRxE	r	r	When set to '1', automatic amalgamation of bulk packets is selected
[6]	MPTxE	r	r	When set to '1', automatic splitting of bulk packets is selected
[5]	BigEndian	r	r	When set to '1' indicates Big Endian ordering is selected.
[4]	HBRxE	r	r	When set to '1' indicates High-bandwidth Rx ISO Endpoint Support selected.
[3]	HBTxE	r	r	When set to '1' indicates High-bandwidth Tx ISO Endpoint Support selected.
[2]	DynFIFO Sizing	r	r	When set to '1' indicates Dynamic FIFO Sizing option selected.
[1]	SoftConE	r	rw	When set to '1' indicates Soft Connect/Disconnect option selected.
[0]	UTMI DataWidth	r	r	Indicates selected UTMI+ data width. 0 ⇒ 8 bits; 1 ⇒ 16 bits. starts/ends. It is also set by the CPU to initiate the Session Request Protocol, or cleared by the CPU when in Suspend mode to perform a software disconnect.

NAKLIMIT0 (Host Mode only)

Bit	Name	from CPU	from USB	Function

[4:0]		rw	r	5-bit register that sets the number of frames/microframes (High-Speed transfers) after which Endpoint 0 should timeout on receiving a stream of NAK responses.

TXMAXP

Bit	Name	from CPU	from USB	Function
[15:11] Or [12:11]	multiplier	rw	r	Where the option of High-bandwidth Isochronous/Interrupt endpoints or of packet splitting on Bulk endpoints has been taken when the core is configured, the register includes either 2 or 5 further bits that define a multiplier m which is equal to one more than the value recorded.
[10:0]	Maximum Payload/transaction	rw	r	Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full speed and High-speed operations.

TXCSR In Peripheral mode

Bit	Name	from CPU	from USB	Function
[15]	AutoSet	rw	r	If the CPU sets this bit, TxPktRdy will be automatically set when data of the maximum packet size (value in TxMaxP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then TxPktRdy will have to be set manually.
[14]	ISO	rw	r	The CPU sets this bit to enable the Tx endpoint for Isochronous transfers, and clears it to enable the Tx endpoint for Bulk or Interrupt transfers
[13]	Mode	rw	r	The CPU sets this bit to enable the endpoint direction as Tx, and clears it to enable the endpoint direction as Rx.
[12]	DMAReqEnab	rw	r	The CPU sets this bit to enable the DMA request for the Tx endpoint.
[11]	FrcDataTog	rw	r	The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx

				endpoints that are used to communicate rate feedback for Isochronous endpoints.
[10]	DMAReqMode	rw	r	The CPU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
[9:8]	–	r	r	Unused, always return 0.
[7]	IncompTx	r/clear	set	When the endpoint is being used for high-bandwidth Isochronous/Interrupt transfers, this bit is set to indicate where a large packet has been split into 2 or 3 packets for transmission but insufficient IN tokens have been received to send all the parts.
[6]	ClrDataTog	set	r/clear	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
[5]	SentStall	r/clear	set	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the TxPktRdy bit is cleared (see below). The CPU should clear this bit.
[4]	SendStall	rw	r	The CPU writes a 1 to this bit to issue a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition.
[3]	FlushFIFO	set	r	The CPU writes a 1 to this bit to flush the latest packet from the endpoint Tx FIFO. The FIFO pointer is reset, the TxPktRdy bit (below) is cleared and an interrupt is generated. May be set simultaneously with TxPktRdy to abort the packet that is currently being loaded into the FIFO.
[2]	UnderRun	r/clear	set	The USB sets this bit if an IN token is received when the TxPktRdy bit not set. The CPU should clear this bit.
[1]	FIFONotEmpty	r/clear	set	The USB sets this bit when there is at least 1 packet in the Tx FIFO.
[0]	TxPktRdy	r/set	clear	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point (if enabled). TxPktRdy is also automatically cleared prior to loading a second packet into a double-buffered FIFO.

TXCSR In Host mode

Bit	Name	from CPU	from USB	Function
[15]	AutoSet	rw	r	If the CPU sets this bit, TxPktRdy will be automatically set when data of the maximum

				packet size (value in TxMaxP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then TxPktRdy will have to be set manually.
[14]	–	rw	r	Unused, always returns zero.
[13]	Mode	rw	r	The CPU sets this bit to enable the endpoint direction as Tx, and clears it to enable the endpoint direction as Rx.
[12]	DMAReqEnab	rw	r	The CPU sets this bit to enable the DMA request for the Tx endpoint.
[11]	FrcDataTog	rw	r	The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.
[10]	DMAReqMode	rw	r	The CPU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
[9:8]	–	r	r	Unused, always return 0.
[7]	NAK Timeout	r/clear	set	Bulk endpoints only: This bit will be set when the Tx endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the TxInterval register. The CPU should clear this bit to allow the endpoint to continue.
	IncompTx	r/clear	set	High-bandwidth Interrupt endpoints only: This bit will be set if no response is received from the device to which the packet is being sent.
[6]	ClrDataTog	set	r/clear	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
[5]	RxStall	r/clear	set	This bit is set when a STALL handshake is received. When this bit is set, any DMA request that is in progress is stopped, the FIFO is completely flushed and the TxPktRdy bit is cleared (see below). The CPU should clear this bit.
[4]	–	r	r	Unused. Returns zero when read.
[3]	FlushFIFO	set	r	The CPU writes a 1 to this bit to flush the latest packet from the endpoint Tx FIFO. The FIFO pointer is reset, the TxPktRdy bit (below) is cleared and an interrupt is generated. May be set simultaneously with TxPktRdy to abort the

				packet that is currently being loaded into the FIFO.
[2]	Error	r/clear	rw	The USB sets this bit when 3 attempts have been made to send a packet and no handshake packet has been received. When the bit is set, an interrupt is generated, TxPktRdy is cleared and the FIFO completely flushed. The CPU should clear this bit. Valid only when the endpoint is operating in Bulk or Interrupt mode.
[1]	FIFONotEmpty	r/clear	set	The USB sets this bit when there is at least 1 packet in the Tx FIFO.
[0]	TxPktRdy	r/set	clear	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point (if enabled). TxPktRdy is also automatically cleared prior to loading a second packet into a double-buffered FIFO.

RXMAXP

Bit	Name	from CPU	from USB	Function
[15:11] Or [12:11]	multiplier	rw	r	Where the option of High-bandwidth Isochronous/Interrupt endpoints or of combining Bulk packets has been taken when the core is configured, the register includes either 2 or 5 further bits that define a multiplier m which is equal to one more than the value recorded.
[10:0]	Maximum Payload/transaction	rw	r	Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full speed and High-speed operations.

RXCSR In Peripheral mode

Bit	Name	from CPU	from USB	Function
[15]	AutoClear	rw	r	If the CPU sets this bit then the RxPktRdy bit will be automatically cleared when a packet of RxMaxP bytes has been unloaded from the Rx FIFO. When packets of less than the maximum

				packet size are unloaded, RxPktRdy will have to be cleared manually.
[14]	ISO	rw	r	The CPU sets this bit to enable the Rx endpoint for Isochronous transfers, and clears it to enable the Rx endpoint for Bulk/Interrupt transfers.
[13]	DMAReqEnab	rw	r	The CPU sets this bit to enable the DMA request for the Rx endpoint.
[12]	DisNyet	Rw/r	r/rw	Bulk/Interrupt Transactions: The CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received Rx packets are ACK'd including at the point at which the FIFO becomes full.
[12]	PID Error	Rw/r	r/rw	ISO Transactions: The core sets this bit to indicate a PID error in the received packet.
[11]	DMAReqMode	rw	r	The CPU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
[10:9]	–	r	r	Unused, always return 0.
[8]	IncompRx	r/clear	set	This bit is set in a high-bandwidth Isochronous/Interrupt transfer if the packet in the Rx FIFO is incomplete because parts of the data were not received. It is cleared when RxPktRdy is cleared.
[7]	ClrDataTog	set	r/clear	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
[6]	SentStall	r/clear	set	This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.
[5]	SendStall	rw	r	The CPU writes a 1 to this bit to issue a STALL handshake. The CPU clears this bit to terminate the stall condition.
[4]	FlushFIFO	set	r	The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint Rx FIFO. The FIFO pointer is reset and the RxPktRdy bit (below) is cleared.
[3]	DataError	r	set	This bit is set when RxPktRdy is set if the data packet has a CRC or bit-stuff error. It is cleared when RxPktRdy is cleared.
[2]	OverRun	r/clear	set	This bit is set if an OUT packet cannot be loaded into the Rx FIFO. The CPU should clear this bit.
[1]	FIFOFull	r	set	This bit is set when no more packets can be loaded into the Rx FIFO.
[0]	RxPktRdy	r/clear	set	This bit is set when a data packet has been

				received. The CPU should clear this bit when the packet has been unloaded from the Rx FIFO. An interrupt is generated when the bit is set.

RXCSR In host mode

Bit	Name	from CPU	from USB	Function
[15]	AutoClear	rw	r	If the CPU sets this bit then the RxPktRdy bit will be automatically cleared when a packet of RxMaxP bytes has been unloaded from the Rx FIFO. When packets of less than the maximum packet size are unloaded, RxPktRdy will have to be cleared manually.
[14]	AutoReq	rw	r	If the CPU sets this bit, the ReqPkt bit will be automatically set when the RxPktRdy bit is cleared.
[13]	DMAReqEnab	rw	r	The CPU sets this bit to enable the DMA request for the Rx endpoint.
[12]	PID Error	r	rw	ISO Transactions Only: The core sets this bit to indicate a PID error in the received packet. Bulk/Interrupt Transactions: The setting of this bit is ignored.
[11]	DMAReqMode	rw	r	The CPU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
[10:9]	–	r	r	Unused, always return 0.
[8]	IncompRx	r/clear	set	This bit will be set in a high-bandwidth Isochronous/Interrupt transfer if the packet received is incomplete. It will be cleared when RxPktRdy is cleared.
[7]	ClrDataTog	set	r/clear	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
[6]	RxStall	r/clear	set	When a STALL handshake is received, this bit is set and an interrupt is generated. The CPU should clear this bit.
[5]	ReqPkt	rw	rw	The CPU writes a 1 to this bit to request an IN transaction. It is cleared when RxPktRdy is set.
[4]	FlushFIFO	set	r	The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint Rx FIFO. The FIFO pointer is reset and the RxPktRdy bit (below) is cleared.

[3]	DataError/ NAK Timeout	r (/clear)	set	When operating in ISO mode, this bit is set when RxPktRdy is set if the data packet has a CRC or bit-stuff error and cleared when RxPktRdy is cleared. In Bulk mode, this bit will be set when the Rx endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the RxInterval register. The CPU should clear this bit to allow the endpoint to continue.
[2]	Error	r/clear	set	The USB sets this bit when 3 attempts have been made to receive a packet and no data packet has been received. The CPU should clear this bit. An interrupt is generated when the bit is set.
[1]	FIFOFull	r	set	This bit is set when no more packets can be loaded into the Rx FIFO.
[0]	RxPktRdy	r/clear	set	This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the Rx FIFO. An interrupt is generated when the bit is set.

RXCOUNT

Bit	Name	from CPU	from USB	Function
[12:0]	Endpoint Rx Count	r	w	13-bit read-only register that holds the number of received data bytes in the packet currently in line to be read from the Rx FIFO. If the packet was transmitted as multiple bulk packets, the number given will be for the combined packet.

TXTYPE (Host mode only)

Bit	Name	from CPU	from USB	Function
[5:4]	Protocol	rw	r	The CPU should set this to select the required protocol for the Tx endpoint: 00: Illegal 01: Isochronous 10: Bulk 11: Interrupt
[3:0]	Target Endpoint number	rw	r	The CPU should set this value to the endpoint number contained in the Tx endpoint descriptor returned to the MUSBHDRC during device enumeration.

TXINTERVAL (Host mode only)

Bit	Name	from CPU	from USB	Function
[7:0]	Tx Polling Interval/NAK Limit (m)	rw	r	TxInterval is an 8-bit register that, for Interrupt and Isochronous transfers, defines the polling interval for the currently-selected Tx endpoint. For Bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a TxInterval register for each configured Tx endpoint (except Endpoint 0).

RXTYPE (Host mode only)

Bit	Name	from CPU	from USB	Function
[5:4]	Protocol	rw	r	The CPU should set this to select the required protocol for the Rx endpoint: 00: Illegal 01: Isochronous 10: Bulk 11: Interrupt
[3:0]	Target Endpoint number	rw	r	The CPU should set this value to the endpoint number contained in the Rx endpoint descriptor returned to the MUSBHDRC during device enumeration.

RXINTERVAL (Host mode only)

Bit	Name	from CPU	from USB	Function
[7:0]	Rx Polling Interval/NAK Limit (m)	rw	r	RxInterval is an 8-bit register that, for Interrupt and Isochronous transfers, defines the polling interval for the currently-selected Rx endpoint. For Bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a RxInterval register for each configured Rx endpoint (except Endpoint 0).

FIFOSIZE

Bit	Name	from CPU	from USB	Function
[7:4]	Rx FIFO Size	r	r	the sizes of the FIFOs associated with the selected additional Rx endpoints.
[3:0]	Tx FIFO Size	r	r	the sizes of the FIFOs associated with the selected additional Tx endpoints.

HWVERS

Bit	Name	from CPU	from USB	Function
[15]	RC	r	r	Set to ‘1’ if RTL used from a Release Candidate rather than from a full release of the core.
[14:10]	xx	r	r	Major Version Number (Range 0 – 31).
[9:0]	yyy	r	r	Minor Version Number (Range 0 – 999).

EPINFO

Bit	Name	from CPU	from USB	Function
[7:4]	RxEndPoints	r	r	The number of Rx endpoints implemented in the design.
[3:0]	TxEndPoints	r	r	The number of Tx endpoints implemented in the design.

RAMINFO

Bit	Name	from CPU	from USB	Function
[7:4]	DMAChans	r	r	The number of DMA channels implemented in the design.
[3:0]	RamBits	r	r	The width of the RAM address bus – 1.

LINKINFO

Bit	Name	from CPU	from USB	Function
[7:4]	WTCON	rw	r	Sets the wait to be applied to allow for the user’s connect/disconnect filter in units of 533.3ns. (The default setting corresponds to 2.667µs.)
[3:0]	WTID	rw	r	Sets the delay to be applied from IDPULLUP being asserted to IDDIG being considered valid in units of 4.369ms. (The default setting corresponds to 52.43ms.)

VPLEN

Bit	Name	from CPU	from USB	Function
[7:0]	VPLEN	rw	r	Sets the duration of the VBus pulsing charge in units of 546.1 µs. (The default setting corresponds to 32.77ms.)

HS_EOF1

Bit	Name	from CPU	from USB	Function
[7:0]	HS_EOF1	rw	r	Sets for High-speed transactions the time before EOF to stop beginning new transactions, in units of 133.3ns. (The default setting corresponds to 17.07µs.)

FS_EOF1

Bit	Name	from CPU	from USB	Function
[7:0]	FS_EOF1	rw	r	Sets for Full-speed transactions the time before EOF to stop beginning new transactions, in units of 533.3ns. (The default setting corresponds to 63.46µs.)

LS_EOF1

Bit	Name	from CPU	from USB	Function
[7:0]	LS_EOF1	rw	r	Sets for Low-speed transactions the time before EOF to stop beginning new transactions, in units of 1.067µs. (The default setting corresponds to 121.6µs.)

RqPktCount (Host Mode Only)

Bit	Name	from CPU	from USB	Function
[15:0]	RqPktCount	rw	rw	Sets the number of packets of size MaxP that are to be transferred in a block transfer. Only used in Host mode when AutoReq is set. Has no effect in Peripheral mode or when AutoReq is not set.

TxFIFOsz/ RxFIFOsz

Bit	Name	from CPU	from USB	Function
4	DPB	rw	r	Defines whether double-packet buffering supported. When ‘1’, double-packet buffering is supported. When ‘0’, only single-packet buffering is supported.
[3:0]	SZ[3:0]	rw	r	Maximum packet size to be allowed for (before any splitting or after any combination within the FIFO of Bulk/High-Bandwidth packets prior to or following transmission.

SZ[3:0]	Packet Size(Byte)
0000	8
0001	16
0010	32
0011	64
0100	128
0101	256
0110	512
0111	1024

				1000 — 2048 1001 — 4096 If DPB = 0, the FIFO will also be this size; if DPB = 1, the FIFO will be twice this size.

TxFIFOadd/ RxFIFOadd

Bit	Name	from CPU	from USB	Function
[12:0]	AD	rw	r	Start address of the endpoint FIFO in units of 8 bytes as follows:

AD[12:0]	Start Address
0000	0000
0001	0008
0002	0010
...	...
1FFF	FFF8

5.19 POR

Power on reset module is a system asynchronous reset signal generation module, it detect the power status and generate the reset signal when power is supply. Figure 25 is CJC6822 power on reset circuit block diagram and figure 26 is the POR signal timing sequence.

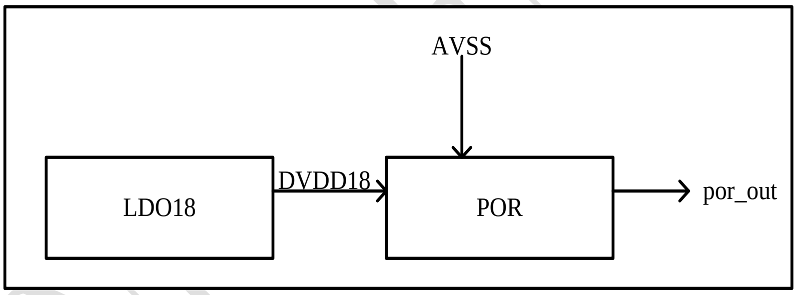


Figure 25 CJC6822 POR circuit block diagram

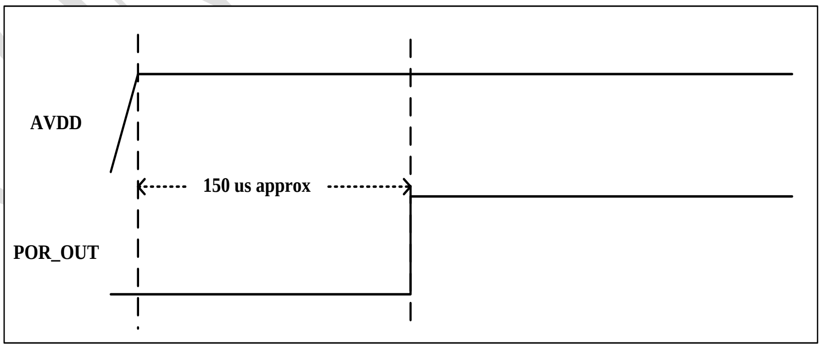


Figure 26 CJC6822 POR timing

5.20 Power control unit

5.20.1 Power supply

Figure 27 is the CJC6822 chip power pad and power supply diagram. The power supply include three parts: 3.3V supply for CJC6822 analog circuit, 3.3V power supply for I/O and 1.8V power pad from internal LDO. 3.3V power supply for analog circuit have 3 pairs power/ground pin, one pair is for USB PHY, another is for PLL which need stable and “clear” power supply to improve jitter and accurate performance, the other is for other analog module in CJC6822 chip such as LDO, APU, SARADC, LVR controller analog circuit. 3.3V power supply for I/O includes 1 pairs. CJC6822 has a internal LDO, which transfer 3.3V power to 1.8V power, LDO output power supply for CJC6822 digital logic and USB PHY digital logic also, a 1.8V pin is output to connect capacitor for decoupling.

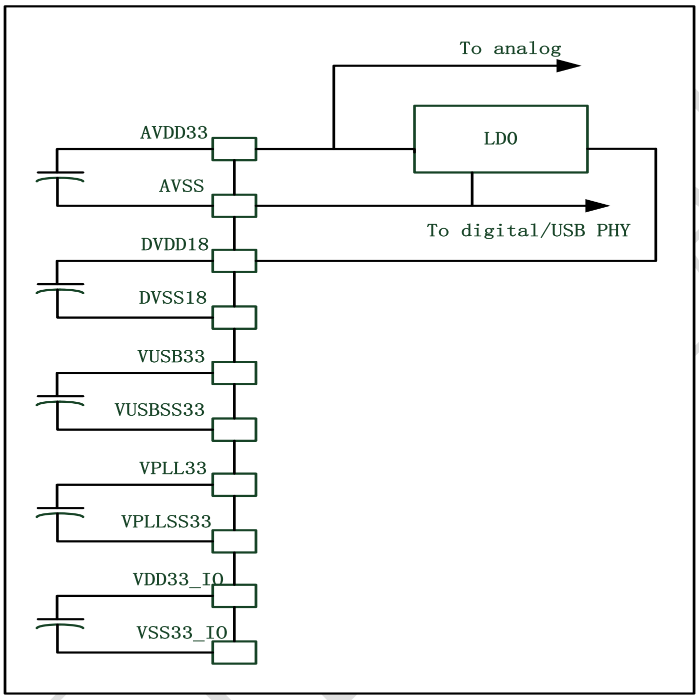


Figure 27 CJC6822 power supply diagram

5.20.2 LVR

CJC6822 has a low voltage reset generation circuit (LVR). The main circuit of LVR is comparator, it comparator the supply voltage with the configured threshold. If the supply is lower than the threshold, reset will be generated and send to all others module, then CJC6822 will enter into reset state.

5.20.3 Register control

Table 16 Power unit register list (BaseAddr = 0x4001_0000)

0x08	R2	R/W	0x0	[2:1]	lvr_in : config the threshold voltage for low voltage reset b00 : 2.0V b01 : 2.4V b10 : 2.7V

					b11 : 3V
				[0]	lvr_en : enable low voltage reset

5.20.4 Power Saving Mode

CJC6822 has the low power management mode that can help reducing power consumption when the device does not require intensive CPU resources and speed. There are one low power modes available: SLEEP mode, as showed in Figure 28.

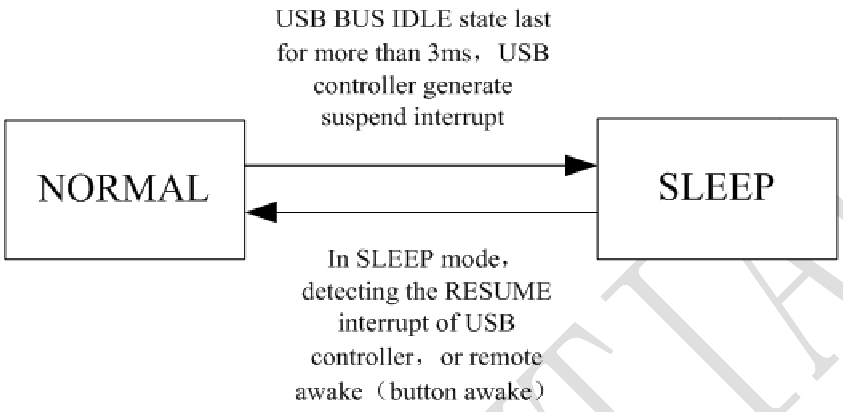

USB BUS IDLE state last for more than 3ms， USB controller generate suspend interrupt
NORMAL
SLEEP
In SLEEP mode, detecting the RESUME interrupt of USB controller， or remote awake（button awake）

Figure 28 CJC6822 power saving mode diagram

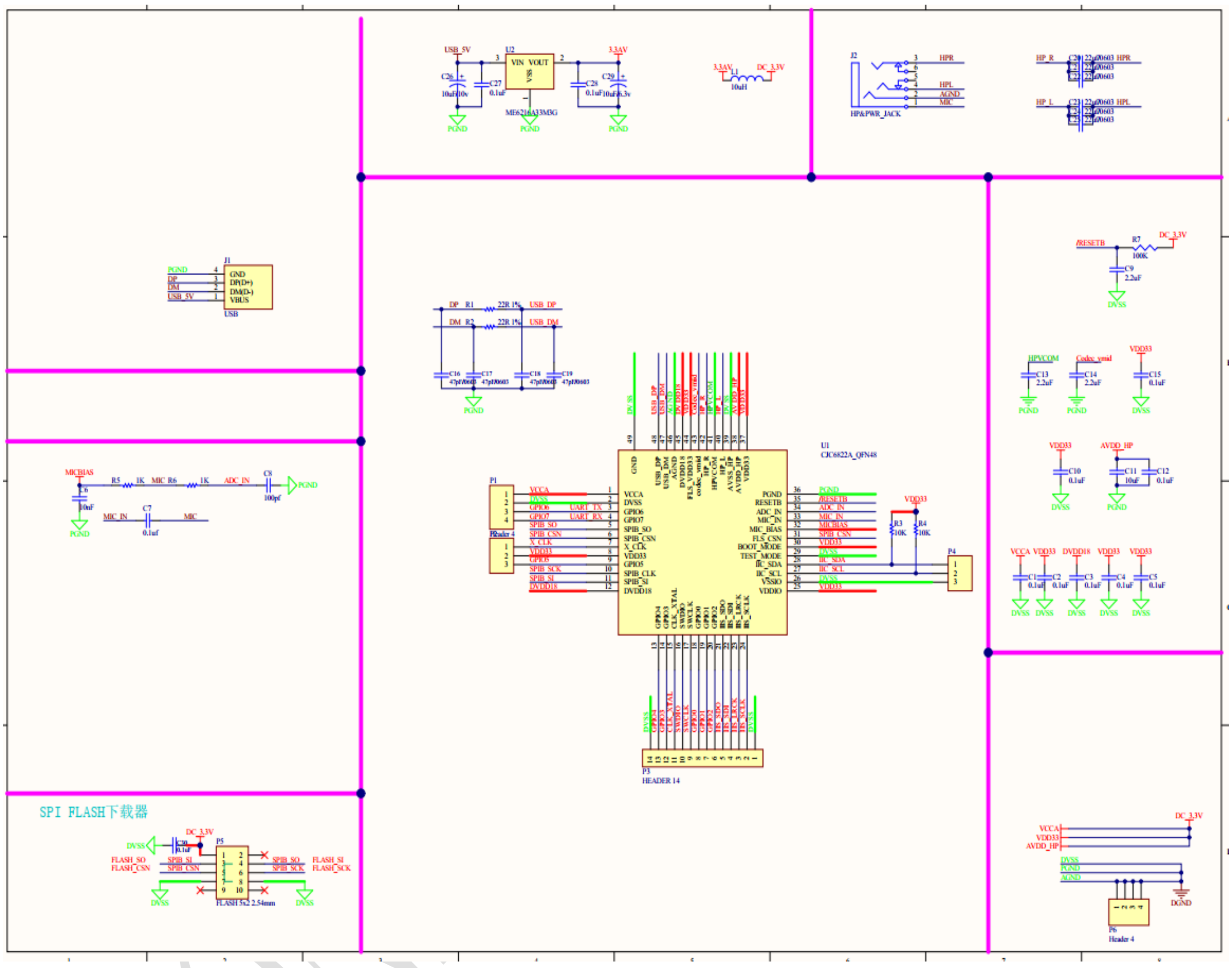
i) Entry into SLEEP mode: When operating as a peripheral, the USB controller monitors the activity on the USB and when no activity has occurred for 3 ms, it goes into SLEEP mode. If the Suspend interrupt has been enabled, an interrupt will be generated at this time. The SUSPENDM output will also go low (If the Enable SuspendM bit is set). Users need to do the following thing: Write "1" to sys_pll_pdn and audio_pll_pdn registers to power down the system PLL and audio PLL;Set the codec power management register to power down the codec circuit; Change the system clock to internal low-power 10KHZ oscillator .

(ii) When Resume signaling occurs on the bus, first the clock to the USB controller must be restarted if necessary. Then the USB controller will automatically exit SLEEP mode. If the Resume interrupt is enabled, an interrupt will be generated. Then users can reconfigure the registers to power on the system PLL , audio PLL, codec circuit and change the system clock to 48MHz.

(iii) Initiating a Remote Wakeup. If the SARADC receive a signal from the button of earphone, CJC6822 should write to the Power register to set the Resume bit to '1'. The software should leave this bit set for approximately 10 ms before resetting it to 0. By this time the hub should have taken over driving Resume signaling on the USB. Then the USB controller will exit SLEEP mode . Users can reconfigure the registers to power on the system PLL , audio PLL, codec circuit and change the system clock to 48MHz.

Applications information

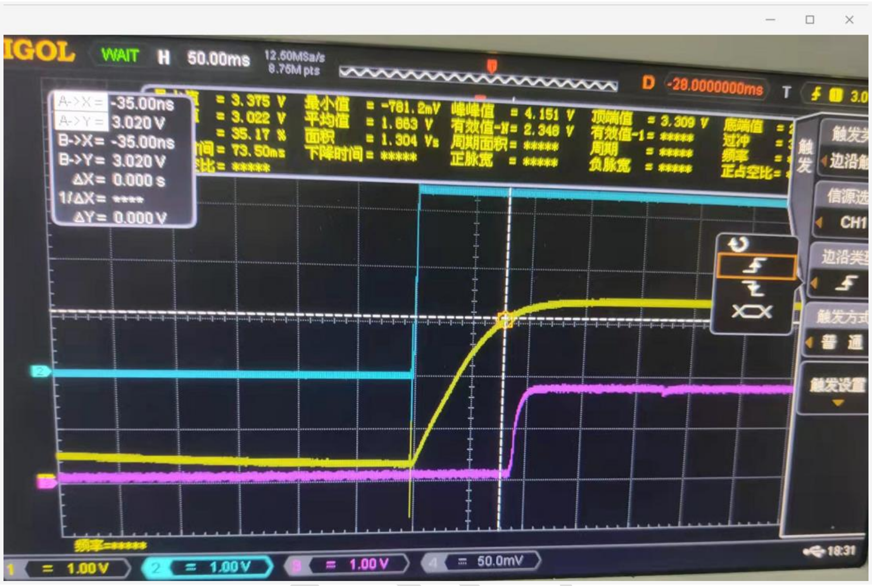
Recommended



5.20.5 Power on sequence

Reference RESETB circuit (2.2 UF + 200K circuit as a reference, Blue Line for VDD33, yellow for RESETB, purple for HP, this can determine whether the MCU is working) , to ensure that power, VDD33 RESETB HP start from 0V, and power supply without Burr, Resetb to 3.3 v between 20ms-200ms, at this point after power-on MCU can work normally.

Below： DVDD33 Resetb HP_L power on sequence

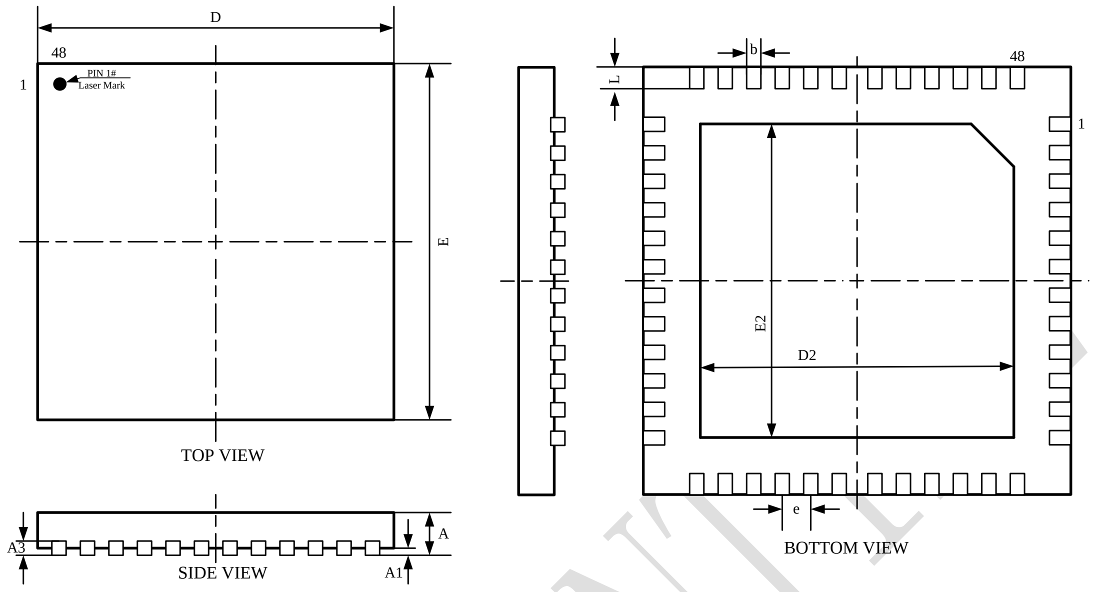


```
IGOL   WAIT   H 50.00ms   12.50MSa/s 8.75M pts      D -28.0000000ms   T 3.0

A->X= -35.00ns      = 3.375 V    最小值    = -781.2mV   峰峰值     = 4.151 V   顶端值     = 3.309 V   底端值
A->Y= 3.020 V       = 3.022 V    平均值    = 1.863 V    有效值-N=  2.348 V   有效值-1=  *****     过冲
B->X= -35.00ns      = 35.17 %    面积      = 1.304 Vs   周期面积=  *****     周期       = *****     频率
B->Y= 3.020 V   间= 73.50ms      下降时间 = *****      正脉宽     = *****     负脉宽     = *****     正占空比=
 ΔX= 0.000 s    比= *****
1/ΔX= ****
 ΔY= 0.000 V

频率=*****

1 = 1.00 V    2 = 1.00 V    3 = 1.00 V    4 = 50.0mV    18:31
```

When communicating with the system, our chips are either more stable than the system (Powered by Stanby) or more stable than the system (increasing the LDO or timing the sound card)

6 .PACKAGE DIMENSIONS

48 PIN COL QFN PLASTIC PACKAGE 6×6mm BODY



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.75	0.85	1.00
A1	0.00	0.02	0.05
A3		0.20REF	
b	0.15	0.20	0.25
D/E		6.00BSC	
D2/E2	4.15	4.40	4.65
e		0.40BSC	
L	0.30	0.40	0.50