产品名称	*HR_C7000*
文档描述	用户使用手册
密　　级	内部公开

HR_C7000 用户手册

拟制	蔡锦恩+24536	日期	2016-08-03
审核人	姓名+工号	日期	yyyy-mm-dd
批准	姓名+工号	日期	yyyy-mm-dd

浙江大华技术股份有限公司

格式编排规定：

（1）编写文档时，请注意“（）”中的提示信息及其他斜体文字，“XXX”表示该部分可由作者自定义。

（2）如果作者认为当前章节有扩充的必要，可以增加子章节，并自拟题目。

（3）请务必遵守文档模版中的样式信息。

（4）目录中必须包含：标题 1，标题 2，标题 3；标题 4 是否包含在目录中由作者决定。

（5）请注意保持图形、表格风格一致（如：不同图形中模块方框的大小基本均匀，线条和图形的基本色为黑色，文字的基本字体字号一致）。

文档修改记录

版本号	修改日期	修订内容	修订人
1.0	2016-08-03	创建文档	蔡锦恩
1.1	2016-12-26	增加焊接温度说明	蔡锦恩
1.2	2017-1-11	增加芯片低功耗流程说明	蔡锦恩
2.0	2017-4-10	更新 PinMap 和 RAM 大小	蔡锦恩
2.1	2017-4-19	增加 IO 参数说明	蔡锦恩
2.2	2017-7-10	修改 IO 复用关系，修改时钟和复位相关描述，解决 Fullmask 版本与 MPW 版本的差异	谢国军

目录

HR_C7000 用户手册 1

1 特征功能 8

1.1 总体框图 9
1.2 CPU 处理器 9
1.2.1 概述 9
1.2.2 特点 10
1.2.3 地址映射关系 10
1.2.4 JTAG 调试 11
1.3 功耗分布和控制 12
1.4 芯片封装 14
1.5 PINMAP 17
1.6 管脚说明 17
1.7 软件开发工具 23

2 芯片配置 23

2.1 芯片启动 23
2.2 时钟配置 24
CLK_MGR_REG0x04 27
CLK_MGR_REG0x08 28
CLK_MGR_REG0x0c 29
CLK_MGR_REG0x10 29
CLK_MGR_REG0x14 29
CLK_MGR_REG0x18 30
CLK_MGR_REG0x1c 30
CLK_MGR_REG0x20 30
CLK_MGR_REG0x24 30
CLK_MGR_REG0x28 31
CLK_MGR_REG0x2c 31
2.3 复位管理 32
SYS_SOFT_RSTN 33
2.4 管脚复用 33
IO_DIPLEX0 36
IO_DIPLEX1 38
IO_DIPLEX2 40

3 参数说明 42

3.1 电压参数 42
3.2 时钟范围 43
3.3 温度参数 44
3.4 指标参数说明 44
3.5 IO 参数说明 46

4 外设使用 47

4.1 SPI 接口 47
4.1.1 概述 47
4.1.2 功能描述 47
4.1.3 工作方式 48
4.1.4 寄存器概述 51
4.2 I80 接口 52
4.2.1 概述 52
4.2.2 功能描述 52
4.2.3 工作方式 52
4.2.4 寄存器概述 52
4.3 UART 接口 53
4.3.1 概述 53
4.3.2 功能描述 53
4.3.3 寄存器概述 53
4.4 I2C 接口 54
4.4.1 概述 54
4.4.2 功能描述 54
4.4.3 工作方式 55
4.4.4 寄存器概述 55
4.5 PWM 接口 56
4.5.1 概述 56
4.5.2 功能描述 56
4.5.3 工作方式 57
4.5.4 寄存器概述 57
4.6 SDIO 接口 57
4.6.1 概述 57
4.6.2 功能描述 58
4.6.3 工作方式 58
4.6.4 寄存器概述 67
4.7 GPIO 68
4.7.1 概述 68
4.7.2 功能描述 68
4.7.3 工作方式 68
4.7.4 寄存器概述 68
4.8 TIMER 69
4.8.1 概述 69

4.8.2 功能描述 69
4.8.3 工作方式 69
4.8.4 寄存器概述 69
4.9 中断（PIC） 70
4.9.1 概述 70
4.9.2 功能描述 70
4.9.4 工作方式 72
4.9.4 寄存器概述 72
4.10 WATCHDOG 73
4.10.1 概述 73
4.10.2 功能描述 73
4.10.3 工作方式 73
4.10.4 寄存器概述 74
4.11 USB 75
4.11.1 概述 75
4.11.2 功能描述 75
4.11.3 工作方式 75
4.11.4 寄存器概述 78
4.12 RTC 79
4.12.1 概述 79
4.12.2 功能描述 80
4.12.3 工作方式 80
4.12.3.1 RTC 初始化 80
4.12.3.2 中断处理 80
4.12.4 寄存器概述 81
4.13 ADC 81
4.13.1 概述 81
4.13.2 功能描述 81
4.13.3 工作方式 82
4.13.3.1 单次扫描处理流程 82
4.13.4 寄存器概述 87
4.14 DAC 88
4.14.1 概述 88
4.14.2 功能描述 88
4.14.3 工作方式 88
4.14.4 寄存器概述 88
4.15 CODEC 89
4.15.1 概述 89
4.15.2 功能描述 89
4.15.3 工作方式 90
4.15.4 寄存器概述 91

5 基带接口使用 92

5.1 发送 DAC 92
5.1.1 两点调制 92
5.1.2 单点调制 94
5.2 接收 ADC 94
5.2.1 AF 接收 94
5.2.2 中频接收 95
5.3 数字接收 96

关键字： DMR，用户手册，HR_C7000

缩略语：

缩略语	英文含义	中文解释
DMR	Digital Mobile Radio	数字移动无线电
TDMA	Time Division Multiple Access	时分多址

1 特征功能

- CPU
  - 集成 CK803S 内核，最高主频 192MHz
  - 内置 SRAM 320 Kbyte
  - 最高支持外挂 16M bit nor Flash
  - 精简指令集处理器架构（RISC）
  - 32 位数据、16/32 位混合编码指令
  - 集成 4KB 指令 Cache
  - 3 级流水线
- 丰富的外设接口
  - 3 组 SPI 接口可选，支持主从模式；其中 NFC SPI 固定用于 nor Flash 接口，支持 QualSPI Flash
  - 1 组 I8080 接口
  - 3 组 I2C 接口
  - 4 组 UART 接口
  - 3 组 PWM 接口
  - 1 组 USB 接口，支持 USB1.1 标准
  - 1 组 SDIO 标准接口，支持单线和 4 线模式
  - 内置 RTC 控制电路，支持 32.768K 晶体
  - 内置 8 路 ADC，10bit 位宽
  - 内置 3 路 DAC，12bit 位宽
- DMR
  - 符合 ETSI TS102 361（DMR） Tier I/II/III 标准的协议设计
  - 支持物理层、数据链路层和呼叫控制层独立控制
  - 支持真双时隙同步头检测
  - 采用 TDMA 技术，支持全双工、半双工语音、数据通信及数话同传业务
  - 支持 IP 数据业务
  - 支持单频、双频中继
  - 支持 4.8Kbps 和 9.6Kbps 数据传输
  - 支持数字模拟智能检测
  - 支持中继语音和数据功能
  - 支持语音加密功能
- 调制解调及信道编解码
  - 高性能 4FSK 调制解调
  - 集成协议规定的信道编解码器
- 声码器支持
  - 支持 HR_V3000(宏睿 AMBE+2)、SELP 声码器（清华）、AVDS 声码器（712）等 SPI 接口的声码器，同时为数字录音、回放及提示音输入提供接口
  - 支持数字话音加密
- 射频接口
  - 发送射频接口采用单端输出，模拟两点调制和数字 AK2401 接口
  - 接收射频接口采用差分输入方式，支持基带 IQ、中频和 AF，支持 AK2401 数字接口
  - 发送两路信号偏置、幅度大小可独立调节

- 支持用户配置 GPIO 控制射频通道

- 模拟 FM
  - 支持 12.5KHz/25KHz 信道通信
  - 支持加重、去加重
  - 支持压缩、解压缩
  - 支持 CDCSS/CTCSS 亚音处理
  - 支持 2-tone/5-tone 处理
  - 支持 DTMF 处理
  - 支持模拟静噪功能
- 内置高性能 IP
  - 高性能 ADC/DAC
  - LDO，采用 3.3V 供电，单一电源设计
  - 高性能 PLL
  - 高性能 Codec，支持差分或单端 Mic 输入和 Line_out 输出

1.1 总体框图

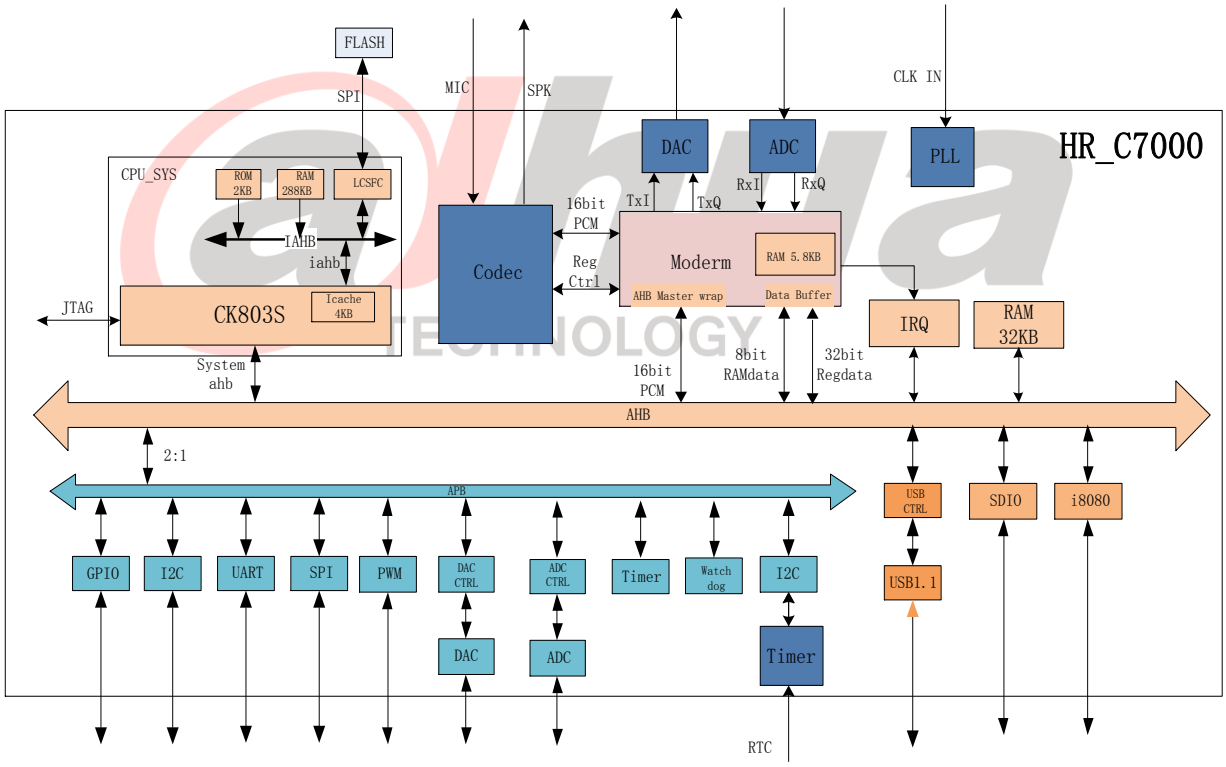


图 1.1 芯片总体框图

1.2 CPU 处理器

1.2.1 概述

HR_C7000 处理器采用中天 CK803S，32 位地址与数据通路，AHB 接口，工作频率最高为 192M。CK803S 是面向控制领域的 32 位高效能嵌入式 CPU 核，具有低成本、低功耗、高代码密度的特点。CK803S 采用 16/32 位混合编码指令系统，具有精简高效的 3 级流水线。同时系统集成片上 RAM 供 CPU 进行使用，SRAM 容量为 288KB。整体框架如下图：

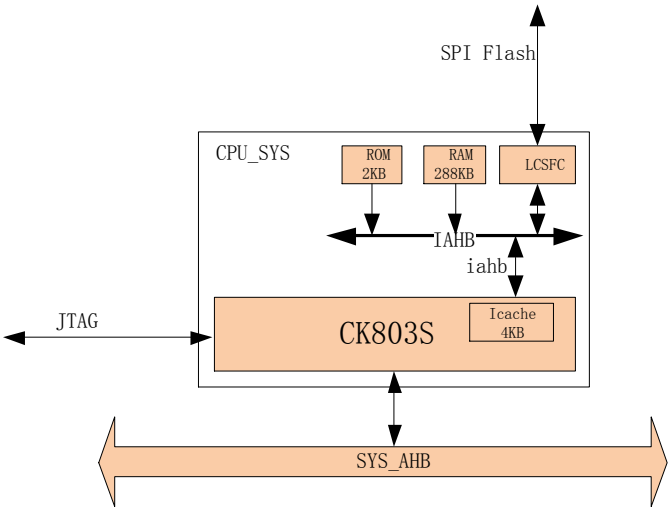


图 1.2 CPU 系统整体框图

1.2.2 特点

- 精简指令集处理器架构（RISC）
- 32 位数据、16/32 位混合编码指令
- 16 个 32 位通用寄存器
- 集成 4KB 指令 Cache
- 3 级流水线
- 按序发射、按序执行、按序退休
- 可配置的多总线接口
- 支持多种处理器时钟和系统时钟比

1.2.3 地址映射关系

HR_C7000 地址映射关系如下图：

表 1 地址映射关系

起始地址	结束地址	功能	大小	说明
0x0000_0000	0x0000_FFFF	BOOTROM 存储空间	2KB	
0x0001_0000	0x0005_7FFF	IRAM 存储空间	288KB	
0x0300_0000	0x03FF_FFFF	LCSFC 地址空间	16MB	Flash 运行空间
0x1100_0000	0x1100_FFFF	Modem	64KB	
0x1200_0000	0x1200_FFFF	i8080	64KB	
0x1300_0000	0x1303_FFFF	USB	256KB	
0x1400_0000	0x1400_FFFF	TIMER	64KB	
0x1401_0000	0x1401_FFFF	WDG	64KB	
0x1402_0000	0x1402_FFFF	GPIOA	64KB	
0x1403_0000	0x1403_FFFF	UART0	64KB	
0x1404_0000	0x1404_FFFF	UART1	64KB	
0x1405_0000	0x1405_FFFF	UART2	64KB	
0x1406_0000	0x1406_FFFF	I2C0	64KB	
0x1407_0000	0x1407_FFFF	I2C1	64KB	

0x1408_0000	0x1408_FFFF	I2C2	64KB	内部 RTC 专用
0x1409_0000	0x1409_FFFF	UART3	64KB	
0x140A_0000	0x140A_FFFF	SPI Master 0	64KB	
0x140B_0000	0x140B_FFFF	SPI Master 1	64KB	
0x140C_0000	0x140C_FFFF	PWM	64KB	
0x140D_0000	0x140D_FFFF	ADC	64KB	
0x140E_0000	0x140E_FFFF	SPI2	64KB	
0x140F_0000	0x140F_FFFF	DAC	64KB	
0x1410_0000	0x1410_FFFF	GPIOB	64KB	
0x1411_0000	0x1411_FFFF	GPIOC	64KB	
0x1412_0000	0x1412_FFFF	SPI Slave 0	64KB	
0x1413_0000	0x1413_FFFF	SPI Slave 1	64KB	
0x1414_0000	0x1414_FFFF	SPI Slave 2	64KB	
0x1415_0000	0x1415_FFFF	SPI Master 3	64KB	EFUSE 专用
0x1500_0000	0x1500_FFFF	SDIO	64KB	
0x1600_0000	0x1600_FFFF	Modem_Buffer	64KB	
0x1700_0000	0x1700_FFFF	PIC	64KB	
0x1800_0000	0x1800_7FFF	SRAM	32KB	SAHB 上的 SRAM

1.2.4 JTAG 调试

CK803S CPU 支持 JTAG 调试。CK803S JTAG 调试接口的主要特性如下：

- 使用标准的 JTAG 协议进行调试
- 非侵入式获取 CPU 状态
- 支持软断点、8 个硬断点
- 可以设置多个内存断点
- 检查和设置 CPU 寄存器的值
- 检查和改变内存的值
- 可进行指令单步执行或多步执行
- 快速下载程序
- 可在 CPU 复位之后或在普通用户模式下进入调试模式

CK803S 的调试工作是在调试软件、调试代理服务程序、调试器和调试接口的配合下完成的。调试接口在整个调试环境中的位置如下图所示。其中，调试软件和调试代理服务程序通过网络互联，调试代理服务程序和调试器通过 USB 连接，调试器与 CPU 的调试接口通过 JTAG 连接。

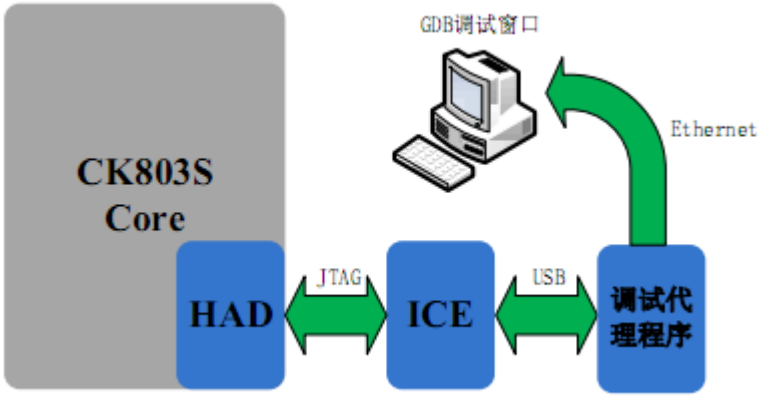


图 1.3 CK803 JTAG 调试

1.3 功耗分布和控制

HR_C7000 芯片内部功耗分布主要在以下图 1.4 所示的功能模块中。

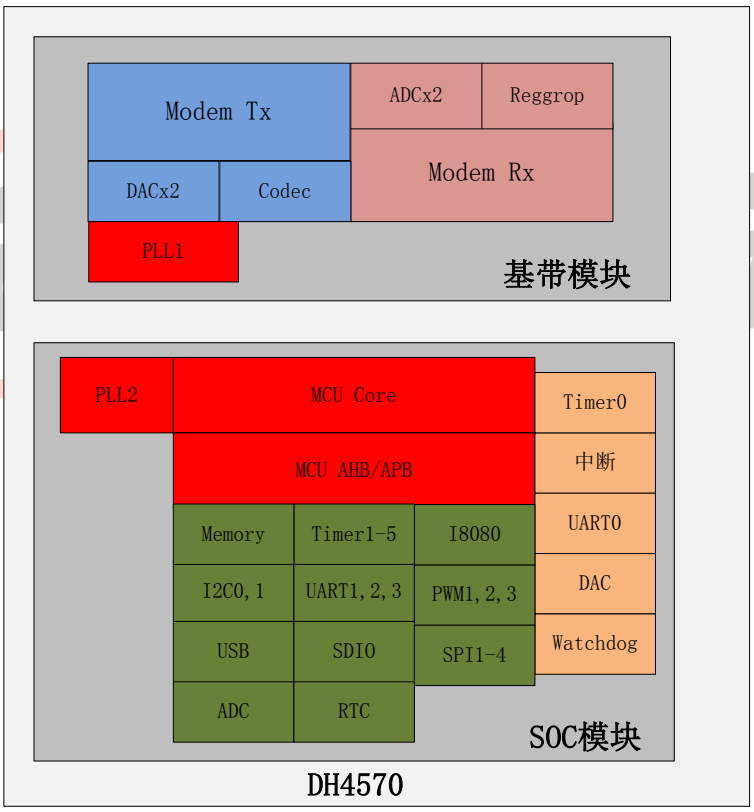


图 1.4 HR_C7000 功能模块划分

HR_C7000 主要功能模块分为两大部分：基带模块和 SOC 模块，其中基带模块包含有数字调制解调部分（ Modem Tx 和 Modem Rx，以及相关的寄存器 Reggrop）；模拟部分包括两路 ADC 和两路 DAC，以及路音频 Codec 和一个 PLL 模块（PLL1）。

SOC 模块由 PLL2 模块产生时钟，主要功能块包括 MCU Core 单元，MCU 总线（AHB/APB）以及总线的外设等部分。

以上这些模块功能在待机情况下，按照功能需要可以对部分模块进行时钟门控，按照图 1 中不同颜色区域模块，可以分别按照不同的控制方式进行处理，具体可以分成图 1.5 所示的 5 个功能类别。

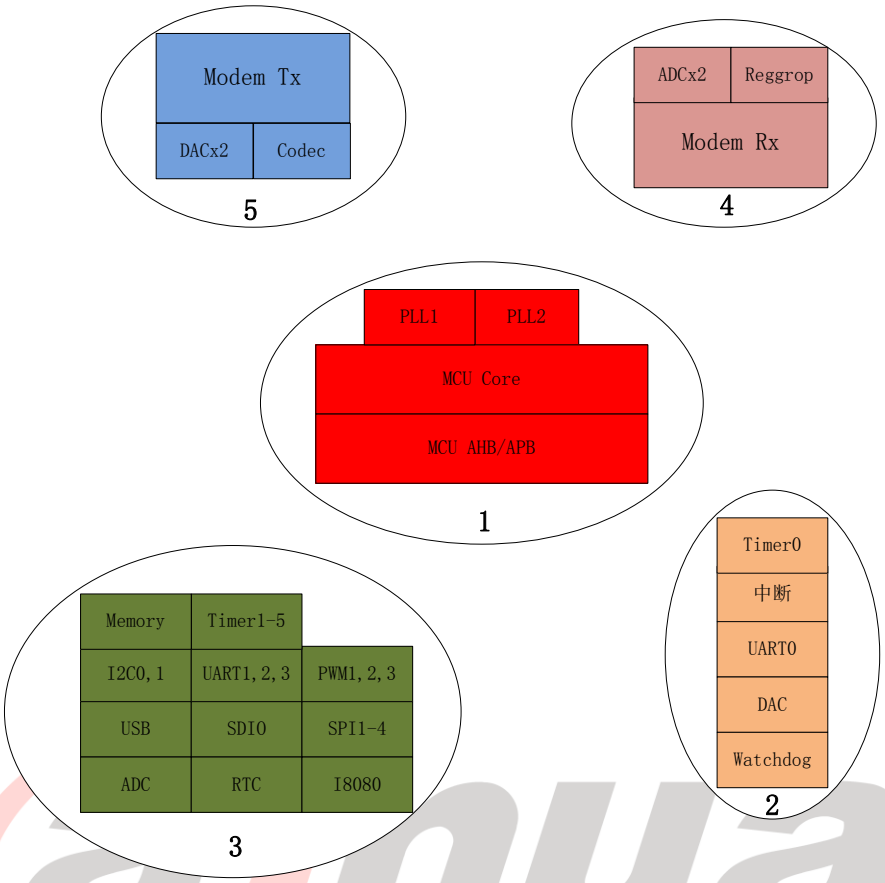


图 1.5　HR_C7000 功能功耗控制分布

在待机模式下，2 和 4 包含的模块需要开启状态，保证正常的信号和中断的接入。1 包含的模块可以工作在低时钟频率（MCUCore 和 AHB 时钟和 APB 时钟均工作在晶振时钟）。其中在中频接收方案时候，4 包含的模块仅仅需要开启一个 ADC 工作，另外一个 ADC 可以关闭。

语音正常接收过程中，根据接收开始对应的提示中断，开关 1 包括的 Codec 的 DAC 的功能；接收完成后，重新关闭 Codec，回到待机状态。

语音发送过程中，根据发送开始对应的提示中断，开启基带的 DAC 和 Codec 的 ADC 功能；传统的单点调制方案仅仅需要开启一路 DAC。发送完成后，重新关闭 Codec 和 DAC；回到待机状态。

数据发送和接收过程中，根据发送和接收开始中断，开启基带的 DAC 功能和 Codec 的 DAC 功能完成基带收发和数据收发的提示音播放；收发完成后，关闭基带 DAC 和 Codec 的 DAC，重新返回待机状态。

具体流程如图 1.6 所示。

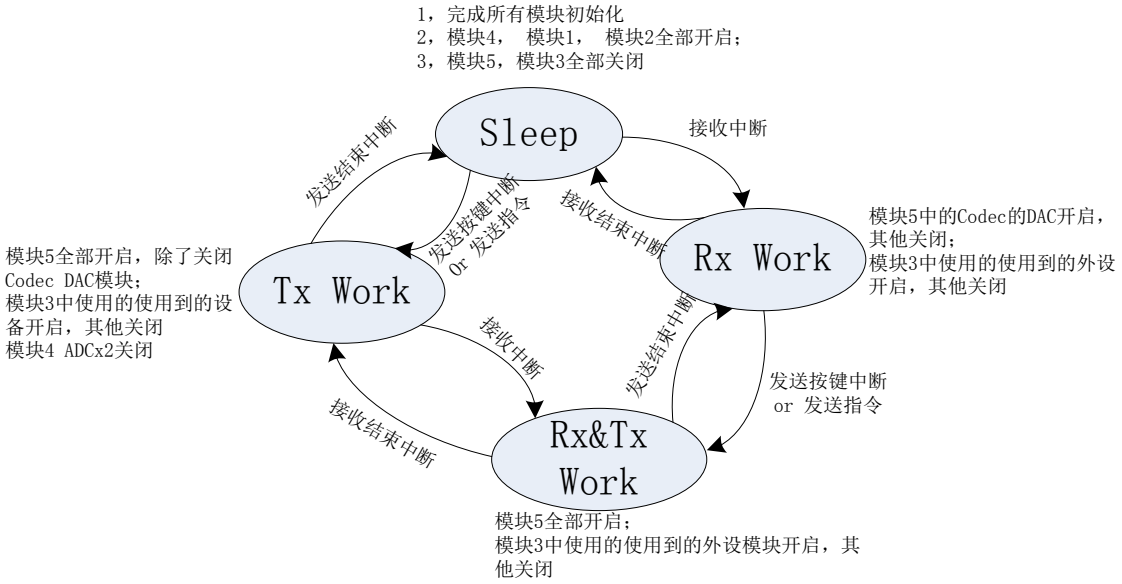
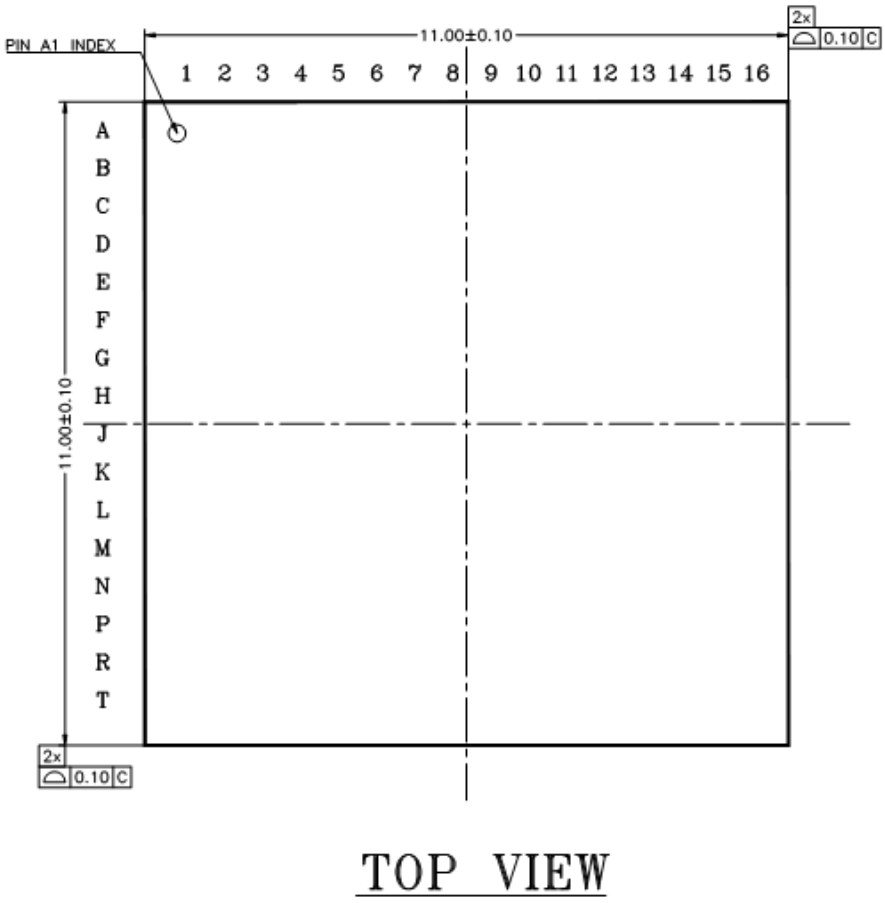


图 1.6　HR_C7000 低功耗工作流程

1.4 芯片封装

HR_C7000 采用 STFBGA 封装，封装尺寸 11mm x 11mm x1.27mm，管脚间距为 0.65mm，管脚总数为 220 个。



TOP VIEW

图 1.7 HR_C7000 芯片封装顶视图

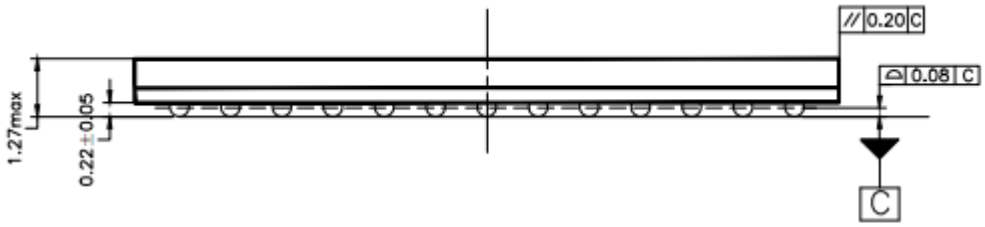


图 1.8 HR_C7000 芯片封装侧视图

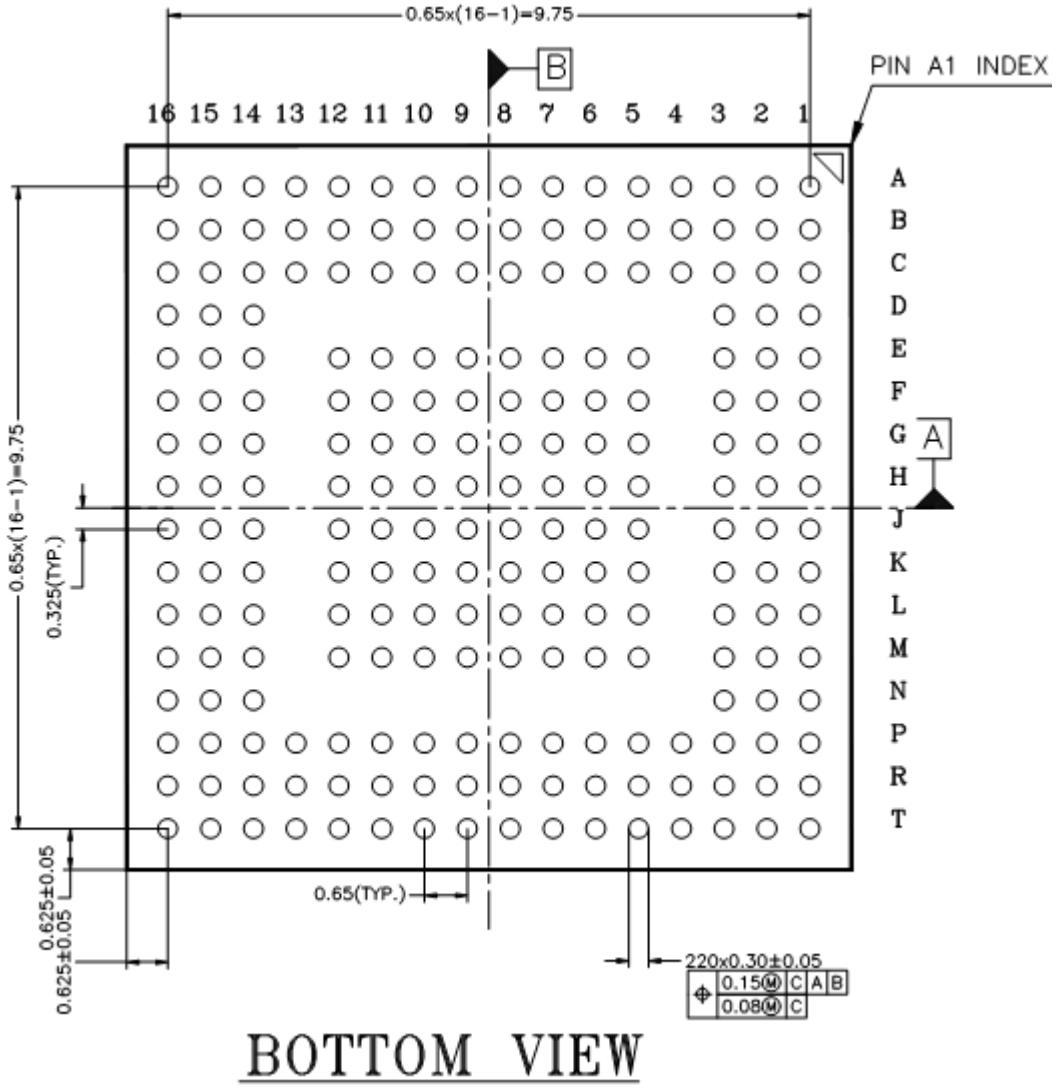


图 1.9 HR_C7000 芯片封装底视图

表 1.2 HR_C7000 芯片封装参数说明表

Symbol	Dimension in mm		
	MIN	NOM	MAX
A			1.27
A1	0.16	0.21	0.26
A2	0.91	0.96	1.01
c	0.22	0.26	0.30
D	10.90	11.0	11.10
E	10.90	11.0	11.10
D1		10.10	
E1		10.10	
e		0.65	
b	0.25	0.30	0.35
aaa	0.15		
bbb	0.10		
ddd	0.08		
eee	0.15		
fff	0.08		
MD/ME	16/16		

1.5 PinMap

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	DVSS	ADC1_IN	ADC3_IN	ADC6_IN	ADC7_IN	JTG_TDI	SPI0_MOSI	SPI0_CSN_0	I2C_SCL_1	UART1_TXD	PTB27	LCD_DB6	Codec_MIC_INP2	Codec_MIC_INP1	Codec_MICBIAS	DVSS	A
B	ADC0_IN	ADC_VRE	ADC2_IN	ADC5_IN	JTG_TCK	JTG_TDO	SPI0_MISO	SPI0_CSN_1	I2C_SDA_1	PTB29	PTB26	LCD_DB7	Codec_MIC_INN2	Codec_MIC_INN1	Codec_VCAP	Codec_VREFP	B
C	DVSS	ADC_AVS	ADC_AVD	ADC4_IN	JTG_TMS	JTG_RST_N	DVSS	SPI0_SCLK	UART1_RXD	PTB28	PTB25	PTB24	Codec_AVDD	Codec_LINOUT2_LP	Codec_LINOUT1_LP	DVDD25	C
D	RTC_CLK	RTC_CLK	LDO_RTC_	AVDD33										Codec_AVSS33	DVSS	Codec_LINOUT_VSS	D
E	DVSS	DVSS	LDO_RTC_	AVSS33	RTC_AVS	DVSS	DVSS	LDO_DVDD12_1	LDO_AVD	LDO_AVS	DVSS	DVDD33		DVSS	LCD_DB5	LCD_DB4	E
F	USB_DP	USB_DM	RTC_DVDD12		DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVDD33		LCD_DB3	LCD_DB2	LCD_DB1	F
G	DVSS	DVSS	USB_AVDD33		DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS		LCD_DB0	LCD_RD	LCD_WR	G
H	PR_G	PR_V	DVSS		DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	LDO_AVDD33_0		LCD_RS	LCD_CS	LCD_NRESET	H
J	DAC_VOU	DAC_VOU	DAC_AVDD33		DAC_VRE	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	LDO_AVSS		PWM_2	DVSS	UART2_TXD	J
K	DAC_VOU	DAC_VOU	DAC_AVSS33		DAC_VRE	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	LDO_DVDD12_0		UART2_RXD	PTB13	PTB14	K
L	DAC_VOU	DVSS	ADC_AVDD33		DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS	DVSS		PTB15	PTB16	UART3_RXD	L
M	ADC_VIN	ADC_VIN	ADC_VREF		LDO_DVDD12_3	PLL_AVD	PLL_AVSS	DVSS	DVSS	DVSS	DVDD33	DVDD33		UART3_TXD	PTB17	PWM_1	M
N	ADC_VIN	ADC_VIN	ADC_AVSS33											NFC_CSN	NFC_SCLK	NFC_MOSI	N
P	ADC_VIN	ADC_VIN	LDO_AVDD33_3	LDO_AVS	PTB20	PTB23	PTB3	PTB6	I2C_SDA_0	UART0_TXD	SPI1_MISO	BS_INTER_IN	PTB8	PTB11	NFC_MISO	NFC_HOLD	P
R	ADC_AVD	POR_EN	NRESET	PTB19	PTB21	CLK_OUT	PTB2	PTB5	I2C_SCL_0	SPI1_CSN_0	SPI1_MOSI	BS_INTER_OUT	PTB7	PTB10	PTB12	NFC_WP	R
T	DVSS	OSC_CLK_IN	TEST_MODE	PTB18	PTB22	PTB0	PTB1	PTB4	UART0_RXD	SPI1_CSN_1	SPI1_SCLK	TIME_SLOT_R_INTER	TIME_SLOT_T_INTER	PTB9	PWM_0	DVSS	T
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

图 1.10 HR_C7000 Pin Map

1.6 管脚说明

表 1.3 管脚排列说明

Ball ID	管脚名	类型	初始状态	管脚说明
B2	ADC_VREF_MCU	AP		MCU 外设接口 ADC 的参考电压。
C3	ADC_AVDD33_MCU	AP		MCU 外设接口 ADC 的供电电压，典型值 3.3V。
C2	ADC_AVSS33_MCU	AG		MCU 外设接口 ADC 的模拟地
E3	LDO_RTC_AVSS33	AG		RTC 模块的模拟地。
D3	LDO_RTC_AVDD33	AP		RTC 模块的供电电压；RTC 需单独供电，典型值 3.3V。
E5	RTC_AVSS	AG		RTC 模块的模拟地。
D1	RTC_CLK_IN	AI	Input	RTC 晶体的时钟输入，频率 32.768KHz
D2	RTC_CLK_OUT	AO	Output	RTC 晶体的时钟输出，频率 32.768KHz

F3	RTC_DVDD12	AP		RTC 电路内置 LDO 的 core1.2 电压输出。
F2	USB_DM	AIO		USB 差分数据 D-
F1	USB_DP	AIO		USB 差分数据 D+
G3	USB_AVDD33	AP		USB 电源，模拟 3.3V 供电
G2	DVSS	AG		USB 模拟地，与数字地共用
H2	PR_V	AI		保留
H1	PR_G	AG		保留
J1	DAC_VOUT_A	AI	Output	基带模拟输出信号端，通常连接单点调制信号或者两点调制信号的 Mod I
J2	DAC_VOUT_B	AI	Output	基带模拟输出信号端，通常连接单点调制信号或者两点调制信号的 Mod Q
K5	DAC_VREFN	AG		基带 DAC 和 MCU DAC 共用的参考电压负端，外部连接点模拟地
J5	DAC_VREFP	AP		基带 DAC 和 MCU DAC 共用的参考电压正端，外部连接点模拟 3.3V
J3	DAC_AVDD33	AP		基带 DAC 和 MCU DAC 共用的模拟 3.3V 电源
K3	DAC_AVSS33	AG		基带 DAC 和 MCU DAC 共用的模拟地
M3	ADC_VREF	AO	Output	基带 ADC 的模拟参考输出电压，可以通过 bg_trim 和 buffer_trim 参考配置调制输出电压，缺省值为 500mV。
N3	ADC_AVSS33	AG		基带 ADC 的模拟地
R1,L3	ADC_AVDD33	AP		基带 ADC 的模拟 3.3V 供电
N1	ADC_VINCM_Q	AI	Input	在差分模式下：Q 路输入共模电压，当 EXTCM=1 时候，有外部提供；当 EXTCM=0 时候，该管脚 floating。 在单端模式下：该管脚连接到模拟地。 在伪差分模式下：该管脚与 VINM 同输入。
M2	ADC_VINM_Q	AI	Input	在差分输入模式下，为 Q 路输入的差分负端； 在伪差分模式下，固定为 3.3V 电压输入
M1	ADC_VINP_Q	AI	Input	在差分输入模式下，为 Q 路输入的差分正端； 在为差分和单端模式下，为信号输入端
N2	ADC_VINP_I	AI	Input	在差分模式下：I 路输入共模电压，当 EXTCM=1 时候，有外部提供；当 EXTCM=0 时候，该管脚 floating。 在单端模式下：该管脚连接到模拟地。 在伪差分模式下：该管脚与 VINM 同输入。
P2	ADC_VINM_I	AI	Input	在差分输入模式下，为 I 路输入的差分负端； 在伪差分模式下，固定为 3.3V 电压输入
P1	ADC_VINCM_I	AI	Input	在差分输入模式下，为 I 路输入的差分正端； 在为差分和单端模式下，为信号输入端
M6	PLL_AVDD_1	AP		PLL 模拟 3.3V 供电
M7	PLL_AVSS_0	AG		PLL 模拟地
P3	LDO_AVDD33_3	AP		内部 LDO 的供电电压，典型值 3.3V。负责内部

				core 电压转换
M5	LDO_DVDD12_3	AP		LDO 输出 1.2 电压参考
P4	LDO_AVSS	AG		LDO 模拟地
R2	POR_EN	DI	Input	上电复位使能，该使能为 1 则芯片使用内部上电复位芯片进行复位；该使能为 0 则芯片需要外边 NRESET 进行复位。
T2	OSC_CLK_IN	DI	Input	芯片主时钟晶振输入，24MHz 固定频率要求
R6	CLK_OUT	DO	Output	内部分频时钟输出，可以通过寄存器控制位不同的 PLL 的分频时钟输出或者 RTC CLK 的分频时钟输出。
T3	TEST_MODE	DI	Input	测试模式控制使能，ATE 测试时候该管脚必须为 1。
R3	NRESET	DI	Input	外部复用管脚，当 POR_EN 为 0 是，作为外部复用使用；其他情况可以作为普通 GPIO。
N14	NFC_CSN	DO	Output	片外 Nor Flash SPI 接口平片选信号，默认主模式
N15	NFC_SCLK	DO	Output	NFC SPI 接口时钟，最高支持 96MHz。
N16	NFC_MOSI	DO	Output	串行 SPI 的数据输出；QualSPI 的数据 IO0
P15	NFC_MISO	DO	Output	串行 SPI 的数据输入；QualSPI 的数据 IO1
P16	NFC_HOLD	DO	Output	串行 SPI 的 Hold 控制信号；QualSPI 的数据 IO3
R16	NFC_WP	DO	Output	串行 SPI 的 WP 控制信号；QualSPI 的数据 IO2
T6	PTB0	DI	Input	GPIO
T7	PTB1	DI	Input	GPIO
R7	PTB2	DI	Input	GPIO
P7	PTB3	DI	Input	GPIO
T8	PTB4	DI	Input	GPIO
R8	PTB5	DI	Input	GPIO
P8	PTB6	DI	Input	GPIO
P9	I2C_SDA0	DI	Input	I2C 接口数据线
R9	I2C_SCL_0	DO	Output	I2C 接口时钟线
T9	UART0_RXD	DI	Input	UART0，可以用于调试和 boot 升级
P10	UART0_TXD	DO	Output	UART0，可以用于调试和 boot 升级
T10	SPI1_CSN_1	DO	Output	SPI1 接口片选信号
R10	SPI1_CSN_0	DO	Output	SPI1 接口片选信号
T11	SPI1_SCLK	DO	Output	SPI1 的串行时钟，最高支持 48MHz
R11	SPI1_MOSI	DO	Output	SPI1 主模式数据输出，从模式数据输入
P11	SPI1_MISO	DI	Input	SPI1 主模式数据输入，从模式数据输出
P12	BS_INTER_IN	DI	Input	BS 同步中断输入，可以用作普通 GPIO
R12	BS_INTER_OUT	DO	Output	BS 同步中断输出，可以用作普通 GPIO
T12	TIME_SLOT_R_INTER	DO	Output	基带接收时隙中断，可以用作普通 GPIO
T13	TIME_SLOT_T_INTER	DO	Output	基带发送时隙中断，可以用作普通 GPIO
R13	PTB7	DI	Input	GPIO
P13	PTB8	DI	Input	GPIO

T14	PTB9	DI	Input	GPIO
R14	PTB10	DI	Input	GPIO
P14	PTB11	DI	Input	GPIO
T15	PWM_0	DO	Output	PWM 波输出
P15	PTB12	DI	Input	GPIO
H12	LDO_AVDD33_0	AP		内部 LDO 的供电电压，典型值 3.3V。负责内部 core 电压转换
K12	LDO_DVDD12_0	AP		LDO 输出 1.2 电压参考
J12	LDO_AVSS	AG		LDO 模拟地
K15	PTB13	DI	Input	GPIO
K16	PTB14	DI	Input	GPIO
L14	PTB15	DI	Input	GPIO
L15	PTB16	DI	Input	GPIO
L16	UART3_RXD	DI	Input	UART3
M14	UART3_TXD	DO	Output	UART3
M16	PWM_1	DO	Output	PWM 波输出
M15	PTB17	DI	Input	GPIO
T4	PTB18	DI	Input	GPIO
R4	PTB19	DI	Input	GPIO
P5	PTB20	DI	Input	GPIO
R5	PTB21	DI	Input	GPIO
T5	PTB22	DI	Input	GPIO
P6	PTB23	DI	Input	GPIO
J16	UART2_RXD	DI	Input	UART2
K14	UART2_TXD	DO	Output	UART2
J14	PWM_2	DO	Output	PWM 波输出
H16	LCD_NRESET	DO	Output	LCD 复位控制信号
H15	LCD_CS	DO	Output	LCD 片选信号
H14	LCD_RS	DO	Output	寄存器选择信号
G16	LCD_WR	DO	Output	写控制信号
G15	LCD_RD	DO	Output	读控制信号
G14	LCD_DB0	DO	Output	数据线
F16	LCD_DB1	DO	Output	数据线
F15	LCD_DB2	DO	Output	数据线
F14	LCD_DB3	DO	Output	数据线
E16	LCD_DB4	DO	Output	数据线
E15	LCD_DB5	DO	Output	数据线
C15	Codec_LINOUT1_LP	DO	Output	Codec LineOut1 输出，PWM 信号
C14	Codec_LINOUT2_LP	DO	Output	Codec LineOut2 输出，PWM 信号
D16	Codec_LINOUT_VSS	AG		LineOut1 和 LineOut2 两个 IO 地
C16	DVDD25	AP		LineOut1 和 LineOut2 两个 IO 供电，电压为 2.5V，由 Codec_VREFP 提供
B16	Codec_VREFP	AP		Codec 输出 2.5V 参考电压

C13	Codec_AVDD	AP		Codec 的模拟电源 3.3V
D14	Codec_AVSS33	AG		Codec 的模拟地
B15	Codec_VCAP	AO		Codec 带隙电压
A15	Codec_MICBIAS	AO		Codec Mic 偏置电压
A14	Codec_MiC_INP1	AI	Input	Mic1 差分输入正端，或 Linein1 单端输入端
B14	Codec_MIC_INN1	AI	Input	Mic1 差分输入负端
A13	Codec_MiC_INP2	AI	Input	Mic2 差分输入正端，或 Linein2 单端输入端
B13	Codec_MIC_INN2	AI	Input	Mic2 差分输入负端
E9	LDO_AVDD33_1	AP		内部 LDO 的供电电压，典型值 3.3V。负责内部 core 电压转换
E8	LDO_DVDD12_1	AP		LDO 输出 1.2 电压参考
E10	LDO_AVSS	AG		LDO 模拟地
A12	LCD_DB6	DO	Output	数据线
B12	LCD_DB7	DO	Output	数据线
C12	PTB24	DI	Input	GPIO
C11	PTB25	DI	Input	GPIO
B11	PTB26	DI	Input	GPIO
A11	PTB27	DI	Input	GPIO
C10	PTB28	DI	Input	GPIO
B10	PTB29	DI	Input	GPIO
C9	UART1_RXD	DI	Input	UART1
A10	UART1_TXD	DO	Output	UART1
A9	I2C_SCL_1	DO	Output	I2C1 时钟
B9	I2C_SDA_1	DIO		I2C1 数据
A8	SPI1_CSN_0	DO	Output	SPI1 接口片选信号
B8	SPI1_CSN_1	DO	Output	SPI1 接口片选信号
C8	SPI1_SCLK	DO	Output	SPI1 的串行时钟，最高支持 48MHz
A7	SPI1_MOSI	DO	Output	SPI1 主模式数据输出，从模式数据输入
B7	SPI1_MISO	DI	Input	SPI1 主模式数据输入，从模式数据输出
C6	JTG_RST_N			
B6	JTG_TDO			
A6	JTG_TDI			
B5	JTG_TCK			
C5	JTG_TMS			
A5	ADC7_IN	AI	Input	MCU 模拟输入
A4	ADC6_IN	AI	Input	MCU 模拟输入
B4	ADC5_IN	AI	Input	MCU 模拟输入
C4	ADC4_IN	AI	Input	MCU 模拟输入
A3	ADC3_IN	AI	Input	MCU 模拟输入
B3	ADC2_IN	AI	Input	MCU 模拟输入
A2	ADC1_IN	AI	Input	MCU 模拟输入
B1	ADC0_IN	AI	Input	MCU 模拟输入
E12 ，	DVDD33	AP		数字电源 3.3V

F12 ， M11 ， M12				
A1, A16, C1, C7, D15, E1, E2, E6, E7, E11, E14, F5, F6, F7, F8, F9, F10, F11, G1, G5, G6, G7, G8, G9, G10, G11, G12, H3, H5, H6, H7, H8, H9, H10 H11, J6, J7, J8, J9, J10, J11,	DVSS	AG		数字地

J15, K6, K7, K8, K9, K10, K11, L2, L5, L6, L7, L8, L9, L10, L11, L12, M8, M9, M10, T1, T16				

1.7 软件开发工具

2 芯片配置

2.1 芯片启动

正常工作模式下，CPU 支持两种系统启动方式：

- 串口调试模式：复位后，CPU 判断是否有程序从串口载入，若有则将程序载入内存，否则等待直到超时后进入 Flash 启动模式。调试模式下，串口速率固定设置为 115200bps。
- Flash 启动：用户只需要将预定程序烧到 Flash 中，复位后且启动程序进入到 Flash 模式后，CPU 验证 Flash 程序的有效性。在发现 Flash 中的有效程序后，CPU 会通过 LCSFC 从 Flash 取指执行。

系统启动流程图如下图所示：

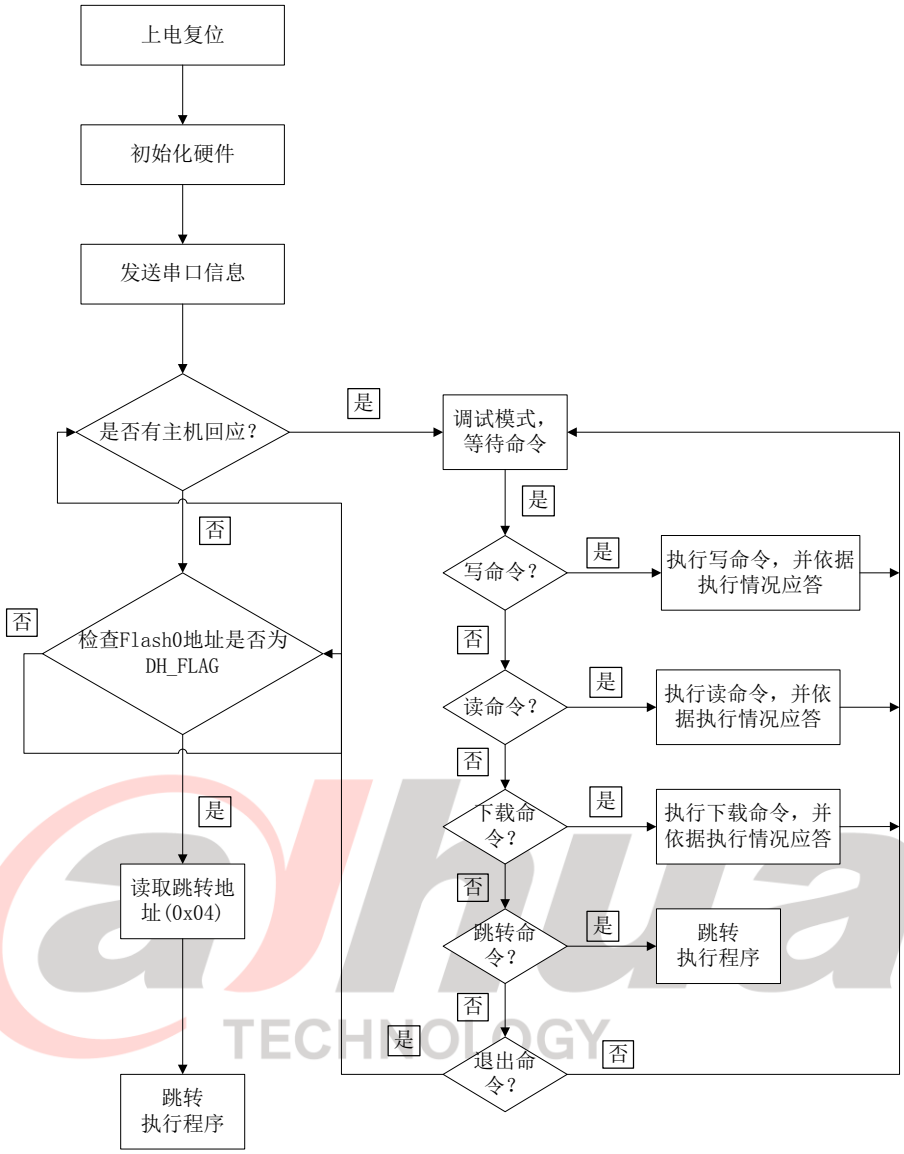


图 2.1 系统启动流

2.2 时钟配置

HR_C7000 需要外置晶振提供 24MHz 时钟工作。

芯片内置两个 PLL，其中 PLLA 的输出时钟以及它的分频时钟主要提供 Modem 模块的工作时钟；PLLB 的输出时钟主要提供 CPU 以及总线时钟和部分外设接口时钟。

Codec 时钟直接由晶振时钟提供。

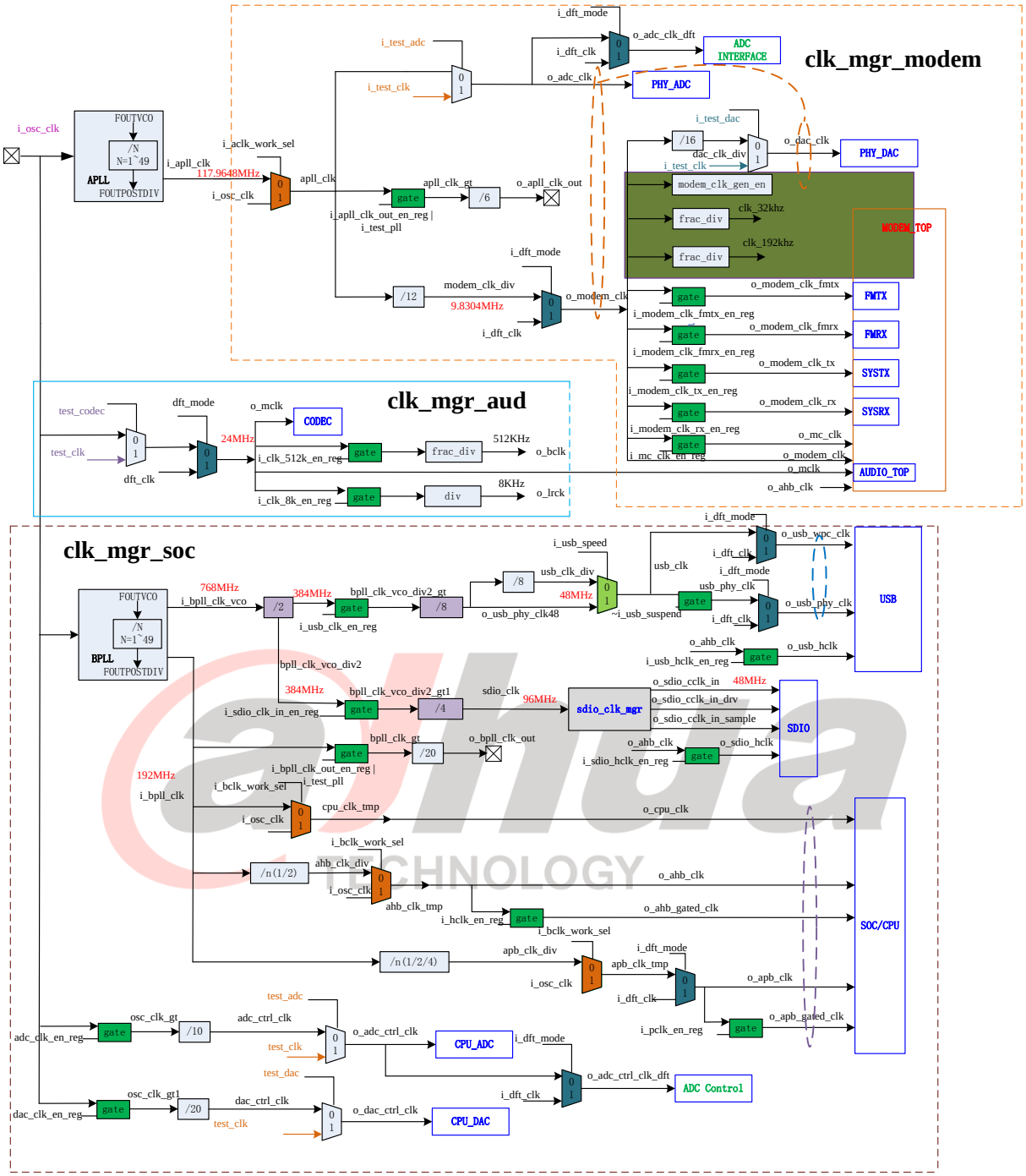


图 2.2 芯片时钟系统框图

上电初始化过程中，在 PLLA 和 PLLB 稳定输出有效时钟之前，芯片内部所有模块的工作时钟是由晶振直接提供。

如图 2.3 描述了 PLL 内部电路示意图。

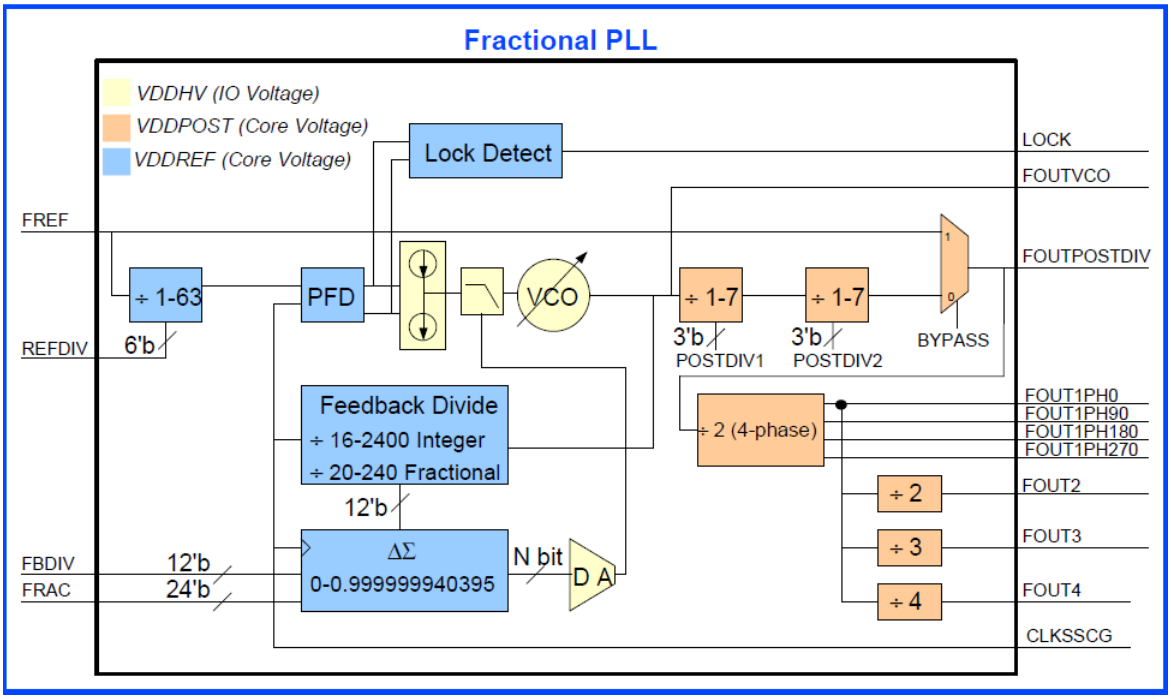


图 2.3 PLL 电路示意图

PLL输出频率计算公式如下：

整数模式（PLLB、DSMPD = 1'b1）

- FOUTVCO = FREF/REFDIV x FBDIV
- FOUTPOSTDIV = (FREF/REFDIV) x FBDIV/POSTDIV1/POSTDIV2

分数模式（PLLA、DSMPD = 1'b0）

- FOUTVCO = (FREF/REFDIV) x (FBDIV + FRAC/2^{24})
- FOUTPOSTDIV = (FREF/REFDIV) x (FBDIV + FRAC/2^{24})/POSTDIV1/POSTDIV2

如图 2.4 描述了上电时钟配置过程。

上电时钟配置工作流程：

1、 上电；

2、 CK 从 bootrom 启动；

3、 开始进行时钟配置；

4、 时钟管理模块复位后的默认配置可产生主流工作场景下的工作频率，但各个时钟的门控是关闭的，需要 CPU 通过寄存器配置将其打开，开关时钟可在线实时进行

5、 任何时候只有配置寄存器选择 PLL 输出时钟，工作时钟配置才会有效，系统起来后默认工作在 osc 时钟

6、 若偏好默认配置，只需配置寄存器 CLK_MGR_REG0x04 的 bit0 和 bit1 为 1 表示开启 PLL 配置，如希望自定义配置，则需通过 CK 一一操作相应的 PLL 分频及相应工作时钟分频系数寄存器，再配置寄存器 CLK_MGR_REG0x04 的 bit0 和 bit1 为 1 表示开启 PLL 配置，之后 PLL 进入 POWER DOWN 状态(无论 PLL 当前是否为 POWER DOWN，该动作都会被执行)

7、 芯片内部自动进行 PLL 配置过程，其中 PLL 执行 POWER DOWN 的时间约 2us；

8、 通过读寄存器 CLK_MGR_REG0x04 的 Bit31，判断两个 PLL 是否都锁定标志，由于寄存器读写时间的需求，要求 PLL 锁定标志连续判断 10 次以上才可以认定 PLL 锁定，或者之

间要求等待 100us 以后读取寄存器判断是否锁定；

9、 PLL 锁定后，开始切换系统工作时钟，从晶振时钟切换到 PLL 输出的时钟，配置寄存器 CLK_MGR_REG0x04 的 Bit3 和 Bit2 为 0；

10、 等待时钟准备 OK 标志，读取寄存器 CLK_MGR_REG0x04 的 Bit30 为高即可，由于寄存器读写时间的需求，要求 clk_rdy 标志连续判断 10 次以上才可以认定；

11、 时钟准备 OK 后，系统全局软复位操作；

12、 若要重新配置时钟频率，重新执行 3—11。

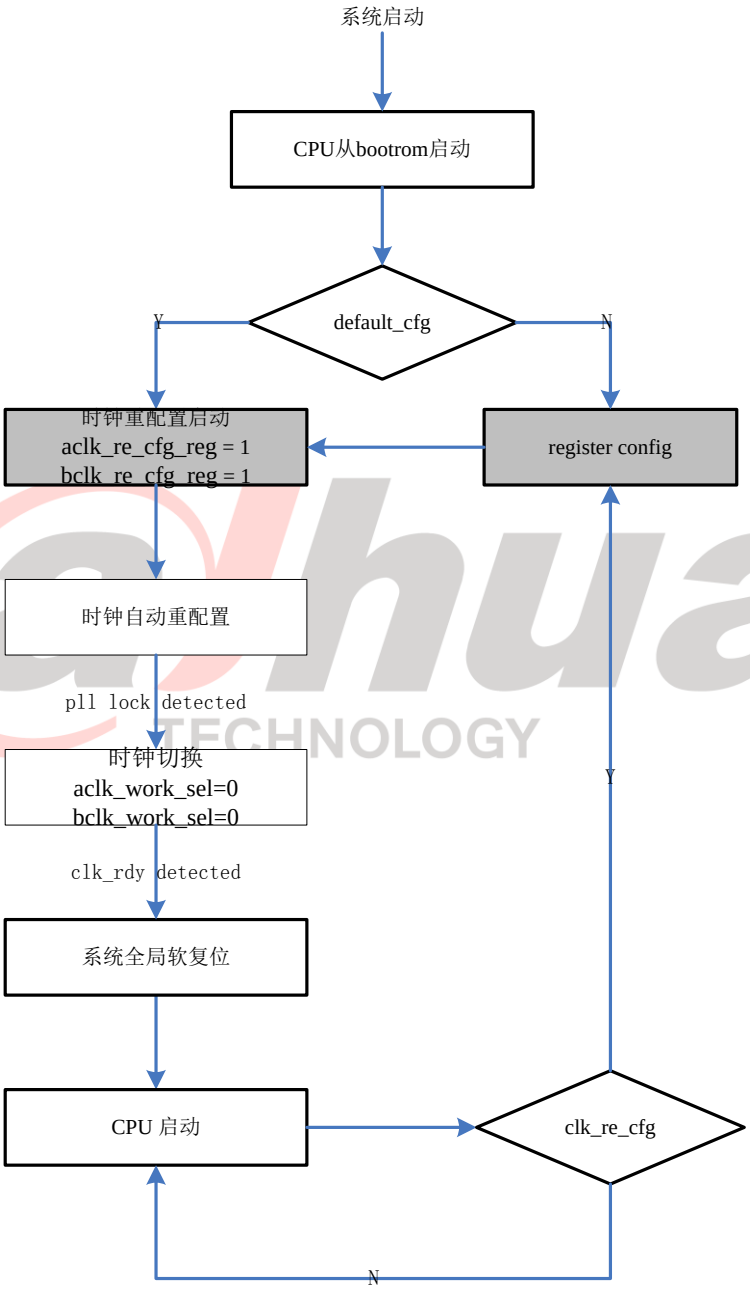


图 2.4 上电时钟配置流程图

CLK_MGR_REG0x04

Name：CLK_MGR_REG0x04

Address：0x11000000+0x4				
Bits	Access	Name	Default	Description
[31]	RO	pll_ld	0x0	总的PLL锁定标志(apll_ld&bpll_ld)
[30]	RO	clk_rdy	0x0	时钟切换准备好标志
[29:4]	-	reserved		
[3]	RW	bclk_work_sel	0x1	SOC系统时钟控制信号，复位后默认值为1。 0：SOC系统工作在PLL时钟； 1：SOC系统工作在晶振时钟。
[2]	RW	aclk_work_sel	0x1	基带系统时钟控制信号，复位后默认值为1。 0：基带系统工作在PLL时钟； 1：基带系统工作在晶振时钟。
[1]	RW	bclk_re_cfg_reg	0x0	BPLL时钟重配置信息准备好标志，由CPU发出，高有效
[0]	RW	aclk_re_cfg_reg	0x0	APLL时钟重配置信息准备好标志，由CPU发出，高有效

CLK_MGR_REG0x08

Name：CLK_MGR_REG0x08 Address：0x11000000+0x8				
Bits	Access	Name	Default	Description
[31]	RO	apll_ld	0x0	APLL 时钟输出锁定标志（系统起来后一定为1）
[30]	RO	apll_pd	0x0	APLL POWER DOWN标志： 1：power down；0：非power down。
[29]	RW	apll_dsmpd_reg	0x0	Power down Delta-Sigma Modulator： 0：DSM is active； 1：DSM is powered down。
[28]	RW	apll_dacpd_reg	0x0	Power down noise canceling DAC in FRAC mode： 0：DAC is active (default mode)； 1：DAC is not active (test mode only)。
[27]	RW	apll_bypass_reg	0x0	APLL FOUTPOSTDIV旁路到FREF，高有效
[26]	RW	apll_foutvcopd_reg	0x1	APLL post vco输出关闭使能，高有效
[25]	RW	apll_foutpostdivpd_reg	0x0	APLL输出分频控制
[24]	RW	apll_fout4phasepd_reg	0x1	4phase 时钟产生器关闭使能(4phase输出也可以由apll_foutpostdivpd关闭)，高有效
[23:18]	RW	apll_refdiv_reg	0x1	APLL参考时钟分频系数(1~63)
[17:6]	RW	apll_fbdiv_reg	0x3a	APLL feedback div value(16~2400 in

				integer mode, 20~240 in fraction mode)
[5:3]	RW	apll_postdiv1_reg	0x6	APLL post div1(1~7)
[2:0]	RW	apll_postdiv2_reg	0x2	APLL post div2(1~7)

CLK_MGR_REG0x0c

Name：CLK_MGR_REG0x0c Address：0x11000000+0xc				
Bits	Access	Name	Default	Description
[31:24]	-	reserved		
[23:0]	RW	apll_frac	0xfb7e91	APLL小数部分分频值

CLK_MGR_REG0x10

Name：CLK_MGR_REG0x10 Address：0x11000000+0x10				
Bits	Access	Name	Default	Description
[31]	RO	bpll_ld	0x0	BPLL 时钟输出锁定标志（系统起来后一定为1）
[30]	RO	bpll_pd	0x0	BPLL POWER DOWN标志： 1：power down；0：非power down。
[29]	RW	bpll_dsmpd_reg	0x1	Power down Delta-Sigma Modulator： 0：DSM is active； 1：DSM is powered down。
[28]	RW	bpll_dacpd_reg	0x1	Power down noise canceling DAC in FRAC mode： 0：DAC is active (default mode)； 1：DAC is not active (test mode only)。
[27]	RW	bpll_bypass_reg	0x0	BPLL FOUTPOSTDIV旁路到FREF，高有效
[26]	RW	bpll_foutvcopd_reg	0x0	BPLL post vco输出关闭使能，高有效
[25]	RW	bpll_foutpostdivpd_reg	0x0	BPLL输出分频控制
[24]	RW	bpll_fout4phasepd_reg	0x1	4phase 时钟产生器关闭使能(4phase输出也可以由bpll_foutpostdivpd关闭)，高有效
[23:18]	RW	bpll_refdiv_reg	0x2	BPLL参考时钟分频系数(1~63)
[17:6]	RW	bpll_fbdiv_reg	0x28	BPLL feedback div value(16~2400 in integer mode, 20~240 in fraction mode)
[5:3]	RW	bpll_postdiv1_reg	0x2	BPLL post div1(1~7)
[2:0]	RW	bpll_postdiv2_reg	0x2	BPLL post div2(1~7)

CLK_MGR_REG0x14

Name：CLK_MGR_REG0x14 Address：0x11000000+0x14				
Bits	Access	Name	Default	Description

[31:24]	-	reserved		
[23:0]	RW	bpll_frac	0x0	BPLL小数部分分频值

CLK_MGR_REG0x18

Name：CLK_MGR_REG0x18 Address：0x11000000+0x18				
Bits	Access	Name	Default	Description
[31:24]	RW	dac_clk_div_num	0x10	IP DAC工作时钟分频系数设置
[23:16]	RW	bclk_out_div_num	0xa	BPLL管脚输出时钟分频系数设置
[15:8]	RW	aclk_out_div_num	0xc	APLL管脚输出时钟分频系数设置
[7:0]	RW	modem_clk_div_num	0xc	基带工作系统时钟分频系数设置

CLK_MGR_REG0x1c

Name：CLK_MGR_REG0x1c Address：0x11000000+0x1c				
Bits	Access	Name	Default	Description
[31:27]	RW	clk_32k_div_denom	0x5	32K小数分频系数——分母
[26:16]	RW	clk_32k_div_numer	0x600	32K小数分频系数——分子
[15:11]	RW	clk_192k_div_denom	0x5	192K小数分频系数——分母
[10:0]	RW	clk_192k_div_numer	0x100	192K小数分频系数——分子

CLK_MGR_REG0x20

Name：CLK_MGR_REG0x20 Address：0x11000000+0x20				
Bits	Access	Name	Default	Description
[31]	RW	mclk_sel_reg	0x1	Codec MCLK 时钟选择： 0：PLL输出时钟分频所得； 1：晶振时钟。
[30:26]	RW	mclk_div_num	0x9	Codec MCLK所需24MHz时钟分频系数，mclk_div_num+1对应分频，时钟源为BPLL的VCO输出时钟的2分频所得。
[25]	-	reserved		
[24:13]	RW	clk_8k_div_num	0xbb8	8K分频系数
[12:9]	RW	clk_512k_div_denom	0x8	512K小数分频系数——分母
[8:0]	RW	clk_512k_div_numer	0x177	512K小数分频系数——分子

CLK_MGR_REG0x24

Name：CLK_MGR_REG0x24 Address：0x11000000+0x24				
Bits	Access	Name	Default	Description
[31:24]	-	reserved		
[23:16]	RW	cpu_dac_clk_div_num	0x18	CPU DAC工作时钟分频系数设置，0表示不分频，dac_clk_div_num+1对应分频倍数

[15:12]	-	reserved		
[11:10]	RW	sdio_clk_dly_reg_drv	0x2	SDIO的cclk_in_drv时钟延时 延时 = n/4*cclk_in，列如cclk_in=20ns，n=1时，延时=5ns，也即90° 相移
[9:8]	RW	sdio_clk_dly_reg_sample	0x0	SDIO的cclk_in_sample时钟延时 延时 = n/4*cclk_in，列如cclk_in=20ns，n=1时，延时=5ns，也即90° 相移
[7:4]	RW	adc_clk_div_num	0xa	CPU ADC工作时钟分频系数设置，0表示不分频，adc_clk_div_num+1对应分频倍数
[3:1]	RW	apb_clk_div_num	0x1	AHB和APB时钟频率关系配置寄存器： 0，AHB:APB=1:1；1，AHB:APB=1:2； 2，AHB:APB=1:4；
[0]	RW	cpu_clk_ratio_reg	0x0	AHB和CPU时钟频率关系配置寄存器： 0，AHB:CPU=1:1；1，AHB:CPU=1:2；

CLK_MGR_REG0x28

Name：CLK_MGR_REG0x28 Address：0x11000000+0x28				
Bits	Access	Name	Default	Description
[31:21]	-	reserved		
[20:16]	RW	sdio_clk_div_num	0x2	SDIO时钟的2倍时钟96MHz分频系数,sdio_clk_div_num+1对应分频
[15:5]	-	reserved		
[4:0]	RW	usb_clk_div_num	0x4	USB所需48MHz时钟分频系数,usb_clk_div_num+1对应分频

CLK_MGR_REG0x2c

Name：CLK_MGR_REG0x2c Address：0x11000000+0x2c				
Bits	Access	Name	Default	Description
[31:24]	-	reserved		
[23]	RW	dac_clk_en	0x0	CPU DAC时钟门控使能，高有效
[22]	RW	adc_clk_en	0x0	CPU低速ADC时钟门控使能，高有效
[21]	RW	hclk_en	0x0	可控AHB总线上的外设的hclk时钟门控使能，高有效
[20]	RW	pclk_en	0x0	可控APB总线上的外设的pclk时钟门控使能，高有效
[19]	RW	sdio_hclk_en	0x0	sdio hclk输入时钟门控使能，高有效
[18]	RW	sdio_clk_en	0x0	sdio时钟门控使能，高有效
[17]	RW	usb_hclk_en	0x0	usb hclk输入时钟门控使能，高有效

[16]	RW	usb_clk_en	0x0	usb时钟门控使能，高有效
[15:11]	-	reserved		
[10]	RW	mc_clk_en	0x0	Codec MC接口配置时钟使能，高有效
[9]	RW	modem_clk_rx_en	0x0	基带接收时钟门控使能，高有效
[8]	RW	modem_clk_tx_en	0x0	基带发送时钟门控使能，高有效
[7]	RW	modem_clk_fmrx_en	0x0	FM接收时钟门控使能，高有效
[6]	RW	modem_clk_fmtx_en	0x0	FM发送时钟门控使能，高有效
[5]	-	reserved		
[4]	-	reserved		
[3]	RW	clk_512k_en_reg	0x0	512K时钟门控使能，高有效
[2]	RW	clk_8k_en_reg	0x0	8K时钟门控使能，高有效
[1]	RW	bclk_out_en_reg	0x0	BPLL管脚输出时钟门控使能，高有效
[0]	RW	aclk_out_en_reg	0x0	APLL管脚输出时钟门控使能，高有效

2.3 复位管理

HR_C7000 芯片内置上电自动复位模块，在芯片管脚 POR_EN 处于高电平有效的情况下，内部自动复用模块工作，芯片被全局复位，同时外部复位管脚 NRESET 也能起到复位作用。

当芯片管脚 POR_EN 处于低电平的时候，内部自动复位电路不工作，芯片的全局复位由外部复位管脚 NRESET 完成。为保证外部复位有效，要求 NRESET 的低电平持续时间超过 400ns 以上。

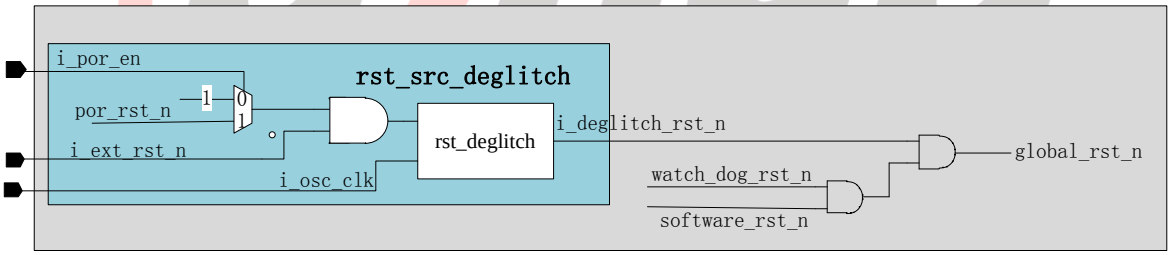


图 2.4 系统全局复位框图

芯片除了外部管脚或者内部 POR 模块进行的全局硬件复位之外，watch_dog 异常也能触发系统的全局复位。另外，系统内部设置全部软复位寄存器，也能对系统各个模块（除了寄存器配置模块之外）进行主动复位。

图 2.4 主要描述系统级的全局复位操作。在实际使用过程中，也可以通过寄存器配置对单个模块进行软件复位操作，具体的模块分布如图 2.5。其中软复位过程不能对寄存器模块进行复位。

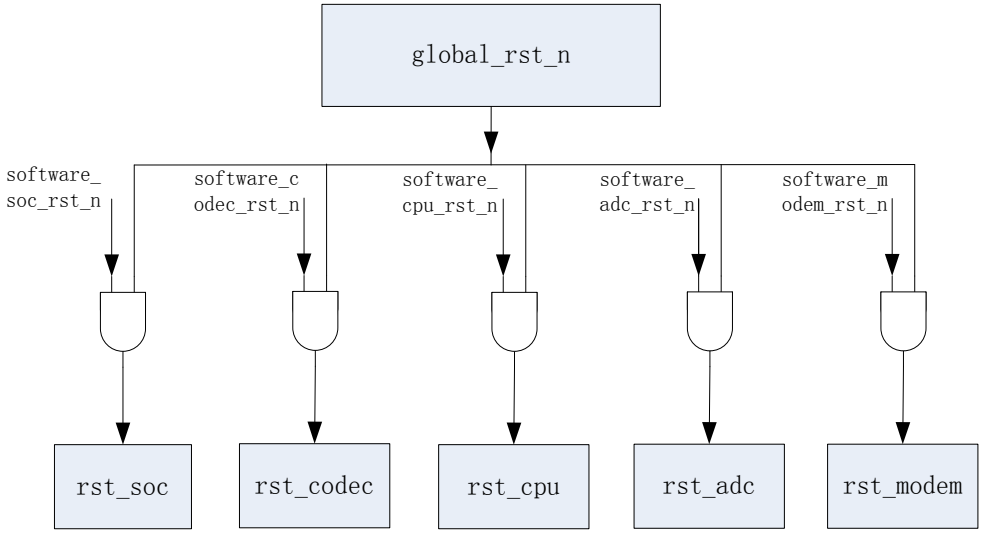


图 2.5 各个模块单独软件复位

SYS_SOFT_RSTN

Name：SYS_SOFT_RSTN Address：0x11000000+0x0				
Bits	Access	Name	Default	Description
[31:9]	-	reserved		
[8]	RW	cpu_soft_rstn	0x1	CPU 软复位控制，低有效
[7]	RW	sys_soft_rstn	0x1	系统全局软复位控制，低有效
[6]	RW	adc_ctrl_soft_rstn	0x1	内置CPU IP ADC软复位控制，低有效
[5]	RW	adc_soft_rstn	0x1	内置基带IP ADC软复位控制，低有效
[4]	RW	codec_soft_rstn	0x1	内置IP CODEC软复位控制，低有效
[3]	RW	audio_soft_rstn	0x1	音频接口模块软复位控制，低有效
[2]	RW	fm_soft_rstn	0x1	FM模块软复位控制，低有效
[1]	RW	phy_soft_rstn	0x1	物理层模块软复位控制，低有效
[0]	RW	protocol_soft_rstn	0x1	协议层模块软复位控制，低有效

2.4 管脚复用

Ball No.	Default Func	Func2	Func3	Func4	Func5
T2	OSC_CLK_IN	--	--	--	--
R6	PTC0	PLLA_CLK_OUT	PLLB_CLK_OUT	RTC_CLK	--
R3	NRESET	--	--	--	--
R2	POR_EN	--	--	--	--
T3	TEST_MODE	--	--	--	--
N14	NFC_CSN	--	--	--	--
N15	NFC_SCLK	--	--	--	--

N16	NFC_MOSI	--	--	--	--
P15	NFC_MISO	--	--	--	--
P16	NFC_HOLD	--	--	--	--
R16	NFC_WP	--	--	--	--
D1	RTC_CLK_IN	--	--	--	--
D2	RTC_CLK_OUT	--	--	--	--
F1	USB_DP	--	--	--	--
F2	USB_DM	--	--	--	--
C5	JTG_TMS	PTA0	--	--	--
B5	JTG_TCK	PTA1	--	--	--
A6	JTG_TDI	PTA2	--	--	--
B6	JTG_TDO	PTA3	--	--	--
C6	JTG_RST_N	PTA4	--	--	--
R9	PTA5	I2C_SCL_0	--	--	--
P9	PTA6	I2C_SDA_0	--	--	--
A9	PTA7	I2C_SCL_1	--	--	--
B9	PTA8	I2C_SDA_1	--	--	--
T9	UART0_RXD	--	--	--	--
P10	UART0_TXD	--	--	--	--
C9	PTA9	UART1_RXD	--	--	MODEM_SPI_CSN
A10	PTA10	UART1_TXD	--	--	MODEM_SPI_SCLK
K14	PTA11	UART2_RXD	--	--	MODEM_SPI_MOSI
J16	PTA12	UART2_TXD	--	--	MODEM_SPI_MISO
L16	PTA13	UART3_RXD	RF_TX_INTER	USB_DM	RF_RX_INTER
M14	PTA14	UART3_TXD	RF_RX_INTER	USB_DP	RF_TX_INTER
T15	PTA15	PWM_0	--	--	--
M16	PTA16	PWM_1	--	--	--
J14	PTA17	PWM_2	--	--	--
A8	PTA18	SPI0_CSN_0_M	SPI0_CSN_S	--	--
B8	PTA19	SPI0_CSN_1_M	--	--	--
C8	PTA20	SPI0_SCLK_M	SPI0_SCLK_S	--	--
A7	PTA21	SPI0_MOSI_M	SPI0_MOSI_S	--	--

B7	PTA22	SPI0_MISO_M	SPI0_MISO_S	--	--
R10	PTA23	SPI1_CSN_0_M	SPI1_CSN_S	SDIO_DAT3	--
T10	PTA24	SPI1_CSN_1_M	SYS_INTER	--	--
T11	PTA25	SPI1_SCLK_M	SPI1_SCLK_S	SDIO_CLK	--
R11	PTA26	SPI1_MOSI_M	SPI1_MOSI_S	SDIO_CMD	--
P11	PTA27	SPI1_MISO_M	SPI1_MISO_S	SDIO_DAT0	--
P12	PTA28	--	--	SDIO_DAT2	BS_INTER_IN
R12	PTA29	--	--	SDIO_DAT1	BS_INTER_OUT
T12	PTA30	--	--	SDIO_CARD_DETECT	TIME_SLOT_R_INTER
T13	PTA31	POR_RST_N	--	SDIO_CARD_CWPR	TIME_SLOT_T_INTER
H16	PTC2	LCD_NRESET	--	--	--
H15	PTC3	LCD_CS	SPI2_CSN_M	SPI2_CSN _S	--
H14	PTC4	LCD_RS	SPI2_SCLK_M	SPI2_SCLK_S	--
G16	PTC5	LCD_WR	--	--	--
G15	PTC6	LCD_RD	--	--	--
G14	PTC7	LCD_DB0	SPI2_MISO_M	SPI2_MISO_S	--
F16	PTC8	LCD_DB1	SPI2_MOSI_M	SPI2_MOSI_S	--
F15	PTC9	LCD_DB2	--	--	--
F14	PTC10	LCD_DB3	--	--	--
E16	PTC11	LCD_DB4	--	--	--
E15	PTC12	LCD_DB5	--	--	--
A12	PTC13	LCD_DB6	--	--	--
B12	PTC14	LCD_DB7	--	--	--
T6	PTB0	--	--	--	--
T7	PTB1	--	--	--	--
R7	PTB2	--	AK_DAC_FS	--	BCLK(M/S)
P7	PTB3	--	AK_DAC_SCLK	--	LRCK(M/S)
T8	PTB4	--	AK_DAC_SDI	--	CODEC_DAC_DATA
R8	PTB5	--	AK_ADC_FS	--	--
P8	PTB6	--	AK_ADC_SCLK	--	SCLIN
R13	PTB7	--	AK_ADC_SDO	--	SDAIN
P13	PTB8	--	--	--	--
T14	PTB9	--	--	--	--

R14	PTB10	--	--	--	--
P14	PTB11	--	--	--	--
R15	PTB12	--	--	--	--
K15	PTB13	--	--	--	--
K16	PTB14	--	--	--	--
L14	PTB15	--	--	--	SDAOUT
L15	PTB16	--	--	--	CODEC_ADC_DATA
M15	PTB17	--	--	--	--
T4	PTB18	--	--	--	--
R4	PTB19	--	--	--	--
P5	PTB20	--	--	--	--
R5	PTB21	--	--	--	--
T5	PTB22	--	--	--	--
P6	PTB23	--	--	--	--
C12	PTB24	--	--	--	--
C11	PTB25	--	--	--	--
B11	PTB26	--	--	--	--
A11	PTB27	--	--	--	--
C10	PTB28	--	--	--	--
B10	PTB29	--	--	--	--
B1	PTC15	ADC0_IN	--	--	--
A2	PTC16	ADC1_IN	--	--	--
B3	PTC17	ADC2_IN	--	--	--
A3	PTC18	ADC3_IN	--	--	--
C4	PTC19	ADC4_IN	--	--	--
B4	PTC20	ADC5_IN	--	--	--
A4	PTC21	ADC6_IN	--	--	--
A5	PTC22	ADC7_IN	--	--	--
K1	PTC23	DAC_VOUT_MCUA	--	--	--
K2	PTC24	DAC_VOUT_MCUB	--	--	--
L1	PTC25	DAC_VOUT_MCUC	--	--	--

IO_DIPLEX0

Name：IO_DIPLEX0 Address：0x11000000+0x34				
Bits	Access	Name	Default	Description
[31]	RW	ptb7_sel	0x1	PTB7复用选择 1：GPIO1_PTB7

				0：AK_ADC_SDO
[30:20]	-	reserved		
[19]	RW	pwm2_sel	0x1	PWM_2复用选择 1：GPIO_PTA17 0：PWM_2
[18]	RW	pwm1_sel	0x1	PWM_1复用选择 1：GPIO_PTA16 0：PWM_1
[17]	RW	pwm0_sel	0x1	PWM_0复用选择 1：GPIO_PTA15 0：PWM_0
[16:15]	RW	uart3_txd_sel	0x3	UART3_TXD复用选择 3：GPIO_PTA14 2：保留 1：基带rf_rx_inter 0：UART3_TXD
[14:13]	RW	uart3_rxd_sel	0x3	UART3_RXD复用选择 3：GPIO_PTA13 2：保留 1：基带rf_tx_inter 0：UART3_RXD
[12]	RW	uart2_txd_sel	0x1	UART2_TXD复用选择 1：GPIO_PTA12 0：UART2_TXD
[11]	RW	uart2_rxd_sel	0x1	UART2_RXD复用选择 1：GPIO_PTA11 0：UART2_RXD
[10]	RW	uart1_txd_sel	0x1	UART1_TXD复用选择 1：GPIO_PTA10 0：UART1_TXD
[9]	RW	uart1_rxd_sel	0x1	UART1_RXD复用选择 1：GPIO_PTA9 0：UART1_RXD
[8]	RW	i2c1_sda_sel	0x1	I2C_SDA_1 复用选择 1：GPIO_PTA8 0：I2C_SDA_1
[7]	RW	i2c1_scl_sel	0x1	I2C_SCL_1复用选择 1：GPIO_PTA7 0：I2C_SCL_1
[6]	RW	i2c0_sda_sel	0x1	I2C_SDA_0 复用选择 1：GPIO_PTA6 0：I2C_SDA_0
[5]	RW	i2c0_scl_sel	0x1	I2C_SCL_0复用选择

				1：GPIO_PTA5 0：I2C_SCL_0
[4]	RW	jtg_rstn_sel	0x0	JTG_RST_N复用选择 1：GPIO_PTA4 0：JTG_RST_N
[3]	RW	jtg_tdo_sel	0x0	JTG_TDO复用选择 1：GPIO_PTA3 0：JTG_TDO
[2]	RW	jtg_tdi_sel	0x0	JTG_TDI复用选择 1：GPIO_PTA2 0：JTG_TDI
[1]	RW	jtg_tck_sel	0x0	JTG_TCK复用选择 1：GPIO_PTA1 0：JTG_TCK
[0]	RW	jtg_tms_sel	0x0	JTG_TMS复用选择 1：GPIO_PTA0 0：JTG_TMS

IO_DIPLEX1

Name：IO_DIPLEX1 Address：0x11000000+0x38				
Bits	Access	Name	Default	Description
[31]	RW	ptb6_sel	0x1	PTB6复用选择 1：GPIO1_PTB6 0：AK_ADC_SCLK
[30]	RW	ptb5_sel	0x1	PTB4复用选择 1：GPIO1_PTB5 0：AK_ADC_FS
[29]	RW	ptb4_sel	0x1	PTB4复用选择 1：GPIO1_PTB4 0：AK_DAC_SDI
[28]	RW	ptb3_sel	0x1	PTB3复用选择 1：GPIO1_PTB3 0：AK_DAC_SCLK
[27]	RW	ptb2_sel	0x1	PTB2复用选择 1：GPIO1_PTB2 0：AK_DAC_FS
[26:25]	RW	time_slot_tx_sel	0x3	TIME_SLOT_T_INTER复用选择 3：GPIO_PTA31 2：SDIO sdio_card_write_prt 1：保留 0：TIME_SLOT_T_INTER
[24:23]	RW	time_slot_rx_sel	0x3	TIME_SLOT_R_INTER复用选择

				3：GPIO_PTA30 2：SDIO sdio_card_detect_n 1：保留 0：TIME_SLOT_R_INTER
[22:21]	RW	bs_inter_out_sel	0x3	BS_INTER_OUT复用选择 3：GPIO_PTA29 2：SDIO sdio_cdata[1] 1：保留 0：BS_INTER_OUT
[20:19]	RW	bs_inter_in_sel	0x3	BS_INTER_IN复用选择 3：GPIO_PTA28 2：SDIO sdio_cdata[2] 1：POR_RST_N 0：BS_INTER_IN
[18:17]	RW	spi1_miso_sel	0x3	SPI1_MISO复用选择 3：GPIO_PTA27 2：SDIO sdio_cdata[0] 1：Slave SPI1_MISO 0：Master SPI1_MISO
[16:15]	RW	spi1_mosi_sel	0x3	SPI1_MOSI复用选择 3：GPIO_PTA26 2： SDIO sdio_ccmd 1：Slave SPI1_MOSI 0：Master SPI1_MOSI
[14:13]	RW	spi1_sclk_sel	0x3	SPI1_SCLK复用选择 3：GPIO_PTA25 2：SDIO sdio_cclk_out 1：Slave SPI1_SCLK 0：Master SPI1_SCLK
[12:11]	RW	spi1_csn1_sel	0x3	SPI1_CSN_1复用选择 3：GPIO_PTA24 2： 保留 1：基带sys_inter 0：Master SPI1_CSN_1
[10:9]	RW	spi1_csn0_sel	0x3	SPI1_CSN_0复用选择 3：GPIO_PTA23 2：SDIO sdio_cdata[3] 1：Slave SPI1_CSN_0 0：Master SPI1_CSN_0
[8:7]	RW	spi0_miso_sel	0x3	SPI0_MISO复用选择 3：GPIO_PTA22 2：保留 1：Slave SPI0_MISO 0：Master SPI0_MISO

[6:5]	RW	spi0_mosi_sel	0x3	SPI0_MOSI复用选择 3：GPIO_PTA21 2： 保留 1：Slave SPI0_MOSI 0：Master SPI0_MOSI
[4:3]	RW	spi0_sclk_sel	0x3	SPI0_SCLK复用选择 3：GPIO_PTA20 2：保留 1：Slave SPI0_SCLK 0：Master SPI0_SCLK
[2]	RW	spi0_csn1_sel	0x1	SPI0_CSN_1复用选择 1：GPIO_PTA19 0：SPI0_CSN_1
[1:0]	RW	spi0_csn0_sel	0x3	SPI0_CSN_0复用选择 3：GPIO_PTA18 2：保留 1：Slave SPI0_CSN_0 0：Master SPI0_CSN_0

IO_DIPLEX2

Name：IO_DIPLEX2 Address：0x11000000+0x3c				
Bits	Access	Name	Default	Description
[31:30]	-	reserved		
[29]	RW	dac_vout_mcuc_sel	0x1	DAC_VOUT_MCUC复用选择 1：GPIO_PTC25 0：DAC_VOUT_MCUC
[28]	RW	dac_vout_mcub_sel	0x1	DAC_VOUT_MCUB复用选择 1：GPIO_PTC24 0：DAC_VOUT_MCUB
[27]	RW	dac_vout_mcua_sel	0x1	DAC_VOUT_MCUA复用选择 1：GPIO_PTC23 0：DAC_VOUT_MCUA
[26]	RW	adc7_in_sel	0x1	ADC7_IN复用选择 1：GPIO_PTC22 0：ADC7_IN
[25]	RW	adc6_in_sel	0x1	ADC6_IN复用选择 1：GPIO_PTC21 0：ADC6_IN
[24]	RW	adc5_in_sel	0x1	ADC5_IN复用选择 1：GPIO_PTC20 0：ADC5_IN
[23]	RW	adc4_in_sel	0x1	ADC4_IN复用选择

				1：GPIO_PTC19 0：ADC4_IN
[22]	RW	adc3_in_sel	0x1	ADC3_IN复用选择 1：GPIO_PTC18 0：ADC3_IN
[21]	RW	adc2_in_sel	0x1	ADC2_IN复用选择 1：GPIO_PTC17 0：ADC2_IN
[20]	RW	adc1_in_sel	0x1	ADC1_IN复用选择 1：GPIO_PTC16 0：ADC1_IN
[19]	RW	adc0_in_sel	0x1	ADC0_IN复用选择 1：GPIO_PTC15 0：ADC0_IN
[18]	RW	lcd_db7_sel	0x1	LCD_DB7复用选择 1：GPIO_PTC24 0：LCD_DB7
[17]	RW	lcd_db6_sel	0x1	LCD_DB6复用选择 1：GPIO_PTC13 0：LCD_DB6
[16]	RW	lcd_db5_sel	0x1	LCD_DB5复用选择 1：GPIO_PTC12 0：LCD_DB5
[15]	RW	lcd_db4_sel	0x1	LCD_DB4复用选择 1：GPIO_PTC11 0：LCD_DB4
[14]	RW	lcd_db3_sel	0x1	LCD_DB3复用选择 1：GPIO_PTC10 0：LCD_DB3
[13]	RW	lcd_db2_sel	0x1	LCD_DB2复用选择 1：GPIO_PTC9 0：LCD_DB2
[12:11]	RW	lcd_db1_sel	0x3	LCD_DB1复用选择 3：GPIO_PTC8 2：Slave SPI2_MOSI 1：Master SPI2_MOSI 0：LCD_DB1
[10:9]	RW	lcd_db0_sel	0x3	LCD_DB0复用选择 3：GPIO_PTC7 2：Slave SPI2_MISO 1：Master SPI2_MISO 0：LCD_DB0
[8]	RW	lcd_rd_sel	0x1	LCD_RD复用选择 1：GPIO_PTC6

				0：LCD_RD
[7]	RW	lcd_wr_sel	0x1	LCD_WR复用选择 1：GPIO_PTC5 0：LCD_WR
[6:5]	RW	lcd_rs_sel	0x3	LCD_RS复用选择 3：GPIO_PTC4 2：Slave SPI2_SCLK 1：Master SPI2_SCLK 0：LCD_RS
[4:3]	RW	lcd_cs_sel	0x3	LCD_CS复用选择 3：GPIO_PTC3 2：Slave SPI2_CS 1：Master SPI2_CS 0：LCD_CS
[2]	-	reserved		
[1:0]	RW	clk_out_sel	0x3	CLK_OUT复用选择 3：GPIO_PTC0 2：RTC_CLK 1：PLLB CLK_OUT 0：PLLA CLK_OUT

3 参数说明

3.1 电压参数

	名称	描述	MIN	TYP	MAX	单位
数字电压	DVDD33	数字电源供电	2.97	3.3	3.63	V
Codec 供电	DVDD25	Codec 的 LineOut 数字 PWM IO 电源供电	2.35	2.5	2.65	V
	Codec_VREFP	Codec 参考电压输出	2.35	2.5	2.65	V
	Codec_MICBIAS	MICBIAS_V=0		2.08V		V
		MICBIAS_V=1		1.66		V
	Codec_AVDD	Codec 模拟电源	2.97	3.3	3.63	V
	Codec_VCAP	Codec 带隙输出电压		2		V
MCU 外设 ADC 电压	ADC_AVDD33_MCU	MCU ADC 供电	2.97	3.3	3.63	V
	ADC_VREF_MCU	MCU ADC 参考电压	0.5* AVDD33		0.9* AVDD33	V
RTC 供电	LDO_RTC_AVDD33	RTC 独立 LDO 供电	2.0	3.3	3.63	V
	RTC_DVDD12	RTC 独立 LDO core 供电	1.08	1.2	1.32	V
USB 供电	USB_AVDD33	USB 供电	2.8	3.3	3.63	V
EFUSE 供电	EFUSE_DVDD25	EFUSE 供电，持续时间效应 1s	2.25	2.5	2.75	V
DAC 供电	DAC_AVDD33	DAC 模拟供电	2.97	3.3	3.63	V
	DAC_VREFP	DAC 参考电压输入	2.97	3.3	3.63	V

PHY ADC 供电	ADC_AVDD33	基带 ADC 模拟供电	3.04	3.3	3.6	V
	ADC_VREF	基带 ADC 的参考电压输出	0.485	0.5	0.515	V
PLL 供电	PLL_AVDD_1	PLL 模拟供电	2.97	3.3	3.63	V
LDO 供电	LDO_AVDD33_1	LDO 1 模拟供电	2.97	3.3	3.63	V
	LDO_AVDD33_2	LDO 2 模拟供电	2.97	3.3	3.63	V
	LDO_AVDD33_3	LDO 3 模拟供电	2.97	3.3	3.63	V
	LDO_DVDD12_1	LDO 1 转换电压输出	1.08	1.2	1.32	V
	LDO_DVDD12_1	LDO 2 转换电压输出	1.08	1.2	1.32	V
	LDO_DVDD12_1	LDO 3 转换电压输出	1.08	1.2	1.32	V
参考地	DVSS	数字地		0		V
	Codec_LINOUT_VSS	Codec LineOutIO 的模拟地		0		V
	Codec_AVSS33	Codec 模拟地		0		V
	ADC_AVSS33_MCU	MCU ADC 模拟地		0		V
	LDO_RTC_AVSS33	RTL 自带 LDO 模拟地		0		V
	RTC_AVSS	RTC 模拟地		0		V
	USB_AVSS33	USB 模拟地		0		V
	EFUSE_AVSS	EFUSE 模拟地		0		V
	DAC_AVSS33	DAC 模拟地		0		V
	DAC_VREFN	DAC 负端参考		0		V
	ADC_AVSS33	ADC 模拟地		0		V
	PLL_AVSS_0	PLL 模拟地		0		V
	LDO_AVSS	LDO 模拟地		0		V
输入输出电压门限	V_{IH}	输入高电压	2.0		3.63	V
	V_{IL}	输入低电压	-0.3		0.8	V
	V_{OL}	输出低电压			0.4	V
	V_{OH}	输出高电压	2.4			V

注：所有数字和模拟 Core 电压均有 LDO 供电，不通过 Pad 外供

3.2 时钟范围

名称	描述	MIN	TYP	MAX	单位
OSC_CLK_IN	主时钟晶振	--	24	--	MHz
Apll_clk	APLL 输出时钟	--	117.9648	--	MHz
mdem_clk	Modem 基带工作时钟	--	9.8304	--	MHz
mdem_clk_fm_tx	Modem 基带 FM 发送时钟	--	9.8304	--	MHz
mdem_clk_fm_rx	Modem 基带 FM 接收时钟	--	9.8304	--	MHz
mdem_clk_tx	Modem 基带发送时钟	--	9.8304	--	MHz
mdem_clk_rx	Modem 基带接收	--	9.8304	--	MHz
Mc_clk	Codec MC 接口时钟	--	9.8304	--	MHz
O_dac_clk	基带 DAC 工作时钟	0.0384	0.6144	1	MHz
O_adc_clk	基带 ADC 工作时钟	--	117.9648	--	MHz
Mclk	Codec 工作主时钟	--	24	--	MHz
Bpll_clk_vco	BPLL VCO 输出时钟	--	--	768	MHz

Usb_wpc_clk	Usb 工作时钟	6	48	--	MHz
Usb_phy_clk	Usb 工作时钟	6	48	--	MHz
O_usb_hclk	Usb 总线接口时钟	24	120	192	MHz
Sdio_cclk_in	Sdio 工作时钟	40	48	48	MHz
O_sdio_hclk	Sdio 总线接口时钟	24	120	192	MHz
Cpu_clk	Cpu 内核工作时钟	24	120	192	MHz
O_ahb_clk	AHB 总线时钟	24	120	192	MHz
O_apb_clk	APB 总线时钟	24	60	96	MHz
O_adc_ctrl_clk	MCU 自带 adc 工作时钟	0.2	--	2	MHz
O_dac_ctrl_clk	MCU 自带 dac 工作时钟			1	MHz

3.3 温度参数

名称	描述	MIN	TYP	MAX	单位
Oprating temp		-40	25	85	° C
Junction temp		-40	25	100	° C
Solder temp	器件可在 260C 过三次回流焊，并保持机械和电性能正常（高温区段时间约为 260° C 10s，）； 手工焊接，建议在高温条件时间尽量短，可参考一般返修条件 340° C 5~10s，		260		° C

3.4 指标参数说明

分类	参数	描述	MIN	TYP	MAX	单位
DAC	Resistive load	Anolog output		1.5		kOhm
	Capacitve load	Anolog output			30	pF
	Voltage swing	Anolog output	VREFN		VREFP	V
	RES			12		bit
	DNL			±1		LSB
	INL			±1.5		LSB
	SNR					
	THD					
	VIN_{diff}	差分输入电压（VINP-VINM）			0.9	V_{pp}
	VIN_{se}	单端或伪差分输入电压，VINM=com mode of VINP			0.45	V_{pp}
	VIN_{cm-dc}	VINCM 共模电压（DC 模式）	0.5		1.5	V
	VIN_{cm-ac}	VINCM 共模电压（AC 模式）		0		V
	Vnol	AVDD Dyamic Noise			50	mV_{pp}
	RES			10		bit
	C_{in}	VINP/VINM 输入电容		2		pF

PHY ADC	C_{ref}	VREF 参考电压电容		100		nF
	DNL			±1		LSB
	INL			±2		LSB
	Gain Error	Internal reference, before BGTRIM correction	-4		+4	%
	Gain Error	Internal reference, after BGTRIM correction	-1		+1	%
	Gain Error	External reference	-1		+1	%
	OS	Input offset error	-5		+5	mV
	LAT	Analog input to digital output latency		12		cycles
	WT	Wake up time		120		cycles
	BTime	Buffer start-up time		15	20	uS
	Total_Stime	Complete ADC start up time		WT+ BTimte		uS
	SNR			56		dB
	ENOB			9		bit
MCU ADC	Input Range	8-channel single-ended input	0.01 VREF		0.99 VREF	V
	Input capacitance			1		pF
	RES				10	bit
	DNL		-0.2852		0.3599	LSB
	INL		-0.7344		0.9160	LSB
	Offset Error			0.5064		%FS
	Gain Error			-0.0432		%FS
	Data Latency			10		cycle
	SINAD	Fin = 1.03K		57.91		dB
	SFDR	Fin = 1.03K		63.81		dB
	THD +3HD	Fin = 1.03K		-64.64		dB
	ENOB	Fin = 1.03K		9.33		bit
USB	V_{OL}	1.5K pull-up resistor on DP to USB_AVDD33			0.3	V
	V_{OH}	15K pull-down reisitor on DP/DM to AGND	2.8			V
	V_{diff}	|V(DP)-V(DM)|	0.2			V
	V_{cm}	USB_AVDD33 > V_{cm}+0.8	0.8		2.5	V
	V_{se-l}	Single-ended recever low level input voltage			0.8	V
	V_{se-h}	Single-ended recever high level input voltage	2.0			V
	Fs	Sampling frequency	8	16	192	KHz
	Mclk			24		MHz
	Duty cycle	Mclk	0.45	0.5	0.55	

Codec	WT	From power-down to playback mode		181	493	ms
		From power-down to record mode		124	230	ms
	Micbias out current				4	mA
	Micbias out noise	A-weighted		20	40	uVrms
	Micbias capacitor	Decoupling capacitor Cmic	0.75	1	1.25	nF
	Input level	ADC full scale, Gain GID=0dB, GIM=20dB	0.189	0.212	0.239	Vpp
	SNR	ADC , A-weighted,1KHz sine wave, full scale and gain 0dB	85	90		dB
	THD	ADC, 1KHz sine wave -1dB and gain 0dB		-80	-70	dB
	Dynamic range	ADC, A-weighted, 1KHz sine wave, full scale and gain 0dB	85	90		dB
	PSRR	ADC, 100mVpp 1KHz sinewave is applied to AVD, Gain GID=0dB, GIM=20dB		90		dB
	Input resisitance	ADC, GIM=0dB, differential config	132	160	200	KOhm
		ADC, GIM=20dB, differential	20	26	30	KOhm
		ADC, GIM=0dB, single-ended	92	115	138	KOhm
		ADC, GIM=20dB, single-ended	19	24	29	KOhm
	Input capacitance	ADC, Cbyline		1		uF
		ADC, includes 10uf for ESD, bongding and package pins cap			25	pF
	SNR	PWM out, A-weighted,1KHz sine wave, full scale and gain 0dB	100	105		dB
	THD	PWM out, 1KHz sine wave -1dB and gain 0dB, Fs <=16KHz		-88	-80	dB
	THD+N	PWM out, 1KHz sine wave -1dB and gain 0dB, Fs <=16KHz		-87	-80	dB
	Dynamic Range	PWM out, 1KHz sine wave, full scale -60dB and volume=0dB	100	105		dB
	Gain range	PWM out, Godl 6-bit programmable range	-31		32	dB
	Gain step	PWM out,		1		dB
	PWM frequency	MCLK =24MHz		750		kHz
	Modulation rate		25		75	%
LDO	I_{max}	Load current			100	mA
	V_{trip}	Max ripple on Vout, Vin = 3.3V			100	mV_{pp}
	C_{out}	External capacitor			4.7	uF

3.5 IO 参数说明

名称	描述	MIN	TYP	MAX	单位

V_T	阈值点	-	1.45	-	V
V_{T+}	Schmitt 触发从低到高阈值点	-	1.59	-	V
V_{T-}	Schmitt 触发从高到低阈值点	-	1.24	-	V
R_{PU}	上拉电阻	28	41	65	KOhm
R_{PD}	下拉电阻	26	41	68	KOhm
V_{IH}	输入高电压	2.0		3.63	V
V_{IL}	输入低电压	-0.3		0.8	V
V_{OL}	输出低电压			0.4	V
V_{OH}	输出高电压	2.4			V
I_{OL}	Low level ouput @ V_{OL} = 0.4V, PBx4RT	3.3	5.7	8.3	mA
	Low level ouput @ V_{OL} = 0.4V, PBx8RT	6.7	11.6	16.7	mA
I_{OH}	Low level ouput @ V_{OL} = 2.4V, PBx4RT	3.8	8.5	15.1	mA
	Low level ouput @ V_{OL} = 2.4V, PBx8RT	7.6	17.2	30.4	mA

4 外设使用

4.1 SPI 接口

4.1.1 概述

SPI控制器实现数据的串并、并串转换，可以作为Master或Slave与外部设备进行同步串行通信。

4.1.2 功能描述

4.1.2.1 功能特性

SPI 接口具有以下特点：

- 多个中断及中断屏蔽寄存器；
- 芯片有两组SPI接口，每个SPI接口支持最大2个slave；
- 支持中断源优先级可配；
- 时钟频率可编程；
- 与APB总线时钟同步；
- 数据发送/接收16bit位宽FIFO独立，FIFO深度128；
- 支持SPI全双工工作模式、支持数据帧传输极性及相位调节；
- 提供内部自回环模式测试；
- 支持4~16bit的软件可配置的SPI传输模式。

4.1.2.2 典型应用

SPI 模块连接两个 slave 设备的典型应用方式如下图所示：

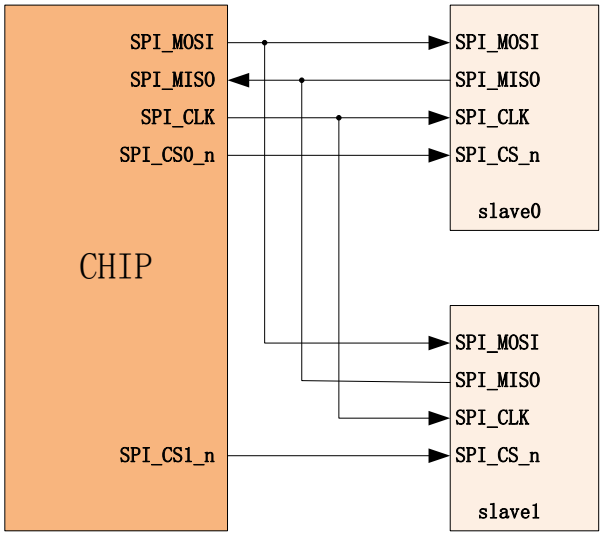


图 4.1 SPI 模块的典型连接方式

4.1.3 工作方式

4.1.3.1 传输时序

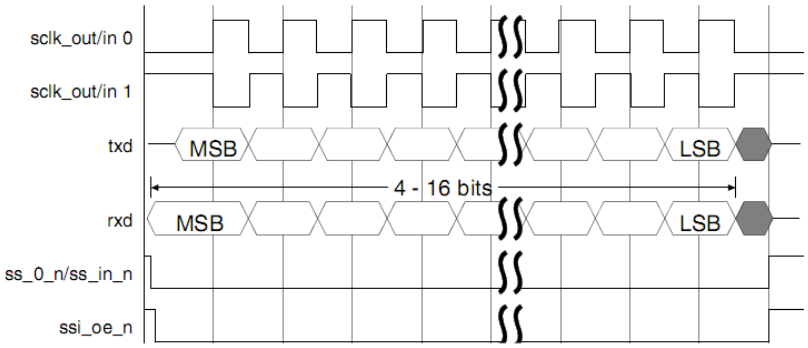


图 4.2 数据串行时序

串行数据在时钟下降沿生成，在时钟上升沿进行采样。

同一个命令执行完毕需要取消片选，下一条命名重新使能片选；同一条命令处理过程中取消片选会中断该命令。

为适应多种设备，该模块可设置数据串行传输顺序可以通过设置 FRF 寄存器实现：0，从 MSB 开始；1，从 LSB 开始。

4.1.3.2 操作流程

SPI 模块的接收 FIFO 寄存器与发送 FIFO 寄存器地址为同一个。当向该地址写入数据时，SPI 模块将数据写入发送 FIFO 中；当读取该地址数据时，SPI 模块将从接收 FIFO 中取出数据到 APB 总线。

软件启动数据传输的流程为：

(1) 禁用 SPI 模块，设置 SSIENR 寄存器为 0x0；

(2) 使能所有 slave 片选，设置 SER 寄存器为 0x0；

(3) 清除所有中断，TXOICR、RXOICR、RXUICR、MSTICR、ICR；

(4) 配置传输模式 CTRLR0[9:8]及传输数据帧长度 CTRLR0[3:0]；

(5) 配置 BAUDR 传输时钟频率及使能时钟；
(6) 设置接收 FIFO 的中断门限值 RXFTLR 及发送 FIFO 的中断门限值 TXFTLR；
(7) 设置中断使能寄存器 IMR；
(8) 设置接收数据个数 RXFLR（Receive Only 和 EEPROM Read Transfer 模式）；
(9) 使能 SPI 模块，设置 SSIENR 寄存器为 0x1；
(10) 使能相应 slave 片选，设置 SER 寄存器为相应 slave 位为 1；
(11) 使能 slave 片选后，SPI 模块自动启动传输，当接收 FIFO 中的数据＞RXFTLR 时触发 RXFIS 中断；当发送 FIFO 中的数据≤TXFTLR 是触发 TXEIS 中断。

注意：当设置数据传输帧的位宽为 X(4~32)时，软件操作数据位宽为 32bit，但 FIFO 中数据只有其中低 X 位有效。

4.1.3.3 SPI 模块 4 种传输模式

(1) Transmit & Receive 模式

Transmit & Receive 模式下，SPI 模块发送数据的同时接收数据，发送数据个数等于接收数据个数。

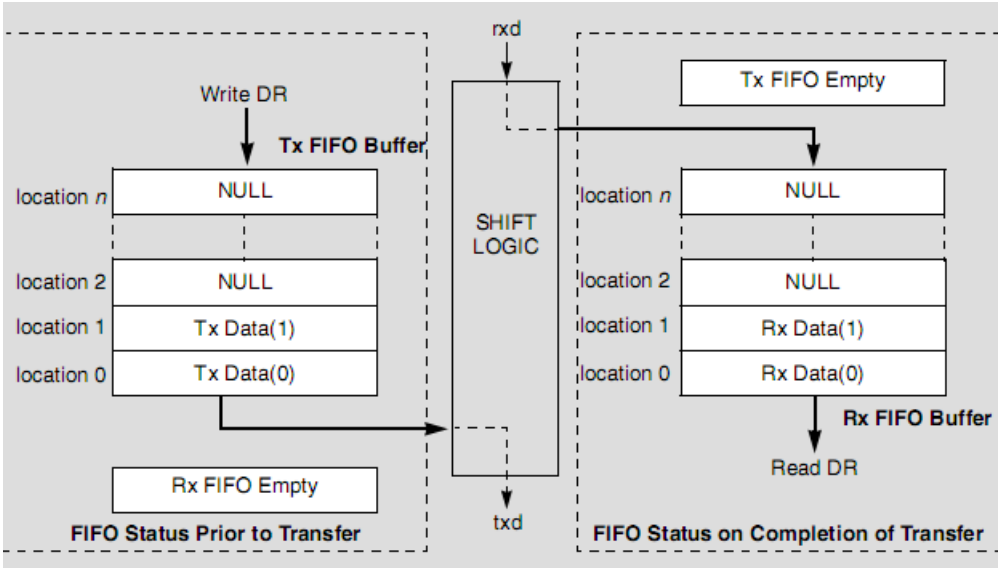


图 4.3 Transmit & Receive 模式

(2) Transmit Only 模式

Transmit Only 模式下，SPI 模块只进行数据发送；对于接收的数据经过移位寄存器后不放入接收 FIFO 中。

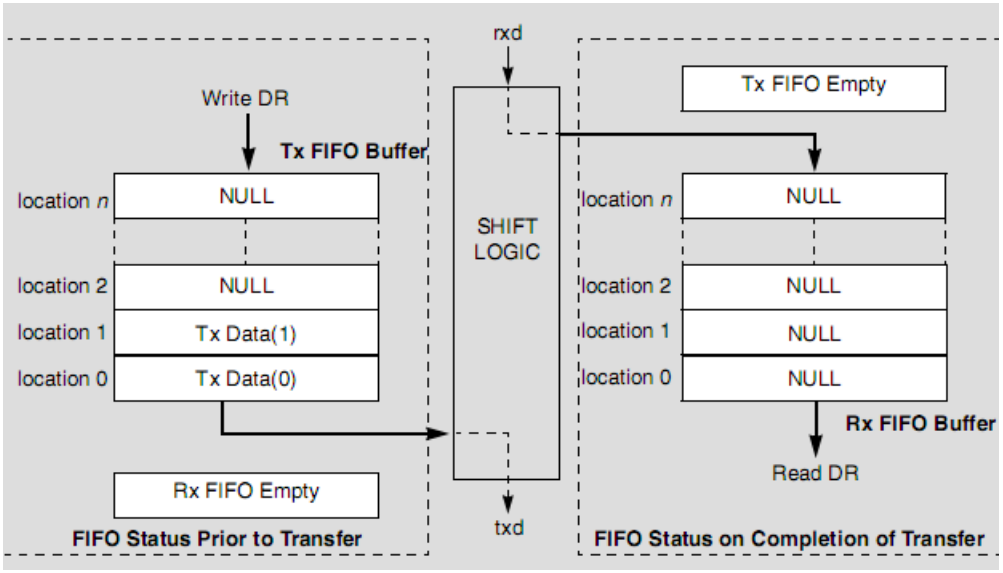


图 4.4 Transmit Only 模式

(3) Receive Only 模式

Receive Only 模式下，SPI 模块只进行数据接收；但是为了启动数据接收，必须要在发送 FIFO 中随意写入一个数据，这个数据会被重复发送出去。

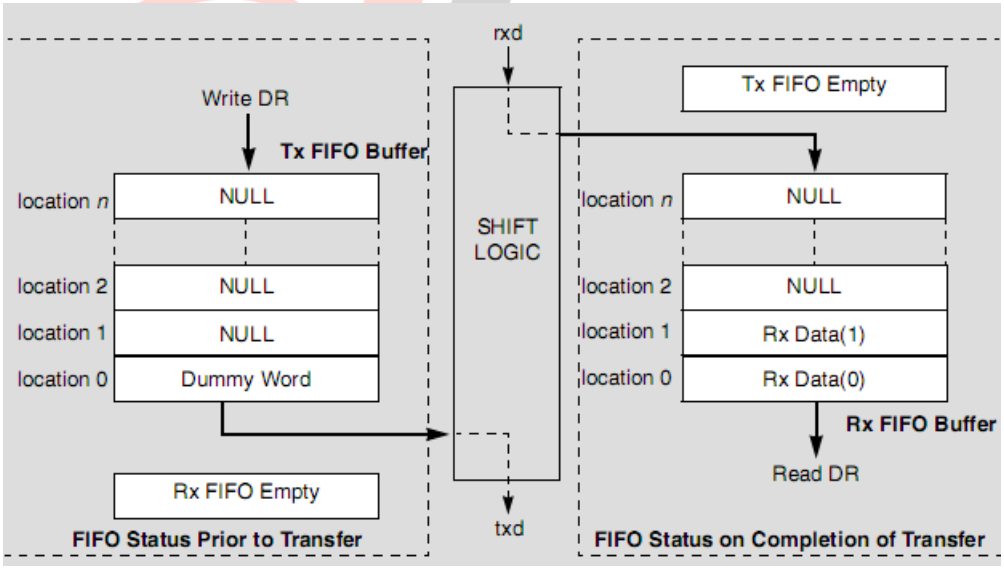


图 4.5 Receive Only 模式

(4) EEPROM Read Transfer 模式

EEPROM Read Transfer 模式下，SPI 模块先将发送 FIFO 中的数据发送出去后，再进行数据的接收；发送数据过程中不接收数据。注意必须要先将发送数据填入 FIFO 中后，在片选使能相应 slave 设备。

注意：EEPROM Read Transfer 模式下，命令长度至少为两拍，即发送命令至少需要写 FIFO 两次；若只有一次，则读回的数据的第一拍（frame 宽度为单位）不会写入到 FIFO 中。

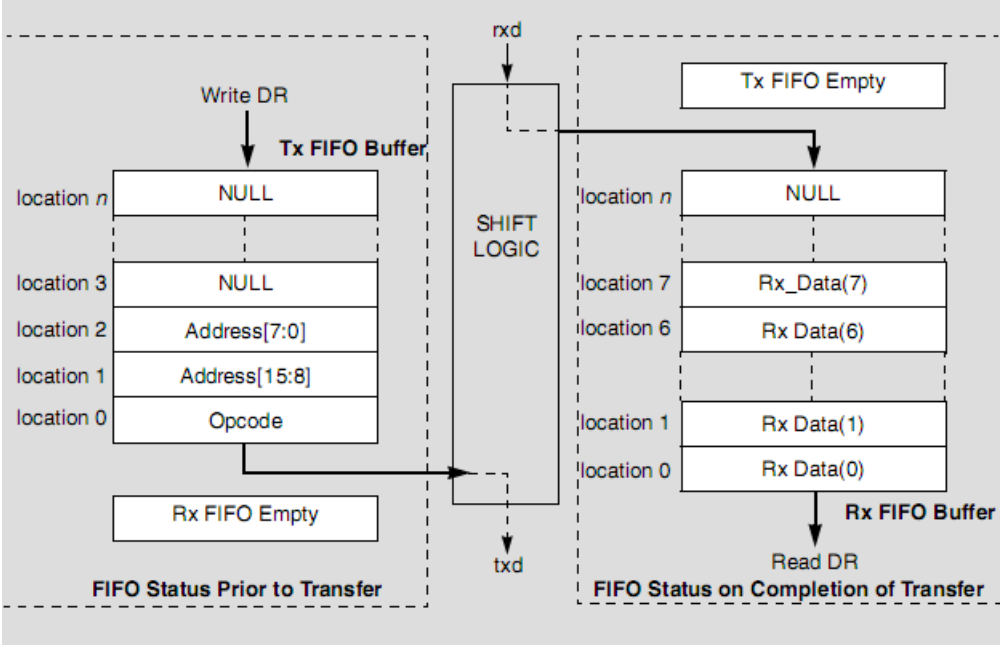


图 4.6 EEPROM Read Transfer 模式

4.1.4 寄存器概述

寄存器表 4.1 SPI 寄存器概述

偏移地址	名称	描述	其他
0x00	CTRLR0	传输方式及帧长度配置	
0x04	CTRLR1	接收数据帧个数	
0x08	SSIENR	SPI 模块使能	
0x10	SER	slave 片选使能	
0x14	BAUDR	时钟使能及分频选择	
0x18	TXFTLR	发送 FIFO 门限值	
0x1C	RXFTLR	接收 FIFO 门限值	
0x20	TXFLR	发送 FIFO 中数据个数	
0x24	RXFLR	接收 FIFO 中数据个数	
0x28	SR	SPI 模块状态	
0x2C	IMR	中断屏蔽寄存器	
0x30	ISR	被屏蔽的中断寄存器	
0x34	RISR	未被屏蔽的中断寄存器	
0x38	TXOICR	发送 FIFO overflow 中断清除	
0x3C	RXOICR	接收 FIFO overflow 中断清除	
0x40	RXUICR	接收 FIFO underflow 中断清除	
0x48	ICR	清除 ssi_txo_intr、ssi_rxu_intr、ssi_rxo_intr 中断	
0x58	IDR	模块 ID 寄存器	
0x5C	SSI_COMP_VERSION	版本号	
0x60	DR	数据发送/接收 FIFO 接口	

4.2 I80 接口

4.2.1 概述

DH4570 内置一 Intel I80 接口控制器，用于与 LCD 等器件接口。

4.2.2 功能描述

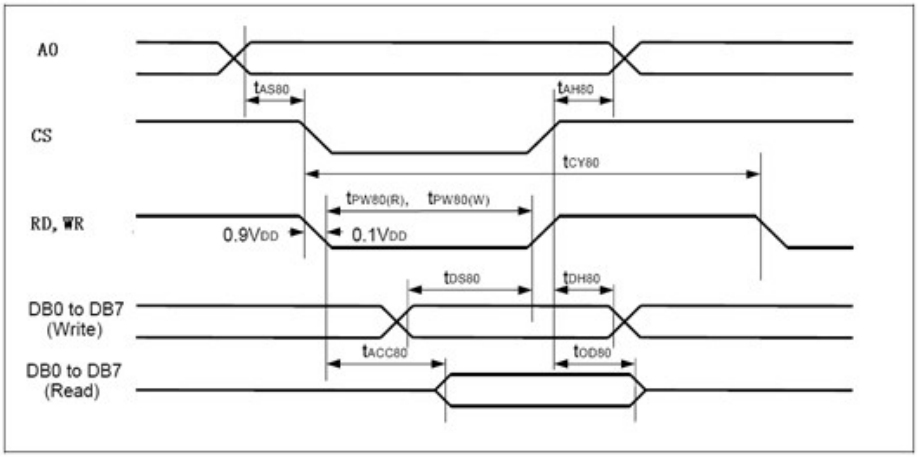
I80 控制器在芯片内连接到系统 AHB 总线，在芯片外连接到外部 I80 接口器件。

4.2.3 工作方式

I80 控制器自动将对其寄存器的操作转换为符合 I80 总线要求的时序。I80 控制器内部有多种配置寄存器，可以实现 I80 各种时序参数和电平极性的灵活配置。

8080并行总线接口

当PS为高电平、C86为低电平时，LCD模块的总线接口为8080并行总线接口，LCD的时序图如下：



当LCD模块处于8080总线模式下时，原LCD模块上的EP引脚将作为8080总线的RD（读使能信号线）。

(VDD = 2.4 to 3.6V, Ta = -40 to +85°C)

Item	Signal	Symbol	Min.	Typ.	Max.	Unit	Remark
Address setup time Address hold time	RS	tAS80 tAH80	0 0	-	-	ns	
System cycle time	RS	tCY80	300	-	-	ns	
Pulse width (WRB)	RW_WRB	tPW80(W)	60	-	-	ns	
Pulse width (RDB)	E_RDB	tPW80(R)	60	-	-	ns	
Data setup time Data hold time	DB7 to DB0	tDS80 tDH80	40 15	-	-	ns	
Read access time Output disable time		tACC80 tOD80	- 10	-	140 100	ns	CL = 100 pF

图 4.7　8080 接口时序

4.2.4 寄存器概述

寄存器表 4.2 I80 寄存器概述

偏移地址	名称	描述	其他
0x0	INDEX	寻址寄存器	
0x4	DATA	数据寄存器	
0x10	WCFG	写时序参数配置	
0x14	RCFG	读时序参数配置	
0x20	SCFG	配置寄存器	

0x2c	AC_MODE	总线模式寄存器	

4.3 UART 接口

4.3.1 概述

通用异步收发器UART（Universal Asynchronous Receiver Transmitter）是一个异步串行的通信接口，主要功能是将来自外围设备的数据进行串并转换之后传入内部总线，以及将数据进行并串转换之后输出到外部设备。UART 的主要功能是和外部芯片的UART 进行对接，从而实现两芯片间的通信。DH4570提供3个UART单元，默认RXD和TXD上下拉使能关闭，当不使用Uart功能时，建议软件配置打开上下拉使能寄存器，并在板级端口上将RXD固定为高电平或者低电平。

4.3.2 功能描述

UART 的一次帧传输主要包括起始信号、数据、校验位和结束信号，如4.3.2功能描述所示。数据帧从某一UART 的TXD 端输出，从另一个UART 的RXD 端输入。

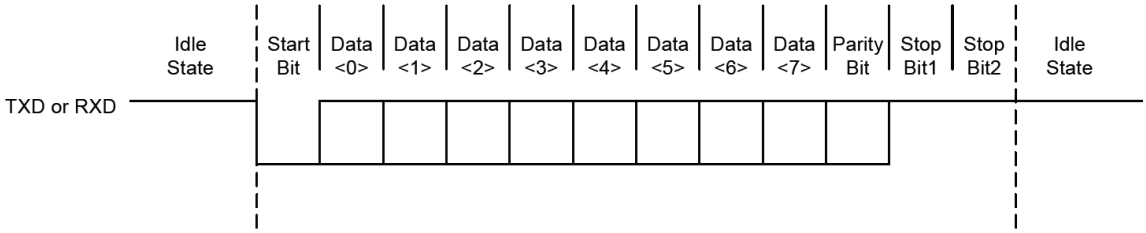


图 1 UART 帧格式

起始信号、数据、校验位和结束信号的含义如下：

- 起始信号（start bit）
  一个数据帧开始的标志，UART 协议规定TXD 信号出现一个低电平就表示一个数据帧的开始。在UART 不传输数据时，应该保持高电平。
- 数据信号（data bit）
  数据位宽可以根据不同的应用要求进行调整，可以配置成5bit/6bit/7bit/8bit 数据位宽。
- 校验位（parity bit）
  校验位是1 比特纠错信号，UART 的校验位有奇校验、偶校验和固定校验位，同时支持校验位的使能和禁止。
- 结束信号（stop bit）
  结束信号即数据帧的停止位，支持1 比特和2 比特停止位两种配置。数据帧的结束信号就是把TXD 拉成高电平。

4.3.3 寄存器概述

寄存器表 4.3 UART 寄存器概述

名称	偏移地址	描述	其他
RBR	0x00	接收缓存寄存器	
THR	0x00	发送保持寄存器	
DLL	0x00	分频锁存低位	
DLH	0x04	分频锁存高位	

IER	0x04	中断使能寄存器	
IIR	0x08	中断标识寄存器	
FCR	0x08	FIFO 控制器寄存器	
LCR	0x0c	线控制寄存器	
MCR	0x10	Modem 控制寄存器	
LSR	0x14	线状态寄存器	
MSR	0x18	Modem 状态寄存器	
SCR	0x1c	临时寄存器	

4.4 I2C 接口

4.4.1 概述

I2C（The Inter-Integrated Circuit）接口有 2 个信号：SDA（串行数据或地址线）和 SCL（串行时钟线）。DH5010M I2C 模块作为 I2C 总线 Master，可对 I2C 从设备进行通信，遵循 I2C 总线协议 2.1 版本。

4.4.2 功能描述

I2C 模块有以下特点：

- 支持标准的 100kbit/s 数据传输速率和快速模式 400kbit/s；
- 提供 TX FIFO 和 RX FIFO；
- 支持中断上报，中断查询。

I2C 在传送数据（Data Transfer）时，SDA 上的数据信号必须在 SCL 处于高电平时保持稳定，而只能在 SCL 处于低电平时变换相位。

START 传送的条件是：在 SCL 处于高电平时，SDA 有一个 High-to-Low 的跳变。

STOP 传送的条件是：在 SCL 处于高电平时，SDA 有一个 Low-to-High 的跳变。这个过程下图所示：

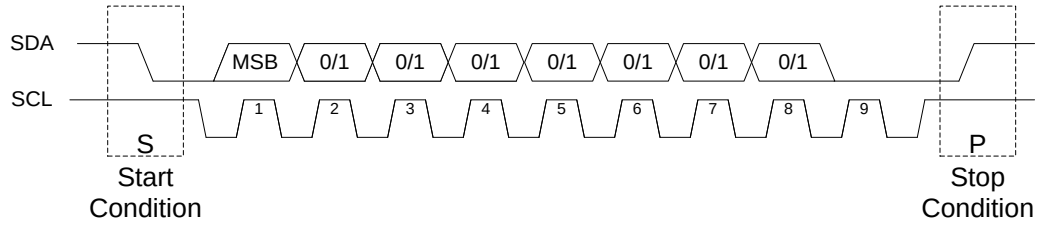


图 4.9 I2C 总线的数据传送过程

I2C 总线上的每个数据包是 8 位位宽，外加一个 ACK 位，所以一个完整的数据传送包括 8 个时钟周期。当 Receiver 接收受完 8 位后，将 SDA 置低来表示 ACK，此时，Transmitter 需要释放 SDA 引线。如果 Slave Receiver 没有发送 ACK，则表示 Receiver 没有收到数据，那么 Master 可以发送 STOP 信号终止数据传送。如果 Master Receiver 没有发送 ACK，则表示传送结束，那么 Slave Transmitter 可以终止传送

表 4.4 I2C 接口信号

信号名	方向	描述
I2C_SCL	B	I2C 串行时钟线
I2C_SDA	B	I2C 串行数据地址线

4.4.3 工作方式

4.4.3.1 初始化

I2C 控制器初始化流程如下：

1） 向 IC_ENABLE 寄存器写入 0，以禁用 I2C 控制器。
2） 写入 IC_CON 寄存器，配置支持的最高速度（第[2:1]位）、7 位或 10 位地址模式（第[4]位）。将第 6 位 IC_SLAVE_DISABLE 设置为 1，第 0 位 MASTER_MODE 设置为 1。
3） 向 IC_TAR 寄存器写入待寻址的 I2C 设备地址。这个寄存器同时用来指示是否发起 General Call 或 START BYTE 命令。
4） 向 IC_ENABLE 寄存器写入 1，以使能 I2C 控制器。
5） 向 IC_DATA_CMD 寄存器写入传输方向和数据。如果此寄存器在 I2C 使能之前被写入，则数据或命令会被丢弃，因为在使能之前 I2C 控制器的缓存是被禁用的。这一步骤会在总线上产生 START 条件及器件地址。
6） IC_START 寄存器写 1，开启 I2C 模块数据传输。

4.4.3.2 数据发送和接收

I2C 控制器支持动态地切换读写操作。为发送数据，只需要向 IC_DATA_CMD 寄存器的最低字节中写入数据。对写操作，第 8 位 CMD 应该置 0。读命令需要向 IC_DATA_CMD 寄存器低字节写入任意数据并把 CMD 位置 1。

4.4.3.3 停用 I2C 控制器

在 IC_ENABLE 寄存器从 1 配为 0 之后，软件可以通过监测 IC_ENABLE_STATUS 寄存器来确定硬件是否完成关闭过程。

停用 I2C 控制器的过程为：

1) 定义一个时间间隔（Ti2c_poll）等于 I2C 最高传输周期的 10 倍。例如如果最高传输速度是 400kbps，则 Ti2c_poll 为 25us。
2) 定义最大超时参数，MAX_T_POLL_COUNT，使任何大于此值的重复查询操作都会报错。
3) 禁止发起新的 I2C 传输，但是允许未完成的传输继续
4) 初始化 POLL_COUNT 变量为 0。
5) 设置 IC_ENABLE 寄存器为 0。
6) 读 IC_ENABLE_STATUE 寄存器，检查器 IC_EN 位（第 0 位）。将 POLL_COUNT 加一。如果 POLL_COUNT 大于等于 MAX_T_POLL_COUNT 则返回错误。
7) 如果 IC_EN 位为 1，则等待 Ti2c_poll 时间，并返回上一步。否则成功退出。

4.4.4 寄存器概述

寄存器表 4.5 I2C 寄存器概述

偏移地址	名称	描述	其他
0x0	IC_CON	I2C 控制寄存器	
0x4	IC_TAR	I2C 目标地址寄存器	
0x10	IC_DATA_CMD	I2C 收发数据缓存和命令	
0x1c	IC_FS_SCL_HCNT	快速 SCL 高电平计数	

0x20	IC_FS_SCL_LCNT	快速 SCL 低电平计数	
0x2c	IC_INTR_STAT	中断状态	
0x30	IC_INTR_MASK	中断屏蔽	
0x34	IC_RAW_INTR_STAT	原始中断状态	
0x38	IC_RX_TL	接收 FIFO 阈值	
0x3c	IC_TX_TL	发送 FIFO 阈值	
0x40	IC_CLR_INTR	清除中断	
0x44	IC_CLR_RX_UNDER	清除中断 RX_UNDER	
0x48	IC_CLR_RX_OVER	清除中断 RX_OVER	
0x4c	IC_CLR_TX_OVER	清除中断 TX_OVER	
0x50	IC_CLR_RD_REQ	清除中断 RD_REQ	
0x54	IC_CLR_TX_ABRT	清除中断 TX_ABRT	
0x58	IC_CLR_RX_DONE	清除中断 RX_DONE	
0x5c	IC_CLR_ACTIVITY	清除中断 ACTIVITY	
0x60	IC_CLR_STOP_DET	清除中断 STOP_DET	
0x64	IC_CLR_START_DET	清除中断 START_DET	
0x6c	IC_ENABLE	I2C 使能寄存器	
0x70	IC_STATUS	I2C 状态寄存器	
0x74	IC_TXFLR	发送 FIFO 水位寄存器	
0x78	IC_RXFLR	接收 FIFO 水位寄存器	
0x80	IC_TX_ABRT_SOURCE	I2C 传输终止状态寄存器	
0x9c	IC_ENABLE_STATUS	I2C 使能状态寄存器	
0xa0	IC_START	控制数据开始发送	

4.5 PWM 接口

4.5.1 概述

PWM 模块产生周期性的脉冲波形，其中周期频率及占空比均可通过寄存器配置。DH4570 提供 3 个独立的 PWM 模块。

4.5.2 功能描述

PWM 模块支持以下功能：

- 周期频率可设，最大支持 54M；
- 占空比可设。

PWM 输出时序图如图所示，其中 period 和高低电平 phase 长度均可配置：

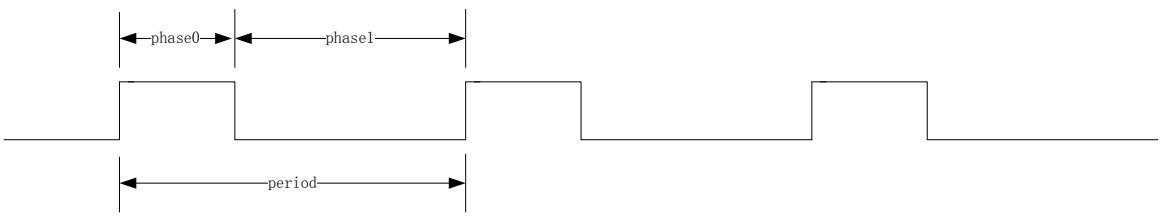


图 2 PWM 输出时序图

4.5.3 工作方式

4.5.3.1 Continuous 模式

配置 PWM_CFG 的 bit[5:4]为 2'b10，表示为 continuous 模式。然后配置一个周期第一个相位持续的时钟拍数，通过配置 PWM_FP 来实现。一个周期持续的时钟拍数可以通过配置 PWM_PER 来实现。

4.5.3.2 One Shot 模式

配置 PWM_CFG 的 bit[5:4]为 2'b01，表示为 one shot 模式。然后配置一个周期第一个相位持续的时钟拍数，通过配置 PWM_FP 来实现。一个周期持续的时钟拍数可以通过配置 PWM_PER 来实现。重复的周期数可以通过配置 PWM_RPT 来实现。

4.5.4 寄存器概述

寄存器表 4.6 PWM 寄存器概述

偏移地址	名称	描述	其他
0x0000	PWM0_CFG	PWM0 配置寄存器	
0x0004	PWM0_RPT	PWM0 ONE_SHOT 传输重复的周期数	
0x0008	PWM0_PER	PWM0 周期波形持续系统时钟的拍数	
0x000C	PWM0_FP	PWM0 一个周期第一个相位持续的拍数	
0x0010	PWM0_STATUS	PWM0 状态寄存器	
0x0020	PWM1_CFG	PWM1 配置寄存器	
0x0024	PWM1_RPT	PWM1 ONE_SHOT 传输重复的周期数	
0x0028	PWM1_PER	PWM1 周期波形持续系统时钟的拍数	
0x002C	PWM1_FP	PWM1 一个周期第一个相位持续的拍数	
0x0030	PWM1_STATUS	PWM1 状态寄存器	
0x0040	PWM2_CFG	PWM2 配置寄存器	
0x0044	PWM2_RPT	PWM2 ONE_SHOT 传输重复的周期数	
0x0048	PWM2_PER	PWM2 周期波形持续系统时钟的拍数	
0x004C	PWM2_FP	PWM2 一个周期第一个相位持续的拍数	
0x0050	PWM2_STATUS	PWM2 状态寄存器	
0xff0	**P**WM_INTR_STA	所有 PWM 中断状态	

4.6 SDIO 接口

4.6.1 概述

DH4570 内置 SDIO 控制器，可以实现与 SD 卡、SDIO 设备等连接。

4.6.2 功能描述

SD/SDIO 控制器用于对 SD/SDIO 协议设备、MMC 协议设备和 CE-ATA 协议设备的读写等操作。控制器软核具有下列功能：

支持接口类型：

- SD 卡（Secure Digital Memory） **version 3.0** 协议
- SDIO（Secure Digital I/O） **version 3.0** 协议
- CE-ATA(Consumer Electronics Advanced Transport Architecture) version 1.1 协议
- Multimedia Cards(MMC version4.41, eMMC version4.5)

支持读写模式：

- 比特流读写（SIDO 设备）
- 单块数据读写（主要面向小容量 SD 卡）
- 连续多块数据读写（主要面向大容量 SD 卡）

支持数据总线模式：

- 1bit 数据模式（SD 模式、SDIO 模式、SPI 模式）
- 4bit 数据总线（SD 模式、SDIO 模式）

其他特点：

- 控制器可支持 1 张 SD 卡或 SDIO 设备
- 支持指令完成信号中断，可编程屏蔽
- cclk_in 时钟标称值 50MHz，可配分频得到 cclk_out，分频倍数为 1 或 2*n（n=1~255）
- cclk_in 时钟 1/4 相位精度可配得到采样时钟和驱动时钟
- 支持低功耗时钟门控

4.6.3 工作方式

4.6.3.1 总体流程

SD/SDIO Controller 从上电到正常数据通信过程的基本流程如下：

a) 打开外部设备的电源，设置设备检测的时钟周期数；
b) 设置时钟分频用于设备枚举（100KHz~400KHz），启动时钟；
c) 使能全局中断，屏蔽不需要的中断，检测设备状态；
d) 进行设备枚举，完成设备完成初始化；
e) 更换时钟频率（标准模式最高 25MHz，高速模式最高 50MHz），
f) 设置与应用层相关的 FIFO 门限、超时门限、中断屏蔽等寄存器；
g) 进入数据通信操作。

注：SD 卡上电完成后，需要维持至少 74 个 cclk_out 时钟后，才允许进入枚举功能，否则存储卡设备不能正常相应。cclk_out 时钟周期远大于系统时钟，故软件需留足够时间。

4.6.3.2 电源管理

SD/SDIO Controller 电源管理功能：

a) 通过寄存器配置 SD 设备的电源开关；
PWREN[0] = 1'b1; //使能#0 卡设备电源

4.6.3.3 时钟控制

更换时钟频率操作需在指令及数据通信空闲状态进行，更换时钟频率之前需先禁用时钟，然后再使能新频率的时钟。

a) 禁用时钟：清零 clk_enable 寄存器（CLKENA@0x10）；然后设置 start_cmd、update_clock_only、wait_data_complete（CMD@0x2c）各位为 1，发送时钟控制指令；等待 start_cmd 清零，表示指令已被发送。

CLKENA[15:0] = 16'h0; //禁用所有存储卡的工作时钟

CMD[31] = 1'b1, CMD[21] = 1'b1,CMD[13] = 1'b1; //更新时钟设置参数

b) 启用时钟：设置时钟分频参数 CLKDIV 寄存器(CLKDIV@0x08)，选定分频时钟源（CLKSRC@0x0c），并设置时钟使能 CLKENA 寄存器；然后设置 start_cmd、update_clock_only、wait_data_complete 各位为 1，发送时钟控制指令；等待 start_cmd 清零，表示指令已被发送。

CLKDIV[7:0] = 8'h40; //设置#0 分频器 128 分频

CLKSRC[1:0] = 2'h0; //选定#0 存储卡的 clk_out 为#0 分频器时钟

CLKENA[0] = 1'b0; //使能#0 存储卡 cclk 时钟

CMD[31] = 1'b1, CMD[21] = 1'b1,CMD[13] = 1'b1; //更新时钟设置参数

c) 设置时钟低功耗模式(可选)：通过设置 CLKEN@0x10 寄存器 bit[31:16]位，使能使设备低功耗模式。在低功耗模式下，卡设备处于 IDLE 状态超过 8 个时钟后，控制器自动停止时钟输出，以降低功耗。配置低功耗模式时，需要使用 CMD 时钟更新指令（start_cmd、update_clock_only）才能有效。而且，低功耗模式只限于 MMC/SD 卡使用；对于 SDIO 设备，时钟停止后将检测不到 SDIO 设备中断。

CLKEN[16] = 1'b1; //使能#0 卡设备时钟低功耗模式

注：时钟控制指令只作用在 SD/SDIO Controller 内部，不发送到外部设备，因而不产生指令完成中断。

4.6.3.4 中断控制

中断使能功能包括：一个全局中断控制寄存器 int_enable(CTRL@0x00, bit4)，控制全局中断使能，只有当 int_enable 有效时，中断才会触发。

三组与中断相关的寄存器：INTMASK@0x24，MINTSTS@0x40 和 RINTSTS@0x044，其中 INTMASK[15:0]位控制屏蔽 16 个特定功能的中断，INTMASK[31:16]位控制屏蔽对应 16 个 SDIO 中断，RINTSTS[31:0]显示中断触发状态（与 INTMAST 状态无关），而 MINTSTS[31:0]标示未被屏蔽的触发中断，三者满足下述关系：

MINTSTS = MINTSTS & RINTSTS;

INTMASK 寄存器清零时，对应中断被屏蔽，该中断触发后，仅在 RINTSTS 寄存器中更新中断状态，在 MINTSTS 无更新。响应 RINTSTS/MINSTS 中断后，向该寄存器内写 1 清除中断状态。

4.6.3.5 状态检测

STATUS@0x48 寄存器用于设备数据读写时 FIFO 状态、数据总线状态、发送状态等信息查询。控制器与设备通信前，应先查询该寄存器，确定控制器状态。

CMD@0x2c 寄存器用于主端发送指令控制、设备初始化、应答格式控制。

4.6.3.6 设备枚举

设备进入数据传输过程之前，必须先进行设备枚举过程，进行设备识别及设备初始化操作。枚举过程时钟频率为 100~400KHz。

SD 设备枚举过程流程图如下，具体参照 SD Specifications Part 1 Physical Layer Specification Version 3.00：

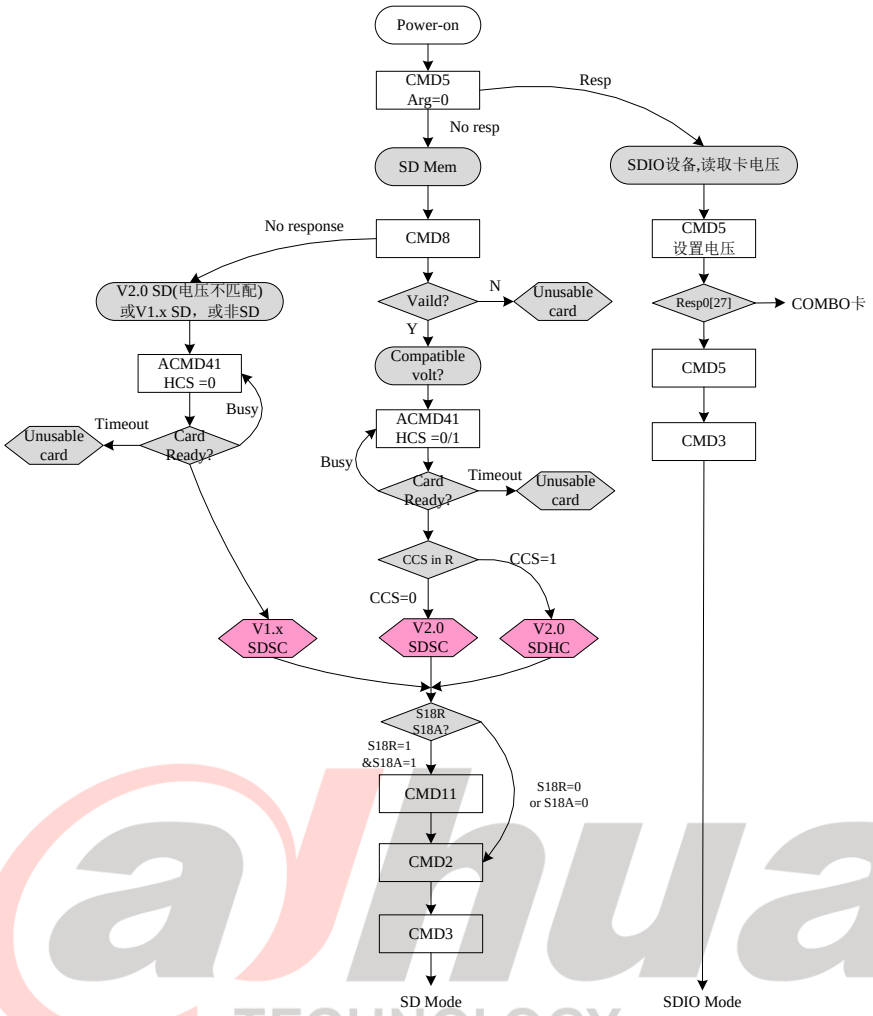


图 3 SD 卡的枚举和初始化流程

a) 发送 CMD0 初始化卡设备。发送 CMD0 时需要设置 send_initialization（CMD@0x2c，bit15）值为 1，控制器自动发送 80 个时钟的初始化序列；

b) 识别卡类型。

首先发送 CMD5，若收到正确应答，则为 SDIO 设备，跳转至 d)步骤进行 SDIO 设备枚举；

若应答超或应答错误，则发送 CMD8，其 Argument 内容段如下：

如果控制器收到 CMD8 的正确应答 R7，则为是支持 SD2.0 以上协议的大容量设备；接着发送 Argument 内容段如下所示的 ACMD41 指令（ACMD41 需先发送 CMD55，然后发送 CMD41，设备自动将 CMD 识别为 ACMD41）；

如果控制器收到 ACMD41 的正确应答 R3，则为 SD 存储卡；否则为 MMC、CEATA 或无效卡（项目中不对支持 MMC、CEATA 设备不做要求，故不再累述这两张卡的枚举和读写操作）；检测应答 R3 中 busy bit 为状态，若为零表示卡设备在初始化过程，需要重复发送 ACMD41 指令，等待应答中 busy bit 置位；若应答超时，则卡无效；

如果控制器未收到 CMD8 的正确应答 R3，则表示设备不是不支持 SD2.0 协议；接着发送 CMD0 复位卡设备；然后重新发送 Argument 如下的 ACMD41 指令；

此时，如控制器收 ACMD41 到应答 R3，则卡为 SD 存储器，否则为 MMC 或者 CEATA；

注意：必须先发送 CMD8 然后发送 ACMD41；

c) 重复发送 ACMD41 直至检测到应答 R3 的 busy 位置位，标志设备上电初始化完成；

d) 按照规定的指令顺序枚举卡设备。CMD2 在应答中获取卡设备的 CID 信息，CMD3 为卡设备提供新的偏移地址。若要发布新的偏移地址需重新发送 CMD3。

SD 卡：发送 CMD0，CMD8，ACMD41，CMD2，CMD3；

SDIO：发送 CMD5，如果功能数量正确则接着发送 CMD3；若是 memory present 置位，则说明是带有存储区的 COMBO 卡，之后按照 SD 存储卡完成枚举；

MMC：发送 CMD0，CMD1，CMD2，CMD3；

e) 识别 MMC/CE-ATA 卡

发送 CMD8 获取设备 EXT_CSD 寄存器的 S_CMD_SET[4]值为 1，则设备支持 ATA 模式。然后通过设置 EXT_CSD 寄存器的 CMD_SET[4]激活 ATA 模式指令集，SD/SDIO Controller 通过 CMD6 更换指令模式；

如果设备应答不支持 ATA 模式：发送 CMD39 如果收到应答并在应答中包含 CE-ATA 标志，则是 CE-ATA v1.0 设备；否则是 MMC 卡。

4.6.3.7 指令操作

SD/SDIO Controller 指令包括时钟更新指令和 SD 协议标准指令。时钟更新指令不发送到外部指令总线上。指令总线上每次只能同时存在一条要求数据传输的指令。通过设置指令寄存器（CMD@8'h2c）发送指令。

某些特定指令在发送前，需先向 argument 寄存器（CMDARG@0x28）填充相应功能值。协议中规定没有 argument 的指令，相应 argument 内容段建议清零。

当从卡设备接收到应答后，控制器自动将 Command done（RINTSTS@0x44)置为 1，触发中断。应答错误可能触发 Response timeout error、Response CRC error 或 Response error 中断。短应答内容存放在 RESP0@0x30；长应答内容依次存放在 RESP0@0x30、RESP1@0x34、RESP2@0x38、RESP3@0x3c 四个寄存器中。

发送指令（CMD@0x2c）需设置的寄存器如下表所示：

表 4.7 发送指令寄存器设置

IP Main Features	Values	Comments
默认属性		
Bit[31], start_cmd	1	设置为 1 表示启动指令发送；当 CIU 取走指令后将 start_cmd 设置为 0；当 start_cmd 值为 1 时软件仍然发送指令时 Hardware Locked Write Error（RINTSTS @0x40）会置为 1，触发错误中断。
Bit[29], use_hold_reg	0	CMD 和 DATA 不经过 HOLD Reg 发送
Bit[28], volt_switch	0	不进行电压切换
Bit[27:22]	0	与 SD 和 SDIO 无关
Bit[21], update_clock_ registers_only	0/1	0：SD 指令，CIU 将指令发送到卡设备； 1：时钟控制指令，CIU 不把指令发送到卡设备，只更新时钟相关寄存器。
Bit[20:16], card_number	x	卡设备偏移地址
Bit[15], send_initalization	0/1	0：不初始化序列 1：发送 80 个时钟的'1'，初始化卡设备
Bit[14], stop_abort_cmd	0	0：对当前指令无影响 1：停止或放弃当前指令
Bit[12], send_auto_stop	0	0：数据传输结束后，无自动停止指令 1：数据传输结束后，有自动停止指令

Bit[11], transfer_mode	0/1	0：数据块传输（SDHC） 1：数据流传输（SDIO）
Bit[10], read/write	0/1	0：从卡设备读取；1：向卡设备写入
Bit[9], data_expected	0/1	0：不进行数据传输 1：数据传输指令。
Bit[7], response_length	0/1	0：短应答；1：长应答。
Bit[6], response_expect	1	0：CMD0/CMD4/CMD15 等指令不需要应答； 1：需要应答。
Bit[5:0], cmd_index	x	指令编号
用户属性		
Bit[13], wait_prvdata_complete	1	等待数据传输操作完成才能发送下一条指令，一般设为 1（除非发送状态查询指令或传输中止指令）
Bit[8], check_response_crc	1	对应答 CRC 校验

4.6.3.8 数据传输

SD 存储器数据传输指令与 SDIO 设备的 I/O 操作所使用的指令集不相同，SD 存储器可使用多条数据读写指令，如 CMD17、CMD18、CMD24、CMD25 等，而 SDIO 设备数据读写操作有且只有 CMD53 指令。另外，SD 存储器和 SDIO 设备在数据传输中状态转移也不同，SD 模式数据传输过程状态如下图所示，SDIO 模式数据传输过程状态如下图所示。SDIO 本质是一种基于 SD 协议的扩展接口技术，故 SD 协会从指令集和状态机等多个方面竭力简化 SDIO 协议，以使得 SDIO 能成为完全兼容 SD 电气标准、简单易用、对上层软件透明 IO 接口。具体说明参照 SD Specifications Part 1 Physical Layer Specification Version 3.00 和 SD Specifications Part E1 SDIO Specification Version 3.00。

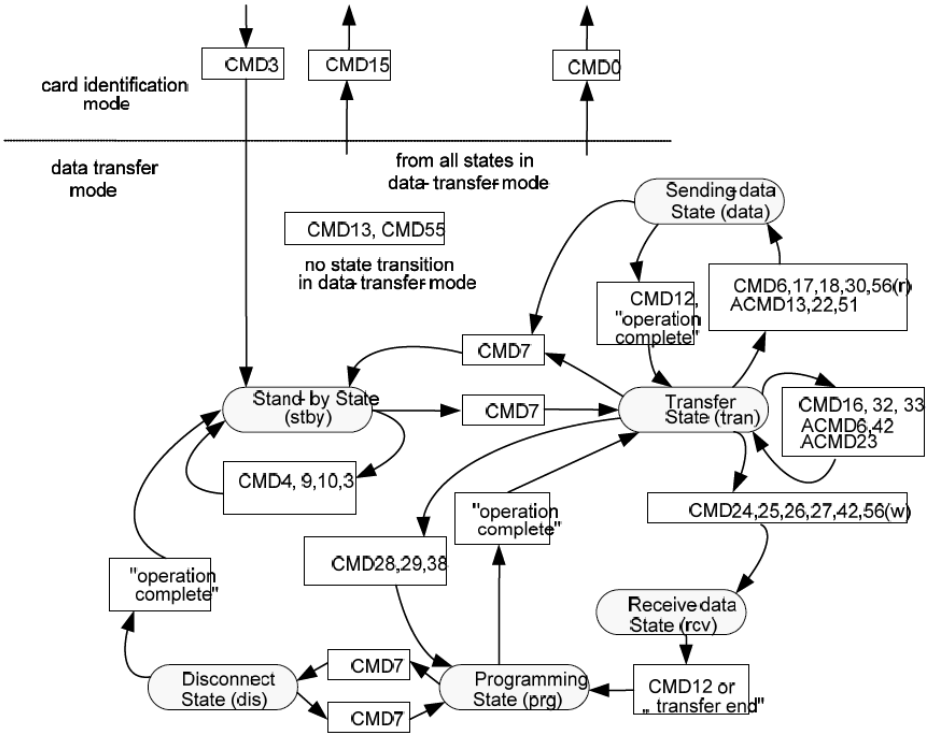


图 4.12 SD 存储器数据传输状态转移图

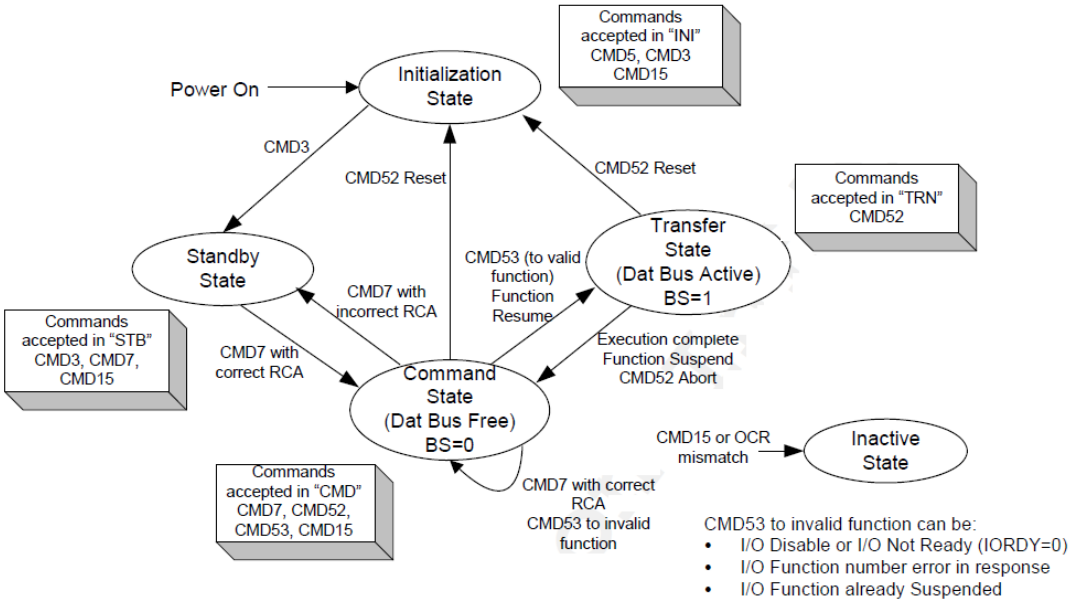


图 4 SDIO 设备数据传输状态转移图

数据传输过程会在 RINTSTS@0x44 寄存器中触发相应中断：

数据传输过程会在 RINTSTS@0x44 寄存器中触发相应中断：

End Bit Error/Write no CRC(EBE)：读卡时结束位错误，或写卡后未收到 CRC 确认；

Auto Command Done：多块数据传输结束后，自动发送 CMD12 指令完成中断；

Start Bit Error(SBE)：4bit 总线位宽读卡开始时，若 DAT0 发出起始位 0，其他数据线未发出 0，触发该中断；

Hardware Locked write Error(HLE)：在 hardware_locked 状态，写寄存器触发该中断；

FIFO Under/Over Run Error(FRUN)：Host 在 FIFO 满/空时，写/读数据触发该中断；

Data Starvation by Host Timeout(HTO)：host 端在 FIFO 满/空下未及时读写 FIFO，触发该中断；

Data Read Timeout(DRTO)：数据读取超时，卡设备没有在规定时间内向 Host 端发送数据；

Data CRC Error(DCRC)：数据接收过程 CRC 校验错误、在 End 标志位错误；

Receive FIFO Data Request(RXDR)：FIFO 中数据超过门限，软件可从 FIFO 中读取数据；

Transmit FIFO Data Request(TXDR)：FIFO 中数据低于门限，软件可写入数据到 FIFO；

Data Transfer Over(DTO)：数据传输完成中断；

注意：

(1) 如果数据传输指令应答超时，表示指令未能生效，不再会有数据传输;

(2) 若数据传输过程中发生 DCRC, SBE 或 EBE 等错误时，可能会也可能不会触发 DTO，故发生任何数据错误时，必须发生 CMD12 或 CMD53 强制终止数据传输，确保 DTO 被触发(DesignWare DWC Mobile Storage Host Databook,Version 2.60a，p56)。

发送数据传输指令之前，需要发送 CMD13 指令读取 SD 存储器当前状态(argument 段的[31:16]为偏移地址，[15:0]填充字节)，应答中 READY_FOR_DATA 位标示存储器当前是否可以接收新的数据。然后发送 CMD7（argument 段[31:16]为偏移地址，[15:0]填充字节）选中卡设备，使其进入传输状态。

1) 单块或多块读数据

a) 向寄存器 CTRL@0x00 fifo_reset 位写“1”，复位 FIFO 指针，查询等待该位自动清零。

b) 向寄存器 BYTCNT@0x20 写入待传输数据的字节数；
c) 向寄存器 BLKSIZ@0x1c 写入块的大小（SDHC 块大小固定为 512 字节）；
d) 向寄存器 CMDARG@0x28 写入数据的起始地址；
e) 设置寄存器 CMD。对于 SD/MMC 存储器，分别使用指令 CMD17/CMD18 进行单块/多块读取操作；对于 SDIO 设备，仅使用指令 CMD53 进行单块/多块读操作。一旦寄存器 CMD 被写入，SD/SDIO Controller 开始执行指令。当指令发送完成且接收到正确应答后，触发 Command Done 中断；
f) 检查寄存器 STATUS@0x48 fifo_empty 状态位清“0”，开始从寄存器 FIFO@0x200 读取数据，以便 SD/SDIO Controller 接收后面的数据；同时检查数据错误中断，即寄存器 RINTSTS bit[7]和 RINTSTS bit[15]。此时，程序可以发送一个停止指令中止数据的传输。
g) 读取 FIFO 数据，直至 STATUS fifo_empty 状态位置“1”。数据读取完成后，向 RINTSTS bit[3] 写“1”,清除传输完成中断；
h) 若进行多块读取操作，且已将寄存器 CMD send_auto_stop 置“1”，SD/SDIO Controller 在数据读取完成后自动发送 CMD12 停止指令。必要时可以主动发送数据传输停止指令 CMD12 停止数据传输。

表 1 SD 存储器读取操作指令设置

IP Main Features	Values	Comments
默认属性		
Bit[31], start_cmd	1	设置为 1 表示启动指令发送；当 CIU 取走指令后将 start_cmd 设置为 0；当 start_cmd 值为 1 时软件仍然发送指令时 Hardware Locked Write Error（RINTSTS @0x40）会置为 1，触发错误中断。
Bit[29], use_hold_reg	0/1	CMD 和 DATA 是否经过 HOLD Reg 发送
Bit[28], volt_switch	0	不进行电压切换
Bit[27:22]	0	与 SD 和 SDIO 无关
Bit[21], update_clock_ registers_only	0	0：SD 指令，CIU 将指令发送到卡设备； 1：时钟控制指令，CIU 不把指令发送到卡设备，只更新时钟相关寄存器。
Bit[20:16], card_number	x	卡设备偏移地址
Bit[15], send_initalization	0	0：不初始化序列 1：发送 80 个时钟的’1’，初始化卡设备
Bit[14], stop_abort_cmd	0	0：对当前指令无影响 1：停止或放弃当前指令
Bit[12], send_auto_stop	0/1	0：数据传输结束后，无自动停止指令 1：数据传输结束后，有自动停止指令
Bit[11], transfer_mode	0	0：数据块传输；1：数据流传输
Bit[10], read/write	0	0：从卡设备读取；1：向卡设备写入
Bit[9], data_expected	1	0：非数据传输操作；1：数据传输操作
Bit[7], response_length	0	0：短应答；1：长应答。
Bit[6], response_expect	1	0：不需应答（如 CMD0/CMD4/CMD15 等） 1：需要应答
Bit[5:0], cmd_index	x	指令编号
用户属性		
Bit[13], wait_prvdata_complete	1	等待数据传输操作完成才能发送下一条指令，一般设

		为 1（除非发送状态查询指令或传输中止指令）
Bit[8], check_response_crc	1	对应答 CRC 校验

2) 单块与多块写数据

a) 向寄存器 CTRL fifo_reset 写“1”，复位 FIFO 指针，查询等待直至该位自动清零；

b) 向寄存器 BYTCNT 写入待传输数据的大小；

c) 向寄存器 BLKSIZ 写入块的大小（SDHC 块大小固定为 512 字节）；

d) 向寄存器 CMDARG 写入数据的起始地址；

e) 读取 STATUS 寄存器信息，判断 FIFO 状态后，将数据写入 FIFO，即写寄存器 0x200，通常在开始时应先写满 FIFO；

f) 设置寄存器 CMD。对于 SD/MMC 卡，分别使用指令 CMD24/CMD25 进行单块/多块写操作；对于 SDIO 卡，使用指令 CMD53 进行单块/多块写操作；

g) 检查寄存器 STATUS@0x48 fifo_empty 状态位置“1”，或 RINTSTS [TXDR]中断位置“1”，写 0x200 寄存器，向 FIFO 填充数据；同时应检查数据错误中断，即检查寄存器 RINTSTS [DCRC]位、RINTSTS [EBE]位、RINTSTS [HLE]位。如果有需要，程序可以发送一个停止指令以中止数据的传输。当寄存器 RINTSTS DTO 置“1”，数据传输结束，回写“1”，清除该中断；

i) 若进行多块写入操作，且已将寄存器 CMD send_auto_stop 置“1”，控制器自动发送停止指令结束一次数据传输。必要时可以主动发送数据传输停止指令 CMD12 停止数据传输；

j) 查询并等待寄存器 CMD [data_busy]由“1”变为“0”。

表 4.9 SD 存储器写入操作指令设置

IP Main Features	Values	Comments
默认属性		
Bit[31], start_cmd	1	设置为 1 表示启动指令发送；当 CIU 取走指令后将 start_cmd 设置为 0；当 start_cmd 值为 1 时软件仍然发送指令时 Hardware Locked Write Error（RINTSTS @0x40）会置为 1，触发错误中断。
Bit[29], use_hold_reg	0/1	CMD 和 DATA 是否经过 HOLD Reg 发送
Bit[28], volt_switch	0	不进行电压切换
Bit[27:22]	0	与 SD 和 SDIO 无关
Bit[21], update_clock_registers_only	0	0：SD 指令，CIU 将指令发送到卡设备； 1：时钟控制指令，CIU 不把指令发送到卡设备，只更新时钟相关寄存器。
Bit[20:16], card_number	x	卡设备偏移地址
Bit[15], send_initalization	0	0：不初始化序列 1：发送 80 个时钟的’1’，初始化卡设备
Bit[14], stop_abort_cmd	0	0：对当前指令无影响 1：停止或放弃当前指令
Bit[12], send_auto_stop	0/1	0：数据传输结束后，无自动停止指令 1：数据传输结束后，有自动停止指令
Bit[11], transfer_mode	0	0：数据块传输；1：数据流传输
Bit[10], read/write	1	0：从卡设备读取；1：向卡设备写入
Bit[9], data_expected	1	0：不进行数据传输；

		1：数据传输指令
Bit[7], response_length	0	0：短应答；1：长应答。
Bit[6], response_expect	1	0：CMD0/CMD4/CMD15 等指令不需要应答； 1：需要应答。
Bit[5:0], cmd_index	x	指令编号
用户属性		
Bit[13], wait_prvdata_complete	1	等待数据传输操作完成才能发送下一条指令，一般设为 1（除非发送状态查询指令或传输中止指令）
Bit[8], check_response_crc	1	对应答 CRC 校验

3) 流数据读写

流数据的读写方式，除了将寄存器 CMD [Transfer_mode]置“1”外，其他与块数据的读写方式一致。对于流数据的传输，通常需要使用 Auto-stop 功能。

4.6.3.9 Auto-stop 操作

在多块读写指令操作中，需使用停止指令完成数据传输。停止指令可以软件通过主动发送 CMD12，也可以在 CMD(0x2c)置位 Auto-stop 功能使 SD/SDIO Controller 自动发送 CMD12，以便卡能返回相应的状态。CMD12 完成后会触发 RINTSTS(0x44) auto_command_done 中断，用以判断该停止指令是否完成，其应答内容保存在寄存器 **RESP1**(0x34)中。Auto-stop 功能的应用场合如下

表 4.10 Auto-Stop 支持的卡类型和传输模式

存储卡类型	传输模式	字节数(0x20)	可否使用 send_auto_stop	说明
MMC	Stream read/write	0	否	Open-end 流传输
MMC	Stream read/write	>0	是	字节传输后 Auto-stop
MMC	Single-block read/write	>0	否	字节数=0 非法
MMC	Multiple-block read/write	>0	是	传输完成后 Auto-stop
SDMEM	Single-block read/write	>0	否	字节数=0 非法
SDMEM	Multiple-block read/write	0	否	Open-end 多块操作
SDMEM	Multiple-block read/write	>0	是	传输完成后 Auto-stop
SDIO	Single-block read/write	>0	否	字节数=0 非法
SDIO	Multiple-block read/write	0, or >0	否	SDIO 不支持该功能

4.6.3.10 停止数据传输

1) 指令 CMD12 用于 MMC/SD 存储卡、SDIO 存储功能的数据传输停止。
注意需要设置 CMD(0x2c) bit14 = 1’b1, bit13 = 1’b0
2) 指令 CMD52 设置 SDIO 卡的寄存器 CCCR 0x06 地址中 bit0-1(AS0-AS1)位，用于 SDIO 的 I/O 功能的传输停止。
注意需要设置 CMD(0x2c) bit14 = 1’b1, bit13 = 1’b0

4.6.3.11 擦除操作

1) 预擦除操作
在多块写入操作前，用 ACMD23 指令设置这些被写入块为可被预擦除模式时，写入速度

会快于未被设置相同操作。主端利用 ACMD23 告知存储卡写入/预擦除块数量。若主端在中间过程发起终止写入指令(CMD12)，余下的已经被设置预擦除模式的数据块的内容将处于未知状态，内容可能已被擦除，也可能未被擦除。如果主端发送的数据块数多于 ACMD23 中给定的数据块数值，当接收到新数据块时，存储卡逐块擦除数据块。完成多块写操作后，这个数值(pre-erase-count)会复位成默认值 1。协议推荐在 CMD25 写入指令前先发送 ACMD23 指令，以提高存储卡写入速度。注意，ACMD23 必须在写入指令前一条发出，否则 pre-erase-count 会被其他指令清空。

2) 正常擦除操作

用 ERASE_WR_BLK_START（CMD32）指令和 ERASE_WR_BLK_END(CMD33)指令要定义擦除数据块的起末地址，随之发送 ERASE(CM38)开始擦除操作。这三条指令必须顺序执行，否则存储卡应答 ERASE_SEQ_ERROR 错误，并复位整个序列。若其他指令插入(SEND_STATUS 指令除外)，存储卡置位 ERASE_RESET，复位擦除指令队列，执行最后一条指令(插入的指令)。若擦除地址范围内含有写保护的数据段，必须跳过这些数据段，仅擦除未写保护的区域，并同时置位 WP_ERASE_SKIP。最小擦除范围由 CSD 寄存器中 Sector 段规定。

另外，擦除操作同样耗时较长，主端可以采用同写入操作类似的交织操作方式，以提高效率。SCR 寄存器中的 DATA_STAT_AFTER_ERASE(bit 55)定义擦除后状态是 1 或是 0。

4.6.4 寄存器概述

寄存器表 1 SDIO 寄存器概述

偏移地址	名称	描述	其他
0x00	CTRL	控制寄存器	
0x04	PWREN	上电使能寄存器	
0x08	CLKDIV	时钟分频寄存器	
0x0c	CLKSRC	时钟源寄存器	
0x10	CLKENA	时钟使能寄存器	
0x14	TMOUT	超时寄存器	
0x18	CTYPE	卡类型寄存器	
0x1c	BLKSIZ	块大小寄存器	
0x20	BYTCNT	字节计数寄存器	
0x24	INTMASK	中断掩码寄存器	
0x28	CMDARG	指令内容段寄存器	
0x2c	CMD	命令寄存器	
0x30	RESP0	响应回复寄存器 0	
0x34	RESP1	响应回复寄存器 1	
0x38	RESP2	响应回复寄存器 2	
0x3c	RESP3	响应回复寄存器 3	
0x40	MINTSTS	中断掩码寄存器	
0x44	RINTSTS	中断状态寄存器	
0x48	STATUS	状态寄存器，主要用于检视控制器状态	
0x4c	FIFOTH	FIFO 门限寄存器	
0x50	CDETECT	卡检测寄存器	
0x54	WRTPRT	写保护寄存器	

0x58	GPIO	不使用	
0x5c	TCBCNT	CIU 模块传输的 byte 个数寄存器	
0x60	TBBCNT	Host 与 BIU_FIFO 交互数据的 byte 个数寄存器	
0x64	DEBNCE	卡检测反跳寄存器（单位为 host clock 数目）	
0x68	USRID	用户 ID 寄存器	
0x6C	VERID	软核版本 ID 寄存器	
0x70	HCON	硬件配置寄存器	
0x74	UHS_REG	不使用	
0x78	RST_n	硬复位寄存器	
0x80~0x98	DMA_CTRL	DMA 相关，不使用	
0x100	CardThrCtl	卡读门限使能寄存器	
0x104	Back_end_power	Back_end Power 寄存器	
0x108	UHS_REG_ECT	不使用	
0x10c	EMMC_DDR_REG	不使用	
0x110	ENABLE_SHIFT	转换相位控制寄存器	
>=0x200	DATA	FIFO 读写空间	

4.7 GPIO

4.7.1 概述

HR_C7000 支持 3 组 GPIO（GPIOA~GPIOC），每组最多可以有 32 个 GPIO。每个 GPIO 口可单独设为输入输出模式，在输入模式支持中断输入。部分 GPIO 和功能管脚复用。

4.7.2 功能描述

GPIO 模块支持以下功能：

- 支持每个 GPIO 管脚可以设置为输入或输出，可以作为外部中断源输入；
- 当设置为输出模式时，每个 GPIO 管脚可以独立置位或清零；
- 在中断模式下，支持上升沿，下降沿，高电平，低电平这 5 种触发中断方式；
- 支持去抖功能，去抖采样时间可配。

4.7.3 工作方式

省略

4.7.4 寄存器概述

寄存器表 4.12 GPIO 寄存器概述

名称	偏移地址	描述	其他
SWPORTA_DR	0x00	GPIO 数据寄存器	
SWPORTA_DDR	0x04	GPIO 方向寄存器	
INTEN	0x30	中断使能寄存器	
INTMASK	0x34	中断屏蔽寄存器	
INTTYPE_LEVEL	0x38	中断类型器	

INT_POLARITY	0x3c	中断极性寄存器	
INTSTATUS	0x40	中断状态寄存器	
RAW_INTSTATUS	0x44	原始中断状态寄存器	
DEBOUNCE	0x48	去抖使能	
PORTA_EOI	0x4c	中断清除寄存器	
EXT_PORTA	0x50	端口状态	

4.8 Timer

4.8.1 概述

HR_C7000 包含 6 个独立的 32 位定时器（Timer1~6），实现定时、计数功能，可以供操作系统用作系统时钟，也可以供应用程序用作定时和计数。

4.8.2 功能描述

定时器有以下特点：

- Timer1~6可单独使用，各自产生定时器中断；
- 支持自由运行和用户定义计数两种工作模式；
- 支持查询四个定时器的当前计数值。

4.8.3 工作方式

定时器从一个固定的值向下计数，当计数值变为 0 时会产生中断。

每个定时器的初始值会从相应的加载计数寄存器（TimerNLoadCount）里加载。以下两种事件会触发从 TimerNLoadCount 寄存器加载初始值：

- 定时器在复位或被禁用后使能
- 定时器计数到 0

所有的状态寄存器和中断寄存器都可以在任何时刻访问。

4.8.4 寄存器概述

寄存器表 4.13 Timer 寄存器概述

偏移地址	名称	描述	其他
0x00	Timer1LoadCount	定时器 1 加载寄存器	
0x04	Timer1CurrentValue	定时器 1 当前值寄存器	
0x08	Timer1ControlReg	定时器 1 控制寄存器	
0x0c	Timer1EOI	定时器 1 中断清除寄存器	
0x10	Timer1IntStatus	定时器 1 中断状态寄存器	
0x14	Timer2LoadCount	定时器 2 加载寄存器	
0x18	Timer2CurrentValue	定时器 2 当前值寄存器	
0x1c	Timer2ControlReg	定时器 2 控制寄存器	
0x20	Timer2EOI	定时器 2 中断清除寄存器	
0x24	Timer2IntStatus	定时器 2 中断状态寄存器	
0x28	Timer3LoadCount	定时器 3 加载寄存器	

0x2c	Timer3CurrentValue	定时器 3 当前值寄存器	
0x30	Timer3ControlReg	定时器 3 控制寄存器	
0x34	Timer3EOI	定时器 3 中断清除寄存器	
0x38	Timer3IntStatus	定时器 3 中断状态寄存器	
0x3c	Timer4LoadCount	定时器 4 加载寄存器	
0x40	Timer4CurrentValue	定时器 4 当前值寄存器	
0x44	Timer4ControlReg	定时器 4 控制寄存器	
0x48	Timer4EOI	定时器 4 中断清除寄存器	
0x4c	Timer4IntStatus	定时器 4 中断状态寄存器	
0x3c	Timer5LoadCount	定时器 5 加载寄存器	
0x40	Timer5CurrentValue	定时器 5 当前值寄存器	
0x44	Timer5ControlReg	定时器 5 控制寄存器	
0x48	Timer5EOI	定时器 5 中断清除寄存器	
0x4c	Timer5IntStatus	定时器 5 中断状态寄存器	
0x3c	Timer6LoadCount	定时器 6 加载寄存器	
0x40	Timer6CurrentValue	定时器 6 当前值寄存器	
0x44	Timer6ControlReg	定时器 6 控制寄存器	
0x48	Timer6EOI	定时器 6 中断清除寄存器	
0x4c	Timer6IntStatus	定时器 6 中断状态寄存器	
0xa0	TimersIntStatus	所有定时器的中断状态	
0xa4	TimersEOI	清除所有寄存器的中断	
0xa8	TimersRawIntStatus	所有定时器的未屏蔽的中断状态	
0xac	TIMERS_COMP_VERSION	定时器版本号	

4.9 中断（PIC）

4.9.1 概述

中断控制器是用来管理来自内部/外部的异常事件，中断CPU的正常执行程序，使CPU进入指定中断服务函数的模块。

4.9.2 功能描述

中断控制器具有以下特点：

- 支持64个中断源；
- 支持高电平、低电平触发，支持上升沿、下降沿触发；
- 支持中断源优先级可配；
- 支持中断源屏蔽；

中断控制器的功能框图如图所示：

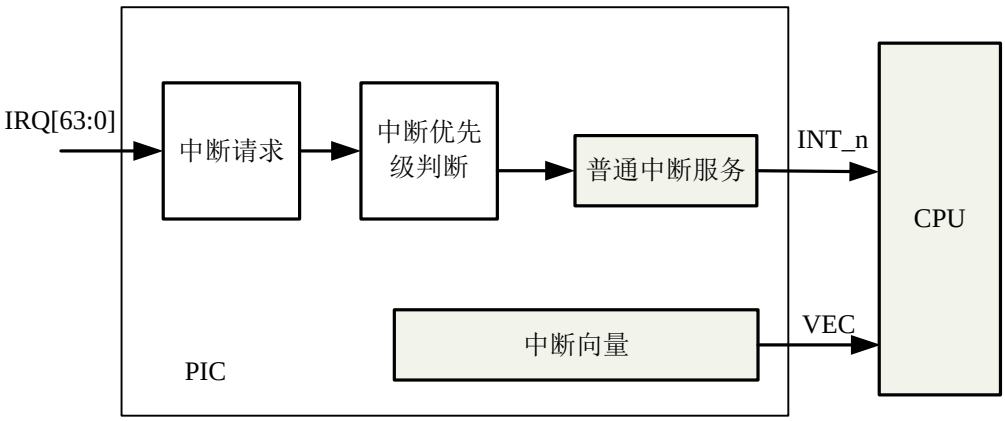


图 4.14 中断控制器功能框图

4.9.2.1 中断源分配

表 4.14 中断源列表

序号	中断源	默认中断向量	中断触发方式
0	保留	32	
1	TIMER_1	33	电平触发，高有效
2	TIMER_2	34	电平触发，高有效
3	TIMER_3	35	电平触发，高有效
4	TIMER_4	36	电平触发，高有效
5	TIMER_5	37	电平触发，高有效
6	TIMER_6	38	电平触发，高有效
7	I2C1	39	电平触发，高有效
8	I2C0	40	电平触发，高有效
9	I2C2	41	电平触发，高有效
10	UART1	42	电平触发，高有效
11	UART2	43	电平触发，高有效
12	UART0	44	电平触发，高有效
13	UART3	45	电平触发，高有效
14	GPIOB	46	电平触发，高有效
15	GPIOA	47	电平触发，高有效
16	SPI0	48	电平触发，高有效
17	GPIOC	49	电平触发，高有效
18	PWM	50	电平触发，高有效
19	RTC	51	电平触发，高有效
20	SDIO	52	电平触发，高有效
21	USB	53	电平触发，高有效
22	SPI1	54	电平触发，高有效
23	SPI2	55	电平触发，高有效
24	LCSFC	56	电平触发，高有效
25	DAC	57	电平触发，高有效
26	ADC	58	电平触发，高有效

27	PCM_rd_data_interp	59	电平触发，高有效
28	PCM_wr_data_interp	60	电平触发，高有效
29	System_inter	61	电平触发，高有效
30	Time_slot_inter_tx	62	电平触发，高有效
31	Time_slot_inter_rx	63	电平触发，高有效
32	Rf_tx_inter	64	电平触发，高有效
33	Rf_rx_inter	65	电平触发，高有效
34	SlaveSPI0	66	电平触发，高有效
35	SlaveSPI1	67	电平触发，高有效
36	SlaveSPI2	68	电平触发，高有效
37	Codec_interp	69	电平触发，高有效

4.9.4 工作方式

4.9.3.1 初始化

中断控制器默认屏蔽所有中断请求，使用前需先配置中断控制器寄存器，步骤如下：

1. 配置中断触发方式：
   通过配置 PIC_MODE 可设置中断源的触发方式；
2. 配置中断极性：
   通过配置 PIC_PO 可设置中断源的触发极性；
3. 配置中断源优先级、快速中断等
   - 配置 PIC_PRIOR0-3～PIC_PRIOR60-63，表示各个中断源优先级计数器的初始值。初始值小表示优先级高。如果不配置，则使用默认优先级；
   - 配置 PIC_FFLAG_L，PIC_FFLAG_H 可以将任意中断源配置成快速中断，默认全部是普通中断。
4. 清中断状态（非必要）
   配置 PIC_INT_ST，PIC_INT_ST_1 等于 0xffffffff，清楚中断记录寄存器 PIC_INT_ST，PIC_INT_ST_1，PIC_INT_ST_CNT。
5. 开启中断检测（重要）
   配置 PIC_MASK，PIC_MASK_1 对应 bit 为 0，开启中断源中断请求检测功能；

4.9.3.2 中断状态查询

通过查询PIC_INT_ST，可获知哪些中断源发出过中断请求，每个中断源只能记录一次中断请求。对应位写1可清除状态。

4.9.4 寄存器概述

寄存器表 4.15 PIC 寄存器概述

偏移地址	名称	描述	其他
0x00	PIC_MODE	低 32 位中断源触发方式选择	
0x04	PIC_PO	低 32 位中断源的触发极性选择	
0x08	PIC_MASK	低 32 位中断源是否屏蔽	
0x0c	PIC_VECTOR	中断向量基址	
0x10	PIC_COW1	中断结束	
0x14	PIC_PRIOR0-3	中断源 0-3 对应的计数器初始值	
0x18	PIC_PRIOR4-7	中断源 4-7 对应的计数器初始值	

0x1c	PIC_PRIOR8-11	中断源 8-11 对应的计数器初始值	
0x20	PIC_PRIOR12-15	中断源 12-15 对应的计数器初始值	
0x24	PIC_PRIOR16-19	中断源 16-19 对应的计数器初始值	
0x28	PIC_PRIOR20-23	中断源 20-23 对应的计数器初始值	
0x2c	PIC_PRIOR24-27	中断源 24-27 对应的计数器初始值	
0x30	PIC_PRIOR28-31	中断源 28-31 对应的计数器初始值	
0x34	PIC_COW2	控制寄存器	
0x38	PIC_SYNC	异步处理控制寄存器	
0x3c	PIC_FFLAG_L	低 32 位快速中断设置	
0x40	PIC_RECORD_SEL	选择一个中断源记录中断次数	
0x44	PIC_INT_ST	0-31 中断请求记录	
0x48	PIC_INT_ST_1	32-63 中断请求记录	
0x4c	PIC_INT_CNT	中断源中断次数计数	
0x60	PIC_MODE_1	高 32 位中断源触发方式选择	
0x64	PIC_PO_1	高 32 位中断源的触发极性选择	
0x68	PIC_MASK_1	高 32 位中断源是否屏蔽	
0x6c	PIC_PRIOR32-35	中断源 32-35 对应的计数器初始值	
0x70	PIC_PRIOR36-39	中断源 36-39 对应的计数器初始值	
0x74	PIC_PRIOR40-43	中断源 40-43 对应的计数器初始值	
0x78	PIC_PRIOR44-47	中断源 44-47 对应的计数器初始值	
0x7c	PIC_PRIOR48-51	中断源 48-51 对应的计数器初始值	
0x80	PIC_PRIOR52-55	中断源 52-55 对应的计数器初始值	
0x84	PIC_PRIOR56-59	中断源 56-59 对应的计数器初始值	
0x88	PIC_PRIOR60-63	中断源 60-63 对应的计数器初始值	
0x8c	PIC_FFLAG_H	高 32 位快速中断设置	

4.10 WatchDog

4.10.1 概述

看门狗用于在系统死机时对系统重新复位。

4.10.2 功能描述

看门狗主要有两个特点：

- 一旦有外部物理复位信号到来，看门狗输出低电平复位信号并持续 510 个系统时钟时间，延长物理复位时间，保证系统充分复位；
- 防止系统死机，软件需要在一定时间周期内发出敲门信号给看门狗，在预定时间内看门狗没有得到激励，则认为系统死机，将自动对系统进行复位，复位电平持续时间为 510 个系统时钟。

4.10.3 工作方式

4.10.3.1 开启看门狗

系统启动后配置 WD_CW.wde=1，即可开启看门狗。

4.10.3.2 关闭看门狗

关闭看门狗的操作略显繁琐，这是为了防止软件误操作或者电磁干扰等原因而意外关闭看门狗。关闭过程如下：

1. 在wde为高的情况下，将WD_CW.wdtoe置1；
2. 将WD_CW.wde置0，此时看门狗被关闭；
3. 将WD_CW.wdtoe置0为初始值。

关闭之后，看门狗的初始计数值，以及其他控制字均未被改变，可以在开启看门狗后继续使用，时序图如下：

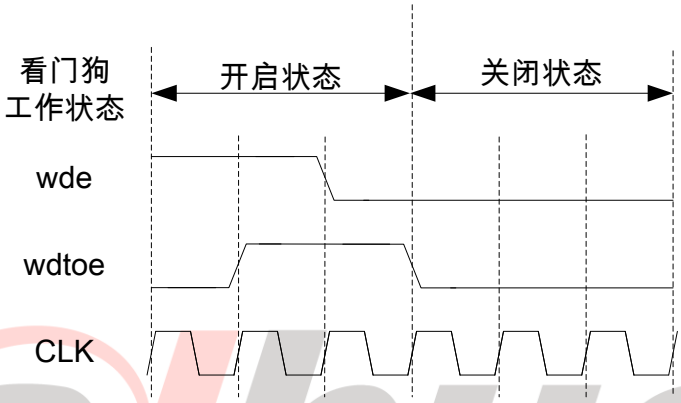


图 4.15 看门狗关闭时序图

表 4.16 看门狗工作状态表

wdtoe	wde	看门狗工作情况
0	0	不允许
0	1	开启
1	0	关闭（从11跳入才关闭，直接从10跳入不会关闭）
1	1	开启

4.10.3.3 正常工作

正常工作模式下，需要系统周期性地写看门狗寄存器 WD_CW.knocker，如果超过指定周期没有写该寄存器，则看门狗认为系统死机，会重新复位系统。

1. 配置WD_TIMER，设置看门狗周期初始值；
2. 配置WD_CW.wde=1开启看门狗；
3. 在内部计数器从初始值WD_TIMER计数到溢出值之前，配置一次WD_CW.knocker=1，完成敲门动作。敲门后，内部计数器重新从WD_TIMER开始计数。

4.10.4 寄存器概述

寄存器表 4.17 WDT 寄存器概述

偏移地址	名称	描述	其他
0x38	WD_TIMER	看门狗时间周期初始值	

0x3c	WD_CW	看门狗控制	

4.11 USB

4.11.1 概述

HR_C7000 采用 Non-OTG Device 配置模式，实现 USB1.1 Device 功能。

4.11.2 功能描述

Non-OTG Device 主要配置参数如下：

- 工作模式：Non-OTG Device
- 结构：内部 DMA
- SG-DMA：无
- HS PHY 接口：无
- USB1.1 FS 收发器接口：专用 FS 接口
- 总 EP 数目：1 CTRL IN/OUT(EP0) + 2 IN(EP1/3) + 2 OUT(EP2/4)，共计 5 个 EP
- TxFIFO 模式：专用 TxFIFO
- 总 FIFO 深度：128(FIFO 位宽为 35bit)

4.11.3 工作方式

Dev 模式下，软核 MAC 接收 Host 发来的各种包，并检测包的完整性，具体流程如下：

- 当接收到 OUT 或 SETUP 信令包时，软核 MAC 等待并检测随之而来的数据包 PID 号，然后将数据写入 RxFIFO(可用)。数据包接收完成后，软核 MAC 检测数据包的完整性，并向 Host 做相应的应答。若软核在收到 OUT 信令但 RxFIFO 不可用时，MAC 应答 NAK；
- 当接收到 IN 信令包且 TxFIFO 可用时，MAC 自动从 TxFIFO 读取数据，并发送数据包，接着等待 Host 可能的应答。若收到 IN 信令但 TxFIFO 无数据时，MAC 应答 NAK。

软核收发数据包时，各种中断和配置有**严格**的前后时序关系，若配置内容或配置时间点不正确，则软核无法正常工作。本小节抽出通用的中断触发时序和配置时序，方便上层软件开发者查询，图例说明如下：

表 4.18 图例符号定义

编号	用例名称	描述
1		Host 包
2		Dev 包
3	黄色文字	触发 GINTSTS.RxFLvl 中断，有包写入 RxFIFO
4	蓝色文字	设置 IN/OUT 端口传输属性 若有数据发送，可向 TxFIFO 压入数据
5	黑色文字	传输完成中断，标示传输完成
6	时间先后关系	从左到右，从上到下

1) 带有数据输出的 CTRL 传输

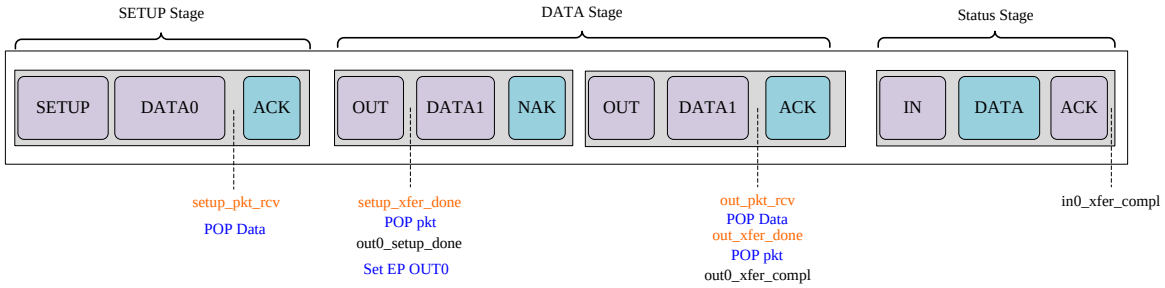


图 5 CTRL OUT 传输时序

2) 带有数据输入的 CTRL 传输时序

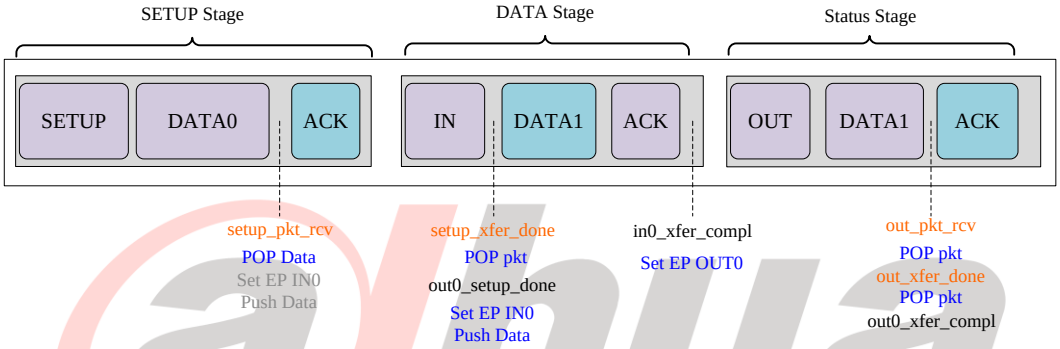


图 6 CTRL IN 传输时序

3) 无数据的 CTRL 传输时序

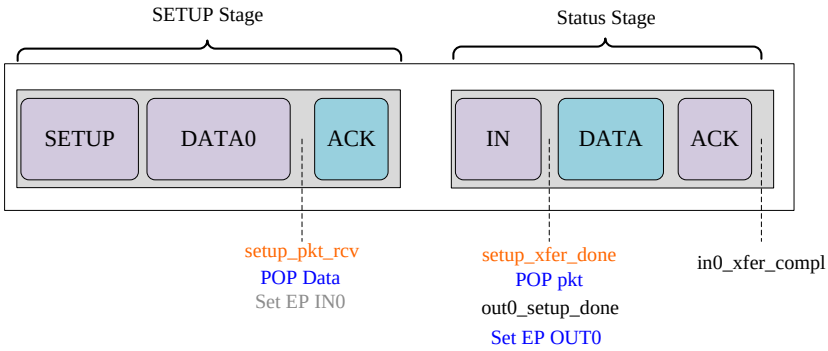


图 4.18 无数据 CTRL 传输时序

4) 单次 OUT 事务

在接收到 OUT 信令时，尚未正确设置 OUT 端口参数时的时序；若接收前已经配置好 OUT 端口参数，则 OUT 端口直接应答 ACK；

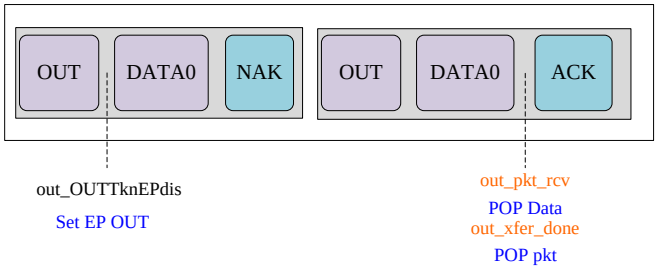


图 7 单次 OUT 事务时序

5) 多次 OUT 事务

设置 OUT 端口接收多个包后，应用软件在 RxFLvl 中断服务程序中从 RxFIFO 中不断读取数据，当全部数据传输完成后，软核触发完成中断。若在此过程中，Host 未收到某个 ACK，Host 会重发该数据包，软核自动应答 ACK、且无 RxFLvl 中断触发，软件无需干预；

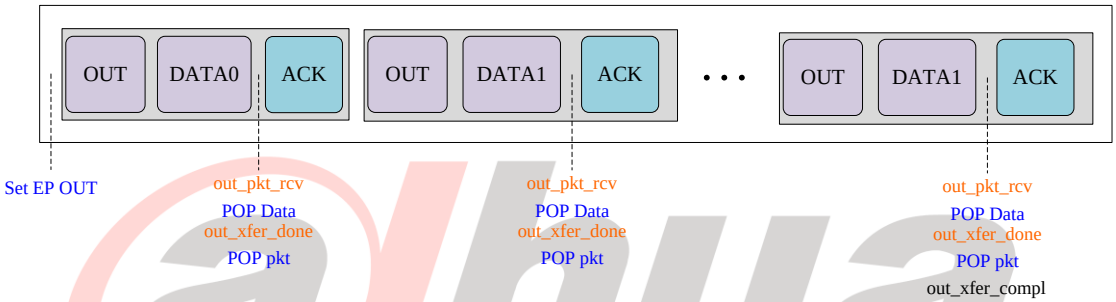


图 4.20 多次 OUT 事务时序

6) 单次 IN 事务时序

IN 端口在接收到 IN 信令时为 TxFIFO 未就绪时的时序；若接收 IN 信令前，IN 端口设置及 TxFIFO 均就绪时，则 IN 端口直接应答数据包；

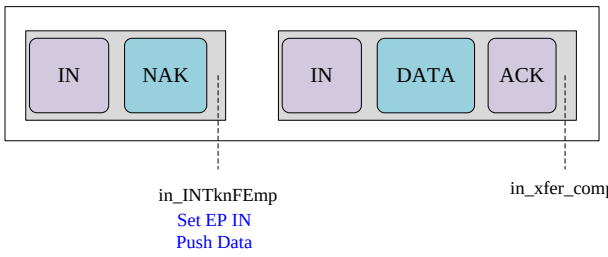


图 4.21 单次 IN 事务时序

7) 多次 IN 事务时序

设置 IN 端口连续发送多个包后，应用软件在 TxFEmp 中断服务程序中向 TxFIFO 不断压入数据，当全部数据传输完成后(也即均收到 Host 发送的 ACK)，软核触发完成中断。若在此过程中，Dev 未收到某个 ACK，软核自动重发该数据包，不需要软件干预；

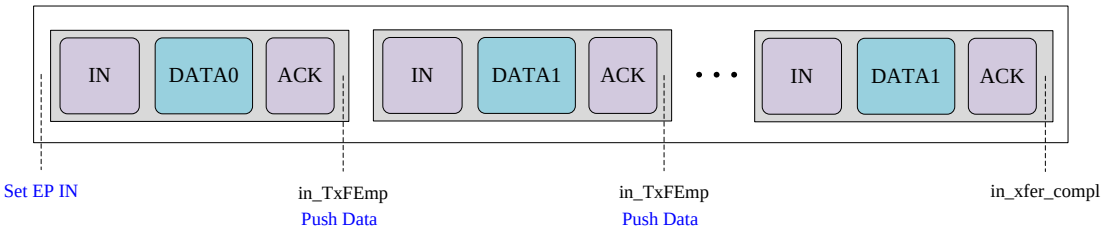


图 4.22 多次 IN 事务时序

4.11.4 寄存器概述

寄存器表 4.19 USB 寄存器概述

偏移地址	名称	描述
0x008	GAHBCFG	AHB Configuration，全局 AHB 配置寄存器
0x00c	GUSBCFG	USB Configuration，全局 USB 配置寄存器
0x010	GRSTCTL	Reset，全局复位控制寄存器
0x014	GINTSTS	Interrupt，全局中断状态寄存器
0x018	GINTMSK	Interrupt Mask，全局中断屏蔽控制寄存器
0x01c	GRXSTSR	Receive Status Debug Read，全局接收状态读取寄存器
0x020	GRXSTSP	Status Read and Pop，全局接收状态读出寄存器
0x024	GRXFSIZ	Receive FIFO Size，RxFIFO 分配控制寄存器
0x028	GNPTXFSIZ	Non-Periodic Transmit FIFO Size，Non-Periodic TxFIFO 分配控制寄存器
0x038	GGPIO	General Purpose IO，通用 IO 寄存器
0x03c	GUID	User ID，用户 ID 寄存器
0x040	GSNPSID	Synopsys ID，版本号只读寄存器(0x4f54_270a)
0x044	GHWCFG1	User HW Config1，软核配置状态寄存器 1
0x048	GHWCFG2	User HW Config2，软核配置状态寄存器 2
0x04c	GHWCFG3	User HW Config3，软核配置状态寄存器 3
0x050	GHWCFG4	User HW Config4，软核配置状态寄存器 4
0x054	GUSRSTS	User States，用户定义观察信号
0x104	DIEPTXF1	IN Endpoint1 TxFIFO Depth，端口 1 的 TxFIFO 深度配置寄存器
0x108	DIEPTXF3	IN Endpoint3 TxFIFO Depth，端口 3 的 TxFIFO 深度配置寄存器
0x800	DCFG	Device Configuration，设备总体配置寄存器
0x804	DCTL	Device Control，设备总体控制寄存器
0x808	DSTS	Device Status，设备总体状态寄存器
0x810	DIEPMSK	Device IN Endpoint Common Interrupt Mask，设备 IN 端口总体中断屏蔽
0x814	DOEPMSK	Device OUT Endpoint Common Interrupt Mask，设备 OUT 端口总体中断屏蔽寄存器
0x818	DAINT	Device All Endpoints Interrupt，设备全部端口中断状态寄存器
0x81c	DAINTMSK	Device All Endpoints Interrupt Mask，设备全部端口中断屏蔽控制寄存器
0x830	DTHRCTL	Device Threshold Control，设备门限控制寄存器
0x834	DIEPEMPMSK	Device IN Endpoint FIFO Empty Interrupt Mask，设备 IN 端口 FIFO 空中断屏蔽寄存器

0x900	DIEPCTL0	Device Control IN Endpoint0 Control，设备 IN EP0 控制寄存器
0x908	DIEPINT0	Device IN Endpoint0 Interrupt，设备 IN EP0 中断状态寄存器
0x910	DIEPTSIZ0	Device IN Endpoint0 Transfer Size，设备 IN EP0 传输大小设置寄存器
0x914	DIEPDMA0	Device IN Endpoint0 DMA Addr，设备 IN EP0 DMA 地址
0x918	DTXFSTS0	Device IN Endpoint0 Transmit FIFO Status，设备 IN EP0 TxFIFO 状态
0xb00	DOEPCTRL0	Device Control OUT Endpoint0 Control，设备 OUT EP0 控制寄存器
0xb08	DOEPINT0	Device OUT Endpoint0 Interrupt，设备 OUT EP0 中断状态寄存器
0xb10	DOEPTSIZ0	Device OUT Endpoint0 Transfer Size，设备 OUT EP0 传输大小设置寄存器
0xb14	DOEPDMA0	Device OUT Endpoint0 DMA Addr，设备 OUT EP0 DMA 地址
0x920	DIEPCTL1	Device Control IN Endpoint1 Control，设备 IN EP1 控制寄存器
0x928	DIEPINT1	Device IN Endpoint1 Interrupt，设备 IN EP1 中断状态寄存器
0x930	DIEPTSIZ1	Device IN Endpoint1 Transfer Size，设备 IN EP1 传输大小设置寄存器
0x934	DIEPDMA1	Device IN Endpoint1 DMA Addr，设备 IN EP1 DMA 地址
0x938	DTXFSTS1	Device IN Endpoint1 Transmit FIFO Status，设备 IN EP1 TxFIFO 状态
0xb40	DOEPCTRL2	Device Control OUT Endpoint2 Control，设备 OUT EP2 控制寄存器
0xb48	DOEPINT2	Device OUT Endpoint2 Interrupt，设备 OUT EP2 中断状态寄存器
0xb50	DOEPTSIZ2	Device OUT Endpoint2 Transfer Size，设备 OUT EP2 传输大小设置寄存器
0xb54	DOEPDMA2	Device OUT Endpoint2 DMA Addr，设备 OUT EP2 DMA 地址
0x960	DIEPCTL3	Device Control IN Endpoint3 Control，设备 IN EP3 控制寄存器
0x968	DIEPINT3	Device IN Endpoint3 Interrupt，设备 IN EP3 中断状态寄存器
0x970	DIEPTSIZ3	Device IN Endpoint3 Transfer Size，设备 IN EP3 传输大小设置寄存器
0x974	DIEPDMA3	Device IN Endpoint3 DMA Addr，设备 IN EP3 DMA 地址
0x978	DTXFSTS3	Device IN Endpoint3 Transmit FIFO Status，设备 IN EP3 TxFIFO 状态
0xb80	DOEPCTRL4	Device Control OUT Endpoint4 Control，设备 OUT EP4 控制寄存器
0xb88	DOEPINT4	Device OUT Endpoint4 Interrupt，设备 OUT EP4 中断状态寄存器
0xb90	DOEPTSIZ4	Device OUT Endpoint4 Transfer Size，设备 OUT EP4 传输大小设置寄存器
0xb94	DOEPDMA4	Device OUT Endpoint4 DMA Addr，设备 OUT EP4 DMA 地址
0xe00	PCGCCTL	Power and Clock Gating Control，电源和时钟门控控制寄存器
0x1000	DEVEP0FF	EP0 FIFO PUSH/POP 入口地址
0x2000	DEVEP1FF	EP1 FIFO PUSH/POP 入口地址
0x3000	DEVEP2FF	EP2 FIFO PUSH/POP 入口地址
0x4000	DEVEP3FF	EP3 FIFO PUSH/POP 入口地址
0x5000	DEVEP4FF	EP4 FIFO PUSH/POP 入口地址

4.12 RTC

4.12.1 概述

DH4570 包含一个实时时钟模块，负责系统的计时，可保存当前世界时间，可以设置某个未来时间点产生中断信号。

4.12.2 功能描述

RTC 主要有以下特点：

- RTC 内部使用 6 位宽的秒分寄存器，5 位宽的时寄存器，16 位宽的日寄存器，秒寄存器使用 1hz 的时钟计数，当计数到 60 时分寄存器加一，秒寄存器自身清零。依次类推。其中天寄存器有 16 位，可以计 65526 天，相当于 179 年。
- 中断产生时间可配置，并可以控制中断是否产生。
- RTC 初始时间可配置。
- 可以控制 RTC 开始或停止。
- 芯片掉电以后 RTC 依然工作，使用外部纽扣电池为其供电。

4.12.3 工作方式

DH4570 中 RTC 挂载于内部 I2C2 上，通过 I2C 总线进行操作。RTC 计时时钟为外部提供的 32.768KHz 晶振。

4.12.3.1 RTC 初始化

RTC 在首次上电时，系统需要将 RTC 初始化。RTC 的初始化过程如下：

1. 拉低 rtc_rst_n，复位 RTC 的 I2C 接口模块。
2. 等待 60ns（3 个 apb 时钟周期）。
3. 配置 RTC_SOFT_RST_N 寄存器为 0，执行软复位。（设备地址 7’b1110000）
4. 等待 25ms（计数器在 128hz 的时钟下工作，异步复位的同步撤离需要 23.44ms）。
5. 配置 RTC_SOFT_RST_N 寄存器为 1，撤销软复位。
6. 依次配置 RTC_MATCH_S、RTC_MATCH_M、RTC_MATCH_H、RTC_MATCH_D_L、RTC_MATCH_D_H，设置 RTC 中断产生的时间，配置 RTC_MATCH_VEN[0]为 1 更新中断产生时间（此位会自动清零），配置 RTC_MATCH_VSTAT[0]为 1 表示当前中断产生时间已经配置，如果需要重新配置中断产生时间时先将此位配置为 0，配置中断产生时间以后再将此位配置为 1。
7. 依次配置 RTC_LOAD_S、RTC_LOAD_M、RTC_LOAD_H、RTC_LOAD_D_L、RTC_LOAD_D_H，设置 RTC 的初始时间，配置 RTC_CCR[2]为 1 更新当前的时间（此位会自动清零），配置 RTC_LOAD_VSTAT[0]为 1 表示初始时间已经设置，如果需要重新配置初始时间时先将此位配置为 0，配置初始时间以后再将此位配置为 1。
8. 配置 RTC_CCR[0]为 1，使能中断。
9. 配置 RTC_CCR[1]为 1，让计时器开始计时。
10. RTC 以 1HZ 的计数时钟频率，从 RTC_LOAD_S、RTC_LOAD_M、RTC_LOAD_H、RTC_LOAD_D_L、RTC_LOAD_D_H 中的值开始计数，当计数值到达 RTC_MATCH_S、RTC_MATCH_M、RTC_MATCH_H、RTC_MATCH_D_L、RTC_MATCH_D_H 中的值时，将根据 RTC_CCR[0]的设置，决定是否产生中断。

4.12.3.2 中断处理

系统收到 RTC 发出的中断以后，表示定时的时间到，用户可以执行相应的的自定义操作，RTC 计数器任保持计数。RTC 中断的处理过程如下：

1. 读 RTC_RSTAT，清除 RTC 的中断状态。
2. 如果还需要设置定时时间，先关闭中断使能，配置 RTC_MATCH_VSTAT[0]=0，向 RTC_MATCH_S、RTC_MATCH_M、RTC_MATCH_H、RTC_MATCH_D_L、RTC_MATCH_D_H 设置新的 RTC 比较值，配置 RTC_MATCH_VEN[0]=1，RTC_MATCH_VSTAT[0]=1 再开启中断使能。

4.12.4 寄存器概述

寄存器表 4.20 RTC 寄存器概述

偏移地址	名称	描述	其他
0x00	RTC_VAL_S	RTC_VAL_S 的总体参数配置	
0x01	RTC_VAL_M	RTC_VAL_M 的总体参数配置	
0x02	RTC_VAL_H	RTC_VAL_H 的总体参数配置	
0x03	RTC_VAL_D_L	RTC_VAL_D_L 的总体参数配置	
0x04	RTC_VAL_D_H	RTC_VAL_D_H 的总体参数配置	
0x05	RTC_MATCH_S	RTC_MATCH_S 的总体参数配置	
0x06	RTC_MATCH_M	RTC_MATCH_M 的总体参数配置	
0x07	RTC_MATCH_H	RTC_MATCH_H 的总体参数配置	
0x08	RTC_MATCH_D_L	RTC_MATCH_D_L 的总体参数配置	
0x09	RTC_MATCH_D_H	RTC_MATCH_D_H 的总体参数配置	
0x0a	RTC_MATCH_VEN	RTC_MATCH_VEN 的总体参数配置	
0x0b	RTC_MATCH_VSTAT	RTC_MATCH_VSTAT 的总体参数配置	
0x0c	RTC_LOAD_S	RTC_LOAD_S 的总体参数配置	
0x0d	RTC_LOAD_M	RTC_LOAD_M 的总体参数配置	
0x0e	RTC_LOAD_H	RTC_LOAD_H 的总体参数配置	
0x0f	RTC_LOAD_D_L	RTC_LOAD_D_L 的总体参数配置	
0x10	RTC_LOAD_D_H	RTC_LOAD_D_H 的总体参数配置	
0x11	RTC_CCR	RTC_CCR 的总体参数配置	
0x12	RTC_LOAD_VSTAT	RTC_LOAD_VSTAT 的总体参数配置	
0x13	RTC_RSTAT	RTC_RSTAT 数配置	
0x14	RTC_CLKDIV_NUMER_L	RTC_CLKDIV_NUMER_L 数配置	
0x15	RTC_CLKDIV_NUMER_H	RTC_CLKDIV_NUMER_H 数配置	
0x16	RTC_CLKDIV_NUMER_VLD	RTC_CLKDIV_NUMER_VLD 数配置	
0x17	RTC_SOFT_RST_N	RTC_SOFT_RST_N 数配置	

4.13 ADC

4.13.1 概述

HR_C7000 内置 ADC。ADC 控制器用于根据上层软件配置，控制 ADC 正常采样，并将采样数据上传。

4.13.2 功能描述

- ADC 控制器具有如下功能：
- 数据转换速率 200KSPS
- 采样精度 10 bit
- 8 个 ADC 通道
- 支持单通道扫描功能

- 支持连续扫描功能
- 支持连续扫描时低功耗模式，ADC 自动开启或退出低功耗状态
- 采用可屏蔽中断方式上报 ADC 采样数据
- 支持采样初始值及变化阈值配置，自动过滤采样信号都抖动
- 支持 ADC 异常状态中断，并设计 ADC 强制软复位功能
- 支持可调 ADC 上电/掉电保护时间间隔
- 支持将 ADC 采样数据直接串行输出到片外

4.13.3 工作方式

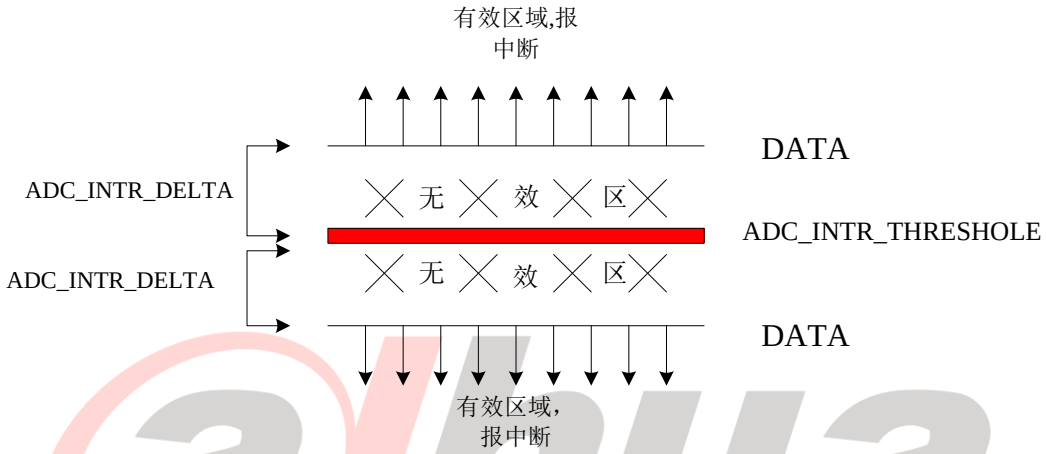


图 4.23 ADC_INTR_THRESHOLE 与 ADC_INTR_DELTA 关系示意图

上图中，从 ADC 采样到的数据 DATA，与 ADC_INTR_THRESHOLE 做差值，差值的绝对值大于 ADC_INTR_DELTA 的数据，上报中断。

4.13.3.1 单次扫描处理流程

在单次扫描模式下，配置扫描模式（ADC_CTRL[2]=1'b0），是否支持低功耗（ADC_CTRL[0]）、有效通道号 ADC_CH_VLD 等相关控制信息，ADC_CTRL 启动（ADC_START），启动一次扫描，等待 ADC 转换结果，ADC_CTRL 获取 ADC 转换值，上报中断。单次扫描不滤毛刺。
具体操作流程如下：

1) 配置全局控制器寄存器 0x1100_0010=0x0，关闭 ADC 模块低功耗模式；
2) 配置控制器 ADC_CTRL(0x0) = 0x8，软复位 ADC 模块和控制器；
3) 配置控制器 ADC_CTRL_STOP（0x2c）寄存器的 bit1，软复位控制器。
4) 读取 ADC_CTRL_STATE(0x20)和 ADC_INTR_STA(0x14)，确认控制器当前状态，若 ADC_CTRL_STATE 寄存器 ADC_CTRL_BUSY 位和 SAMP_FSM_STATE 位不为零，则需要重复步骤 3)，继续复位控制器直至控制器进入 IDLE 态；
5) 配置 ADC_CTRL(0x0) = 0x0，使能单次扫描模式；
6) 配置 ADC_INTR_THRESHOLE(0x24)，设置采样数据触发中断水平；
7) 配置 ADC_DATA_DELTA(0x4)，设置采样信号与触发水平值误差范围；
8) 配置 ADC_PD_SEOC_TIME(0x34)，设置 PD 上电和掉电保护时间段；
9) 配置 ADC_P2S_EN(0x38)，开启或关闭 ADC 数据串行输出；

10) 配置 ADC_INTR(0x10)[8] = 1’b1，及 ADC_INTR[4:0]使能相应中断；
11) 配置 ADC_CH_VLD(0x30)，使能采样通道。若此时配置多通道有效，控制器仅输出最低通道号数据，例如配置 ADC_CH_VLD=0x6，则仅输出 B 通道数据；
12) 设置 ADC_START(0x1c)=0x1，启动 ADC 单次单通道采样；
13) 等待 ADC_INTR_SAT(0x14)相应通道号中断置位后，从 0x8 或 0xc 寄存器中读取相应采样数据，并清除中断寄存器(0x14)。

注意：

a) 步骤 1)-2)和 5)-9)仅需要在第一次操作时配置，之后操作无需再重复；每次启动新一次扫描，其他步骤都需要重复操作。
b) 务必在开启采样前读清除 ADC_INTR_STA(0x14)以及配置 ADC_STOP=0x2 复位控制器，
c) 不得在 ADC_START 后至 ADC_CTRL_BUSY 清零前，写入软复位（也即 ADC_CTRL(0x00) bit4 和 ADC_STOP(0x2c)）之外的任何寄存器。

4.13.3.2 单次扫描软件操作流程：

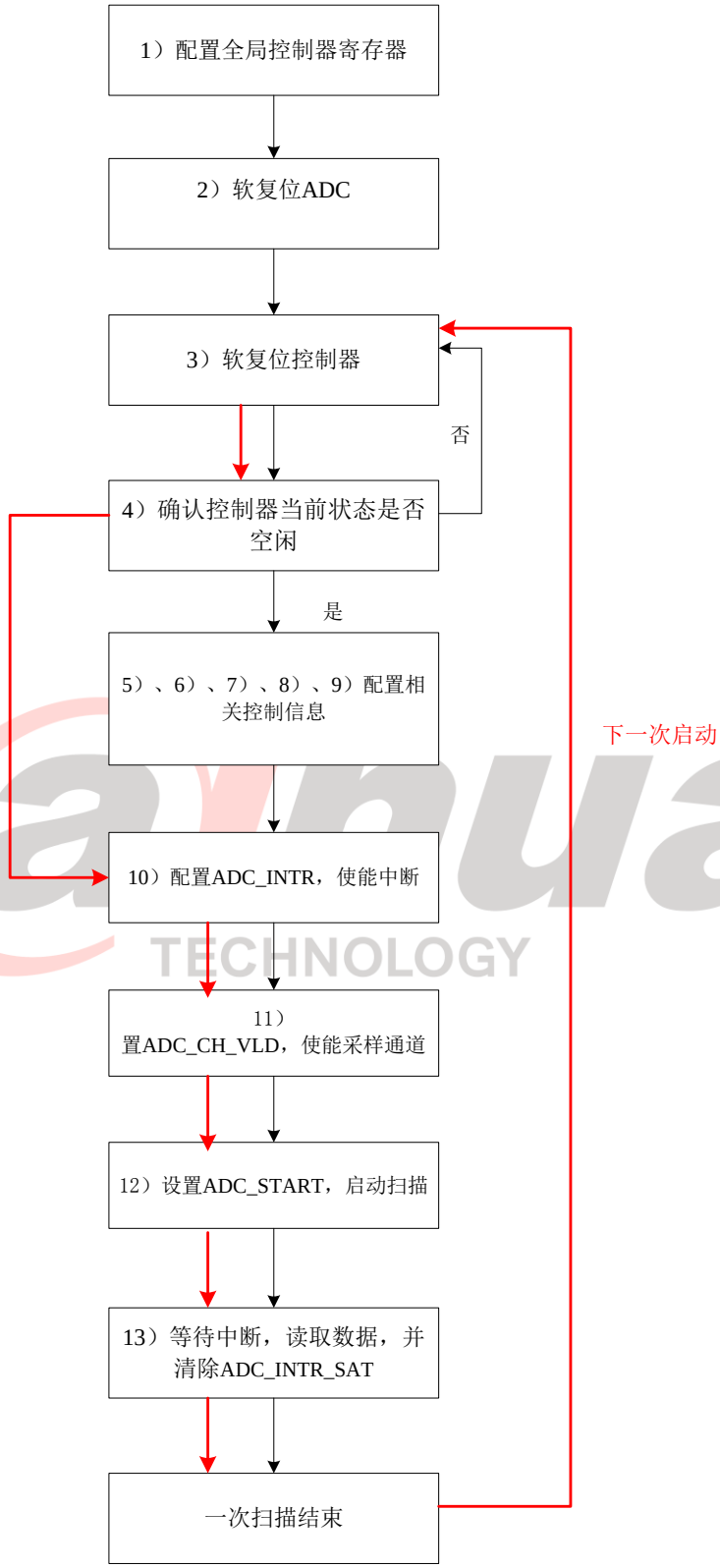


图 4.24 ADC_CTRL 单次扫描软件操作流程

上图中，黑色箭头流程为复位以后，第一次扫描软件操作流程。红色箭头部分为再次启动软件操作流程。

4.13.3.3 连续扫描处理流程

在连续扫描模式下，配置扫描模式（ADC_CTRL[2]=1’b1），是否滤毛刺（ADC_CTRL[1]）、是否支持低功耗模式（ADC_CTRL[0]）连续扫描时间间隔 ADC_SCAN_TIME、有效通道号 ADC_CH_VLD，ADC_CTRL 启动（ADC_START），启动扫描，在一个时间间隔 Tscan 内完成一个有效通道的扫描，在下一个扫描时刻到来时，启动对下一个有效通道的扫描。

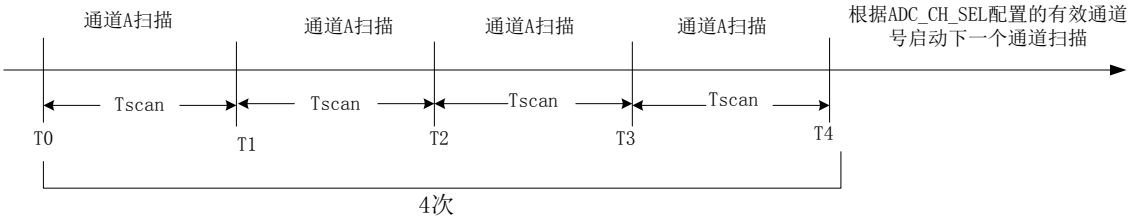


图 4.25 ADC_CTRL 连续扫描通道轮询示意图

具体操作流程如下：

1) 配置全局寄存器 0x1100_0010=0x0，关闭 ADC 模块低功耗模式；
2) 配置控制器 ADC_CTRL(0x0) = 0x8，复位 ADC 模块和控制器；
3) 配置控制器 ADC_CTRL_STOP（0x2c）寄存器的 bit1，软复位控制器
4) 读取 ADC_CTRL_STATE(0x20)和 ADC_INTR_SAT(0x14)，确认控制器当前状态，若 ADC_CTRL_STATE 寄存器 ADC_CTRL_BUSY 位和 SAMP_FSM_STATE 位不为零，则需要重复步骤 3），继续复位控制器直至控制器进入 IDLE 态；
5) 配置 ADC_CTRL(0x0)，使能连续扫描模式，其中 ADC_CTRL= 0x4 表示使能连续扫描模式；ADC_CTRL= 0x5 表示使能低功耗连续扫描模式；ADC_CTRL= 0x6 表示使能滤毛刺连续扫描模式；ADC_CTRL= 0x7 表示使能低功耗滤毛刺连续扫描模式；
6) 配置 ADC_INTR_THRESHOLE(0x24)，设置采样数据触发中断水平；
7) 配置 ADC_DATA_DELTA(0x4)，设置采样信号与触发水平值误差范围；
8) 配置 ADC_PD_SEOC_TIME(0x34)，设置 PD 上电和掉电保护时间段；（步骤 8）必须要先于步骤 9）配置）
9) 配置 ADC_SCAN_TIME(0x18)，设置扫描时间，若扫描时间值小于 ADC_PD_SEOC_TIME (0x34) EOC_PD_TIME+12，则实际写入值为 EOC_PD_TIME+12，且 ADC_CTRL_STATE (0x20)寄存器 SCAN_TIME_ERR 置位;
10) 配置 ADC_P2S_EN(0x38)，开启或关闭 ADC 数据串行输出；
11) 配置 ADC_INTR(0x10)[8] = 1’b1，及 ADC_INTR[4:0]使能相应中断；
12) 配置 ADC_CH_VLD(0x30)，使能采样通道。控制器按照从低到高的顺序依次采样各通道，例如配置 ADC_CH_VLD=0x6，控制器依次输出 B、C 通道数据；
13) 设置 ADC_START(0x1c)=0x1，启动 ADC 连续采样；
14) 等待 ADC_INTR_SAT(0x14)相应通道号中断置位后，从 0x8 或 0xc 寄存器中读取相应采样数据，并清除中断寄存器(0x14)。

注意：

a) 若相关配置参数不变，步骤 1)-2)和 5)-10)仅需要在第一次操作时配置，之后操作无需再重复；每次启动新一次扫描， 其他步骤都需要重复操作
b) 务必在开启采样前读清除 ADC_INTR_STA(0x14)以及配置 ADC_STOP=0x3 复位控制器；

c) 连续采样结束时务必配置 ADC_STOP=0x1，终止连续采样模式；
d) 不得在 ADC_START 后至 ADC_CTRL_BUSY 清零前，写入软复位（也即 ADC_CTRL(0x00) bit4 和 ADC_STOP(0x2c)）之外的任何寄存器。

4.13.3.4 连续扫描软件操作流程

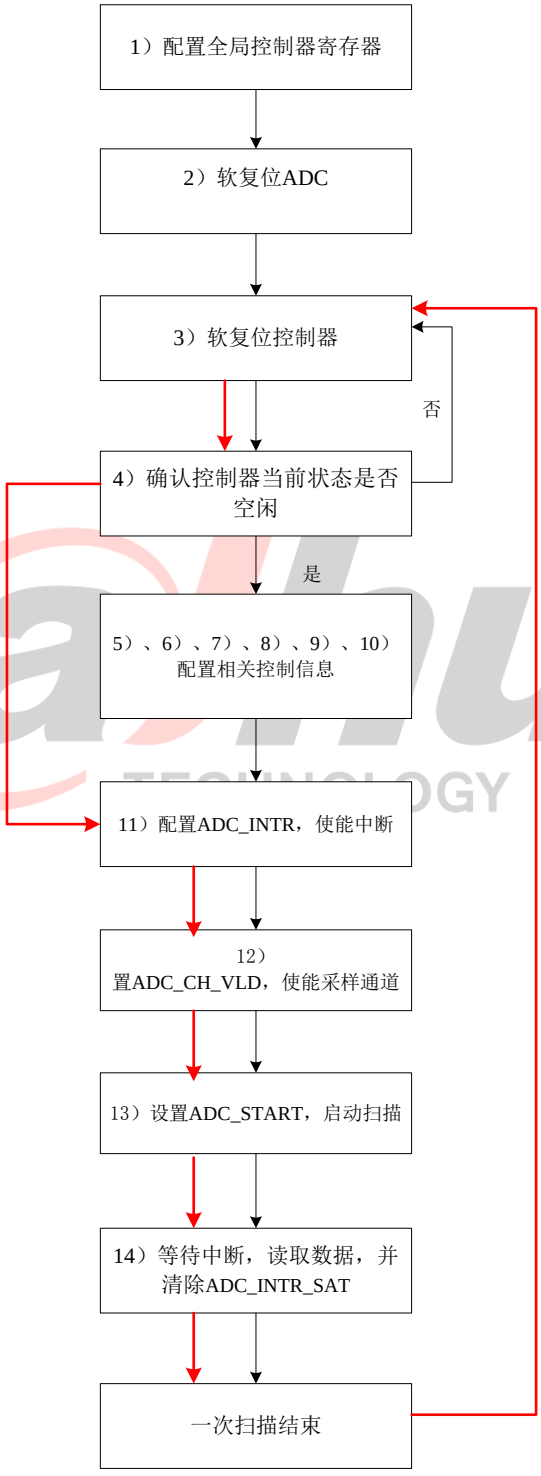


图 8 ADC_CTRL 连续扫描软件操作流程图

上图中，ADC_CTRL 连续扫描软件操作流程图中，下一次启动时，配置相关控制步骤，若是

否滤毛刺、是否支持低功耗等控制信息不变，则只需要重新配置 ADC_CH_VLD 和 ADC_START 寄存器即可。如果其他配置需要改变，也可重新配置其他寄存器。

黑色箭头流程为复位以后，第一次扫描软件操作流程。红色箭头部分为再次启动软件操作流程。

4.13.3.5 滤毛刺流程

连续扫描模式下，滤毛刺过程为，每个通道都以 4 次数据为一个单位，每采样 4 次数据取平均值输出。对滤毛刺后得到的有效扫描值，上报中断并置扫描完成标志位。置位 ADC_CTRL(0x1) GLITCH 位使能该功能。

4.13.3.6 低功耗流程

连续扫描模式下，当使能低功耗模式时，每次 ADC 转换完成后进入低功耗模式，下次扫描时候到来时，现将 ADC 从低功耗模式唤醒，再次启动 ADC 转换。置位 ADC_CTRL(0x1) POWRR_DOWN 位使能该功能。

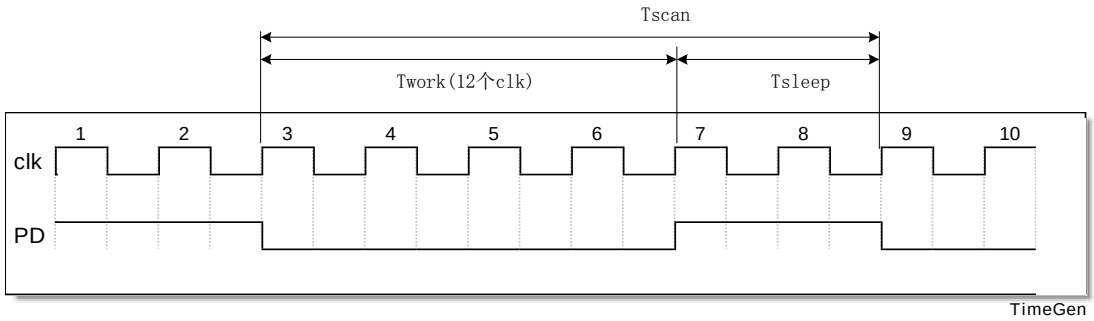


图 4.27 ADC_CTRL 低功耗模式时序简图

ADC_CTRL 的扫描时间间隔是 Tscan，ADC 的转换时间为 Twork，Twork 以后，ADC 进入低功耗模式，低功耗的时间为 Tsleep，等下次 Tscan 时间到，再次进入 Twork。

4.13.4 寄存器概述

寄存器表 4.22 ADC 寄存器概述

偏移地址	名称	描述	其他
0x0	ADC_CTRL	ADC 控制寄存器	
0x4	ADC_INTR_DELTA	ADC 中断误差范围	
0x8	ADC_INTR	ADC 中断控制寄存器	
0xc	ADC_INTR_STA	ADC 中断状态寄存器	
0x10	ADC_SCAN_TIME	ADC 扫描时间间隔设置	
0x14	ADC_START	ADC 启动	
0x18	ADC_CTRL_STATE	ADC 控制和状态寄存器	
0x1c	ADC_INTR_THRESHOLE	ADC 中断初始值	
0x20	ADC_CTRL_STOP	ADC 配置寄存器	
0x24	ADC_CH_VLD	ADC 通道配置寄存器	
0x28	ADC_PD_SEOC_TIME	ADC 掉电时间寄存器	

0x2c	ADC_P2S_EN	ADC 串行数据输出寄存器	
0x30	ADC_DATA_AB	ADC 通道 AB 扫描值	
0x34	ADC_DATA_CD	ADC 通道 CD 扫描值	
0x38	ADC_DATA_EF	ADC 通道 EF 扫描值	
0x3c	ADC_DATA_GH	ADC 通道 GH 扫描值	
0x40	ADC_CTRL_VERID	ADC 版本寄存器	
0x44	ADC_DIR_DAT_OP	ADC EOC 后直接输出 ADC 数据	

4.14 DAC

4.14.1 概述

HR_C7000 内置一个 3 通道 DAC，用于输出低速模拟控制信号。

4.14.2 功能描述

DAC 控制器用于与内部 DAC 接口，实现 CPU 对 DAC 的控制。

4.14.3 工作方式

HR_C7000 的 DAC 控制器通过 CPU 配置寄存器，直接与内部 DAC 交互，向 DAC 发送数据、使能及低功耗控制信号。

4.14.4 寄存器概述

寄存器表 4.23 DAC 寄存器概述

偏移地址	名称	描述	页码
0x0	DAC_PD_CTRL	DAC 使能	错误！未定义书签。
0x4	DAC_PD_MODE_EN	DAC 低功耗使能	错误！未定义书签。
0x8	DAC_DATA_A	DAC 通道 A 数据	错误！未定义书签。
0xc	DAC_DATA_B	DAC 通道 B 数据	错误！未定义书签。
0x10	DAC_DATA_C	DAC 通道 C 数据	错误！未定义书签。
0x14	DAC_CTRL_VERID	DAC 控制器版本	错误！

			未定义书签。

4.15 Codec

4.15.1 概述

HR_C7000 芯片内置高性能音频 Codec IP，支持差分和单端 Mic 输入，ADC 端支持 AGC 自动增益控制。输出支持两路 lineout 单端输出，PWM 输出模式。

4.15.2 功能描述

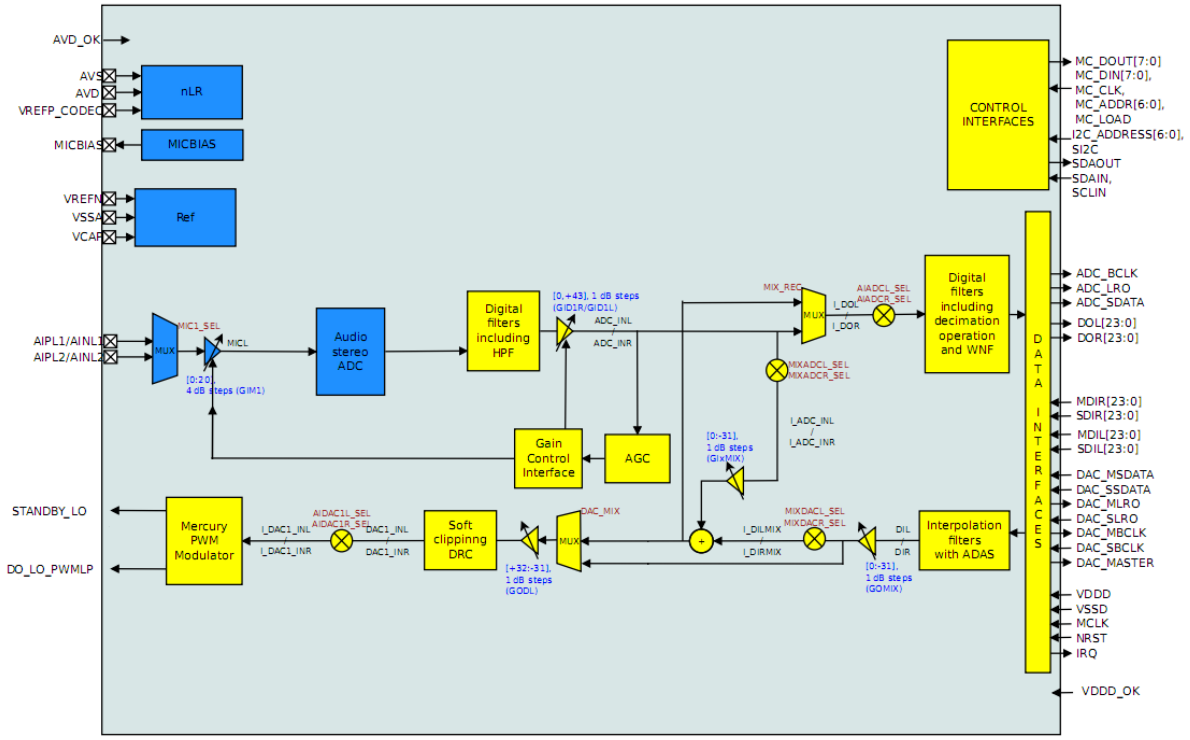


图 4.28 Codec 模块功能框图

4.15.2.1 功能特性

Codec 具有以下特点：

- 主时钟频率24MHz；
- 支持Pop噪声消除；
- 24bit-16bit 线性 PCM 模式输出；
- 支持8K，16K，24K，48K采用频率；
- Fast ADC 唤醒时间典型值100ms；
- 支持自动增益控制（AGC）；
- 支持可编程风噪滤波（WNF）；
- 两路差分/单端Mic输入支持；
- 两路单端Lineout输出支持，Lineout为PWM调制输出。
- ADC工作主模式

- DAC可以选择工作在主模式和从模式

4.15.3 工作方式

4.15.3.1 ADC 输入

Codec 模块的 ADC 输入支持差分 Mic 的输入方式和单端 Mic 的。下图为两种不同的连接方式的原理图示意图。

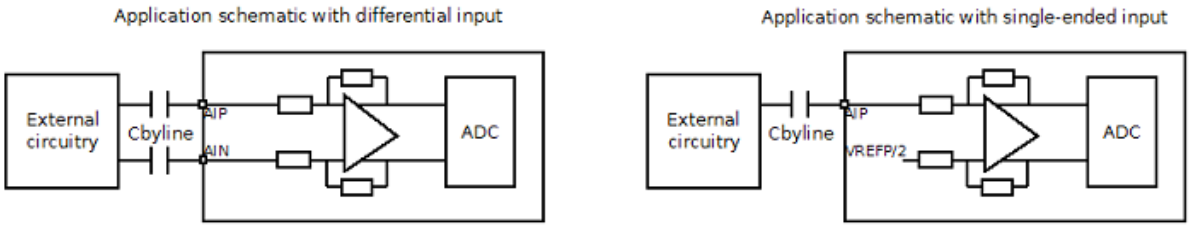


图 4.29 Codec Mic 差分和单端原理图

外部的拾音电路需要通过一组滤波电容 Cbyline 与 Codec 内部隔离，电容大小推荐 1uF。对上述差分或者单端的 Microphone 应该，为了获得更好的性能，减少高频部分干扰，需要有效利用 Micbias 应用电路。

Micbias 应用电路如下图，

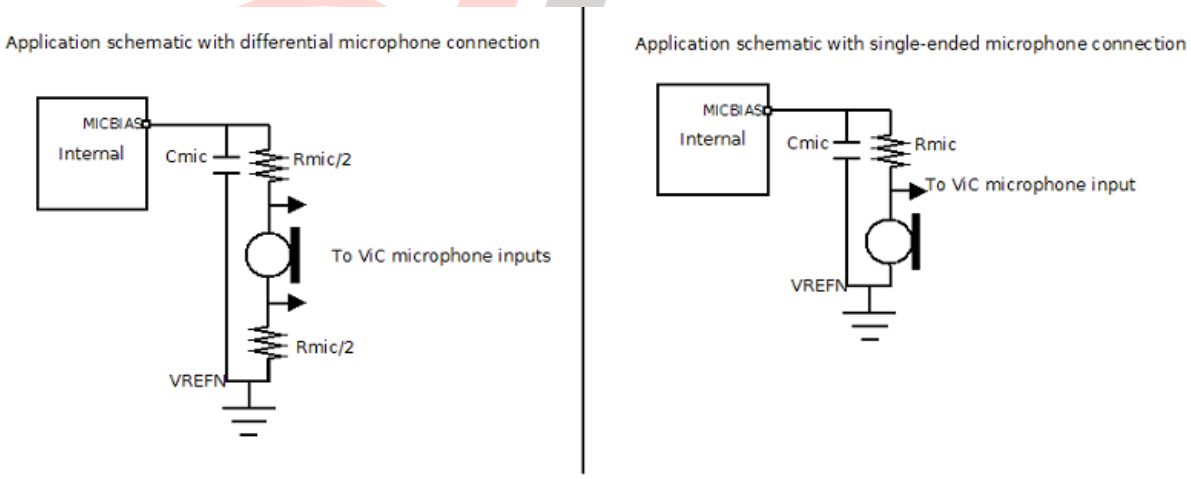


图 4.30 Codec Micbias 差分和单端原理图

Micbias 输出电压依据配置参数的不同，可以配置不同的输出典型值。滤波电路的 Cmic 和 Rmic

表 4.24 Codec Micbias 参数说明

Parameter	Test conditions	Min.	Typ	Max.	Unit
Micbias ouput level	MICBIAS_V = 0		2.08		V
	MICBIAS_V = 1		1.66		
Micbias output current				4	mA
Micbias output noise	A-Weighted		20	40	uVrms
Micbias decoupling capacitor	Cmic	0.75	1	1.25	nF
VCAP output voltage			2		V

4.15.3.2 DAC 输出

HR_C7000 内置的 Codec 模块的 DAC 输出是通过 PWM 数字输出的信号调制后实现 DAC 的功能。

芯片的两路 Lineout 输出（Lineout1 和 Lineout2 为同一个 PWM 调制输出相同波形，通过各自的使能控制，可以使得两路 Lineout 同时输出，也可以控制其中任意一路 Lineout 单端输出。

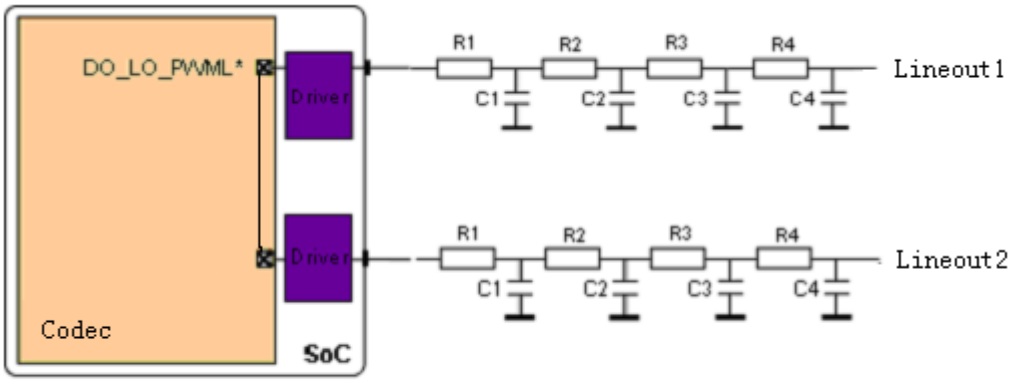


图 4.31 Codec PWM 输出原理图

上述 PWM 输出的外部整流电路的参考器件推荐如下。

表 4.25 Codec PWM 输出参数说明

R1	PWM outputs external resistors	2.7	±1%	kOhm
R2		2.7		
R3		2.7		
R4		2.7		
C1	PWM outputs external capacitors	220	±10%	pF
C2		220		
C3		220		
C4		220		

4.15.4 寄存器概述

寄存器表 4.26 Codec 寄存器概述

偏移地址	名称	描述	其他
0x0	SR1	状态寄存器	
0x1	SR2	状态寄存器	
0x7	MR	ADC 静音控制	
0x8	AICR_DAC	DAC 音频接口控制	
0x9	AICR_ADC	ADC 音频接口控制	
0xB	CR_MIC1	MIC1 控制寄存器	
0xC	CR_MIC2	MIC2 控制寄存器	
0xD	CR_DAC	DAC 控制寄存器	
0xF	CR_ADC	ADC 控制寄存器	
0x10	CR_MIX	数字混叠控制寄存器	

0x11	DR_MIX	数字混叠控制间接寄存器	
0x12	CR_VIC	Codec 的 sleep 模式控制寄存器	
0x13	CR_CK	MCLK 控制寄存器	
0x14	FCR_DAC	DAC 采样时钟配置寄存器	
0x17	FCR_ADC	ADC 采样时钟配置寄存器	
0x18	CR_TIMER_h	计数器高 8bit	
0x19	CR_TIMER_l	计数器低 8bit	
0x1A	ICR	中断控制寄存器	
0x1B	IMR	中断 MASK 控制寄存器	
0x1C	IFR	中断标志定义	
0x1D	IMR2	定时器标志 MASK 控制寄存器	
0x1E	IFR2	定时器标志定义	
0x1F	GCR_DACL	DAC 增益控制	
0x23	GCR_MIC1	Mic1 增益控制	
0x25	GCR_ADCL	ADC 增益控制	
0x27	GCR_MIXDACL	混叠 DAC 增益控制	
0x29	GCR_MIXADCL	混叠 ADC 增益控制	
0x2B	CR_DAC_AGC	DAC 增益控制寄存器	
0x2C	DR_DAC_AGC	DAC AGC 间接地址的读写数据	
0x2F	CR_ADC_AGC	ADC 增益控制寄存器	
0x30	DR_ADC_AGC	ADC AGC 间接地址的读写数据	

5 基带接口使用

5.1 发送 DAC

HR_C7000 内置两个高性能 DAC，采用单端输出，支持两点调制和单点调制射频接口，两路信号幅度、偏置分别可调。

用户可以通过配置寄存器选择相应的发送接口，两路输出信号偏置和两路输出信号的幅度。

此外，为了控制芯片的功耗，用户可以通过设置寄存器在 DAC 不工作时将其关闭。

表 5.1 HR_C7000 基带发送控制寄存器地址

偏移地址	名称	描述	其他
0x70	DAC_CONTROL	DAC 控制寄存器	
0x104	RF_MODE	发送 RF 模式设置	
0x108	SIG_CENTER	发送 IQ 两路偏置设置	
0x10c	TX_IF_FREQ	发送中频频率字配置	
0x110	RF_CONTROL	RF 发送提前量、发送中断提前量设置	
0x114	RF_MOD_BIAS_CTRL	发送信号幅度大小、两点调制偏置	

5.1.1 两点调制

通过过设置寄存器 RF_MODE[6:5]为 0x3，使 HR_C7000 工作于两点调制模式，可以通过配置

SIG_CENTER 寄存器分别调节两点调制信号的偏置值。如下图所示，两点调制信号经过两个运放调节其信号偏置及信号幅度得到 MOD1、MOD2 两路信号分别控制晶振及 VCO，实现两点调制，其中两路 APC 由 HR_C7000 自带的 DAC_MCU 输出模拟信号，可用于调节电压幅度。

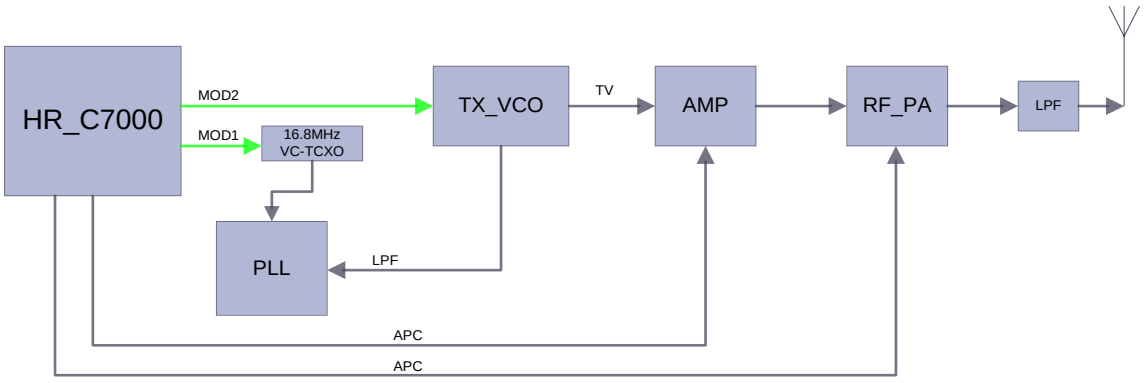


图 5.1　两点调制接口信号

两点调制模式下，发送随机信号时差分输出幅度最大约为 2240mV。

通过配置寄存器 RF_MODE[7]可以调整调制频偏映射关系，配置 RF_MODE[7]=0，对应符号与调制频偏的关系为：

表 5.2 信号的符号与调制频偏对应关系

符号	调制频偏
+3	1944 Hz
+1	648 Hz
-1	-648 Hz
-3	-1944 Hz

配置 RF_MODE[7]=1，对应符号与调制频偏的关系为：

表 5.3 信号的符号与调制频偏对应关系

符号	调制频偏
+3	-1944 Hz
+1	-648 Hz
-1	648 Hz
-3	1944 Hz

通过设置寄存器 SIG_CENTER[31:24]可调整输出 MOD1 路偏置，调整范围约为±422mV，最小调整步径为 3.3mV。

通过设置寄存器 SIG_CENTER[23:16]可调整输出 MOD2 路偏置，调整范围约为±422mV，最小调整步径为 3.3mV。

通过设置寄存器 RF_MODE[9]，可选择发送平缓上升起点，配置 RF_MODE[9]=1’b0，则平缓上升起点为 1.65V，若配置 RF_MODE[9]=1’b1，则平缓上升起点为 0V。

通过设置寄存器 RF_MODE[10]=1’b1，可发送 40Hz 方波，用于两点调制调试。

通过设置寄存器 RF_CONTROL [13:8]，可以配置射频控制中断 RF_TX_INTER 相对 30ms 时隙边界提前量，可调范围为 0μs~6300μs，最小调整步径为 100μs。

通过设置寄存器 RF_MOD_BIAS_CTR[7:0]可调整输出 MOD1 路幅度，可调范围约为 8.75mV~2240mV，最小调整步径为 8.75mV。

通过设置寄存器 RF_MOD_BIAS_CTR[15:8]可调整输出 MOD2 路幅度，可调范围约为 8.75mV~2240mV，最小调整步径为 8.75mV。

当发送采用两点调试模式，如果接收射频通道需要 HR_C7000 输出直流电压用于控制晶振的电压，则需要配置 DAC_CONTROL 寄存器使该路 DAC 处于常开状态。此时，可以通过配置寄存器 RF_MOD_BIAS_CTR[25:16]，设置 MOD1 在接收状态下的输出电压值，调节范围为 0~3.3V。

5.1.2 单点调制

通过过设置寄存器 RF_MODE[6:5]为 0x3，使 HR_C7000 工作于两点调制模式，在两点模式下如果仅仅使用 HR_C7000 基带 DAC 输出的 1 路 Mod1 信号进行发送信号调制，则为单点调制方式。可以通过配置 SIG_CENTER[31:24]寄存器调节单点调制信号的偏置值。如下图所示，单点调制信号经过 RF 专用芯片 RDA1846S 后，在经过 PA 放大后低通发射。其中 MOD1、信号控制 VCO。APC 由 HR_C7000 自带的 DAC_MCU 输出模拟信号，可用于调节电压幅度。

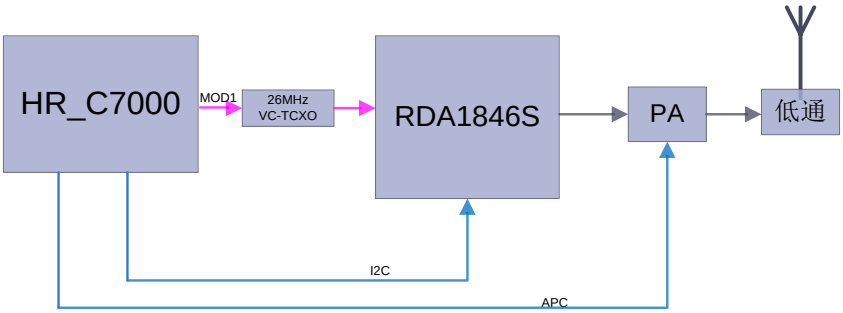


图 5.2　单点调制接口信号

5.2 接收 ADC

HR_C7000 内置两个高性能 ADC，支持 AF 和中频等射频接口，并支持两路幅度和偏置分别可调。

通过寄存器配置控制两路 ADC 信号满量程输入时的电压。

此外，为了控制芯片的功耗，可以通过配置选择是由 MCU 自动根据接收时隙将相应的 ADC 在发送时隙置为休眠模式，或者由 MCU 控制相应 ADC 的工作状态。

表 5.4 HR_C7000 基带发送控制寄存器地址

偏移地址	名称	描述	其他
0x74	ADC_CONTROL	ADC 控制寄存器	
0x104	RF_MODE	接收 RF 模式设置	
0x108	SIG_CENTER	接收 IQ 两路偏置设置	
0x1b0	RX_IF_FREQ	接收中频频率字配置	
0x120	THRESHOLD_VALUE	配置到达检测模块检测门限、定时同步模块检测门限	
0x128	PHASE_OUT	AF 接收模式下的相位幅度（含增益）的区间。分别由最大值、最小值表示。	

5.2.1 AF 接收

通过设置 RF_MODE[24]=1’b1，可使 HR_C7000 工作于 AF 模式。

使用 AF 模式的接收框图如下所示。

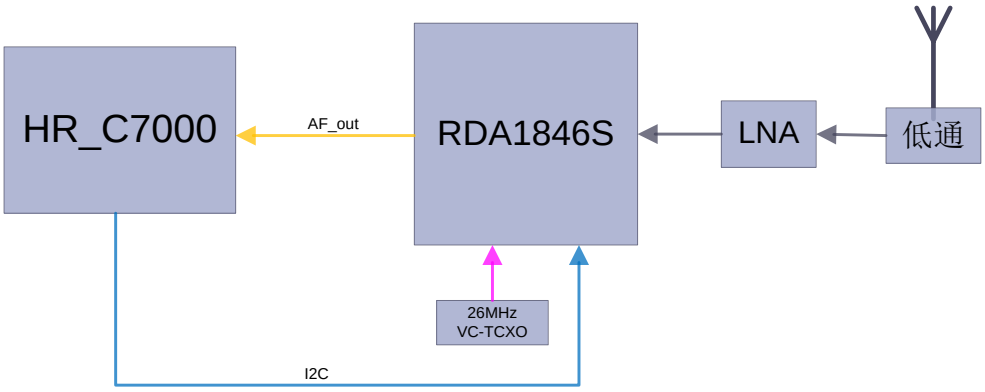


图 5.3 AF 接收参考接口电路

AF 接收模式，进入 HR_C7000 信号为调频后的信号，需要在芯片内部调制 AF 信号的增益和偏置。

其中 AF 信号增益由寄存器 RF_MODE[23:16]进行调节。0~255 分别表示 0~15.9375 倍幅度增益，即单位步进为 1/16 倍。通过实时读取寄存器 PHASE_OUT 获取当前增益配置下的 AF 相位幅度区间（对于标准的信号强度为-47dBm，调制频偏 2.749KHz 的正负 3 信号，AF 相位幅度区间在-1300~1300 附近），从而实现对 AF 幅度粗调。为进一步提升数字接收性能，AF 的增益微调，可以借助于接收误码率测试同步进行，以此获取 RF_MODE[23:16]与当前外围硬件设计相匹配的经验配置值。

AF 信号偏置则由寄存器 SIG_CENTER 配置发送 I/Q 路偏置进行调节。

除上述幅度及偏置的调节外，建议开启寄存器 AF_MODE[25]，“1”有效，可以有效避免射频强信号下，高频信号的包络对 AF 相位信号的干扰。

5.2.2 中频接收

通过设置 RF_MODE[14:13]=2’b00，可使 HR_C7000 工作于中频模式，通过配置 SIG_CENTER 寄存器可对接收中频信号的偏置进行调节，同时可通过配置 RX_IF_FREQ 对 HR_C7000 接收的中频频率进行设置，其计算公式见 4.7.3。设置附表寄存器 THRESHOLD_VALUE，可设定定时同步模块检测门限、到达检测模块检测门限。

使用中频模式的接收框图如下所示。

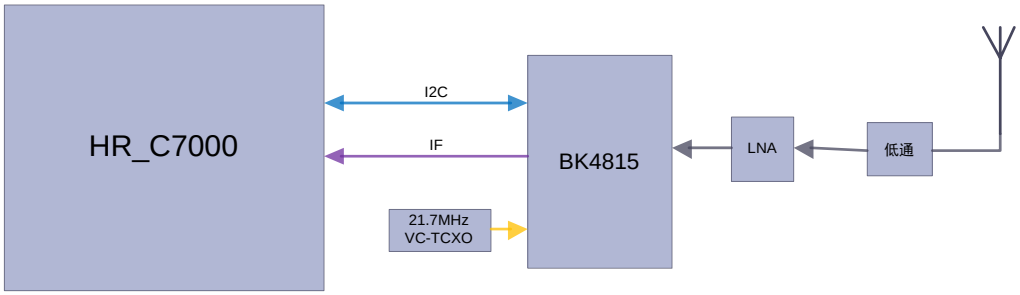


图 5.4　中频接收参考接口电路

接收信号经滤波、放大后送入 BK4815 专用芯片解调得到的中频 IF 信号直接送入 HR_C7000 的 AD 管脚 ADC_IVINP，而 ADC 的负端 ADC_IVINN 可以接地或者接收中频信号的偏置电压。需要注意的是，低中频滤波器主要用于滤除邻道干扰信号，当信道间隔为 25KHz 时，滤波器带宽可选±7.5KHz，当信道间隔为 12.5KHz 时，滤波器带宽可选±3.75KHz 或±4.5KHz。

5.3 数字接收

HR_C7000 支持数字接口接收模式。数字接口采用 SPI 单向接收接口，接口示意图如下图所示。

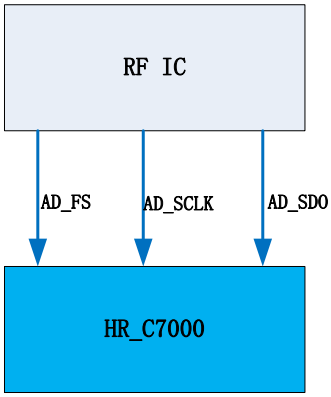


图 5.5 HR_C7000 数字接口示意图

接收如上图，其中 AD_FS 为 ADC 采样时钟，频率为 38.4KHz；AD_SCLK 为 ADC 串行数据时钟，频率为 2.4576MHz；AD_SDO 为 ADC 串行输出数据。

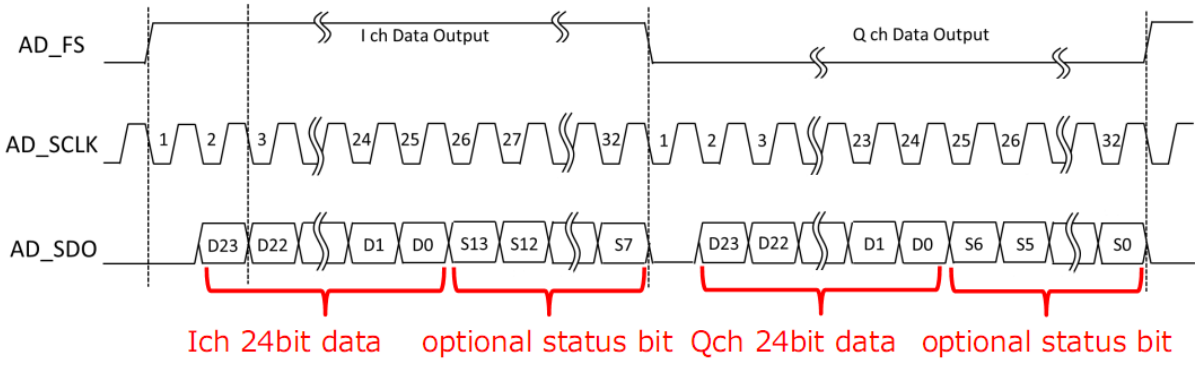


图 5.5 数字 SPI 接口时序图

详细接口时序如上图。

AD_FS 用于控制 ADC 的 IQ 两路通道输入，AD_SDO 最高支持 24bit 精度输入，但是由于 HR_C7000 接收通路支持最高 16bit 接收精度，可以支持选择 AD_SDO 串行数据的 24bit 中的任意 16bit 数据作为 ADC 采样数据输入。

AD_SDO 中的可选状态数据 S13-S0 包括 RSSI 值，AGC 状态值和其他信息，通过 SPI 输入到 HR_C7000，可以通过内部相关寄存器获取到这些信息。