XC6XXX 参考手册

南京新向远微电子有限公司

2024 年 02 月

目录

第 1 章 系统概述

1.1. 概述

XC65xx 芯片是一款低功耗、高性能和高度集成的 SoC，带有蓝牙 5.2 BLE/2.4G 收发器。它集成了高性能 2.4GHz 射频收发器、丰富的基带功能、32 位 MCU 和各种外围 IO。它支持 128/256/512KB 的 flash 和 48KB 的 RAM，以实现可编程协议和配置文件，支持定制应用程序。芯片结构如图 1-1 所示。

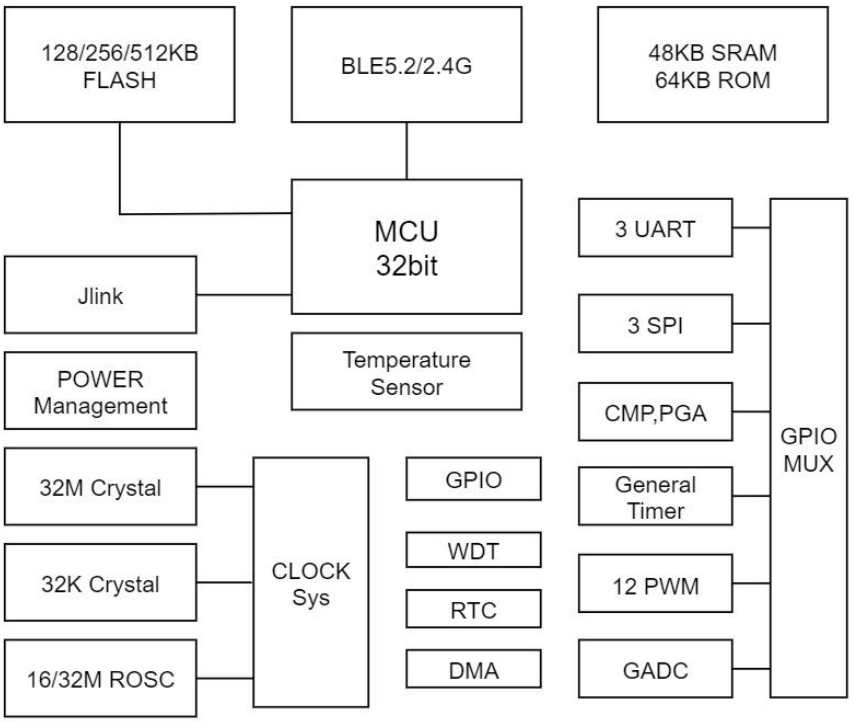

图 1-1 XC65xx 芯片结构

1.2. 特性

- **内核 Core：**

32bit MCU max 96MHz，支持 XIP，128KB 闪存，48 KB SRAM+64KB ROM；

- **模数转换 ADC：**

16 路（外部 12 路+内部 4 路）12bit 通用 ADC，最高速率 1M 采样率；

- **定时器 Timer：**

4*32bit 通用定时器+2 个 32 位 aotimer 定时器+24 位 systick；

- **脉宽调制 PWM：**

PWM(12 路，其中 4 路带死区控制互补输出)，最高 16 位连续可调，支持中心对齐模式，支持刹车，支持 pwm 和 adc 联动；

- **串口：**

XC65xx 芯片包含 1 路 I2C、3 路高速 SPI、3 路 UART；最多 40 个 GPIO；

- **实时时钟 RTC：**

支持定时和触发两种模式；

- **独立看门狗 WDT：**

支持 2 种工作模式，用户可以根据需要选择不同的工作模式。

第 2 章 芯片地址映射

本章节描述各个模块的地址映射。

表 2-1 中描述了各个模块在芯片中的地址映射。

表 2-1 芯片总地址映射表

模块名		地址范围	大小
ROM		0x00000000~0x00018000	96K
RAM		0x10000000~0x10018000	96K
DMA		0x50001000~0x50001FFF	4KB
USB		0x50010000~0x5004FFFF	256KB
2.4G		0x50003000~0x50003FFF	4KB
AUDIO_ADC		0x50004000~0x50004FFF	4KB
CTL_APB	CPR	0x40000000~0x40000FFF	4KB
	GPIO	0x40001000~0x40001FFF	4KB
	RTC	0x40002000~0x400023FF	1KB
	CPR_AO	0x40002400~0x400027FF	1KB
	TIMER_AO	0x40002800~0x40002BFF	1KB
	TIMER	0x40003000~0x40003FFF	4KB
	WDT	0x40004000~0x40004FFF	4KB
	I2C	0x40006000~0x40006FFF	4KB
	UART0	0x40010000~0x40010FFF	4KB
	UART1	0x40011000~0x400113FF	1KB
	UART2	0x40011400~0x400117FF	1KB
	I2S	0x40011800~0x40011BFF	1KB
	SSI0	0x40013000~0x40013FFF	4KB
	SSI1	0x40014000~0x400147FF	2KB
	SSI2	0x40014800~0x40014FFF	2KB
	QDEC	0x40016000~0x40016FFF	4KB
	PWM	0x40017000~0x40017FFF	4KB
	GPADC	0x40018000~0x40018FFF	4KB
	BT	0X40020000~0x4002FFFF	64KB

第 3 章 中断系统

Cortex-M0 处理器内部集成 NVIC 中断控制器，这种中断控制器与 Cortex-M0 紧密耦合，可以提供更快更有效的中断处理进程，NVIC 支持：

1. 可嵌套中断支持。
2. 向量中断支持。
3. 可屏蔽中断支持。
4. 动态优先级调整支持。

3.1. M0 中断说明

NVIC 的核心工作原理是对一张中断向量表的维护上，M0 最多支持 32+16 个中断向量，而这些中断向量默认的优先级是向量号越小的优先级越高，但是我们肯定不会满足于默认的状态，而 NVIC 则提供了这种灵活性，即支持动态优先级调整，无论是 M0 还是 M4 除了三个中断向量之外（复位、NMI 和 HardFault 它们的优先级为负数，它们三个优先级最高且不可以更改），其他中断向量都是额可以动态调整的。

需要注意的是，中断向量表的前 16 个为内核级中断，之后的为外部中断，而内核级中断和外部中断是由两套不同的寄存器组来控制，其中内核级中断由 SCB_SHPRx 寄存器来控制，外部中断是由 NVIC_IPRx 来控制，M4 支持的动态优先级范围为 0-15（8bits 中只有高四位[7:4]才有效），M0 支持的动态优先级范围为 0-3（8bits 中只有高两位[7:6]才有效），秉承着号越小优先级越高的原则（0 最高，15 或 3 为最小）。

3.2. M0 中断分配

M0 支持 32 个 IRQ 中断源，如表 3-1 所示。

表 3-1 M0 中断分配表

中断源	中断信号	中断序号分配	缺省中断优先级
IRQ			
BT	tab_intr_n	0	2
DMA	dmas_int	1	2
CPR	cpr_intr	2	2
GPIO	gpio_intr	3	2
RTC	rtc_intr	4	2

TIMER0	timer0_intr	5	2
TIMER1	timer1_intr	6	2
TIMER2	timer2_intr	7	2
TIMER3	timer3_intr	8	2
WDT	wdt_intr	9	2
I2C	i2c_intr	10	2
UART0	uart0_intr	11	2
UART1	uart1_intr	12	2
SPI0	ssi0_intr	13	2
SPI1	ssi1_intr	14	2
保留	保留	15	2
QDEC	qdec_intr	16	2
GPADC	gpadc_intr	17	2
PWM	pwm_intr	18	2
保留	保留	19	2
USB	usb_intr	20	2
AUDIO ADC	cdc_intr	21	2
2.4G	rf24g_intr_n	22	2
SPI2	ssi2_intr	23	2
保留	保留	24	2
UART2	uart2_intr	25	2
I2S	i2s_intr	26	2
TIMER_AO0	timer_ao_intr[0]	27	2
TIMER_AO1	timer_ao_intr[1]	28	2
CMP	cmp_intr	29	2
FMC	fmc_intr	30	2
CAN	can_intr	31	2

第 4 章 时钟管理模块（CPR）

4.1. 概述

时钟分频模块负责管理挂载在总线上的外设时钟。用于控制全局功耗，产生各模块所需时钟及复位信号，负责特定模块的电源管理，一些模块的 CTL 寄存器的配置等。M0 是 CPR 的唯一 Master，用来配置 CPR 的各个寄存器。

CPR 划分为 AlwaysOn（AO）域和 PowerDown（PD）域，包括 AO 域和 PD 域两个顶层，分别包含不同的功能单元。CPR 的两部分与 PWR3V 模块配合实现系统整体的控制功能。

PD 域在系统进入睡眠状态时，可以关闭电源以节省功耗。

在深度睡眠状态下，仅 PWR3V 模块保持供电，AO 域和 PD 域均会断电。在此场景下，PWR3V 负责系统的深度睡眠的进入和唤醒。在普通睡眠状态下，系统 PD 域断电时，AO 域由 32K 输入时钟维持系统运行，AO 域的 CPR 负责监视中断，从而实现系统的唤醒。除深度睡眠状态之外，AO 域在系统运行过程中保持供电。对于某些影响到系统唤醒的寄存器，AO 域对这些寄存器的值进行单独配置，保证这些配置在 PD 域断电时有效。

●CPR 基地址为：0x40000000（4KB）

●CPR_AO 基地址为：0x40002400（1KB）

4.2. 主要特性

- 时钟管理：产生各模块所需时钟，并控制时钟的打开和关闭；
- memory（RAM）的低功耗控制；
- 各功能模块复位控制；
- 睡眠唤醒电路，用于处理普通睡眠状态的进入和唤醒过程；
- 各功能模块的 CTL 寄存器配置，包括 PINMUX，BOOTCTL 等。

4.3. 时钟结构

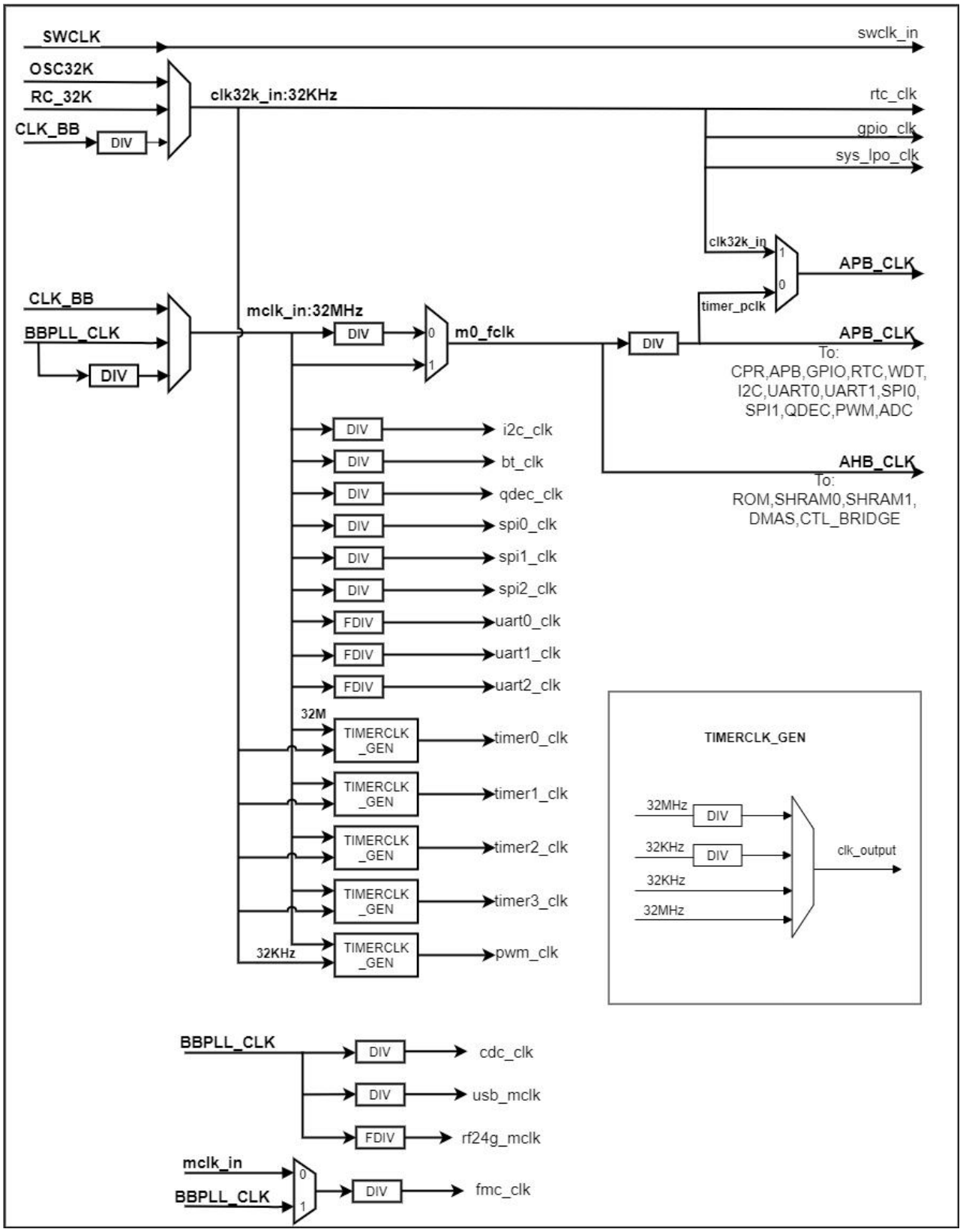

图 4-1 系统时钟树

4.4. 功能描述

芯片上电后，CPR 负责完成上电开机所必需的准备工作，然后 M0 即开始工作。工作状态可分为开机、启动、工作、关机四个阶段，如图 4-2 所示。CPR 模块在各个阶段中起着重要作用，下面具体描述每个阶段中 CPR 的具体工作。

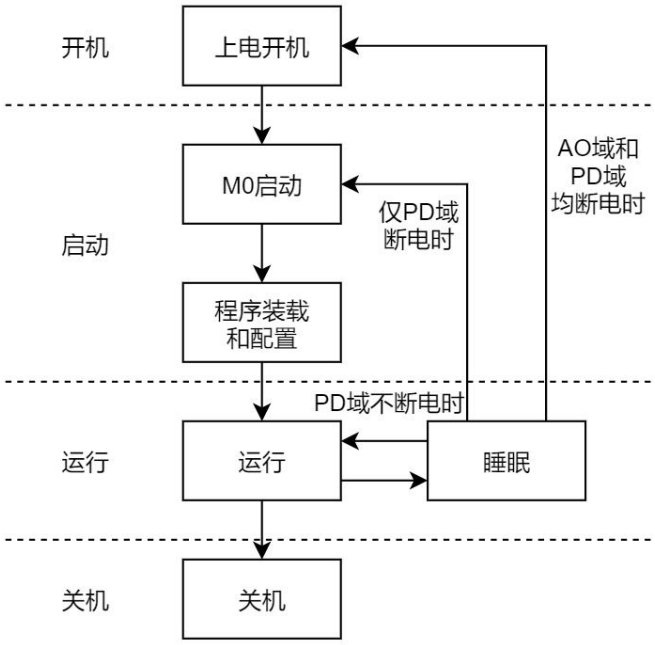

图 4-2 工作状态图

4.4.1. 开机阶段

芯片一上电就进入开机阶段，在此阶段的启动步骤为：

(1) PMU 的 POR3V 单元首先监测电源，如果电源稳定输出则拉高输出信号 POR_PVDD；

(2) POR_PVDD 信号拉高后，PWR3V 模块开始进入工作状态，拉高输出信号；LDO_AO_ON/LDO_DIG_ON/BUCK_ON/BOOST_ON 等 PMU 控制信号；

(3) LDO_AO_ON 信号使能 PMU 的 LDO_AO 单元，开始为 AO 域的逻辑供电，PMU 的其它单元如 LDO_DIG/BUCK/BOOST 也开始供电；

(4) AO 域开始供电后，32K 晶振起振，同时 PMU 的 PORAO 单元监测 LDO_AO 的供电，当输出稳定后拉高输出信号 POR_AO；

(5) POR_AO 信号拉高后，CPR_AO_TOP 模块开始执行 SYS_CTRL 状态机；

(6) SYS_CTRL 状态机首先拉高 OSCEN，使能 32M 晶振；

(7) 在计数器确保 32M 晶振输出稳定之后，PRSTN_DIG 拉起，系统 PD 域复位完成，进入工作状态；

(8) 再经过若干 32M 时钟周期，M0 的复位 M0_SYSRESETN 拉起，系统开始工作。

具体的开机时序如下图所示。

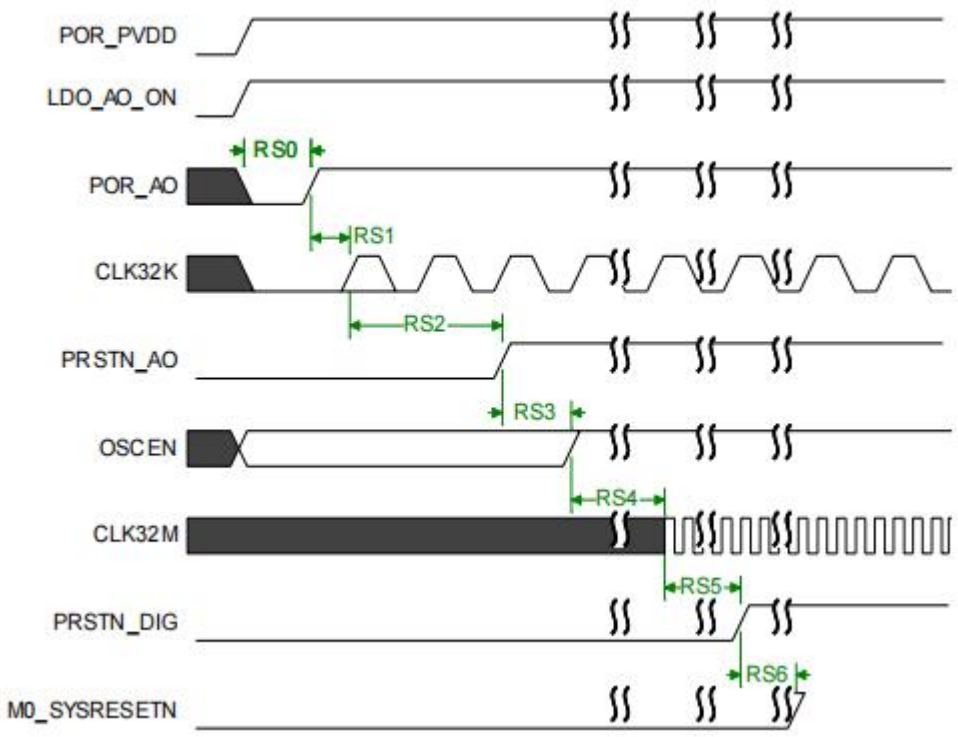

图 4-3 开机时序图

注：

1.管脚说明：

POR_PVDD：PMU 的 POR3V 单元的输出信号，PWR3V 数字域输入的上电复位信号

LDO_AO_ON：PWR3V 数字域的输出信号，使能 PMU 的 LDO_AO 单元供电

POR_AO：PMU 的 PORAO 单元的输出信号，AO 数字域输入的上电复位信号

PRSTN_AO：芯片 AO 域复位信号

PRSTN_DIG：芯片 PD 域复位信号

CLK32K：数字域输入管脚，32K 晶振输入

CLK32M：数字域输入管脚，32M 晶振输入

OSCEN：芯片输出管脚，32M 晶振使能

M0_SYSRESETN：M0 复位信号，拉高后 M0 开始 boot

2.时序说明

RS0：PORAO 单元输出稳定时间/信号 POR_AO 拉高时间

RS1：从供电稳定到 32K 晶振起振时间

RS2：2 个 32K 周期，AO 域内部复位信号 PRSTN_AO 生成

RS3：1 个 32K 周期，32M 晶振使能

RS4：32M 晶振起振时间（注意：确保 PD 域供电在此时间范围内稳定）

RS5：PRSTN_DIG 生成时间，等于 8 个 32KHz 时钟周期

RS6：从 PRSTN_DIG 拉高到 M0 复位拉高时间，大概 6 个 32M 时钟周期

3.开机流程详细说明

a.在 POR_AO 为高 2 个 32K 时钟周期后，系统复位信号 PRSTN_AO 生成。再过 1 个 32K 时钟周期后，CPR 配置芯片管脚 OSCEN 输出高电平，进入外部 32M 晶振起振过程。

b.在 32M 晶振起振过程中，晶振使能信号 OSCEN 始终为高电平。32M 晶振起振时间在开机阶段为 16ms，这是由 CPR 内部计数器规定的，所以用户必须保证 32M 晶振时钟在 16ms 内稳定。注意：晶振稳定时间包括晶振供电稳定时间加上晶振起振稳定时间。

c.16ms 后，CPR 为各个模块提供稳定的时钟，系统进入启动阶段。先拉高 PRSTN_DIG，开启 PD 域信号的初始化。过了若干个 32MHz 时钟周期，拉高 M0_SYSRESETN，M0 开始从 BOOTROM 取程序执行。

4.4.2. 启动阶段

在启动阶段整个芯片得以初始化。启动阶段分为 M0 启动、程序装载和配置。

4.4.2.1. M0 启动

芯片上电开机完成后进入 M0 启动过程，即由 CPR 控制结束 M0 的复位状态并且为 M0 提供时钟，使 M0 处于工作状态，开始执行程序。

芯片的各种启动模式通过管脚 bootctl（管脚名称为 BOOT_CTL）进行配置，bootctl 状态可通过读取寄存器 CPR_CTL_BOOTCTL 获悉。BOOTROM 执行后将进入如表所示。所示的两种启动模式。

表 4-1 系统下载模式表

BOOTCTL	描述
0	UART0 下载
1	SSI0 下载

当 bootctl=1 时，BOOTROM 程序将进入 SPIFLASH 启动模式，BOOTROM 程序通过 SSI0 接口将 FLASH 中数据搬移到内部 M0_RAM（0x10000000）为起始的地址上，搬移完成后，M0 的 PC 指针跳转至 M0_RAM 地址处，开始执行用户程序。

当 bootctl=0 时，BOOTROM 程序将进入 UART0 下载启动模式，BOOTROM 程序将从 UART0 接收的数据搬移到内部 M0_RAM 为起始的地址上，搬移完成后，M0 的 PC 指针跳转至 M0_RAM 地址处，开始执行用户程序。

4.4.2.2. 程序装载和配置

程序装载和配置指的是软件为了 M0 启动所需要做的工作，例如把处理器的执行程序包搬移到相应的目标地址，为了其他核或模块正常工作所需要进行的寄存器配置等。

M0 启动后，应用程序配置 CPR 内部的寄存器，配置系统总线时钟频率、M0 核时钟频率及其它功能单元的时钟频率，完成系统配置。

4.4.3. 工作阶段

工作阶段中，分为运行和睡眠两种状态。当处于运行状态时，CPR 模块监视 M0 及 DMA（详细参见睡眠进入部分）的工作状态，如果 M0 及 DMA 进入休眠状态，且睡眠的其它条件满足，则 CPR 启动睡眠流程。

在睡眠状况下，PWR3V 域恒开，AO 域和 PD 域可以根据软件配置决定是否关闭，32M 时钟和 32K 时钟可以通过软件决定开闭。PMU 根据配置决定各个供电单元的开关。需要注意的是，32K 时钟需要 AO 域供电才能工作，32M 时钟需要 PD 域供电才能工作。

系统存在四种睡眠状态，按照功耗从低到高分别为：

- PWR3V 域供电+AO 域断电+PD 域断电（深度睡眠）
- PWR3V 域供电+AO 域供电（32K 关）+PD 域断电（普通睡眠）
- PWR3V 域供电+AO 域供电（32K 开）+PD 域断电（普通睡眠）
- PWR3V 域供电+AO 域供电+PD 域供电（32M 关）（浅度睡眠）

注意：睡眠状态的选择，需要配合 PWR3V 域的部分断电 Mask 寄存器实现。

在睡眠状态下，PWR3V 域始终保持上电状态。

AO 域断电的深度睡眠状态中，系统会通过关闭 LDO_AO 的供电，直接关闭整个 AO 域的供电。在 AO 域断电的情况下，GPIO0-4 和 PWR3V 域的 TIMER 的能够恢复 PMU 的 LDO_AO 等供电，重新执行上电初始化流程启动系统。

仅 PD 域断电的普通睡眠状态中，除 AO 域的模块（如 RTC、CPR 的 AO 部分）外，其他功能模块处于睡眠且断电状态。AO 域的 RTC 模块（包括 RTC 计时中断、AO_TIMER 定时中断、GPIO0-4 输入检测中断等）发起的中断可以唤醒系统。在收到有效中断后，CPR 执行唤醒流程，控制系统从睡眠状态进入启动阶段重新 BOOT，最终返回到运行状态。

在 PD 域不断电的浅度睡眠状态中，除 AO 域的模块外，其它模块均处于睡眠（时钟可配置为开或关）但不断电状态。CPR 在睡眠状态中监视中断状态（此处中断指的是任意中断源发出的中断），例如 TIMER 中断、GPIO 中断等。在收到有效中断后，CPR 执行唤醒流程，控制从睡眠状态直接回到运行状态。TIMER 在睡眠状态下需要将总线时钟切换为 32KHz 时钟，才能正常的发出中断，如 4.4.3.2.5 所描述。

4.4.3.1. 运行状态

在运行状态中，CPR 能够完成时钟管理、复位控制、睡眠控制、电源管理及中断事件发出等任务。关于睡眠的进入和唤醒将在 3.4.2.3.2 中具体介绍。

4.4.3.1.1. 时钟概述

CPR 中所有时钟如下表 4-2 所示。

表 4-2 时钟列表

时钟	时钟源头	默认频率	上电状态	描述	寄存器控制
M0 时钟					
m0_fclk	mclk_in	32MHz	恒开	M0 工作时钟	CPR_M0_FCLK_CTL
m0_hclk	m0_fclk	32MHz	打开	M0 工作时钟	CPR_M0_FCLK_CTL
m0_stclk	m0_fclk	32MHz	恒开	M0tick 时钟	
Debug 模块时钟					
dap_clk	m0_fclk	32MHz	打开	DAP 时钟	CPR_DAP_CLK_CTL
Bluetooth 模块时钟					
bt_clk	mclk_in	8MHz	关断	bt 工作时钟	CPR_BT_CLK_CTL
bt_pclk	ctl_pclk	16MHz	关断	bt 配置 APB 时钟	CPR_CTLAPBCLKEN_GRCTL
bt_modem_clk	mclk_in	8MHz	关断	btmodem 工作时钟	CPR_BT_MODEM_CLK_CTL
bt32k_clk	clk32k_in	32KHz	关断	bt32K 时钟	CPR_OTHERCLKEN_GRCTL
MEMORY 时钟					
rom_hclk	m0_fclk	32MHz	打开	BOOTROM 工作时钟，频率同 m0_fclk	CPR_AHBCLKEN_GRCTL

m0_ram_hclk	m0_fclk	32MHz	打开	M0 指令和数据 RAM 工作时钟，频率同 m0_fclk	CPR_AHBCLKEN_GRCTL
shram0_hclk	bus_clk	32MHz	打开	SHRAM0 工作时钟，频率同 bus_clk	CPR_AHBCLKEN_GRCTL
总线时钟					
bus_clk	m0_fclk	32MHz	恒开	总线时钟，同 m0_fclk	
ctl_bridge_hclk	bus_clk	32MHz	恒开	CTL_APB 桥时钟，频率同 bus_clk	
ctl_pclk	bus_clk	16MHz	恒开	CTL_APB 总线时钟	CPR_CTL_PCLK_GRCTL
DMA 时钟					
dmas_hclk	bus_clk	32MHz	打开	DMAS 工作时钟，频率同 bus_clk	CPR_AHBCLKEN_GRCTL
CPR 模块时钟					
cpr_pclk	ctl_pclk	16MHz	恒开	CPRAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
WDT 模块时钟					
wdt_pclk	ctl_pclk	16MHz	关断	WDTAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
UART 模块时钟					
uart0_clk	mclk_in	1.8432MHz	打开	UART0 工作时钟	CPR_UART0_CLK_CTL CPR_UART0_CLK_GRCTL
uart1_clk	ctl_pclk	1.8432MHz	关断	UART1 工作时钟	CPR_UART1_CLK_CTL CPR_UART1_CLK_GRCTL
uart2_clk	ctl_pclk	1.8432MHz	关断	UART2 工作时钟	CPR_UART2_CLK_CTL CPR_UART2_CLK_GRCTL
uart0_pclk	ctl_pclk	16MHz	打开	UART0APB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
uart1_pclk	ctl_pclk	16MHz	关断	UART1APB 总线	CPR_CTLAPBCLKEN_GR

				时钟	CTL
uart2_pclk	ctl_pclk	16MHz	关断	UART2APB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
SPI 模块时钟					
ssi0_mclk	mclk_in	8MHz	打开	SSI0 工作时钟	CPR_SSI0_MCLK_CTL
ssi0_pclk	ctl_pclk	16MHz	打开	SSI0APB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
ssi1_mclk	mclk_in	8MHz	关断	SSI1 工作时钟	CPR_SSI1_MCLK_CTL
ssi1_pclk	ctl_pclk	16MHz	关断	SSI1APB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
ssi2_mclk	mclk_in	8MHz	关断	SSI2 工作时钟	CPR_SSI2_MCLK_CTL
ssi2_pclk	ctl_pclk	16MHz	关断	SSI2APB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
I2C 模块时钟					
i2c_pclk	ctl_pclk	16MHz	关断	I2CAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
i2c_clk	ctl_pclk	16MHz	关断	I2C 工作时钟	CPR_I2C_CLK_CTL
GPIO 模块时钟					
gpio_pclk	ctl_pclk	16MHz	打开	GPIOAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
gpio_clk	clk32k_in	32.768KHz	打开	GPIO 工作时钟	CPR_OTHERCLKEN_GRCTL
RTC 模块时钟					
rtc_clk	clk32k_in	32KHz	关断	RTC 工作时钟	CPR_OTHERCLKEN_GRCTL
rtc_pclk	ctl_pclk	16MHz	关断	RTCAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
QDEC 模块时钟					
qdec_clk	mclk_in	4MHz	关断	QDEC 工作时钟	CPR_QDEC_CLK_CTL
qdec_pclk	ctl_pclk	16MHz	关断	QDECAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
GPADC 模块时钟					

gpadc_pclk	ctl_pclk	16MHz	关断	GPADCAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
PWM 模块时钟					
pwm_clk	多时钟源时钟分频	1.6MHz	关断	PWM 工作时钟	CPR_PWM_CLK_CTL
pwm_pclk	ctl_pclk	16MHz	关断	PWMAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
Timer 模块时钟					
timer_pclk	ctl_pclk	16MHz	关断	TIMERAPB 总线时钟	CPR_CTLAPBCLKEN_GRCTL
timer0_clk	多时钟源时钟分频	1.6MHz	关断	TIMER0 工作时钟	CPR_TIMER0_CLK_CTL
timer1_clk	多时钟源时钟分频	1.6MHz	关断	TIMER1 工作时钟	CPR_TIMER1_CLK_CTL
timer2_clk	多时钟源时钟分频	1.6MHz	关断	TIMER2 工作时钟	CPR_TIMER2_CLK_CTL
timer3_clk	多时钟源时钟分频	1.6MHz	关断	TIMER3 工作时钟	CPR_TIMER3_CLK_CTL

4.4.3.1.2. 时钟分配

芯片管脚有 32K 和 32M 两个晶振时钟输入。芯片内部直接使用 32MHz 时钟作为工作时钟，然后分频得到芯片内部大多数模块的工作时钟。

- 32KHz 时钟（来源可选：外部 32KHz 晶振/32MHz 时钟分频/内部 RC 振荡器时钟）
  - CPR 睡眠唤醒电路
  - PWM/TIMER 工作时钟
  - RTC 工作时钟
  - GPIO 工作时钟
  - BT32K 工作时钟
- 32MHz 晶振
  - 作为系统工作主时钟
  - CPR 所需 32MHz 时钟源，分频得到

- 32KHz 工作时钟
- M0 工作时钟
- 总线时钟
- UART 工作时钟
- SSI 工作时钟
- QDEC 工作时钟
- BT 工作时钟
- I2C 工作时钟

- 多时钟源时钟分频

PWM/TIMER 模块的工作时钟可以选择由多个不同的时钟源分频。

pwm/timerX_clk（X=0~3）的分频时钟源有如下几个。

- mclk_in：外部输入的 32MHz 晶振时钟
- clk32k_in：外部输入的 32KHz 晶振时钟

4.4.3.1.3. 时钟分频逻辑

4.4.3.1.3.1. 通用整数分频

通用整数分频，分频参数如下

- CLKx_EN：时钟使能
- CLKx_DIV[x-1:0]：时钟分频比，默认值为 y。若设为 n，则为 n+1 分频。

4.4.3.1.3.2. G8 整数分频

G8 整数分频，分频时钟为源时钟频率的 x/8（x 为分频系数）。分频时钟一个周期的高电平宽度和源时钟的一个时钟周期的高电平宽度相同

分频参数如下

- CLKx_GR[4:0]：
  - 0：关时钟
  - 1~8：分频时钟为源时钟频率的 x/8，x=(1~8)，如 x=4，则如果源时钟为 80MHz，目的时钟为 40MHz。
  - 其他值：关时钟。
- CLKx_GR_UPD：分频参数更新信号，1 有效自清 0。

4.4.3.1.3.3. G8 浮点分频

G8 浮点分频用于 UART 等对波特率要求比较精确的模块。uartx_clk(x=0,1)使用了这种分频。使用两级分频得到分频时钟，第一级为 G8 整数分频，第二级为浮点分数分频。分频结构如下图 4-4 所示。

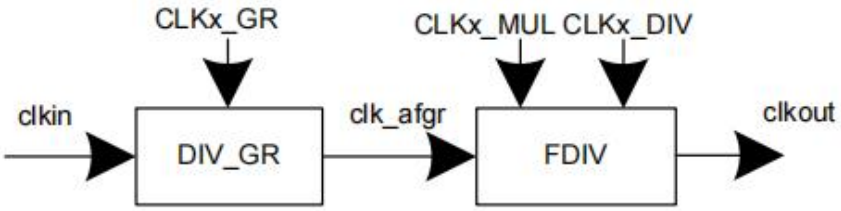

图 4-4 G8 浮点分频逻辑结构

分频参数如下：

- CLKx_GR[3:0]：一级分频系数
  - 0：关时钟
  - 1~8：分频时钟为源时钟频率的 x/8，x=(1~8)，如 x=4，则如果源时钟为 80MHz，目的时钟为 40MHz。
  - 其他值：关时钟。
- CLKx_MUL[15:0]：二级分频乘数
- CLKx_DIV[15:0]：二级分频除数
- CLKx_GR_UPD：分频参数更新信号，1 有效自清 0。

一级分频得到时钟频率为：

$$freq(clk\_afgr) = \frac{freq(clkin) * CLKx\_GR}{8}$$

最终二级分频得到时钟频率为：

$$freq(clkout) = \frac{freq(clk\_afgr) * CLKx\_MUL}{2 * CLKx\_DIV}$$

总的分频公式：

$$freq(clkout) = \frac{freq(clkin) * CLKx\_GR * CLKx\_MUL}{16 * CLKx\_DIV}$$

注：

1.clkin 表示时钟源头，clkout 表示目标时钟，clk_afgr 为内部时钟

2.如果希望配置出占空比为 1:1 的 clkout，则 CLK_GR 允许的配置范围为 1、2、4 或 8，这种情况下占空比仅决定于 MUL 和 DIV；否则，CLK_GR 可任意配置

3.MUL 推荐设置为偶数，否则分数分频的结果可能不对；DIV 和 MUL 都不能置为 0。

4.推荐的配置顺序：

a.配置 CLKx_GR 为 0

b.配置 CLKx_MUL 和 CLKx_DIV 为期望值

c.配置 CLK_GR 为期望值

d.配置 CLKx_GR_UPD 为 1（CLKx_GR_UPD 会自清 0）。

UART0 工作时钟(uart0_clk)分频示例：

浮点分频逻辑的源时钟均为 mclk_in（默认频率为 32MHz）。

当配置 UART0 的波特率为 1152000 时，由于

$$\mathrm{BaudRate} = \frac{\mathrm{f_{uart0\_mclk}}}{16 * \mathrm{divisor}}$$

当 divisor 为 1 时，需要得到的$\mathrm{f_{uart0_mclk}}$=115200*16=1.8432Mhz，（divisor 在 uart 寄存器中可以配置）。（其中，mclk_in*(UART0_CLK_GR)/8 表示 clk_afgr）由此得到：

$$\frac{\mathrm{UART0_{MUL}}}{\mathrm{UART0_{DIV}}} = \frac{\mathrm{f_{uart0\_mclk}} * 16}{\mathrm{f_{mclk\_in}} * \mathrm{UART0\_CLK\_GR}} = \frac{\mathrm{f_{uart0\_mclk}} * 16}{\mathrm{f_{mclk\_in}} * 8} = \frac{1.8432\mathrm{M} * 16}{32\mathrm{M} * 8} = \frac{72}{625}$$

最终 UART0 波特率为 115200 时的 CPR 寄存器的配置为

- CPR_UART0_CLK_GRCTL.UART0_CLK_GR=8
- CPR_UART0_CLK_CTL.UART0_CLK_MUL=72
- CPR_UART0_CLK_CTL.UART0_CLK_DIV=625

同理可以得到 UART0 波特率为 1MHz 时的 CPR 寄存器的配置为：

- CPR_UART0_CLK_GRCTL.UART0_CLK_GR=8
- CPR_UART0_CLK_CTL.UART0_CLK_MUL=1
- CPR_UART0_CLK_CTL.UART0_CLK_DIV=1

4.4.3.2. 睡眠唤醒机制

4.4.3.2.1. 睡眠的进入

CPR 提供了两种进入睡眠的方式，一种方式为硬件自启动睡眠，另外一种方式为软件启动睡眠，图 4-5 给出了关于睡眠进入的逻辑条件图。

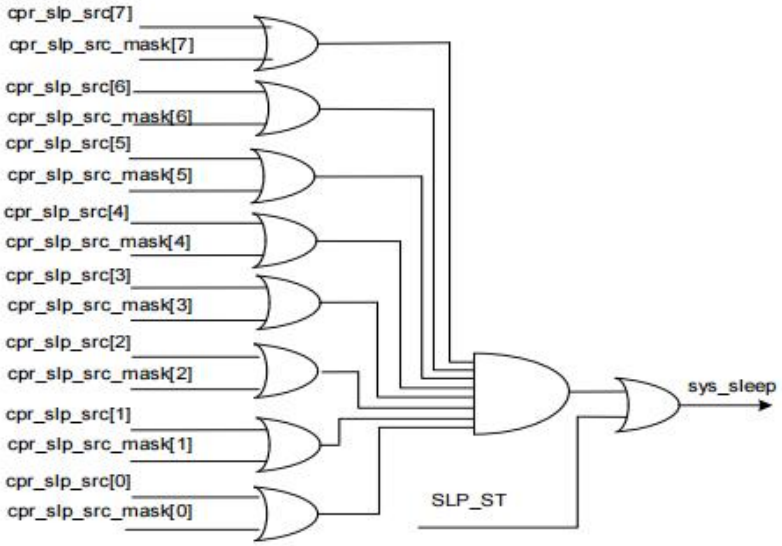

图 4-5 CPR 的 sys_sleep 产生逻辑图

CPR_SLP_SRC_MASK 高 16 位写使能，对应位屏蔽后则该空闲状态信号不作为启动睡眠的条件信号。比如 m0_sleeping MASK 如果置为 1，则 WFI 指令产生的信号不进入睡眠。影响硬件睡眠的信号及其描述如下表 4-3 所示。

表 4-3 硬件睡眠条件指示信号表格

硬件睡眠条件指示信号	描述	屏蔽位
m0_sleeping	M0 睡眠信号，指示 M0 处于睡眠状态，由 M0 硬件产生。写 WFI 指令可以使该信号拉高。 1：处于睡眠状态 0：处于工作状态	slp_src_mask[0]
cdbgpwrupreq	DEBUG 上电请求信号，指示当前连接仿真器并处于 DEBUG 状态。 1：发出 DEBUG 上电请求 0：无效	slp_src_mask[1]

dmas_idle	DMASIDLE 指示信号，DMAS 不工作时拉高。 1：处于 IDLE 状态 0：处于工作状态	slp_src_mask[2]

当硬件或软件睡眠条件满足并产生 sys_sleep 变为 1 后，标志硬件允许进入睡眠。接下来，由图 4-6 所示电路单元负责将睡眠满足的条件传递给睡眠状态机：

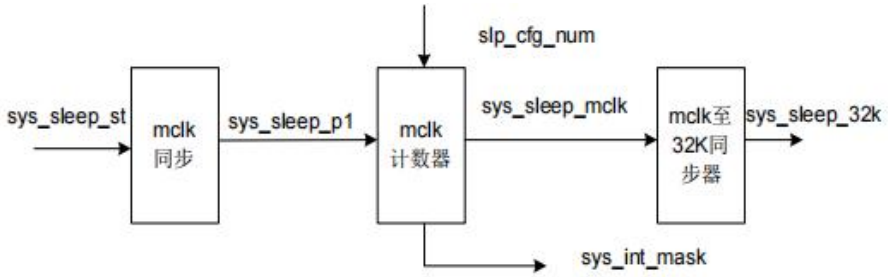

图 4-6 睡眠进入信号传递机制

对上图信号进行说明如下：

- sys_sleep_st 变高两个 clk32m 时钟周期之后，sys_sleep_p1 变为高电平
- slp_cfg_num 为配置寄存器，对应 CPR_SLP_CNT_LIMIT.SLP_CONFIG_NUM
- 当 sys_sleep_p1 变为高电平后，CPR 内部 clk32m 计数器开始计数，当计数到 SLP_CONFIG_NUM 所规定的值时，给出信号 sys_sleep_mclk，同时将 sys_int_mask 变为高电平；sys_int_mask 在芯片进入睡眠后自动变为低电平
- sys_sleep_mclk 变为高电平后，经过同步机制，产生一个脉冲 sys_sleep_32k，从而引发睡眠状态机进入睡眠流程

注：

slp_cfg_num 对应寄存器 CPR_SLP_CNT_LIMIT.SLP_CONFIG_NUM 的值。

sys_intr_mask_cycle 对应寄存器 CPR_SLP_CNT_LIMIT.SYS_INTR_MASK_CYCLE 的值。

sys_int_mask 为 CPR 内部信号，作用是屏蔽所有到 CPU 的中断。

4.4.3.2.2. 睡眠时 PD 域的断电控制

PWR3V 模块可以通过检测 CPR 发出的断电控制信号 sys_pw_sw，关闭 PD 域的供电。寄存器 CPR_SLP_PD_MASK 的 SYS_SLP_PD_MASK 位可以禁止 CPR 在睡眠时发出的对 PD 域的断电控制信号 sys_pw_sw。

4.4.3.2.3. 睡眠的唤醒

在睡眠状态下，由芯片的中断事件唤醒系统。

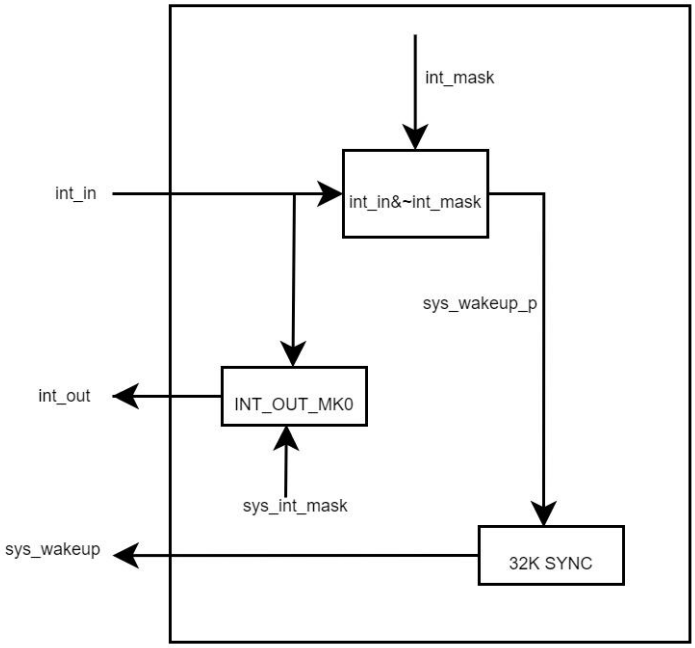

图 4-7 唤醒机制

相关描述如下：

- int_in[31:0]的每一比特对应各模块提供给 CPR 中断请求信号；int_out[31:0]为最终 CPR 送给 M0 的中断请求信号；
- INT_MASK[31:0]为 CPR_SLPCTL_INT_MASK 寄存器，用以屏蔽 int_in[31:0]中各中断状态信号对 SYS_CTRL 状态机的影响。SYS_CTRL 根据寄存器配置确定需要断电和关闭时钟的模块；
- sys_int_mask 为用以保护睡眠进入状态的全局中断屏蔽信号，在睡眠进入过程中产生。
- int_in 经过 int_mask 后，产生 sys_wakeupl_p，经过 32K 同步器产生 sys_wakeup 信号。该信号传送至 SYS_CTRL 状态机，用以唤醒系统。

4.4.3.2.4. 睡眠唤醒中信号关系图

图 4-8 给出了在睡眠唤醒中电源使能控制信号 pw_sw、晶振使能控制信号 OSCEN、中断屏蔽信号 sys_int_mask 信号与睡眠启动信号 sys_sleep 的关系。

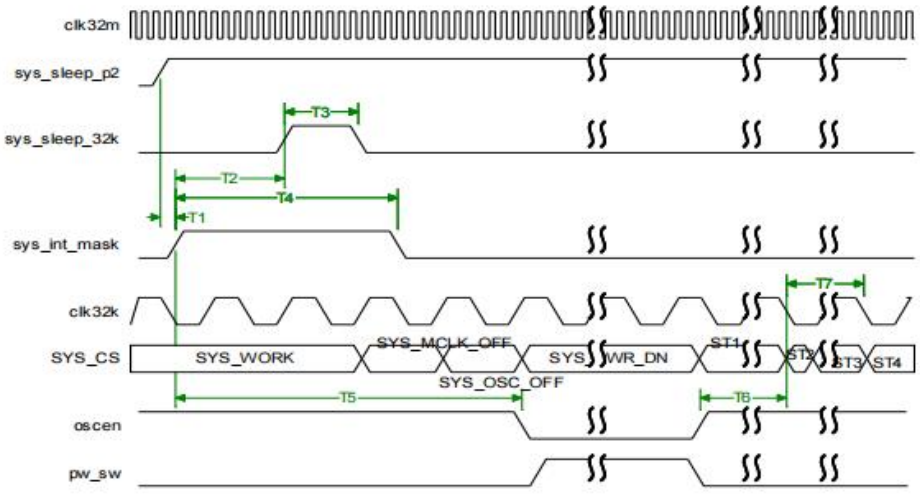

图 4-8 睡眠唤醒过程中关键信号关系图

注：

ST1：SYS_OSC_ON 状态

ST2：SYS_MCLK_ON 状态

ST3：SYS_MRST_GEN 状态

ST4：SYS_WORK 状态

对图中 T1~T7 的时间长度介绍如下：

T1：表示“从 sys_sleep_p2 变高（即睡眠的软硬件条件满足）到 sys_int_mask 变高的时间”，其长度由（CPR_SLP_CNT_LIMIT.SLP_CONFIG_NUM-CPR_SLP_CNT_LIMIT.SYS_INTR_MASK_CYCLE）大小决定，单位是 32M 时钟周期；

T2：表示“从 sys_int_mask 变高到 sys_sleep_32k 变高的时间”，其长度等于 CPR_SLP_CNT_LIMIT.SYS_INTR_MASK_CYCLE 个 CLK32M 周期，加上 32K 同步器的时延；

T3：表示“sys_sleep_32k 持续为高的时间”，其长度等于 1 个 32KHz 时钟周期；

T4：表示“sys_int_mask 持续为高的时间”，其长度约等于 3 到 4 个 32KHz 时钟周期；

T5：表示“从 sys_int_mask 变高到 OSCEN 变低的时间”，其长度在 4 至 5 个 32KHz 时钟周期之间；

T6：表示 32MHz 晶振稳定时间；

T7：表示系统复位稳定时间

芯片的 PMU 控制需要软件控制 PWR3V 模块的寄存器，对 BOOST、BUCK、LDO_AO、LDO_DIG 的开关等进行配置。芯片管脚 OSCEN 上电复位时为 1，可用来控制 32MHz 晶振的打开，OSCEN 信号变化与睡眠相关：默认情况下（CPR_SLP_CTL.OSCEN_CTL 为 0），当芯片进入睡眠后 OSCEN 变低，当唤醒中断到来后 OSCEN 变高；当 CPR_SLP_CTL.OSCEN_MASK=1 时，OSCEN 在睡眠中不发生变化。

如果芯片在睡眠过程中断电，则需要根据睡眠的状态配置 PMU 的供电单元的开关以及设置适当的 MASK 位。详细流程请参考 PWR3V 模块的文档。

4.4.3.2.5. 睡眠及其唤醒过程中 TIMER 接口总线时钟切换机制

TIMERx（x=0/1/2/3）的中断（当 TIMERx 的工作时钟是 clk32k 时产生的中断）只有在 TIMER 模块有接口总线时钟的情况下才能产生。在睡眠时，为节省功耗，只有 32K 时钟（clk32k）在工作，其他时钟都不工作。为了允许使用 TIMER 中断的唤醒，CPR 提供了 TIMER 接口总线时钟在睡眠唤醒过程中的切换机制，即允许该接口总线时钟在非睡眠时为总线时钟 timer_pclk，在睡眠中切换为 clk32k。以睡眠中 TIMER0 中断的产生为例，说明接口总线时钟切换的具体操作：

- 在工作状态中
  - 配置 CPR 的 TIMER0 工作时钟控制寄存器 CPR_TIMER0_CLK_CTL.TIMER0CLK_SEL 为期望值，选择 32KHz 时钟；
  - 根据定时器（TIMER）中工作流程配置 TIMER0 的中断产生条件；
  - 配置 CPR_LP_CTL.TIMER_SYSCLK_SEL 位为 1，在睡眠时，TIMER 总线时钟选择 clk32k；
- 芯片进入睡眠，硬件自动将 TIMER 的总线时钟从 timer_pclk 切换到 clk32k；
- 当 TIMER0 的中断产生时，唤醒芯片，进入唤醒过程，如上一节所描述；
- 芯片醒来后，硬件自动将 TIMER 的接口总线时钟从 clk32k 切换到 timer_pclk；

4.4.3.3. AO 域的控制和寄存器

AO 域是系统中保持运行状态单元的集合，在普通睡眠模式状态下，AO 域时钟保持上电状态。CPR 的系统控制状态机和睡眠唤醒模块均属于 AO 域，负责系统的状态切换，睡眠唤醒，AO 时钟控制，AO 域寄存器管理等重要功能。

4.4.3.3.1. AO 时钟与复位控制

除了 CPR_AO 外，工作在 AO 域的模块主要是 RTC。这些单元的时钟和复位控制均处于 CPR 的 AO 域中。在系统 PD 域断电时，AO 单元的模块能够保持正常的工作。例如，CPR_OTHERCLKEN_GRCTL 控制 RTC 的时钟使能。

4.4.3.3.2. AO 域寄存器

部分系统重新启动相关的寄存器的配置放置在 AO 域。在系统更新配置后，能够在系统唤醒时，使用新的配置用于重新启动，例如 CPR_SYS_TIME 等。CPR 还为软件预留了两个通用的 AO 寄存器，用来保存状态或者标志信息。

AO 域寄存器控制通过单独的 APB 接口进行。AO 域和 PD 域的 CPR 寄存器有着不同的基地址。详见章节 AO 域寄存器地址映射和其它控制寄存器。

4.4.3.4. 低功耗及 UART 时钟保护

4.4.3.4.1. UART 时钟保护

本节专门介绍 CPR_LP_CTL.UARTx_CLK_OFF_PROTECT_EN 的功能（其中 x=0，1）。UART 时钟保护目的主要是为了防止 uart 在数据传输的过程中工作频率改变。

- UARTx_CLK_OFF_PROTECT_EN 配置为 0，软件可直接配置 UARTx 工作时钟，并且软件的配置立即生效；
- UARTx_CLK_OFF_PROTECT_EN 配置为 1：
  - 如果 UARTx 处于非工作状态（不使能），则软件不能修改 UARTx 的时钟配置；
  - 如果 UARTx 处于工作状态且没有数据传输，则软件可以立即修改 UARTx 的时钟配置，且配置立即生效；
  - 如果 UARTx 时钟当前处于工作状态，且 UARTx 当前在进行数据传输状态，软件可以修改 UARTx 的时钟配置，但新的 UARTx 时钟配置会等到 UARTx 空闲（数据传输完成且没有后续的数据传输）后，才会生效，从而对 UARTx 与外界交互实现保护。

4.4.3.4.2. 总线低功耗和模块低功耗控制

总线互联模块的时钟可以在总线的 MASTER 和 SLAVE 不工作的情况下自动将时钟关断。

- 配置 CPR_LP_CTL.x_BUS_LP_EN（x=M0/DMA）为 1 可以使能 MASTER 侧的总线单元时钟在 MASTER 不工作时硬件自动关断。
- 配置 CPR_LP_CTL.x_ICM_LP_EN(x=SHRAM0/CTLAPB)为 1 可以使能 SLAVE 侧的总线单元时钟在该 SLAVE 的所有 MASTER 不工作时硬件自动关断。

- 配置 CPR_LP_CTL.M0_HCLK_LP_EN 可以使能 M0 总线时钟自动关断。
- 配置 CPR_LP_CTL.DAP_CLK_LP_EN 可以使能 DAP 时钟自动关断。

4.4.4. 关机阶段

一般应用场景中，芯片不会全部断电。如果需要全芯片断电，则需要通过外置开关切断 VBAT3V（BUCK 模式）或者 VINBST（BOOST 模式）实现关机。

4.4.5. 控制功能

除了对芯片的整体的时钟和功耗进行控制以外，CPR 还有一些芯片全局控制功能，可以实现一些全局控制寄存器的配置。

4.4.5.1. 各功能模块的 CTL 寄存器配置

包括 PAD 的上下拉配置（CPR_CTL_PECTLn），PAD 的复用控制（CPR_CTL_MUXCTLn），M0 的配置（CPR_CTL_M0_MIS/CPR_CTL_M0_CURRPRI）等。详见各 CTL 寄存器列表。

4.4.6. 中断描述

CPR 能产生的中断及各中断代表的意义如表所示。

表 4-4 CPR 中断模式表

中断原始状态	中断使能	中断状态	描述
cdbgpwrupreq_intr_raw	cdbgpwrupreq_intr_en	cdbgpwrupreq_intr	系统调试中断

与中断相关的各寄存器描述如下：

- CPR_INT_RAW 中各比特位为中断原始状态寄存器；
- 通过控制 CPR_INT_EN 中比特，可以控制当发生 CPR_INT_RAW 中所述各中断时，是否将中断送给 M0；
- 通过读取 CPR_INT 中比特，可获悉具体的中断位状态；
- 通过向 CPR_INT 目标比特位写 1，可清除相应中断。

4.5. 信号及接口描述

4.5.1. CPR 部分接口信号

CPR 的接口信号参见表 4-5。

表 4-5 CPR 信号与接口描述表

名称	宽度	方向	描述	备注
CTL_APB 总线接口				内部信号
芯片控制				
testclk	1	I	测试时钟	管脚信号
testmode	1	I	测试模式选择信号	管脚信号
bootctl	1	I	启动控制信号	管脚信号
CPR AO/PD 域交互信号				
wdt_sys_rstn	1	O	WDT 复位 32K 域控制输出	内部信号
prstn_32k_ao	1	I	32K 域复位信号	内部信号
mclk_en	1	I	32M 时钟使能控制	内部信号
prstn_wdt	1	I	WDT 复位控制信号	内部信号
timer_pclk_sw	1	I	TIMER 的 pclk 切换控制信号	内部信号
slp_ret1n_clken	1	I	总线时钟 gating 控制信号	内部信号
mrstn_out	1	I	数字域 32M 域时钟复位信号	内部信号
sys_mclk_off	1	I	数字域 32M 主时钟关闭控制信号	内部信号
sys_cs	3	I	系统状态标示符	内部信号
iso_en	1	I	系统隔离使能	内部信号
pw_sw	1	I	系统断电指示符	内部信号
睡眠唤醒				
slp_src	8	I	硬件睡眠源	内部信号
int_in	32	I	M0 中断源输入，用于睡眠唤醒	内部信号
int_out	32	O	M0 中断输出，送入 M0	内部信号
cpr_intr	1	O	cpr 中断请求，送入 M0	内部信号
sys_sleep	1	O	系统睡眠请求	内部信号
sys_wakeup	1	I	系统唤醒请求	内部信号
dbgpwrupreq	1	I	coresightDEBUG 上电 REQ	内部信号
cdbgpwrupack	1	O	coresightDEBUG 上电 ACK	内部信号

cdbgpwrupreq_32k	1	O	coresightDEBUGREQ 同步到 32K 域	内部信号
cdbgpwrupreq_16p369m	1	O	coresightDEBUGREQ 同步到 32m 域	内部信号
时钟相关				
外部管脚时钟输入				
clk32k	1	I	32k 时钟输入，来自 CPR AO 域	内部信号
mclkin	1	I	32m 时钟输入 管脚信号	管脚信号
32K 晶体接口控制				
osc32k_restart	1	O	32K 晶体接口 port 复位控制	内部信号
osc32k_ds	3	O	32K 晶体接口 port 控制信号	内部信号
时钟控制				
m0_gatehclk	1	I	m0_hclk 关断使能	内部信号
timer0_en	1	I	timer0 使能，用于使能 timer0_clk	内部信号
timer1_en	1	I	timer1 使能，用于使能 timer1_clk	内部信号
timer2_en	1	I	timer2 使能，用于使能 timer2_clk	内部信号
timer3_en	1	I	timer3 使能，用于使能 timer3_clk	内部信号
aotimer0_en	1	I	aotimer0 使能,用于使能 aotimer0_clk	内部信号
aotimer1_en	1	I	aotimer1 使能,用于使能 aotimer1_clk	内部信号
uart0_lp_req_pclk	1	I	uart0pclk 低功耗指示	内部信号
uart0_lp_req_sclk	1	I	uart0sclk 低功耗指示	内部信号
uart1_lp_req_pclk	1	I	uart1pclk 低功耗指示	内部信号
uart1_lp_req_sclk	1	I	uart1sclk 低功耗指示	内部信号
uart2_lp_req_pclk	1	I	uart2pclk 低功耗指示	内部信号
uart2_lp_req_sclk	1	I	uart2sclk 低功耗指示	内部信号
clk32k_glitch_bypass	1	O	32k 滤毛刺功能旁路,0：旁路 32k 滤毛刺功能	内部信号
模块时钟输出				
M0 时钟				
m0_fclk	1	O	M0 FCLK，最高 100M	内部信号
m0_hclk	1	O	M0 总线时钟，最高 100M	内部信号
m0_stclk	1	O	M0 st_clk，32KHz，供 M0 内部 ST 计数器使用	内部信号

总线相关时钟				
rom_hclk	1	O	ROM 总线时钟，频率同 m0_fclk	内部信号
m0_ram_hclk	1	O	M0 RAM 总线时钟，频率同 m0_fclk	内部信号
shram0_hclk	1	O	SHRAM0 总线时钟，频率同 m0_fclk	内部信号
bus_clk	1	O	AHB 总线时钟，最高 32MHz	内部信号
dmas_hclk	1	O	DMAS 总线时钟，频率同 bus_clk	内部信号
ctl_bridge_hclk	1	O	CTL APB 桥总线时钟，频率同 bus_clk	内部信号
ctl_pclken	1	O	CTL APB 桥总线时钟使能	内部信号
ctl_pclk	1	O	CTL APB 桥 APB 时钟，最高 50MHz	内部信号
cpr_pclk	1	O	CPR 总线时钟，频率同 ctl_pclk	内部信号
wdt_pclk	1	O	WDT 总线时钟，频率同 ctl_pclk	内部信号
Bluetooth 相关时钟				
bt_clk	1	O	bluetooth 工作主时钟	内部信号
bt_modem_clk	1	O	bluetooth modem 工作时钟	内部信号
bt_pclk	1	O	Bluetooth APB 时钟	内部信号
bt32k_clk	1	O	Bluetooth 32K 工作时钟	内部信号
DEBUG 模块相关时钟				
dap_clk	1	O	调试 DAP 时钟，最高 100MHz	内部信号
UART 模块相关时钟				
uart0_clk	1	O	UART0 工作时钟，根据 UART0 需要的波特率配置分频	内部信号
uart0_pclk	1	O	UART0 总线时钟，频率同 ctl_pclk	内部信号
uart1_clk	1	O	UART1 工作时钟，根据 UART1 需要的波特率配置分频	内部信号
uart1_pclk	1	O	UART1 总线时钟，频率同 ctl_pclk	内部信号
uart2_clk	1	O	UART2 工作时钟，根据 UART1 需要的波特率配置分频	内部信号
uart2_pclk	1	O	UART2 总线时钟，频率同 ctl_pclk	内部信号
SSI 模块相关时钟				
ssi0_mclk	1	O	SSI0 工作时钟	内部信号

ssi1_mclk	1	O	SSI1 工作时钟	内部信号
ssi2_mclk	1	O	SSI2 工作时钟	内部信号
ssi0_pclk	1	O	SSI0 配置时钟，频率同 ctl_pclk	内部信号
ssi1_pclk	1	O	SSI1 配置时钟，频率同 ctl_pclk	内部信号
ssi2_pclk	1	O	SSI2 配置时钟，频率同 ctl_pclk	内部信号
I2C 模块相关时钟				
i2c_clk	1	O	I2C 工作时钟	内部信号
i2c_pclk	1	O	I2C 总线时钟，频率同 ctl_pclk	内部信号
GPADC 模块相关时钟				
gpadc_pclk	1	O	GPADC 总线时钟，频率同 ctl_pclk	内部信号
QDEC 模块相关时钟				
qdec_clk	1	O	QDEC 工作时钟	内部信号
qdec_pclk	1	O	QDEC 总线时钟，频率同 ctl_pclk	内部信号
PWM 模块相关时钟				
pwm_clk	1	O	PWM 工作时钟	内部信号
pwm_pclk	1	O	PWM 总线时钟，频率同 ctl_pclk	内部信号
GPIO 模块相关时钟				
gpio_clk	1	O	GPIO 工作时钟，固定 32KHz	内部信号
gpio_pclk	1	O	GPIO 总线时钟，频率同 ctl_pclk	内部信号
TIMER 模块相关时钟				
timer_pclk	1	O	TIMER 总线时钟，频率同 ctl_pclk	内部信号
timer0_clk	1	O	TIMER0 工作时钟	内部信号
Timer1_clk	1	O	TIMER1 工作时钟	内部信号
Timer2_clk	1	O	TIMER2 工作时钟	内部信号
Timer3_clk	1	O	TIMER3 工作时钟	内部信号
AOTimer0_clk	1	O	AOTIMER0 工作时钟	内部信号
AOTimer1_clk	1	O	AOTIMER1 工作时钟	内部信号
复位控制				
复位控制信号				
prstn	1	I	上电复位，来自芯片管脚	管脚信号
wdt_sys_rst_n	1	I	WDT 系统复位	内部信号
m0_sysresetreq	1	I	M0 复位请求	内部信号

m0_lockup	1	I	M0 死锁指示	内部信号
M0&DEBUG 模块复位				
dbg_poresetn	1	O	CORESIGHT 调试模块复位	内部信号
m0_poresetn	1	O	M0 上电复位	内部信号
m0_sysresetn	1	O	M0 系统复位	内部信号
AHB 总线模块复位				
rom_rstn	1	O	ROM 模块复位	内部信号
m0_ram_rstn	1	O	M0RAM 模块复位	内部信号
dma_rstn	1	O	DMA 模块复位	内部信号
bt_rstn	1	O	Bluetooth 模块复位	内部信号
CTLAPB 总线模块复位				
uart0_rstn	1	O	UART0 模块复位	内部信号
uart1_rstn	1	O	UART1 模块复位	内部信号
uart2_rstn	1	O	UART2 模块复位	内部信号
ssi0_rstn	1	O	SPI0 模块复位	内部信号
ssi1_rstn	1	O	SPI1 模块复位	内部信号
ssi2_rstn	1	O	SPI2 模块复位	内部信号
ctl_rstn	1	O	CTLAPB 桥模块复位	内部信号
i2c_rstn	1	O	I2C 模块复位	内部信号
qdec_rstn	1	O	QDEC 模块复位	内部信号
pwm_rstn	1	O	PWM 模块复位	内部信号
gpadc_rstn	1	O	GPADC 模块复位	内部信号
timer_presetn	1	O	TIMER 总线复位	内部信号
timer0_rstn	1	O	TIMER0 模块复位	内部信号
timer1_rstn	1	O	TIMER1 模块复位	内部信号
timer2_rstn	1	O	TIMER2 模块复位	内部信号
timer3_rstn	1	O	TIMER3 模块复位	内部信号
wdt_rstn	1	O	WDT 模块复位	内部信号
gpio_rstn	1	O	GPIO 模块复位	内部信号
TESTPIN 观测				
cprtst_slp_pw_sw	1	O	睡眠断电指示	内部信号
cprtst_slp_iso_en	1	O	睡眠断电 ISO 使能	内部信号

cprtst_slp_rstn_out	1	O	睡眠断电复位	内部信号
cprtst_sys_intr_mask	1	O	睡眠中断屏蔽	内部信号
cprtst_sys_cs	3	O	SYS_CTRL 状态机信号	内部信号
cprtst_sys_sleep	1	O	系统睡眠信号	内部信号
cprtst_sys_wakeup	1	O	系统唤醒信号	内部信号
CTL 控制信号				
ssi0_protocol	1	O	SSI0 支持的协议类型选择	内部信号
ssi1_protocol	1	O	SSI1 支持的协议类型选择	内部信号
ssi2_protocol	1	O	SSI2 支持的协议类型选择	内部信号
ssi1_master_en	1	O	SSI1 的 Master/Slave 选择	内部信号
remap	1	O	REMAP	内部信号
remap_base	20	O	REMAP 基地址	内部信号
总线配置控制信号				
shram0_icm_priority	2	O	SHRAM0 的 ICM 优先权设置	内部信号
ctlapb_icm_priority	2	O	ctlapb 的 ICM 优先权设置	内部信号
低功耗控制控制信号				
m0_bus_lp_en	1	O	M0 的总线低功耗使能	内部信号
dmas_bus_lp_en	1	O	dmas 的总线低功耗使能	内部信号
mem_lp_en	1	O	Memory 的低功耗使能	内部信号
ctlapb_icm_lp_en	1	O	ctlapbICM 的总线低功耗使能	内部信号
shram0_icm_lp_en	1	O	SHRAM0 的总线低功耗使能	内部信号
管脚控制信号				
muxctl1	32	O	管脚复用控制 MUXCTL1	内部信号
muxctl2	32	O	管脚复用控制 MUXCTL2	内部信号
muxctl3	32	O	管脚复用控制 MUXCTL3	内部信号
pe_ctrl1	32	O	管脚上下拉控制 PECTL1	内部信号
pe_ctrl2	32	O	管脚上下拉控制 PECTL2	内部信号
GPIO 复用控制信号				
gpio_fun_sel0-gpio_fun_sel31	5	O	管理 GPIO0-31 复用控制信号	内部信号
测试控制信号				
testctrl0 -9	8	O	测试控制信号 0-9	内部信号

M0 CTL 控制信号				
m0_mpudisable	1	O	MPU disable 控制 1：disable 掉 MPU。 0：打开 MPU。	内部信号
m0_fixmastertype	1	O	控制 AHB-AP 的 HMASTER， 1：HMASTER 总是 1. 0：HMASTER 可以是 0 或 1。	内部信号
m0_stcalib	26	O	system tick 控制寄存器。 m0_stcalib [25]：是否使用参考时钟。 m0_stcalib [24]：10ms 是否准确。 m0_stcalib [23：0]：10ms 的周期数。	内部信号
m0_currpri	8	I	M0 当前的中断优先级	内部信号

4.5.2. CPRAO 部分接口信号

CPRAO 的接口信号参见表 4-6。

表 4-6 CPR AO 域信号与接口描述表

名称	宽度	方向	描述	备注
芯片控制				
testclk	1	I	测试时钟	管脚信号
testmode	1	I	测试模式选择信号	管脚信号
oscen	1	O	晶振使能控制信号	管脚信号
rfpmu_spi_sel	1	O	RF 与 PMU 的 SPI 选择信号	内部信号
CPRAO/CPR 交互信号				
wdt_sys_rstn	1	I	WDT 复位 32K 域控制输入	内部信号
prstn_32k_ao	1	O	32K 域复位信号	内部信号
mclk_en	1	O	32M 时钟使能控制	内部信号
prstn_wdt	1	O	WDT 复位控制信号	内部信号
timer_pclk_sw	1	O	TIMER 的 pclk 切换控制信号	内部信号
slp_ret1n_clken	1	O	总线时钟 gating 控制信号	内部信号
mrstn_out	1	O	数字域时钟复位信号	内部信号
sys_mclk_off	1	O	数字域主时钟关闭控制信号	内部信号

sys_cs	3	O	系统状态标示符	内部信号
iso_en	1	O	系统隔离使能	内部信号
pw_sw	1	O	系统断电指示符	内部信号
APB 接口(略)				
系统时钟				
clk32kin	1	I	32K 晶振输入	内部信号
mclk_chip_i	1	I	32M 时钟输入	内部信号
clk32k_rc_i	1	I	内部 32KRC 振荡器时钟输入	内部信号
clk32k	1	O	系统 32K 工作时钟	内部信号
唤醒控制寄存器				
int_in	32	I	M0 中断源输入，用于睡眠唤醒	内部信号
sys_sleep	1	I	系统睡眠请求	内部信号
sys_wakeup	1	O	系统唤醒请求	内部信号
断电相关				
ram_pd_iso_en	1	I	RAM 断电隔离控制，来自 PD_FSM	内部信号
ram_pd_pw_sw	1	I	RAM 断电控制，来自 PD_FSM	内部信号
ram_pd_rstn_out	1	I	RAM 断电复位信号，来自 PD_FSM	内部信号
sys_rstn_out	1	O	系统复位控制信号	内部信号
sys_iso_en	1	O	系统隔离 isolation 控制信号	内部信号
sys_pw_sw	1	O	系统断电控制信号	内部信号
ram_rstn_out	1	O	RAM 复位控制信号	内部信号
ram_iso_en	1	O	RAM 隔离 isolation 控制信号	内部信号
ram_pw_sw	1	O	RAM 断电控制信号	内部信号
RTC 时钟复位				
rtc_clk	1	O	RTC 工作时钟	内部信号
rtc_rstn	1	O	RTC 复位信号	内部信号

4.6. CPR 寄存器映射表

表 4-7 是 CPR 的寄存器概述和地址映射。

- CPR 的基地址为：0x40000000

表 4-7 CPR 寄存器映射表

寄存器名	描述	方向&宽度	地址偏移	复位值
SLP_SRC_MASK	睡眠源屏蔽寄存器	RW 9bit	0x00	0x00
SLP_CNT_LIMIT	睡眠进入阈值寄存器	RW 15bit	0x10	0x3034
SLP_ST	睡眠启动寄存器	RW 2bit	0x14	0x0
SLP_FSM_STATE	睡眠状态机状态寄存器	R 3bit	0x18	0c0
M0_FCLK_CTL	M0 fclk 时钟控制寄存器	RW 12bit	0x20	0x0
CTL_PCLK_GRCTL	CTL_PCLK 时钟控制寄存器	RW 5bit	0x24	0x4
UART0_CLK_GRCTL	UART0 CLK 时钟 GR 控制寄存器	RW 5bit	0x30	0x8
UART0_CLK_CTL	UART0 CLK 时钟控制寄存器	RW 32bit	0x34	0x480271
UART1_CLK_GRCTL	UART1 CLK 时钟 GR 控制寄存器	RW 5bit	0x38	0x8
UART1_CLK_CTL	UART1 CLK 时钟控制寄存器	RW 32bit	0x3C	0x480271
BT_CLK_CTL	BT_CLK 时钟控制寄存器	RW 5bit	0x40	0x3
BT_MODEM_CTL	BT_MODEM_CLK 时钟控制寄存器	RW 5bit	0x48	0x1
QDEC_CLK_CTL	QDEC 时钟控制寄存器	RW 9bit	0x4C	0x7
SSI0_MCLK_CTL	SSI0_MCLK 时钟控制寄存器	RW 5bit	0x50	0x13
SSI1_MCLK_CTL	SSI1_MCLK 时钟控制寄存器	RW 5bit	0x54	0x3

TIMER0_CLK_CTL	TIMER0_CLK 时钟控制寄存器	RW 31bit	0x58	0x909
TIMER1_CLK_CTL	TIMER1_CLK 时钟控制寄存器	RW 31bit	0x5C	0x909
TIMER2_CLK_CTL	TIMER2_CLK 时钟控制寄存器	RW 31bit	0x60	0x909
TIMER3_CLK_CTL	TIMER3_CLK 时钟控制寄存器	RW 31bit	0x64	0x909
PWM_CLK_CTL	PWM_CLK 时钟控制寄存器	RW 32bit	0x68	0x909
AHBCLKEN_GRCTL	AHB 总线时钟使能寄存器	RW 5bit	0x6C	0x1f
CTLAPBCLKEN_GRCTL	CTL_APB 总线时钟使能寄存器	RW 15bit	0x70	0x114
OTHERCLKEN_GRCTL	其他时钟使能寄存器	RW 4bit	0x74	0x1
I2C_CLK_CTL	I2C_CLK 时钟控制寄存器	RW 5bit	0x78	0x1
INT	M0 的 CPR 中断状态寄存器	RC 11bit	0xA0	0x0
INT_EN	M0 的 CPR 中断使能寄存器	RW 11bit	0xA4	0x0
INT_RAW	M0 的 CPR 中断原始状态寄存器	R 11bit	0xA8	0x0
GPIO_FUN_SELx[8]	GPIO_FUNSEL0..7	RW 29bit	0xC0-0xDC	0x0
RSTCTL_CTLAPB_SW	CTLAPB 模块软复位寄存器	RW 13bit	0x100	0x1FFF
RSTCTL_SUBRST_SW	SUB 模块软复位寄存器	RW 10bit	0x104	0x3FFF
RSTCTL_M0RST_SW	M0 相关软复位寄存器	RW 5bit	0x108	0x1F
RSTCTL_M0RST_MASK	M0 复位屏蔽寄存器	RW 5bit	0x10C	0X1F

RSTCTL_LOCKUP_STATE	M0 死锁状态指示寄存器	R 1bit	0x110	0X0
RSTCTL_WDTRST_MASK	WDT 复位屏蔽寄存器	RW 2bit	0x114	0X3
LP_CTL	低功耗控制寄存器	RW 11bit	0x118	0X0
CTL_SHRAM_ICM_PRIORITY	SHRAM 的 ICM 优先级配置寄存器	RW 6bit	0x120	0X0
CTL_CTLAPB_ICM_PRIORITY	CTL_APB 桥的 ICM 优先级配置寄存器	RW 2bit	0x124	0X0
CTL_MUXCTL1_2[2]	MUXPIN 控制寄存器 12	RW 32bit	0x128-0x12C	0X0
RF_REG3	CRP 通用寄存器 3	RW 32bit	0x130	0x0
RF_REG4	CRP 通用寄存器 4	RW 6bit	0x134	0x0
REMAP_CTRL	REMAP 控制寄存器	RW 32bit	0x13C	0x0
CTL_BOOTCTL	BOOTCTL 指示寄存器	R 1bit	0x140	0x0
CTL_M0_STCALIB	M0 配置寄存器	RW 26bit	0x144	0x13F
CTL_M0_IRQLATENCY	M0 IRQ 延迟寄存器	RW 8bit	0x148	0x0
SSI_CTRL	SSI 控制寄存器	RW 6bit	0x160	0x13
TESTCTRL[0-9]	测试控制寄存器 0...9	RW 8bit	0x170-0x194	0X0
CTL_MUXCTL3	MUXPIN 控制寄存器 3	RW 32bit	0x19C	0X0
RF_REG5	USB、CAN 控制寄存器	RW 29bit	0x1A0	0x0
RF_REG0	CRP 通用寄存器 0	RW 30bit	0x1B0	0x13fb0003
RF_REG1	USB、2.4G、ADUDIO_ADC 控制寄存器	RW 29bit	0x1B4	0x0
RF_REG2	Audio ADC 小数分频	RW 32bit	0x1B8	0x0
OPA_CTRL_REG	Audio adc ananlog 控制寄存器	RW 23bit	0x1BC	0x41fc1c
UART2_CLK_GRCTL	uart2_clk GR 参	RW 6bit	0x250	0x8

	数更新寄存器			
UART2_CLK_CTL	二级分频	RW 32bit	0x254	0x480271
I2S_CLK_CTL	I2S_CLK 时钟控制寄存器	RW 14bit	0x258	0x7c
CAN_CLK_CTL	CAN_CLK 时钟控制寄存器	RW 14bit	0x25C	0xff
CMP_CTL	CMP 控制寄存器	RW 32bit	0x260	0x800f0000
CMP_INT	比较器中断状态寄存器	RC 3it 1	0x264	0x0
CMP_INT_RAW	比较器原始中断状态寄存器	R 3bit	0x268	0x0
CMP_INT_EN	比较器中断使能寄存器	RW 3bit	0x26C	0x0
FMC_CTL	FMC 控制寄存器	RW 15bit	0x270	0x3503
OTP_CTL	OTP 控制寄存器	RW 10bit	0x274	0x111

4.7. CPRAO 寄存器映射表

表 4-8 是 CPRAO 寄存器概述和地址映射。

CPRAO 域基地址为：0x40002400

表 4-8CPR AO 地址映射表

寄存器名	描述	方向&宽度	地址偏移	复位值
睡眠控制寄存器				
SLP_CTL	睡眠控制寄存器	RW 1bits	0x4	0x0
SLPCTL_INT_MASK	睡眠唤醒 M0 中断屏蔽寄存器	RW 32bits	0x8	0x0
SLP_PD_MASK	睡眠断电屏蔽寄存器	RW 8bits	0xC	0x1
系统配置寄存器				
MCLK_DIV_CLK_CTL	MCLK_DIV_CLK(32K)分频控制寄存器	RW 13bits	0x10	0x13E7
RFPMU_SPI_CTL	SPI 控制寄存器	RW 1bits	0x14	0x0

SYS_TIME	系统定时控制寄存器	RW 20bits	0x1C	0x7200
BBLDO_ADJ	BBLDO 控制	RW 8bit	0x20	0x1C
RF_AON_REG	RF AON 寄存器	RW 32bit	0x30	0x323fa20
PE_CTRL1	CPR AO GPIO PE Ctrl1	RW 32bit	0X34	0X0
PE_CTRL2	CPR AO GPIO PE Ctrl2	RW 32bit	0X38	0X0
PU_CTRL1	CPR AO GPIO PU Ctrl1	RW 4bit	0X3C	0X0
AON_VDD_SWITCH_EN	内部开关控制,0 模块不使能	RW 5bit	0X40	0X1f
AON_VDD_ISO_EN	内部断电隔离控制，1 断电	RW 5bit	0X44	0X0
LPOLDO_ADJ	LPO LDO 控制	RW 4bit	0X48	0X4
AON_LPOLDO_ADJ	retation memory ldo 控制	RW 2bit	0X4C	0X1
AON_CORERFLDO_EN	RF 数字部分 LDO 控制	RW 1bit	0X50	0X1
DCDC_CTRL0	dcdc 控制寄存器 0	RW 7bit	0X54	0X0
DCDC_CTRL1	dcdc 控制寄存器 1	RW 31bit	0X58	0x420ca83
AOCLKEN_GRCTL	AO 域时钟使能寄存器	RW 5bit	0X7C	0x12
CTL_CLK32K	32K 时钟配置寄存器	RW 10bits	0X14C	0x109
ALWAYS_ON_REG0	AO 通用器 0	RW 32bit	0X154	0x0
ALWAYS_ON_REG1	AO 通用器 1	RW 32bit	0X158	0x32000000
AO_CTRL	AO 控制寄存器	RW 1bit	0X15C	0x1
ALWAYS_ON_REG2	AO 通用器 2	RW 32bit	0X160	0x0
ALWAYS_ON_REG3	AO 通用器 3	RW 32bit	0X16C	0x0
AON_BOR_CTRL	BOR 控制器	RW 8bit	0x170	0xd8
ALWAYS_ON_REG4	AO 通用器 4	RW 3bit	0x174	0x0

4.8. CPR 寄存器描述

4.8.1. 睡眠控制寄存器描述

4.8.1.1. SLP_SRC_MASK 睡眠源屏蔽寄存器

- 地址偏移：0x0

位数	名称	方向	复位值	描述
8	SLP_MASK	RW	0x0	全局屏蔽信号，屏蔽全部睡眠源
7:0	SLP_SRC_MASK	RW	0x0	睡眠源屏蔽，对应 bit 为 1 屏蔽相应的睡眠来源

4.8.1.2. SLP_CNT_LIMIT 睡眠进入阈值寄存器

- 地址偏移：0x10

位数	名称	方向	复位值	描述
14:12	SYS_INTR_MASK_CYCLE	RW	0x3	配置睡眠计数器计数到 SLP_CONFIG_NUM 之前的多少个周期将中断屏蔽信号 sys_intr_mask 拉高。
7:0	SLP_CONFIG_NUM	RW	0x34	睡眠条件满足的情况下，26M 计数器需要计数到多少，才将睡眠满足信号传递给睡眠状态机：最小值为 0x6，最大值为 0xFFFF

4.8.1.3. SLP_ST 睡眠启动寄存器

- 地址偏移：0x14

位数	名称	方向	复位值	描述
14:12	SLP_ST_FINAL	R	0x0	只读，内部软件睡眠信号的状态，该位在 SLP_ST 置为 1 时拉高，在进入睡眠之后拉低。可以用来监测睡眠的进入。
7:0	SLP_ST	WC1	0x0	软件睡眠进入控制位： 1：启动睡眠控制器进入睡眠；经睡眠过程之后，此位自清零； 0：无影响

4.8.1.4. SLP_FSM_STATE 睡眠状态机状态寄存器

- 地址偏移：0x18

位数	名称	方向	复位值	描述
2:0	SYS_CS	R	0x0	睡眠状态机状态： 0x0：状态机处于工作状态； 0x1：状态机晶振起振阶段； 0x2：状态机主时钟关闭阶段； 0x4：状态机复位系统阶段； 0x5：状态机主时钟打开阶段； 0x6：状态机晶振关闭阶段； 0x7：状态机断电状态； 其他值：不会出现

4.8.2. 时钟分频和控制寄存器

4.8.2.1. M0_FCLK_CTL 时钟控制寄存器

- 地址偏移：0x20

注：BIT16 写 1 解屏蔽 BIT[3:0]，BIT20 写 1 解屏蔽 BIT[7:4]，BIT24 写 1 解屏蔽 BIT[11:8]，BIT28 写 1 解屏蔽 BIT[15:12]。

M0_FCLK 在 M0_FCLK_DIRECT_SW 为 0 时，新配置的分频系数在系统经过睡眠流程后才能发生作用

位数	名称	方向	复位值	描述
11:8	M0_FCLK_DIV	R	0x0	M0_FCLK 当前起作用的时钟分频系统
4	M0_FCLK_DIRECT_SW	RW	0x0	M0_FCLK 时钟分频立即作用
3:0	M0_FCLK_DIV_P	RW	0X0	M0_FCLK 当前配置的时钟分频系数

4.8.2.2. CTL_PCLK_GRCTL 时钟控制寄存器

- 地址偏移：0x24

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	CTL_PCLK_GR_UPD	WC1	0x0	CTL_PCLKGR 参数更新寄存器 1 有效，自清 0
3:0	CTL_PCLK_GR	RW	0x4	CTL_PCLK 频率为 BUS_CLK 时钟频率的 CTL_PCLK_GR/8

4.8.2.3. UART0_CLK_GRCTL 时钟 GR 控制寄存器

- 地址偏移：0x30

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	UART0_CLK_GR_UPD	WC1	0x0	uart0_clkGR 参数更新寄存器 1 有效，自清 0
3:0	UART0_CLK_GR	RW	0x8	uart0_clk 相对于 32MHz 时钟的一级分频比为:UART0_CLK_GR/8

4.8.2.4. UART0_CLK_CTL 时钟控制寄存器

- 地址偏移：0x34

位数	名称	方向	复位值	描述
31:16	UART0_CLK_MUL	RW	0x48	uart0_clk 相对于 32MHz 时钟的二级分频系数 MUL

15:0	UART0_CLK_DIV	RW	0x0271	uart0_clk 相对于 32MHz 时钟的二级分频系数 DIV

4.8.2.5. UART1_CLK_GRCTL 时钟 GR 控制寄存器

- 地址偏移：0x38

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	UART1_CLK_GR_UPD	RW	0x0	uart1_clkGR 参数更新寄存器，1 有效，自清 0
3：0	UART1_CLK_GR	RW	0x8	uart1_clk 相对于 32MHz 时钟的一级分频为:UART1_CLK_GR/8

4.8.2.6. UART1_CLK_CTL 时钟控制寄存器

- 地址偏移：0x3C

位数	名称	方向	复位值	描述
31：16	UART1_CLK_MUL	RW	0x48	uart1_clk 相对于 32MHz 时钟的二级分频系数 MUL
15：0	UART1_CLK_DIV	RW	0x0271	uart1_clk 相对于 32MHz 时钟的二级分频系数 DIV

4.8.2.7. BT_CLK_CTL 时钟控制寄存器

- 地址偏移：0x40

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	BT_CLK_EN	RW	0x0	BT_CLK 时钟使能
3：0	BT_CLK_DIV	RW	0x3	BT_CLK 时钟分频系数

4.8.2.8. BT_MODEM_CLK 时钟控制寄存器

- 地址偏移：0x48

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	BT_MODEM_CLK_EN	RW	0x0	BT_MODEM_CLK 时钟使能
3：0	BT_MODEM_CLK_DIV	RW	0x1	BT_MODEM_CLK 时钟分频系数

4.8.2.9. QDEC_CLK 时钟控制寄存器

- 地址偏移：0x4C

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	QDEC_CLK_EN	RW	0x0	QDEC_CLK 时钟使能
3：0	QDEC_CLK_DIV	RW	0x7	QDEC_CLK 时钟分频系数

4.8.2.10. SSI0_MCLK_CTL 时钟控制寄存器

- 地址偏移：0x50

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	SSI0_MCLK_EN	RW	0x1	SSI0_MCLK 时钟使能
3：0	SSI0_MCLK_DIV	RW	0x3	SSI0_MCLK 时钟分频系数

4.8.2.11. SSI1_MCLK_CTL 时钟控制寄存器

- 地址偏移：0x54

备注：BIT16 写 1 解屏蔽 BIT[3:0]，BIT20 写 1 解屏蔽 BIT[7:4]，BIT24 写 1 解屏蔽 BIT[11:8]，BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
12	SSI2_MCLK_EN	RW	0x0	SSI2_MCLK 时钟使能
11：8	SSI2_MCLK_DIV	RW	0x0	SSI2_MCLK 时钟分频系数

4	SSI1_MCLK_EN	RW	0x0	SSI1_MCLK 时钟使能
3：0	SSI1_MCLK_DIV	RW	0x3	SSI1_MCLK 时钟分频系数

4.8.2.12. TIMER0_CLK_CTL 时钟控制寄存器

- 地址偏移：0x58

位数	名称	方向	复位值	描述
30：28	TIMER0_CLKSEL	RW	0x0	选择 timer0_clk 的分频来源。 0：选择由 mclk_in(32MHz)分频得到。 1：选择由 32KHz 时钟分频得到。 3：选择 32KHz 时钟。 2、4 和 5~7：输出常 0。
27：16	NA	--	--	保留
15：8	TIMER0_CLK1_DIV	RW	0x9	控制 timer0_clk 相对于 32KHz 时钟的分频比 2*(TIMER0_CLK1_DIV+1)。
7：0	TIMER0_CLK0_DIV	RW	0x9	控制 timer0_clk 相对于 mclk_in(32MHz)时钟的分频比为 2*(TIMER0_CLK0_DIV+1)。

4.8.2.13. TIMER1_CLK_CTL 时钟控制寄存器

- 地址偏移：0x5C

位数	名称	方向	复位值	描述
30：28	TIMER1_CLKSEL	RW	0x0	选择 timer1_clk 的分频来源。 0：选择由 mclk_in(32MHz)分频得到。 1：选择由 32KHz 时钟分频得到。 3：选择 32KHz 时钟。 2、4 和 5~7：输出常 0。
27：16	NA	--	--	保留
15：8	TIMER1_CLK1_DIV	RW	0x9	控制 timer1_clk 相对于 32KHz 时钟的分频比 2*(TIMER1_CLK1_DIV+1)。
7：0	TIMER1_CLK0_DIV	RW	0x9	控制 timer1_clk 相对于 mclk_in(32MHz)时钟的分频比为 2*(TIMER1_CLK0_DIV+1)。

4.8.2.14. TIMER2_CLK_CTL 时钟控制寄存器

- 地址偏移：0x60

位数	名称	方向	复位值	描述
30：28	TIMER2_CLKSEL	RW	0x0	选择 timer2_clk 的分频来源。 0:选择由 mclk_in(32MHz)分频得到。 1：选择由 32KHz 时钟分频得到。 3：选择 32KHz 时钟。 2、4 和 5~7：输出常 0。
27：16	NA	--	--	保留
15：8	TIMER2_CLK1_DIV	RW	0x9	控制 timer2_clk 相对于 32KHz 时钟的分频比 2*(TIMER2_CLK1_DIV+1)。
7：0	TIMER2_CLK0_DIV	RW	0x9	控制 timer2_clk 相对于 mclk_in(32MHz)时钟的分频比为 2*(TIMER2_CLK0_DIV+1)。

4.8.2.15. TIMER3_CLK_CTL 时钟控制寄存器

- 地址偏移：0x64

位数	名称	方向	复位值	描述
30：28	TIMER3_CLKSEL	RW	0x0	选择 timer3_clk 的分频来源。 0:选择由 mclk_in(32MHz)分频得到。 1：选择由 32KHz 时钟分频得到。 3：选择 32KHz 时钟。 2、4 和 5~7：输出常 0。
27：16	NA	--	--	保留
15：8	TIMER3_CLK1_DIV	RW	0x9	控制 timer3_clk 相对于 32KHz 时钟的分频比 2*(TIMER3_CLK1_DIV+1)。
7：0	TIMER3_CLK0_DIV	RW	0x9	控制 timer3_clk 相对于 mclk_in(32MHz)时钟的分频比为 2*(TIMER3_CLK0_DIV+1)。

4.8.2.16. PWM_CLK_CTL 时钟控制寄存器

- 地址偏移：0x68

位数	名称	方向	复位值	描述
31	PWM_CLK_EN	RW	0x0	PWM_CLK 时钟使能。 1：使能时钟
30：28	PWM_CLKSEL	RW	0x0	选择 pwm_clk 的分频来源。 0：选择由 mclk_in(32MHz)分频得到。 1：选择由 32KHz 时钟分频得到。 3：选择 32KHz 时钟。 2、4/5~7：输出常 0。
27：16	NA	--	--	保留。
15：8	PWM_CLK1_DIV	RW	0x9	控制 pwm_clk 相对于 32KHz 时钟的分频比为 2*(PWM_CLK1_DIV+1)。
7：0	PWM_CLK0_DIV	RW	0x9	控制 pwm_clk 相对 mclk_in(32MHz)时钟的分频比为 2*(PWM_CLK0_DIV+1)

4.8.2.17. AHBCLKEN_GRCTL 总线时钟使能寄存器

- 地址偏移：0x6C

注：高 16BITS 写 1 解屏蔽低 16BITS。

位数	名称	方向	复位值	描述
4	ROM_HCLK_EN	RW	0x1	ROM_HCLK 时钟使能。 1：使能时钟。
3	M4_RAM_HCLK_EN	RW	0x1	M4_RAM_HCLK 时钟使能。 1：使能时钟。
2	AES_HCLK_EN	RW	0x1	AES_HCLK 时钟使能。 1：使能时钟。
1	SHRAM0_HCLK_EN	RW	0x1	SHRAM0_HCLK 时钟使能。 1：使能时钟。
0	DMAS_HCLK_EN	RW	0x1	DMAS_HCLK 时钟使能。 1：使能时钟。

4.8.2.18. CTLAPBCLKEN_GRCTL 总线时钟使能寄存器

- 地址偏移：0x70

注：高 16BITS 写 1 解屏蔽低 16BITS。

位数	名称	方向	复位值	描述
15	SSI2_PCLK_EN	RW	0x0	SSI2_PCLK 时钟使能。 1：使能时钟。
14	NA	--	--	--
13	GPADC_PCLK_EN	RW	0x0	GPADC_PCLK 时钟使能。 1：使能时钟。
12	PWM_PCLK_EN	RW	0x0	PWM_PCLK 时钟使能。 1：使能时钟。
11	I2C_PCLK_EN	RW	0x0	I2C_PCLK 时钟使能。 1：使能时钟。
10	BT_PCLK_EN	RW	0x0	BT_PCLK 时钟使能。 1：使能时钟。
9	SSI1_PCLK_EN	RW	0x0	SSI1_PCLK 时钟使能。 1：使能时钟。
8	SSI0_PCLK_EN	RW	0x1	SSI0_PCLK 时钟使能。 1：使能时钟。
7	NA	--	--	--
6	QDEC_PCLK_EN	RW	0x0	QDEC_PCLK 时钟使能。 1：使能时钟。
5	UART1_PCLK_EN	RW	0x0	UART1_PCLK 时钟使能 1：使能时钟
4	UART0_PCLK_EN	RW	0x1	UART0_PCLK 时钟使能。 1：使能时钟。
3	TIMER_PCLK_EN	RW	0x0	TIMER_PCLK 时钟使能。 1：使能时钟。
2	GPIO_PCLK_EN	RW	0x1	GPIO_PCLK 时钟使能。 1：使能时钟。
1	RTC_PCLK_EN	RW	0x0	RTC_PCLK 时钟使能。 1：使能时钟。

0	WDT_PCLK_EN	RW	0x0	WDT_PCLK 时钟使能。 1：使能时钟。

4.8.2.19. OTHERCLKEN_GRCTL 其他时钟使能寄存器

- 地址偏移：0x74

注：高 16BITS 写 1 解屏蔽低 16BITS。

位数	名称	方向	复位值	描述
3	NA	--	--	--
2	BT32K_CLK_EN	RW	0x0	BT32K_CLK 时钟使能。 1：使能时钟。
1	NA	--	--	保留。
0	GPIO_CLK_EN	RW	0x0	GPIO_CLK 时钟使能。 1：使能时钟。

4.8.2.20. I2C_CLK_CTL 时钟控制寄存器

- 地址偏移：0x78

注:BIT16 写 1 解屏蔽 BIT[3:0],BIT20 写 1 解屏蔽 BIT[7:4],BIT24 写 1 解屏蔽 BIT[11:8],BIT28 写 1 解屏蔽 BIT[15:12]。

位数	名称	方向	复位值	描述
4	I2C_CLK_EN	RW	0x0	I2C_CLK 时钟使能。 1：使能时钟。
3：0	I2C_CLK_DIV	RW	0x1	I2C_CLK 时钟分频系数。

4.8.3. CPR 中断相关寄存器

4.8.3.1. CPR 中断状态寄存器

- 地址偏移：0xA0

位数	名称	方向	复位值	描述
10	CDBGPWRUPREQ_INTR	RW	0x0	CDBGPWRUPREQ 唤醒中断
9：0	NA	--	--	保留

4.8.3.2. CPR 中断使能寄存器

- 地址偏移：0xA4

位数	名称	方向	复位值	描述
10	CDBGPWRUPREQ_INTR_EN	RW	0x0	CDBGPWRUPREQ 唤醒中断使能
9：0	NA	--	--	保留

4.8.3.3. CPR 中断原始状态寄存器

- 地址偏移：0xA8

位数	名称	方向	复位值	描述
10	CDBGPWRUPREQ_INTR_RAW	R	0x0	CDBGPWRUPREQ 唤醒中断原始状态
9：0	NA	--	--	保留

4.8.4. GPIO 复用控制寄存器

4.8.4.1. GPIO_FUN_SEL0GPIO 复用控制寄存器 0

- 地址偏移：0xC0

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL3	RW	0x0	GPIO3 的功能选择： 0:GPIO_Dx（x 为 GPIO 序号） 1:UART0_TX 2:UART0_RX 3:UART0_CTS 4:UART0_RTS 5:I2C_SCL 6:I2C_SDA 7:UART1_RX 8:UART1_TX 9:SIM_IO

				10:SIM_RST 11:SIM_CLK_OUT 12:PWM0 13:PWM1 14:SSI1_CLK 15:SSI1_SSN 16:SSI1_RX 17:SSI1_TX 其它：GPIO_Dx 说明：当输入有多个源的时候，编号最小的生效。例如：GPIO0 和 GPIO1 同时选择为 UART0_RX，则 UART0_RX 实际上由 GPIO0 输入；输出则允许多个端口输出同一个信号
20:16	GPIO_FUN_SEL2	RW	0x0	GPIO2 的功能选择：描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL1	RW	0x0	GPIO1 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL0	RW	0x0	GPIO0 的功能选择：描述同 GPIO_FUN_SEL3

4.8.4.2. GPIO_FUN_SEL1 GPIO 复用控制寄存器 1

- 地址偏移：0xC4

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL7	RW	0x0	GPIO7 的功能选择：描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL6	RW	0x0	GPIO6 的功能选择：描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL5	RW	0x0	GPIO5 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL4	RW	0x0	GPIO4 的功能选择：描述同 GPIO_FUN_SEL3

4.8.4.3. GPIO_FUN_SEL2 GPIO 复用控制寄存器 2

- 地址偏移：0xC8

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL11	RW	0x0	GPIO11 的功能选择:描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL10	RW	0x0	GPIO10 的功能选择:描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL9	RW	0x0	GPIO9 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL8	RW	0x0	GPIO8 的功能选择：描述同 GPIO_FUN_SEL3

4.8.4.4. GPIO_FUN_SEL3 GPIO 复用控制寄存器 3

- 地址偏移：0xCC

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL15	RW	0x0	GPIO14 的功能选择:描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL14	RW	0x0	GPIO13 的功能选择:描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL13	RW	0x0	GPIO13 的功能选择:描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL12	RW	0x0	GPIO12 的功能选择:描述同 GPIO_FUN_SEL3

4.8.4.5. GPIO_FUN_SEL4 GPIO 复用控制寄存器 4

- 地址偏移：0xD0

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL19	RW	0x0	GPIO19 的功能选择:描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL18	RW	0x0	GPIO18 的功能选择:描述同 GPIO_FUN_SEL3

12:8	GPIO_FUN_SEL17	RW	0x0	GPIO17 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL16	RW	0x0	GPIO16 的功能选择：描述同 GPIO_FUN_SEL3

4.8.4.6. GPIO_FUN_SEL5 GPIO 复用控制寄存器 5

- 地址偏移：0xD4

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL23	RW	0x0	GPIO23 的功能选择：描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL22	RW	0x0	GPIO22 的功能选择：描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL21	RW	0x0	GPIO21 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL20	RW	0x0	GPIO20 的功能选择：描述同 GPIO_FUN_SEL3

4.8.4.7. GPIO_FUN_SEL6 GPIO 复用控制寄存器 6

- 地址偏移：0xD8

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL27	RW	0x0	GPIO27 的功能选择：描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL26	RW	0x0	GPIO26 的功能选择：描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL25	RW	0x0	GPIO25 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL24	RW	0x0	GPIO24 的功能选择：描述同 GPIO_FUN_SEL3

4.8.4.8. GPIO_FUN_SEL7 GPIO 复用控制寄存器 7

- 地址偏移：0xDC

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL31	RW	0x0	GPIO31 的功能选择：描述同 GPIO_FUN_SEL3
20:16	GPIO_FUN_SEL30	RW	0x0	GPIO30 的功能选择：描述同 GPIO_FUN_SEL3
12:8	GPIO_FUN_SEL29	RW	0x0	GPIO29 的功能选择：描述同 GPIO_FUN_SEL3
4:0	GPIO_FUN_SEL28	RW	0x0	GPIO28 的功能选择：描述同 GPIO_FUN_SEL3

4.8.5. 复位控制寄存器

4.8.5.1. RSTCTL_CTLAPB_SW 模块软复位寄存器

- 地址偏移：0x100

位数	名称	方向	复位值	描述
12	GPADC_RSTN	RW	0x1	GPADC 模块软复位： 0：复位； 1：不复位
11	PWM_RSTN	RW	0x1	PWM 模块软复位： 0：复位； 1：不复位
10	QDEC_RSTN	RW	0x1	QDEC 模块软复位： 0：复位； 1：不复位
9	NA	--	--	--
8	CTL_APB_RSTN	RW	0x1	CTL_APB 模块软复位： 0：复位； 1：不复位
7	TIMER_PRESETN	RW	0x1	TIMER 模块软复位： 0：复位； 1：不复位

6	TIMER3_RSTN	RW	0x1	TIMER3 模块软复位： 0：复位； 1：不复位；
5	TIMER2_RSTN	RW	0x1	TIMER2 模块软复位： 0：复位； 1：不复位
4	TIMER1_RSTN	RW	0x1	TIMER1 模块软复位： 0：复位； 1：不复位
3	TIMER0_RSTN	RW	0x1	TIMER0 模块软复位： 0：复位； 1：不复位
2	WDT_RSTN	RW	0x1	WDT 模块软复位： 0：复位； 1：不复位
1	GPIO_RSTN	RW	0x1	GPIO 模块软复位： 0：复位； 1：不复位
0	NA	--	--	保留

4.8.5.2. RSTCTL_SUBRST_SW 模块软复位寄存器

- 地址偏移：0x104

位数	名称	方向	复位值	描述
11	UART2_RSTN	RW	0x1	UART2 模块软复位： 0：复位； 1：不复位
10	SSI2_RSTN	RW	0x1	SSI2 模块软复位： 0：复位； 1：不复位
9:8	NA	--	--	--
7	SHRAM0_RSTN	RW	0x1	SHRAM0 模块软复位： 0：复位； 1：不复位
6	I2C_RSTN	RW	0x1	I2C 模块软复位： 0：复位； 1：不复位；
5	BT_RSTN	RW	0x1	BT 模块软复位： 0：复位； 1：不复位

4	DMAS_RSTN	RW	0x1	DMA 模块软复位： 0：复位； 1：不复位
3	SSI1_RSTN	RW	0x1	SSI1 模块软复位： 0：复位； 1：不复位
2	SSI0_RSTN	RW	0x1	SSI0 模块软复位： 0：复位； 1：不复位
1	UART1_RSTN	RW	0x1	UART1 模块软复位： 0：复位； 1：不复位
0	UART0_RSTN	RW	0x1	UART0 模块软复位： 0：复位； 1：不复位

4.8.5.3. RSTCTL_M0RST_SW 相关软复位寄存器

- 地址偏移：0x108

位数	名称	方向	复位值	描述
4	M0_RAM_RSTN	RW	0x1	M0_RAM 模块软复位： 0：复位； 1：不复位
3	ROM_RSTN	RW	0x1	ROM 模块软复位： 0：复位； 1：不复位
2	DBG_PORESETN	WC0	0x1	DEBUG 模块软复位，持续 5 个 M0_FCLK 周期，自动恢复为 1： 0：复位； 1：不复位
1	M0_PORESETN	WC0	0x1	M0 上电复位软复位，持续 5 个 M0_FCLK 周期，自动恢复为 1： 0：复位； 1：不复位
0	M0_SYSRESETN	WC0	0x1	M0 系统软件复位，持续 5 个 M0_FCLK 周期，自动恢复为 1： 0：复位； 1：不复位

4.8.5.4. RSTCTL_M0RST_MASK 复位屏蔽寄存器

- 地址偏移：0x10C

位数	名称	方向	复位值	描述
4	M0_LOCKUP_MASK	RW	0x1	屏蔽 M0 的 LOCKUP 信号引起的复位 0：不屏蔽； 1：屏蔽
3	M0_SYSRESETREQ_MASK	RW	0x1	屏蔽 M0 的 SYSRESETREQ 信号引起的复位 0：不屏蔽； 1：屏蔽
2	DBG_PORESETN_MASK	RW	0x1	屏蔽 M0 的 DEBUG 模块软复位 0：不屏蔽； 1：屏蔽
1	M0_PORESETN_MASK	RW	0x1	屏蔽 M0 的上电软复位 0：不屏蔽； 1：屏蔽
0	M0_SYSRESETN_MASK	RW	0x1	屏蔽 M0 的系统软复位 0：不屏蔽； 1：屏蔽

4.8.5.5. RSTCTL_LOCKUP_STATE 死锁状态指示寄存器

- 地址偏移：0x110

位数	名称	方向	复位值	描述
0	M0_LOCKUP	R	0x0	M0 的 LOCKUP 状态指示符 0：M0 处于正常状态； 1：M0 处于 LOCKUP 状态

4.8.5.6. RSTCTL_WDTRST_MASK 复位屏蔽寄存器

- 地址偏移：0x114

位数	名称	方向	复位值	描述
1	WDT_M0_RSTN_MASK	R	0x1	屏蔽 WDT 引起的 M0 软复位 0：不屏蔽； 1：屏蔽
0	WDT_SYS_RSTN_MASK		0x1	屏蔽 WDT 引起的系统软复位 0：不屏蔽； 1：屏蔽

4.8.6. 低功耗控制寄存器

4.8.6.1. LP_CTL 低功耗使能寄存器

- 地址偏移：0x118

位数	名称	方向	复位值	描述
12	wdt_tclk_en	RW	0x0	WDT tclk 时钟使能 0：不使能 1：使能
11	wdt_pclk_sel	RW	0x0	wdt_pclk 时钟选择 1：选择 32k 时钟 0：选择 ctl_pclk 睡眠时，可以切换为 32k 时钟，读写 wdt 寄存器时，需要切换为 ctl_pclk 的时钟。
10	DAP_CLK_LP_EN	RW	0x0	dap_clk 的自动关电使能 0：不使能； 1：使能
9	NA	--	--	保留
8	SHRAM0_ICM_LP_EN	RW	0x0	SHRAM0 的 ICM 单元低功耗使能 0：不使能； 1：使能
7	CTLAPB_ICM_LP_EN	RW	0x0	CTL_APB 模块的 ICM 单元低功耗使能 0：不使能； 1：使能
6	M0_BUS_LP_EN	RW	0x0	内核 Mx 的 DW_AHB 单元低功耗使能 0：不使能； 1：使能
5	DMAS_BUS_LP_EN	RW	0x0	M0 系统软件复位，持续 5 个 M0_FCLK 周期，自动恢复为 1： 0：复位； 1：不复位

4	MEM_LP_EN	RW	0x0	DMA 的 DW_AHB 单元低功耗使能 0：不使能 1：使能
3	M0_HCLK_LP_EN	RW	0x0	MEMORY 的低功耗使能 0：不使能; 1：使能
2	TIMER_SYSCLK_SEL	RW	0x0	TIMER 睡眠时 TIMER_PCLK 时钟选择 1：睡眠时切换到 32K 时钟; 0：睡眠时不切换 32K 时钟
1	UART0_CLK_OFF_PROTECT_EN	RW	0x0	UART0 时钟保护功能使能 1：使能; 0：不使能
0	UART1_CLK_OFF_PROTECT_EN	RW	0x0	UART1 时钟保护功能使能 1：使能; 0：不使能

4.8.7. CTL 总线配置寄存器

4.8.7.1. CTL_SHRAM_ICM_PRIORITY 优先级配置寄存器

- 地址偏移：0x120

位数	名称	方向	复位值	描述
5:2	NA	--	--	保留
1:0	SHRAM0_ICM_PRIORITY	RW	0x0	SHRAM0 总线 ICM 的 Master 访问优先级配置

4.8.7.2. CTL_CTLAPB_ICM_PRIORITY 优先级配置寄存器

- 地址偏移：0x124

位数	名称	方向	复位值	描述
1:0	CTLAPB_ICM_PRIORITY	RW	0x0	CTLAPB 总线 ICM 的 Master 访问优先级配置

4.8.8. CTL 管脚配置相关寄存器

4.8.8.1. CTL_MUXCTL1 控制寄存器 1

- 地址偏移：0x128

位数	名称	方向	复位值	描述
31:30	MUXCTL[31:30]	RW	0x0	PAD GPIO15 的复用设置： 00：连接到 GPIO 的 gpio_d[15]； 01：NA； 10：连接到蓝牙模块的 gc1； 11：连接到 test_pin[7]
29:28	MUXCTL[29:28]	RW	0x0	PAD GPIO14 的复用设置： 00：连接到 GPIO 的 gpio_d[14]； 01：连接到 OSC_EN； 10：连接到蓝牙模块的 gc0； 11：连接到 test_pin[6]
27:26	MUXCTL[27:26]	RW	0x0	PAD GPIO13 的复用设置： 00：连接到 GPIO 的 gpio_d[13]； 01：连接到 QDEC 的 qdbz； 10：连接到蓝牙模块的 bt_bdata1； 11：连接到 test_pin[5]
25:24	MUXCTL[25:24]	RW	0x0	PAD GPIO12 的复用设置： 00：连接到 GPIO 的 gpio_d[12]； 01：连接到 QDEC 的 qdaz； 10：连接到蓝牙模块的 txen； 11：连接到 test_pin[4]
23:22	MUXCTL[23:22]	RW	0x0	PAD GPIO11 的复用设置： 00：连接到 GPIO 的 gpio_d[11]； 01：连接到 QDEC 的 qdby； 10：连接到蓝牙模块的 txen； 11：连接到 test_pin[3]
21:20	MUXCTL[21:20]	RW	0x0	PAD GPIO10 的复用设置： 00：连接到 GPIO 的 gpio_d[10]； 01：连接到 QDEC 的 qday； 10：连接到蓝牙模块的 bt_bddata； 11：连接到 test_pin[2]
19:18	MUXCTL[19:18]	RW	0x0	PAD GPIO9 的复用设置： 00：连接到 GPIO 的 gpio_d[9]； 01：连接到 QDEC 的 qdbx； 10：连接到蓝牙模块的 bt_bdclk； 11：连接到 test_pin[1]
17:16	MUXCTL[17:16]	RW	0x0	PAD GPIO8 的复用设置： 00：连接到 GPIO 的 gpio_d[8]； 01：连接到 QDEC 的 qdax； 10：连接到蓝牙模块的 bt_bnden； 11：连接到 test_pin[0]

15:14	MUXCTL[15:14]	RW	0x0	PAD GPIO7 的复用设置： 00：连接到 GPIO 的 gpio_d[7]； 01：连接到 KBS 的 kbs_mko[7]； 10：连接到 iqclk； 11：NA
13:12	MUXCTL[13:12]	RW	0x0	PAD GPIO6 的复用设置： 00：连接到 GPIO 的 gpio_d[6]； 01：连接到 KBS 的 kbs_mko[6]； 10：连接到蓝牙模块的 iq[9]； 11：NA
11:10	MUXCTL[11:10]	RW	0x0	PAD GPIO5 的复用设置： 00：连接到 GPIO 的 gpio_d[5]； 01：连接到 KBS 的 kbs_mko[5]； 10：连接到蓝牙模块的 pta_pri； 11：NA
9:8	MUXCTL[9:8]	RW	0x0	PAD GPIO4 的复用设置： 00：连接到 GPIO 的 gpio_d[4]； 01：连接到 KBS 的 kbs_mko[4]； 10：连接到蓝牙模块的 pta_req； 11：NA
7:6	MUXCTL[7:6]	RW	0x0	PAD GPIO3 的复用设置： 00：连接到 GPIO 的 gpio_d[3]； 01：连接到 KBS 的 kbs_mko[3]； 10：连接到蓝牙模块的 pta_grant； 11：NA
5:4	MUXCTL[5:4]	RW	0x0	PAD GPIO2 的复用设置： 00：连接到 GPIO 的 gpio_d[2]； 01：连接到 KBS 的 kbs_mko[2]； 其它：NA
3:2	MUXCTL[3:2]	RW	0x0	PAD GPIO1 的复用设置： 00：连接到 GPIO 的 gpio_d[1]； 01：连接到 KBS 的 kbs_mko[1]； 其它：NA
1:0	MUXCTL[1:0]	RW	0x0	PAD GPIO0 的复用设置： 00：连接到 GPIO 的 gpio_d[0]； 01：连接到 KBS 的 kbs_mko[0]； 其它：NA

4.7.8.2 CTL_MUXCTL2 控制寄存器 2

- 地址偏移：0x12c

位数	名称	方向	复位值	描述
31:30	MUXCTL[63:62]	RW	0x0	PAD SWD 的复用设置： 00：连接到 SWI 的 SWD； 01：连接到 GPIO 的 gpio_d[31]； 其它：NA
29:28	MUXCTL[61:60]	RW	0x0	PAD SWCK 的复用设置： 00：连接到 SWI 的 SWCK； 01：连接到 GPIO 的 gpio_d[30]； 其它：NA
27:26	MUXCTL[59:58]	RW	0x0	PAD BOOTCTL 的复用设置： 00：连接到 BOOT_CTL； 01：连接到 GPIO 的 gpio_d[29]； 10：连接到蓝牙模块的 iq[0]； 11：NA
25:24	MUXCTL[57:56]	RW	0x0	PAD GPIO28 的复用设置： 00：连接到 GPIO 的 gpio_d[28]； 其它：NA
23:22	MUXCTL[55:54]	RW	0x0	PAD GPIO27 的复用设置： 00：连接到 GPIO 的 gpio_d[27]； 其它：NA
21:20	MUXCTL[53:52]	RW	0x0	PAD GPIO26 的复用设置： 00：连接到 GPIO 的 gpio_d[26]； 01：NA； 10：连接到蓝牙模块的 gc4； 11：NA
19:18	MUXCTL[51:50]	RW	0x0	PAD GPIO25 的复用设置： 00：连接到 GPIO 的 gpio_d[25]； 01：连接到 KBS 的 kbs_mki[7]； 10：连接到蓝牙模块的 iq[8]； 11：NA

17:16	MUXCTL[49:48]	RW	0x0	PAD GPIO24 的复用设置： 00：连接到 GPIO 的 gpio_d[24]； 01：连接到 KBS 的 kbs_mki[6]； 10：连接到蓝牙模块的 iq[7]； 11：NA
15:14	MUXCTL[47:46]	RW	0x0	PAD GPIO23 的复用设置： 00：连接到 GPIO 的 gpio_d[23]； 01：连接到 KBS 的 kbs_mki[5]； 10：连接到蓝牙模块的 iq[6]； 11：NA
13:12	MUXCTL[45:44]	RW	0x0	PAD GPIO22 的复用设置： 00：连接到 GPIO 的 gpio_d[22]； 01：连接到 KBS 的 kbs_mki[4]； 10：连接到蓝牙模块的 iq[5]； 11：NA
11:10	MUXCTL[43:42]	RW	0x0	PAD GPIO21 的复用设置： 00：连接到 GPIO 的 gpio_d[21]； 01：连接到 KBS 的 kbs_mki[3]； 10：连接到蓝牙模块的 iq[4]； 11：NA
9:8	MUXCTL[41:40]	RW	0x0	PAD GPIO20 的复用设置： 00：连接到 GPIO 的 gpio_d[20]； 01：连接到 KBS 的 kbs_mki[2]； 10：连接到蓝牙模块的 iq[3]； 11：NA
7:6	MUXCTL[39:38]	RW	0x0	PAD GPIO19 的复用设置： 00：连接到 GPIO 的 gpio_d[19]； 01：连接到 KBS 的 kbs_mki[1]； 10：连接到蓝牙模块的 iq[2]； 11：NA
5:4	MUXCTL[37:36]	RW	0x0	PAD GPIO18 的复用设置： 00：连接到 GPIO 的 gpio_d[18]； 01：连接到 KBS 的 kbs_mki[0]；

				10：连接到蓝牙模块的 iq[1]； 11：NA
3:2	MUXCTL[35:34]	RW	0x0	PAD GPIO17 的复用设置： 00：连接到 GPIO 的 gpio_d[17]； 01：NA； 10：连接到蓝牙模块的 gc3； 11：连接到 test_pin[9]
1:0	MUXCTL[33:32]	RW	0x0	PAD GPIO16 的复用设置： 00：连接到 GPIO 的 gpio_d[16]； 01：NA； 10：连接到蓝牙模块的 gc2； 11：连接到 test_pin[8]

4.8.9. RF_REG3 通用寄存器 3

- 地址偏移：0x130

位数	名称	方向	复位值	描述
31:8	cossin_freq_in	RW	0x0	正弦波频率调整
6	cossin_unsigned	RW	0x0	1：正弦波输出为原码输出 0：正弦波输出为补码输出
5	tx_cossin_en	RW	0x0	正弦波发生器发射选择 1：发射 DAC 数据源为正弦波发生器 0：发射 DAC 数据源为基带
4	cossin_enable	RW	0x0	正弦波发生器使能，1 使能
2:1	mclk_src_sel	RW	0x0	时钟源选择 0:32M osc 1：BBPLL 2：BBPLL 分频 3： 16M RC
0	BB_CBUS_MODE	RW	0x0	RF 寄存器控制： 1：SPI 模式 0：APB 模式

4.8.10. RF_REG4 通用寄存器 4

- 地址偏移：0x134

位数	名称	方向	复位值	描述
5	clkout12m_en	RW	0x0	12M 输出片外时钟使能，1：使能
4	bbpll_clk_en	RW	0x0	BBPL 时钟使能
3:0	bbpll_clk_div	RW	0x0	BBPL 时钟分频系数

4.8.11. REMAP_CTRL 指示寄存器

- 地址偏移：0x13C

位数	名称	方向	复位值	描述
31:12	REMAP_BASE	RW	0x10000	总线 REMAP 的高位地址配置寄存器
11:1	NA	--	--	保留
0	REMAP	RW	0x0	总线 REMAP 模式选择 0：BOOT Mode 1：Normal Mode

4.8.12. BOOTCTL 指示寄存器

- 地址偏移：0x140

位数	名称	方向	复位值	描述
0	BOOTCTL	R	0x0	BOOTCTL 管脚的当前值 0：执行从 UART0 的 BOOT 下载流程； 1：执行从 SSI0 的 BOOT 下载流程

4.8.13. M0 配置相关寄存器

4.8.13.1. M0_STCALIB 配置寄存器

- 地址偏移：0x144

位数	名称	方向	复位值	描述
25：0	M0_STCALIB	RW	0x000013f	systemtick 控制寄存器。 m0_stcalib[25]：是否使用参考时钟 M0_STCLK： 0：使用 1：不使用 m0_stcalib[24]：10ms 是否准确： 0：准确 1：不准确 m0_stcalib[23：0]：10ms 的周期数

4.8.13.2. M0_IRQLATENCY 延迟寄存器

- 地址偏移：0x148

位数	名称	方向	复位值	描述
7：0	M0_IRQLATENCY	RW	0x0	M0 的 IRQ 响应延迟周期数，0 表示最快响应

4.8.14. SSI_CTRL 控制寄存器

- 地址偏移：0x160

位数	名称	方向	复位值	描述
5	SSI1_MASTER_EN	RW	0x0	SS1 的主从配置： 0：Slave； 1：Master
4	SSI1_PROTOCOL	RW	0x1	SSI1 支持协议选择： 0：SSP； 1：SPI
1	SSI2_PROTOCOL	RW	0x1	SSI2 支持协议选择： 0：SSP； 1：SPI
0	SSI0_PROTOCOL	RW	0x1	SSI0 支持协议选择： 0：SSP； 1：SPI

4.8.15. 测试控制寄存器

4.8.15.1. TESTCTRL0 测试控制寄存器 0

- 地址偏移：0x170

位数	名称	方向	复位值	描述
7：0	TESTCTRL0	RW	0x0	控制寄存器 0

4.8.15.2. TESTCTRL1 测试控制寄存器 1

- 地址偏移：0x174

位数	名称	方向	复位值	描述
7：0	TESTCTRL1	RW	0x0	控制寄存器 1

4.8.15.3. TESTCTRL2 测试控制寄存器 2

- 地址偏移：0x178

位数	名称	方向	复位值	描述
7：0	TESTCTRL2	RW	0x0	控制寄存器 2

4.8.15.4. TESTCTRL3 测试控制寄存器 3

- 地址偏移：0x17C

位数	名称	方向	复位值	描述
7：0	TESTCTRL3	RW	0x0	控制寄存器 3

4.8.15.5. TESTCTRL4 测试控制寄存器 4

- 地址偏移：0x180

位数	名称	方向	复位值	描述
7：0	TESTCTRL4	RW	0x0	控制寄存器 4

4.8.15.6. TESTCTRL5 测试控制寄存器 5

- 地址偏移：0x184

位数	名称	方向	复位值	描述
7：0	TESTCTRL5	RW	0x0	控制寄存器 5

4.8.15.7. TESTCTRL6 测试控制寄存器 6

- 地址偏移：0x188

位数	名称	方向	复位值	描述
7：0	TESTCTRL6	RW	0x0	控制寄存器 6

4.8.15.8. TESTCTRL7 测试控制寄存器 7

- 地址偏移：0x18C

位数	名称	方向	复位值	描述
7：0	TESTCTRL7	RW	0x0	控制寄存器 7

4.8.15.9. TESTCTRL8 测试控制寄存器 8

- 地址偏移：0x190

位数	名称	方向	复位值	描述
7：0	TESTCTRL8	RW	0x0	控制寄存器 8

4.8.15.10. TESTCTRL9 测试控制寄存器 9

- 地址偏移：0x194

位数	名称	方向	复位值	描述
7：0	TESTCTRL9	RW	0x0	控制寄存器 9

4.8.16. 其它控制寄存器

4.8.16.1. CTL_MUXCTL3 控制寄存器 3

- 地址偏移：0x19C

位数	名称	方向	复位值	描述
31：16	NA	--	--	保留
15:14	MUXCTL[79:78]	RW	0x0	gpio[39]；
13；12	MUXCTL[77:76]	RW	0x0	gpio[38]；
11:10	MUXCTL[75:74]	RW	0x0	gpio[37]；
9:8	MUXCTL[73:72]	RW	0x0	gpio[36]；
7:6	MUXCTL[71:70]	RW	0x0	PADSSITX 的复用设置： 00：连接到 SSI0 的 ssi0_tx； 01：连接到 GPIO 的 gpio_d[35]； 其它：NA
5:4	MUXCTL[69:68]	RW	0x0	PADSSIRX 的复用设置： 00：连接到 SSI0 的 ssi0_rx； 01：连接到 GPIO 的 gpio_d[34]； 其它：NA
3:2	MUXCTL[67:66]	RW	0x0	PADSSISSN 的复用设置： 00：连接到 SSI0 的 ssi0_ssn； 01：连接到 GPIO 的 gpio_d[33]； 其它：NA
1:0	MUXCTL[65:64]	RW	0x0	PADSSICLK 的复用设置： 00：连接到 SSI0 的 ssi0_clk； 01：连接到 GPIO 的 gpio_d[32]； 其它：NA

4.8.16.2. USB、CAN 控制寄存器

- 地址偏移：0x1a0

位数	名称	方向	复位值	描述
28	fmc_rdata_reverse	RW	0x0	fmc 读数据翻转
27	can_int_in	RW	0x0	can int 输入
26	can_test	RW	0x0	can test mode
25	can_nint_en_bypass	RW	0x0	can_nint_en bypass
24	can_lp_en	RW	0x0	can 接口低功耗使能
23	NA	--	--	保留
22:21	usbphy_usb_ibc	RW	0x0	usb_phy ibc 控制
20	usbphy_rxrcv_inv	RW	0x0	usbphy RXRCV 反向，1 反向

19	usbphy_rxdpm_inv	RW	0x0	usbphy RXDP，RXDM 反向，1 反向
18	usbphy_txdat_inv	RW	0x0	usbphy TX_DAT 反向，1 反向
17	usb_sel	RW	0x0	不使用
16:0	usb_rwreg	RW	0x0	usb controller 通用控制

4.8.17. RF_REG0 寄存器

- 地址偏移：0x1b0

位数	名称	方向	复位值	描述
31:30	NA	--	--	保留
29	rc16m_disalbe	RW	0x0	1：强制关闭 RC 16M 时钟，0：无效
28	boot_done	RW	0x0	boot done use for software indicate after boot
27:20	rc16m_rtune	RW	0x3F	rc 16m config
19	NA	--	--	--
18:17	rc16m_en_soft	RW	0x1	rc 16m config
16	rf24g_dac_sample_edge	RW	0x0	rc 16m 软件启用，“或”与硬件 oscen
15	rf24g_hwagc_en	RW	0x0	2.4g dac 采样边缘，1:负边缘，0:正边缘
14	rom_ls	RW	0x0	rf_reg0[14] ? 1'b0 : rf24g_rx_on;
13	rom_pd	RW	0x0	rom 浅睡, 1 enable
12	usbphy_debug	RW	0x0	rom 电源关闭, 1 enable
11	sel_24g_agc	RW	0x0	不使用
10	sel_24g_hwpin	RW	0x0	1: 射频 agc 采用 2.4g 信号
9	sel_24g_iq BB_BT_AGC_MODE	RW	0x0	1:rf iq 使用 2.4g 信号 1:rf 硬件引脚使用 2.4g 信号
8	BB_WFRX_MODE_SEL	RW	0x0	NA
7	BB_WB_SEL	RW	0x0	NA
6	BB_PLL2_480M_EN	RW	0x0	NA
5	BB_PLL1_64M_EN	RW	0x0	NA
4	AES_RET1N	RW	0x1	NA
1	USB_RET1N	RW	0x1	AES 内部 memory ret1n，0 有效
0	rc16m_disalbe	RW	0x0	USB 内部 memory ret1n，0 有效

4.8.18. RF_REG1 寄存器

- 地址偏移：0x1b4

位数	名称	方向	复位值	描述
31:29	NA	--	--	保留
28	usb_utmi_clk_en	RW	0x0	usb 的 utmi 时钟时钟，1 使能
27:24	usb_utmi_clk_div	RW	0x3F	usb 的 utmi 时钟分频比，分频值=寄存器值+1
23	adcclk_inv_en	RW	0x1	audio ADC 模拟时钟反向：1：反向
22	cdc_rstn_reg	RW	0x1	audio ADC 软复位，0：复位有效
21	rf24g_rstn_reg	RW	0x0	2.4G 软复位，0：复位有效
20	usb_rstn_reg	RW	0x0	usb 软复位，0：复位有效
19:18	NA	--	--	保留
17	cdc_mclk_gr_upd	RW	0x0	audio ADC 分频寄存器更新，写 1 自清 0
16	cdc_hclk_en	RW	0x0	audio ADC hclk 使能，1 使能
15	rf24g_mclk_en	RW	0x0	2.4G mclk 使能
14	rf24g_hclk_en	RW	0x0	2.4G hclk 使能
13	usb_mclk_en	RW	0x0	usb mclk 使能
12	usb_hclk_en	RW	0x0	usb hclk 使能
11:8	cdc_mclk_gr	RW	0x0	audio ADC mclk 一级分频
7:4	rf24g_mclk_div	RW	0x0	2.4G mclk 分频比，分频值=寄存器值+1
3:0	usb_mclk_div	RW	0x0	usb mclk 分频比，分频值=寄存器值+1

4.8.19. RF_REG2 寄存器

- 地址偏移：0x1b8

位数	名称	方向	复位值	描述
31:16	cdc_mclk_mul	RW	0x0	audio ADC 小数分频 MUL 值
15:0	cdc_mclk_div	RW	0x0	audio ADC 小数分频 DIV 值

4.8.20. audio_adc_ctrl_reg0 控制寄存器

- 地址偏移：0x1bc

位数	名称	方向	复位值	描述
28	micgain2	RW	0x0	opa 第 1 级增益，默认 0,1=20db，,0=0db
27:24	volr2	RW	0x0	opa 第 2 级增益，默认 0000， 0~22.3db,1.6db step
23:19	resadj	RW	0x1e	bias 寄存器控制信号，默认 11110， <23:21> pga bias，默认 111，111 电流最小，000 电流最大 <20>用来控制 PGA 差分和单端，默认 1=差分，0=单端 <19>默认 0，没有使用。
18:17	cmp_ctrl	RW	0x1	比较器迟滞电压控制，默认 01， 00=0mv,01=12mv,10=23mv,11=36mv
16:15	cmp_ibc	RW	0x1	cmp_ibc<1:0>,比较器电流控制， 默认 01，00=10ua 最大,01=3.33ua,10=2ua,11=1.43ua 最小
14	bg5v_ctrl	RW	0x0	没有使用，是 BG 为了兼容 5V 电压加的寄存器
13	en_temp_sensor	RW	0x0	temp_sensor 使能，默认 0,1=en
12	pdbias	RW	0x1	pga bias pd 信号，默认 1，0=en
11:8	volr	RW	0x0	没有使用,codec pga 第 2 级增益，默认 0000，0~22.3db,1.6db step
7:5	opa_ctrl	RW	0x4	独立运放控制信号，量产版默认 100， <7>是运放 pd 信号，1=pd，0=en; <6>是运放正负极切换信号，默认 0=不切换，1=正负极反； <5>运放电流控制信号，默认 0=大电流，1=小电流；
4	pdopa	RW	0x1	pga pd 信号，默认 1，1=pd,0=en
3：0	micgain	RW	0x0	没有使用，codec pga 第一级增益，默认 0，1=20db,0=0db

4.8.21. UART2_CLK_GRCTL 控制寄存器

- 地址偏移：0x250

位数	名称	方向	复位值	描述
5	UART2_PCLK_EN	RW	0x0	UART2 PCLK 时钟使能。 0：关闭时钟。 1：使能时钟。
4	UART2_CLK_GR_UPD	WC1	0x0	uart2_clk GR 参数更新寄存器，1 有效，自清 0
3:0	UART2_CLK_GR	RW	0x8	uart2_clk 相对于 32MHz 时钟的一级分频比为:UART2_CLK_GR/8

4.8.22. UART2_CLK_CTL 控制寄存器

- 地址偏移：0x254

位数	名称	方向	复位值	描述
31:16	UART2_CLK_MUL	RW	0x48	uart2_clk 相对于 32MHz 时钟的二级分频系数 MUL
15:0	UART2_CLK_DIV	RW	0x0271	uart2_clk 相对于 32MHz 时钟的二级分频系数 DIV

4.8.23. I2S_CLK_CTL 控制寄存器

- 地址偏移：0x258

位数	名称	方向	复位值	描述
13	I2S_PCLK_EN	RW	0x0	I2S_PCLK 时钟使能
12	I2S_MCLK_EN	RW	0x0	I2S_MCLK 时钟使能
11:0	I2S_MCLK_DIV	RW	0x7C	I2S_MCLK 时钟分频系数

4.8.24. CAN_CLK_CTL 时钟控制寄存器

- 地址偏移：0x25C

位数	名称	方向	复位值	描述
13	CAN_PCLK_EN	RW	0x0	CAN_PCLK 时钟使能
12	CAN_MCLK_EN	RW	0x0	CAN_MCLK 时钟使能
11:0	CAN_MCLK_DIV	RW	0xFF	CAN_MCLK 时钟分频系数

4.8.25. CMP_CTL 控制寄存器

- 地址偏移：0x260

位数	名称	方向	复位值	描述
31	clkout_en	RW	0x0	芯片内部高速时钟输出到 GPIO2 的使能控制。
30	cmp_pclk_en	RW	0x0	比较器 pclk 使能
29:20	clkout_div	RW	013F	GPIO2 输出时钟的分频控制。
19	cmp_rstn	RW	0x1	cmp 软复位：:0 复位
18:16	cmp_debounce_en	RW	0x7	去抖动使能
13:8	cmp_edge_sel	RW	0x0	13:12: cmp2, 11:10: cmp1, 9:8: cmp0 0: dual edge 1: rise edge 2/3: fall edge
7	cmp_vref_sel	RW	0x0	CMP_VREF_SEL
6:4	cmp_en_ana	RW	0x0	cmp 模拟部分使能: 1 enable
2:0	cmp_en	RW	0x0	cmp 数字部分使能 1 enable

4.8.26. CMP_INT 寄存器

- 地址偏移：0x264

位数	名称	方向	复位值	描述
2	cmp_out2_edge	RC1	0x0	中断状态寄存器
1	cmp_out1_edge	RC1	0x0	中断状态寄存器
0	cmp_out0_edge	RC1	0x0	中断状态寄存器

4.8.27. CMP_INT_RAW 寄存器

- 地址偏移：0x268

位数	名称	方向	复位值	描述
2	cmp_out2_edge_raw	R	0x0	原始中断状态寄存器
1	cmp_out1_edge_raw	R	0x0	原始中断状态寄存器
0	cmp_out0_edge_raw	R	0x0	原始中断状态寄存器

4.8.28. CMP_INT_EN 寄存器

- 地址偏移：0x26C

位数	名称	方向	复位值	描述
2	cmp_out2_edge_en	RW	0x0	中断使能
1	cmp_out1_edge_en	RW	0x0	中断使能
0	cmp_out0_edge_en	RW	0x0	中断使能

4.8.29. FMC_CTL 控制寄存器

- 地址偏移：0x270

位数	名称	方向	复位值	描述
14	fmc_hmaster	RW	0x0	fmc hmaster 信号，现在不使用
13	fmc_cache_rstn	RW	0x1	fmc 内部缓存和总线软复位，0 复位激活
12	fmc_ssi_rstn	RW	0x1	fmc 内部 xip ssi 软复位，0 复位激活
11	fmc_xip_disable	RW	0x0	1 : disable xip 0: enable xip
10	fmc_ss_in_n	RW	0x1	ss in : 1 enable, 0 disable
9	fmc_lp_en	RW	0x0	low power enable 1 enable, 0 disable
8	fmc_ret1n	RW	0x1	fmc mem retation 模式，1: retation
7:4	fmc_mclk_div	RW	0x0	分频比为 div+1
2	fmc_mclk_src_sel	RW	0x0	1: sel bbpll clock 0: sel mclk in
1	fmc_mclk_en	RW	0x1	fmc mclk enalbe , 1 enable
0	fmc_hclk_en	RW	0x1	fmc hclk enalbe , 1 enable

4.8.30. OTP_CTL 控制寄存器

- 地址偏移：0x274

位数	名称	方向	复位值	描述
9	otp_div_bypass	RW	0x0	otp_gclk 时钟分频器旁路 1：绕过 otp_gclk 除法器，然后 otp_gclk_div 必须配置为零 0：使用 otp_gclk 除法器，然后必须将 otp_gclk_div 配置为非 0
8	otp_rstn	RW	0x1	otp s 软件复位
7:4	otp_gclk_div	RW	0x1	otp_gclk 时钟分频器，div+1
1	otp_gclk_upd	WC1	0x0	otp_gclk_div 参数更新，write to 1 auto clear 0
0	otp_hclk_en	RW	0x1	otp_hclk 使能，1 enable

4.9. CPRAO 寄存器描述

寄存器位于 AO 域，始终保持上电状态。

4.9.1. 睡眠控制寄存器

4.9.1.1. SLP_CTL 睡眠控制寄存器

- 地址偏移：0x4

位数	名称	方向	复位值	描述
0	OSCEN_CTL	RW	0x0	OSCEN 控制： 0：OSCEN 在进入睡眠时关闭 OSC 晶振； 1：OSCEN 在进入睡眠时不关闭 OSC 晶振

4.9.1.2. SLPCTL_INT_MASK 睡眠唤醒 M0 中断屏蔽寄存器

- 地址偏移：0x8

位数	名称	方向	复位值	描述
31：0	INT_MASK	RW	0x0	屏蔽唤醒 M0 的中断，对应 bit 为 1 屏蔽 bit 0 -BLE_Handler bit 1 -DMA_Handler bit 3 -GPIO_Handler --- 参考中断号列表

4.9.1.3. SLP_PD_MASK 睡眠断电屏蔽寄存器

- 地址偏移：0xC

位数	名称	方向	复位值	描述
8	sys_slp_pmu_pd_mask	RW	0x0	系统 pmu 睡眠断电屏蔽 1：屏蔽深睡状态下的 pmu 断电 0：不屏蔽
7	sys_slp_pd_mask	RW	0x0	系统睡眠断电屏蔽 1：屏蔽深睡状态下的大数字断电 0：不屏蔽
0	ram_slp_pd_mask	RW	0x1	ram 睡眠断电屏蔽 1：屏蔽深睡状态下的 ram 断电 0：不屏蔽

4.9.2. 系统配置寄存器

4.9.2.1. MCLK_DIV_CLK 分频控制寄存器

- 地址偏移：0x10

位数	名称	方向	复位值	描述
12	MCLK_DIV_CLK_EN	RW	0x0	MCLK_DIV_CLK 时钟使能
11	NA	--	--	保留
10:0	MCLK_DIV_CLK_DIV	RW	0x3E7	MCLK_DIV_CLK 时钟分频系数

4.9.2.2. RFPMU_SPI_CTL 选择控制寄存器

- 地址偏移：0x14

位数	名称	方向	复位值	描述
0	RFPMU_SPI_SEL	RW	0x0	RFPMU_SPI 选择： 0：选择 RF SPI 1：选择 PMU SPI

4.9.2.3. SYS_TIME 系统定时控制寄存器

- 地址偏移：0x1C

位数	名称	方向	复位值	描述
19:12	MRST_READY_TIME	RW	0x7	系统复位稳定时间
11:0	MCLK_STABLE_TIME	RW	0x200	OSC32M/16M 晶振稳定时间：默认值设定为 16ms

4.9.3. BBLDO_ADJ 控制寄存器

- 地址偏移：0x20

位数	名称	方向	复位值	描述
7:4	bb_coreldo_adj_sleep	RW	0b0001	睡眠模式下 coreldo 的工作电压配置
3:0	bb_coreldo_adj_work	RW	0b1100	工作模式下 coreldo 的工作电压配置。 睡眠模式和工作模式的切换由硬件睡眠信号 sys_mclk_off 控制。该信号为 0 时是工作模式，为 1 时是睡眠模式

4.9.4. RF AON 寄存器

- 地址偏移：0x30

位数	名称	方向	复位值	描述
4	DA_BG_EN	RW	0x1	RF BG 软件使能，1 使能
3	OSCEN_AO	RW	0x0	32M 晶体常开使能，睡眠时候不会关断 32M 晶体， 1：使能常开，DA_DCXO_EN 和 DA_CLK_BB_EN 保持为 1 0：DA_DCXO_EN 和 DA_CLK_BB_EN 由硬件状态机控制
2	DA_LDO_EN	RW	0x0	DA_LDO 软件使能，1 使能
1	DA_LDO_SEL	RW	0x0	RF LDO 和 BG 的控制源选择： 0：LDO 和 BG 由硬件自动控制使能 1:LDO 和 BG 由 da_bg_en 和 da_ldo_en 寄存器软件控制使能
0	DA_DCXO_IC	RW	0x0	调整晶振电流，默认 0 小电流，1 大电流。

4.9.5. 管脚上下拉控制寄存器

4.9.5.1. PE_CTRL1 管脚上下拉控制寄存器 1

- 地址偏移：0x34

位数	名称	方向	复位值	描述
31:0	pectrl1	RW	0x400000	GPIO 上下拉控制寄存器，高有效。 GPIO：0 ~ 10，14~15： GPIOx 下拉使能：pe_ctrl1[2*x] GPIOx 上拉使能： pe_ctrl1[2*x+1] BOOTCTL_port 下拉使能： pe_ctrl1[22] BOOTCTL_port 上拉使能： pe_ctrl1[23] SWCK port 下拉使能： pe_ctrl1[24] SWCK port 上拉使能： pe_ctrl1[25] SWD port 下拉使能： pe_ctrl1[26] SWD port 上拉使能： pe_ctrl1[27]

4.9.5.2. PE_CTRL2 管脚上下拉控制寄存器 2

- 地址偏移：0x38

位数	名称	方向	复位值	描述
31:0	Pectrl2	RW	0x0	GPIO 上下拉控制寄存器，高有效。 GPIO：16 ~ 31： GPIOx 下拉使能： pe_ctrl2[2*(16-x)] GPIOx 上拉使能： pe_ctrl2[2*(16-x)+1]

4.9.5.3. PU_CTRL3 管脚上下拉控制寄存器 3

- 地址偏移：0x3C

位数	名称	方向	复位值	描述
11:0	puctrl1	RW	0x400	GPIO 上下拉控制寄存器，高有效。 [0]：SSICLK 下拉使能 [1]：SSISSN 上拉使能 [2]：SSIRX 下拉使能 [3]：SSITX 下拉使能 [4]：GPIO32 下拉使能 [5]：GPIO32 上拉使能 [6]：GPIO33 下拉使能 [7]：GPIO33 上拉使能 [8]：GPIO34 下拉使能 [9]：GPIO34 上拉使能 [10]：GPIO35 下拉使能 [11]：GPIO35 上拉使能

4.9.6. VDD_SWITCH_EN 开关控制寄存器

- 地址偏移：0x40

位数	名称	方向	复位值	描述
4	REG_VDDROM_EN	RW	0x1	ROM 断电
3	REG_VDDRAM0_EN	RW	0x1	SHRAM0 L bank 断电
2	REG_VDDRAM1_EN	RW	0x1	SHRAM0 H bank 断电
1	REG_VDDLTRAM_EN	RW	0x1	BT RAM 断电
0	REG_VDDMDM_EN	RW	0x1	BT MODEM 断电

4.9.7. VDD_ISO_EN 断电隔离控制寄存器

- 地址偏移：0x44

位数	名称	方向	复位值	描述
4	ISO_VDDROM_EN	RW	0x1	ROM 断电隔离
3	ISO_VDDRAM0_EN	RW	0x1	SHRAM0 L bank 断电隔离
2	ISO_VDDRAM1_EN	RW	0x1	SHRAM0 H bank 断电隔离
1	ISO_VDDLTRAM_EN	RW	0x1	BT RAM 断电隔离
0	ISO_VDDMDM_EN	RW	0x1	BT MODEM 断电隔离

4.9.8. LPO LDO 控制寄存器

- 地址偏移：0x48

位数	名称	方向	复位值	描述
2：0	BB_LPOLDO_ADJ	RW	0x0	LPOLDO 档位配置

4.9.9. LDO 控制寄存器

- 地址偏移：0x4C

位数	名称	方向	复位值	描述
2：0	BB_RETMEMLDO_ADJ	RW	0x0	RETMEMLDO 档位配置

4.9.10. AON_CORERFLDO_EN 数字部分 LDO 控制寄存器

- 地址偏移：0x50

位数	名称	方向	复位值	描述
0	BB_CORERFLDO_EN	RW	0x0	RF digital 断电

4.9.11. DCDC_CTRL0 控制寄存器 0

- 地址偏移：0x54

位数	名称	方向	复位值	描述
6	dcdc_en_sel	RW	0x0	dcdc ctrl select, 1:software ctrl,0:hardware ctrl
5	dcdc_bg_en_soft	RW	0x0	dcdc bandgap en 信号，1 work，要比 dcdc_en 早一个 32K 周期，硬件控制，默认 0
4	dcdc_en_soft	RW	0x0	dcdc en 信号，1 work，硬件控制，默认 0
3	dcdc_voutdown	RW	0x0	dcdc 寄存器，控制 dcdc 输出下拉到地，1 下拉到地，默认 0
2	dcdc_cl_ctrl	RW	0x0	dcdc 寄存器，默认 0
1:0	dcdc_pdrslow	RW	0x0	dcdc 寄存器，默认 00

4.9.12. DCDC_CTRL1 控制寄存器 1

- 地址偏移：0x58

位数	名称	方向	复位值	描述
30:28	dcdc_ctrl	RW	0x0	dcdc 寄存器，控制输出电压，默认 100
27	dcdc_dcm	RW	0x0	dcdc 寄存器，默认 1
26:22	dcdc_vref_cal	RW	0x10	dcdc 寄存器,微调参考电压,默认 10000
21:17	dcdc_cal	RW	0x10	dcdc 寄存器,微调输出电压,默认 10000
16:11	dcdc_stbop	RW	0x19	dcdc 寄存器，默认 011001
10	dcdc_deadtime	RW	0x0	dcdc 寄存器，默认 0
9	dcdc_zcd_en	RW	0x0	dcdc 寄存器，默认 0
8:4	dcdc_freqcal	RW	0x8	dcdc 寄存器,控制时钟频率,默认 01000
3	dcdc_dispfm	RW	0x0	dcdc 寄存器，默认 0
2:1	dcdc_cf	RW	0x1	dcdc 寄存器，控制时钟分频比，默认 01
0	dcdc_lp_en	RW	0x1	dcdc 寄存器，默认 1

4.9.13. AOCLKEN_GRCTL AO 域时钟使能寄存器

- 地址偏移：0x7C

位数	名称	方向	复位值	描述
5	bt32k_clk_ao_en	RW	0x0	bt 常开 32k 时钟使能
4	timer_ao_rstn	RW	0x1	timer_aon 软复位，0 复位
3	timer1_ao_clk_en	RW	0x0	timer_ao 1 工作时钟使能
2	timer0_ao_clk_en	RW	0x0	timer_ao 0 工作时钟使能
1	rtc_clk_en	RW	0x0	rtc 工作时钟使能
0	timer_ao_pclk_en	RW	0x0	timer ao 总线时钟使能

4.9.14. CTL_CLK32K 时钟设置寄存器

- 地址偏移：0x14C

位数	名称	方向	复位值	描述
9:8	CLK32K_SEL	RW	0x1	32K 时钟源选择： 00：由 mclk_chip_i 分频得到 01：来自 clk32kin 10：来自 clk32k_rc_i 11：由 mclk_chip_i 分频得到
7:5	NA	--	--	保留
4:2	OSC32K_DS	RW	0x2	OSC32K 晶振控制
1	OSC32K_RESTART	RW	0x0	OSC32K 晶振控制
0	CLK32K_GLITCH_BYPASS	RW	0x1	32K 时钟滤毛刺 BYPASS： 1：使能对 32K 时钟的滤毛刺功能； 0:不使能对 32K 时钟的滤毛刺功能

4.9.15. AO 控制寄存器

4.9.15.1. ALWAYS_ON_REG0 AO 通用寄存器 0

- 地址偏移：0x154

位数	名称	方向	复位值	描述
31:0	AO_REG0	RW	0x0	AO 域通用寄存器 0，为软件预留

4.9.15.2. ALWAYS_ON_REG1 AO 通用寄存器 1

- 地址偏移：0x158

位数	名称	方向	复位值	描述
31:28	DA_BG_RF_RCT	RW	0x3	DA_BG_RF_RCT，晶体控制寄存器
27	retmem0_iso_en	RW	0x0	现在不用
26	retmem1_iso_en	RW	0x0	现在不用
25	BB_CORELDO_DCDC_SW_MUX	RW	0x1	BB_CORELDO_DCDC_SW_MUX
24	BB_CORELDO_DCDC_SW_SOFTSEL	RW	0x0	0: BB_CORELDO_DCDC_SW 由硬件自动开关，在睡眠时候自动关断 1 ： BB_CORELDO_DCDC_SW_MUX 由 BB_CORELDO_DCDC_SW_SOFT 配置
23	BB_CORELDO_DCDC_SW_SOFT	RW	0x0	BB_CORELDO_DCDC_SW 软件模式的配置值。
22	ram_retation_en	RW	0x0	ram retation 使能，1 有效，大数字断电时候 retation memory 该寄存器为 1 时，大数字可以关闭，LDO_DIG 不会关闭，提供 memory retation 的供电。
19	timer_ao_sleep_clksw	RW	0x0	timer ao pclk 切换到 32K 时钟，此时 timer ao 寄存器无法读写了。睡眠时刻需要写改寄存器为 1，唤醒后写该寄存器为 0

4.9.15.3. AO_CTRL AO 控制寄存器

- 地址偏移：0x15C

位数	名称	方向	复位值	描述
0	RTC_RSTN_AO	RW	0x1	RTC 的软复位寄存器

4.9.15.4. ALWAYS_ON_REG2 AO 控制寄存器 2

- 地址偏移：0x160

位数	名称	方向	复位值	描述
31:0	ao_reg2	RW	0x1	aon 通用寄存器 0 bootrom 里的用法就是： 第 31bit 表示热启动使能。 30:0 表示热启动后要跳转的 pc 指针。

4.9.15.5. ALWAYS_ON_REG3 AO 控制寄存器 3

- 地址偏移：0x16C

位数	名称	方向	复位值	描述
31:0	ao_reg3	RW	0x1	aon 通用寄存器 0

4.9.15.6. AON_BOR_CTRL BOR 控制器

- 地址偏移：0x170

位数	名称	方向	复位值	描述
7	bor_out_sync	R	0x1	BOR 当前的状态，经过总线时钟同步
6	bor_out	R	0x1	BOR 当前的原始状态，未经过总线时钟同步
5	bor_rstn_mask	RW	0x0	BOR 复位屏蔽，屏蔽复位数字域
4	bor_intr_mask	RW	0x1	BOR 送入 MCU 的中断屏蔽
3	en_bor	RW	0x1	BOR 使能
2:0	bor_ctrl	RW	0x0	BOR 控制寄存器

4.9.15.7. ALWAYS_ON_REG4 AO 控制寄存器 4

- 地址偏移：0x174

位数	名称	方向	复位值	描述
2	bb_coreldo_normal_sw_mux	RW	0x0	默认 0，BB_COREVDDLDO_NORMAL 不起作用，1=BB_COREVDDLDO_NORMAL 起作用
1	bb_lpoldo_ibc	RW	0x0	默认 0，用来控制 ldo 电流，1 电流加倍
0	bb_corevddldo_ibc	RW	0x0	默认 0，用来控制 ldo 电流，1 电流加倍

第 5 章 通用 IO 控制器（GPIO）

5.1. 概述

通用 IO 控制器（GPIO）是通用管脚输入输出控制器。GPIO 模块内包含 32 个 IO 端口。GPIO 模块可以被 M0 访问。

- GPIO 的基地址为：0x40001000

5.1.1. 主要特性

XC66xx 的每个 GPIO 均可以独立访问。主要有以下功能特点：

- GPIO 均可独立配置。
- GPIO 均可分别配置为数据输入、数据输出、中断输入工作模式。
- GPIO 端口均支持独立配置中断寄存器。
- GPIO 的中断输出到 M0，它们有自己独立的中断屏蔽寄存器和中断状态寄存器。
- GPIO 均支持多种类型的中断输入信号，包括：
  - 高电平中断
  - 低电平中断
  - gpio_clk（32K 时钟）上升沿中断
  - gpio_clk 下降沿中断
  - gpio_clk 沿中断
  - pclk（APB 总线时钟）上升沿中断
  - pclk 下降沿中断
  - pclk 沿中断
- 可以用去毛刺操作来过滤外部中断输入信号
  - 小于一个 gpio_clk（32K 时钟）周期的干扰脉冲输入将被去毛刺操作过滤掉
  - GPIO 的 32 个 IO 端口均可独立设置是否使能去毛刺操作

5.1.2. 功能描述

GPIO 的原理框图如下。

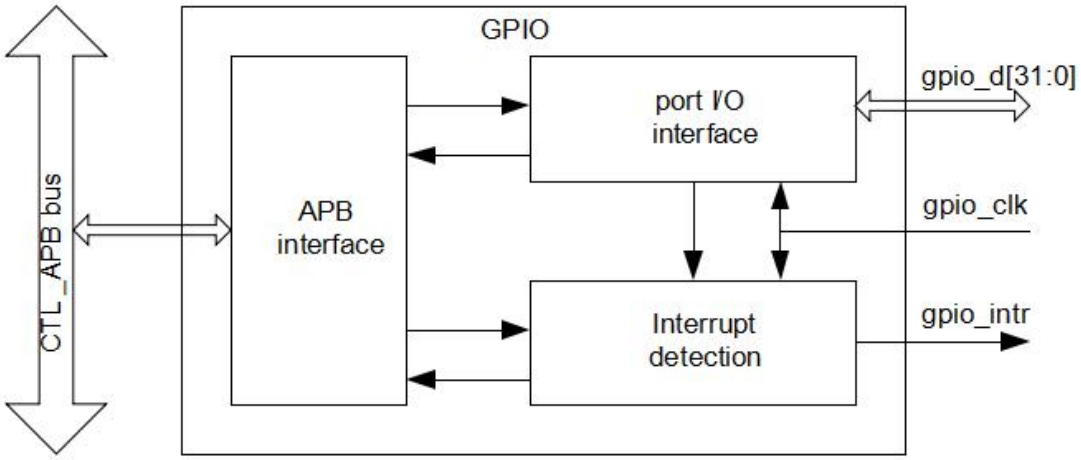

图 5-1 GPIO 模块结构框图

5.1.3. 工作模式

GPIO 根据传输方向和传输的信息类型可以分为数据输入、数据输出、中断输入等三种 工作模式。GPIO 均可独立设置为三种工作模式中的任意一种。

5.1.3.1. 数据输入

GPIO 默认工作在数据输入模式下。

当 GPIO 工作在数据输入模式时，首先通过端口数据方向寄存器（GPIO_PORT_DDRx，x=0~2）来设置对应 IO 端口的方向。当 IO 端口设置为输入方向时，对端口输入寄存器（GPIO_EXT_PORTx，x=0~1）的读操作能够获得对应 IO 端口的输入值。当 IO 端口设置为输出方向时，对端口输入寄存器（GPIO_EXT_PORTx，x=0~1）的读操作获得的是端口数据寄存器（GPIO_PORT_DRx，x=0~2）的值。

由于每次读操作所需要的时钟周期至少为两个 APB 总线时钟周期，所以通过 GPIO 对 IO 端口的输入的最大采样速率为 pclk/2。

5.1.3.2. 数据输出

当 GPIO 工作在数据输出模式时，首先通过端口数据方向寄存器（GPIO_PORT_DDRx，x =0~2）来设置 IO 端口的方向。当 IO 端口设置为输出方向时，可以通过设置端口数据寄存器（GPIO_PORT_DRx，x=0~2）实现输出高电平或者低电平。

5.1.3.3. 中断输入

当 GPIO 工作在中断输入模式时，必须通过端口数据方向寄存器（GPIO_PORT_DDRx，x=0~2）将 IO 端口设置为输入方向。若此时将 IO 端口设置为输出方向，GPIO 无法输入新的中断，但是已有的中断不会丢失（高电平中断、低电平中断除外）。

GPIO 支持多种类型的中断输入，每个 IO 端口均可单独配置成不同的中断输入模式。

中断输入模式的类型如下所示：

- 高电平中断
- 低电平中断
- gpio_clk（32K 时钟）上升沿中断
- gpio_clk 下降沿中断
- gpio_clk 沿中断
- pclk 上升沿中断
- pclk 下降沿中断
- pclk 沿中断

当 GPIO 工作在中断输入模式下时，各种中断类型的中断原始状态产生条件如下所示：

- 对于高电平中断，当 IO 端口输入为高电平时，对应的中断原始状态产生。
- 对于低电平中断，当 IO 端口输入为低电平时，对应的中断原始状态产生。
- 对于 gpio_clk 上升沿中断，当 gpio_clk 有效并且检测到 IO 端口输入信号的上升沿时，对应的中断原始状态产生。
- 对于 gpio_clk 下降沿中断，当 gpio_clk 有效并且检测到 IO 端口输入信号的下降沿时，对应的中断原始状态产生。
- 对于 gpio_clk 沿中断，当 gpio_clk 有效并且检测到 IO 端口输入信号的上升沿或下降沿时，对应的中断原始状态产生。
- 对于 pclk 上升沿中断，当 pclk 有效并且检测到 IO 端口输入信号的上升沿时，对应的中断原始状态产生。
- 对于 pclk 下降沿中断，当 pclk 有效并且检测到 IO 端口输入信号的下降沿时，对应的中断原始状态产生。
- 对于 pclk 沿中断，当 pclk 有效并且检测到 IO 端口输入信号的上升沿或下降沿时，对应的中断原始状态产生。

5.1.4. 去毛刺操作

当 GPIO 工作在中断输入模式时，可以通过设置去毛刺使能寄存器（GPIO_DEBOUNCEx，x=0~2）来选择是否使用去毛刺（Debounce）操作。GPIO 的每个 IO 端口均可独立设置是否使能去毛刺操作。

当去毛刺（Debounce）操作使能时，其效果如下所示：

- 对于小于一个 gpio_clk（32K 时钟）周期的输入脉冲，通过去毛刺（Debounce）操作可以将此输入脉冲过滤掉。
- 对于介于一个和两个 gpio_clk 周期宽度之间的输入脉冲，去毛刺（Debounce）操作无法确保此输入脉冲是否被过滤掉，此输入脉冲是否被过滤掉取决于输入脉冲和 gpio_clk 之间的相位关系。
- 对于大于两个 gpio_clk 周期宽度之间的输入脉冲，去毛刺（Debounce）操作对输入脉冲无影响。

去毛刺（Debounce）操作的使用限制如下所示：

- 去毛刺（Debounce）操作依赖 gpio_clk，因此若希望使能去毛刺（Debounce）操作，gpio_clk 必须有效。
- 使用去毛刺（Debounce）操作会增加三个 gpio_clk 时钟周期的中断响应时间。

5.1.5. 中断说明

当 GPIO 工作在中断输入模式且 IO 端口为输入方向时，首先通过设置中断控制寄存器（GPIO_INTR_CTRLx，x=0~7）设置中断类型并使能中断输入模式。之后当中断原始条件产生时，中断原始状态寄存器（GPIO_INTR_RAW0）中的对应位置‘1’。

GPIO 模块有自己的中断屏蔽寄存器（GPIO_INTR_MASK_C0x，x=0~1）和中断状态寄存器（GPIO_INTR_STATUS_C00）。

当中断原始状态寄存器（GPIO_INTR_RAW0）对应位置‘1’且中断屏蔽寄存器（GPIO_INTR_MASK_C0x，x=0~1）对应位置‘0’时，中断状态寄存器（GPIO_INTR_STATUS_C0）的对应位置‘1’，触发对应的中断管脚的有效状态，引起对应的 CPU 中断。

对于 gpio_clk 上升沿中断、gpio_clk 下降沿中断、gpio_clk 沿中断、pclk 上升沿中断、pclk 下降沿中断和 pclk 沿中断，对相应的中断清除寄存器（GPIO_INTR_CLR0）写‘1’可清除中断。对高电平中断和低电平中断，对相应的中断清除寄存器（GPIO_INTR_CLR0）的操作无效。

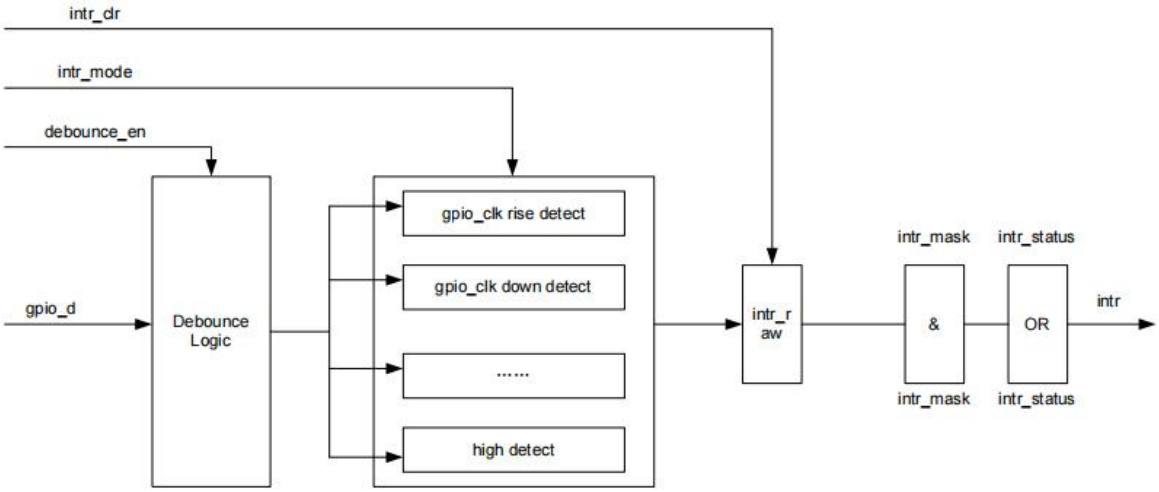

图 5-2 GPIO 中断示意图

GPIO 的 32 个 IO 端口和中断寄存器的对应关系如下所示：

GPIO Portn 的对应寄存器位为 GPIO_INTR_CTRLx[y*4+3:y*4]，其中 x=n/4，y=n%4

GPIO Portn 的对应寄存器位为 GPIO_DEBOUNCEx[y]，其中 x=n/16，y=n%16

GPIO Portn 的对应寄存器位为 GPIO_INTR_RAWx[y]，其中 x=n/32，y=n%32

GPIO Portn 的对应寄存器位为 GPIO_INTR_CLRx[y]，其中 x=n/32，y=n%32

GPIO Portn 的对应寄存器位为 GPIO_INTR_MASK_C0x[y]，其中 x=n/16，y=n%16

GPIO Portn 的对应寄存器位为 GPIO_INTR_STATUS_C0x[y]，其中 x=n/32，y=n%32“/”为除法运算符，“%”为求余运算符。

5.1.6. 信号及接口描述

表 5-1 GPIO 接口信号描述

名称	宽度	方向	描述	备注
CTL_APB 总线接口				内部信号
时钟接口				
gpio_clk	1	I	32K 时钟	内部信号
pclk	1	I	APB 总线时钟	内部信号
数据接口				
gpio_d	32	I/O	GPIO 输入/输出端口	管脚信号
中断				
gpio_intr	1	O	GPIO 中断输出	内部信号

5.2. GPIO 寄存器地址映射

XC66xx 的 GPIO 的基地址为：0x40001000。

GPIO 寄存器地址映射如下表所示。

表 5-2 GPIO 寄存器映射表

寄存器名	描述	方向&宽度	地址偏移	复位值
CPR_CTL_MUXCTL	MUX 控制寄存器	RW 32bits	(0x40000000) 0x128~0x12C	0x0
CPR_GPIO_FUN_SELx (x=0~7)	复用控制寄存器	RW 32bits	(0x40000000) 0xC0~0xDC	0x0
GPIO_PORT_DRx(x=0~1)	端口数据寄存器	RW 32bits	0x00~0x04	0x0
GPIO_PORT_DDRx(x=0~1)	端口数据方向寄存器	RW 32bits	0x20~0x24	0x0
GPIO_EXT_PORT0(x=0~1)	外部端口输入寄存器	R 32bits	0x40	0x0
GPIO_INTR_CTRLx(x=0~1)	中断控制寄存器	RW 20bits	0x100~0x11c	0x0
GPIO_DEBOUNCEx(x=0~1)	去毛刺使能寄存器	RW 32bits	0x180~0x184	0x0
GPIO_INTR_RAW0	中断原始状态寄存器	R 32bits	0x1a0	0x0
GPIO_INTR_CLR0	中断清除寄存器	W 32bits	0x1b0	0x0
GPIO_INTR_MASK_C0x (x=0~1)	中断屏蔽寄存器	RW 32bits	0x200~0x204	0x0
GPIO_INTR_STATUS_C0x (x=0~1)	中断状态寄存器	R 32bits	0x220~0x224	0x0

5.3. GPIO 寄存器描述

5.3.1. GPIO MUX 控制寄存器（CPR_CTL_MUXCTLx）

- 地址偏移
  - CPR_CTL_MUXCTL1：0x40000000+0x128
  - CPR_CTL_MUXCTL2：0x40000000+0x12C

GPIO MUX 可以理解为 GPIO 的一级复用，通常情况下我们将 GPIO 的 MUX 控制寄存器设置为 0，作为普通 GPIO 来使用，但在某些情况下，例如 GPIO 要实现 PWM2 和 PWM3 功能时，需要将 GPIO0、GPIO1 的 MUX 控制寄存器相应位设置为 2。CPR_CTL_MUXCTL1 寄存器控制 GPIO0-GPIO15，CPR_CTL_MUXCTL2 寄存器控制 GPIO15-GPIO31。CPR_CTL_MUXCTL3 寄存器控制 GPIO32-GPIO39。

详细的 GPIO MUX 状态如下表所示：

表 5-3 gpio 复用表

PIN	功能 0	功能 1	功能 2	功能 3
GPIO0	gpio_d[0]	NA	Pwm4	NA
GPIO1	gpio_d[1]	NA	Pwm5	NA
GPIO2	gpio_d[2]	NA	clk_12M_out	NA
GPIO3	gpio_d[3]	NA	pta_grant	NA
GPIO4	gpio_d[4]	NA	pta_req	NA
GPIO5	gpio_d[5]	NA	pta_pri	NA
GPIO6	gpio_d[6]	NA	txen	NA
GPIO7	gpio_d[7]	NA	rxen	NA
GPIO8	gpio_d[8]	NA	can_tx0	NA
GPIO9	gpio_d[9]	NA	can_rx0	NA
GPIO10	gpio_d[10]	NA	i2s_dout	NA
GPIO11	BOOTCTL	gpio_d[11]	NA	NA
GPIO12	SWCK	gpio_d[12]	NA	Pwm2
GPIO13	SWD	gpio_d[13]	NA	Pwm3
GPIO14	gpio_d[14]	NA	ssi2_clk	NA
GPIO15	gpio_d[15]	NA	ssi2_ssn	NA
GPIO16	gpio_d[16]	NA	ssi2_d0	NA

GPIO17	gpio_d[17]	NA	ssi2_d1	NA
GPIO18	gpio_d[18]	NA	NA	NA
GPIO19	gpio_d[19]	NA	NA	NA
GPIO20	gpio_d[20]	NA	NA	NA
GPIO21	gpio_d[21]	NA	NA	NA
GPIO22	gpio_d[22]	NA	NA	NA
GPIO23	gpio_d[23]	NA	NA	NA
GPIO24	gpio_d[24]	NA	NA	NA
GPIO25	gpio_d[25]	NA	NA	NA
GPIO26	gpio_d[26]	NA	ssi2_d2	NA
GPIO27	gpio_d[27]	NA	ssi2_d3	NA
GPIO28	gpio_d[28]	NA	adc_clk	NA
GPIO29	gpio_d[29]	pwmt[0]	adc_data	NA
GPIO30	gpio_d[30]	pwmt[1]	i2s_clk	NA
GPIO31	gpio_d[31]	uart2_rx	i2s_ws	NA
GPIO32	gpio_d[32]	uart2_tx	i2s_din	NA
GPIO33	gpio_d[33]	fmc_d[2]	i2s_dout	NA
GPIO34	gpio_d[34]	fmc_d[3]	can_tx1	NA
GPIO35	gpio_d[35]	pwmt[2]	bootctl1	NA
GPIO36	gpio_d[36]	pwmt[3]		NA
GPIO37	gpio_d[36]	pwmt[4]		NA
GPIO38	gpio_d[36]	pwmt[5]		NA
GPIO39	gpio_d[36]	NA		NA

5.3.2. GPIO 复用控制寄存器（CPR_GPIO_FUN_SELx）

GPIO_FUN_SELx 可以理解为 GPIO 的二级复用，通常情况下我们将 GPIO 的复用控制寄存器 GPIO_FUN_SELx 设置为 0，作为普通 GPIO 来使用，但在某些情况下，例如 GPIO 要实现 UART0_TX 功能时，需要将相应 GPIO 的复用控制寄存器 CPR_GPIO_FUN_SELx 设置为 1。GPIO_FUN_SELx 范围为 GPIO0-28；

注意：说明：当输入有多个源的时候，编号最小的生效。例如：GPIO0 和 GPIO1 同时选择为 UART0_RX，则 UART0_RX 实际上由 GPIO0 输入；输出则允许多个端口输出同一个信号

- 地址偏移：
  - CPR_GPIO_FUN_SEL0：0x40000000+0xC0
  - CPR_GPIO_FUN_SEL1：0x40000000+0xC4
  - CPR_GPIO_FUN_SEL2：0x40000000+0xC8
  - CPR_GPIO_FUN_SEL3：0x40000000+0xCC
  - CPR_GPIO_FUN_SEL4：0x40000000+0xD0
  - CPR_GPIO_FUN_SEL5：0x40000000+0xD4
  - CPR_GPIO_FUN_SEL6：0x40000000+0xD8
  - CPR_GPIO_FUN_SEL7：0x40000000+0xDC

位数	名称	方向	复位值	描述
28:24	GPIO_FUN_SEL3	RW	0x0	GPIO3 的功能选择： 0: GPIO_Dx（x 为 GPIO 序号） 1: UART0_TX 2: UART0_RX 3: UART0_CTS 4: UART0_RTS 5: I2C_SCL 6: I2C_SDA 7: UART1_RX 8: UART1_TX 9: SIM_IO 10: SIM_RST 11: SIM_CLK_OUT 12: PWM0 13: PWM1

				14：SSI1_CLK 15：SSI1_SSN 16：SSI1_RX 17：SSI1_Tx 18：PWM0_INV 19：PWM1_INV 其它：GPIO_Dx
20:16	GPIO_FUN_SEL2	RW	0x0	GPIO2 的功能选择： 描述同 GPIO_FUN_SEL3。
12:8	GPIO_FUN_SEL1	RW	0x0	GPIO1 的功能选择： 描述同 GPIO_FUN_SEL3。
4:0	GPIO_FUN_SEL0	RW	0x0	GPIO0 的功能选择： 描述同 GPIO_FUN_SEL3。

5.3.3. GPIO 上下拉控制寄存器（PE_CTRL1、PE_CTRL2、PU_CTRL1）

5.3.3.1. PE_CTRL1

- 地址偏移：0x40002400+0x34（pe_ctrl1）

位数	名称	方向	复位值	描述
31:30	pe_ctrl1[31:30]	RW	0x0	GPIO15 的上下拉使能： 00：使能浮空 10：使能上拉 01：使能下拉
29:28	pe_ctrl1[29:28]	RW	0x0	GPIO14 的上下拉使能： 10：使能上拉 01：使能下拉
27:26	pe_ctrl1[27:26]	RW	0x0	GPIO13 的上下拉使能： 10：使能上拉 01：使能下拉
25:24	pe_ctrl1[25:24]	RW	0x0	GPIO12 的上下拉使能： 10：使能上拉 01：使能下拉

23:22	pe_ctrl1[23:22]	RW	0x1	GPIO11 的上下拉使能： 10：使能上拉 01：使能下拉
21:20	pe_ctrl1[21:20]	RW	0x0	GPIO10 的上下拉使能： 10：使能上拉 01：使能下拉
19:18	pe_ctrl1[19:18]	RW	0x0	GPIO9 的上下拉使能： 10：使能上拉 01：使能下拉
17:16	pe_ctrl1[17:16]	RW	0x0	GPIO8 的上下拉使能： 10：使能上拉 01：使能下拉
15:14	pe_ctrl1[15:14]	RW	0x0	GPIO7 的上下拉使能： 10：使能上拉 01：使能下拉
13:12	pe_ctrl1[13:12]	RW	0x0	GPIO6 的上下拉使能： 10：使能上拉 01：使能下拉
11:10	pe_ctrl1[11:10]	RW	0x0	GPIO5 的上下拉使能： 10：使能上拉 01：使能下拉
9:8	pe_ctrl1[9:8]	RW	0x0	GPIO4 的上下拉使能： 10：使能上拉 01：使能下拉
7:6	pe_ctrl1[7:6]	RW	0x0	GPIO3 的上下拉使能： 10：使能上拉 01：使能下拉
5:4	pe_ctrl1[5:4]	RW	0x0	GPIO2 的上下拉使能： 10：使能上拉 01：使能下拉
3:2	pe_ctrl1[3:2]	RW	0x0	GPIO1 的上下拉使能： 10：使能上拉 01：使能下拉

1:0	pe_ctrl1[1:0]	RW	0x0	GPIO0 的上下拉使能： 10：使能上拉 01：使能下拉

5.3.3.2. PE_CTRL2

- 地址偏移：0x40002400+0x38（pe_ctrl2）

位数	名称	方向	复位值	描述
31:30	pe_ctrl2[31:30]	RW	0x0	GPIO31 的上下拉使能： 10：使能上拉 01：使能下拉
29:28	pe_ctrl2[29:28]	RW	0x0	GPIO30 的上下拉使能： 10：使能上拉 01：使能下拉
27:26	pe_ctrl2[27:26]	RW	0x0	GPIO29 的上下拉使能： 10：使能上拉 01：使能下拉
25:24	pe_ctrl2[25:24]	RW	0x0	GPIO28 的上下拉使能： 10：使能上拉 01：使能下拉
23:22	pe_ctrl2[23:22]	RW	0x0	GPIO27 的上下拉使能： 10：使能上拉 01：使能下拉
21:20	pe_ctrl2[21:20]	RW	0x0	GPIO26 的上下拉使能： 10：使能上拉 01：使能下拉
19:18	pe_ctrl2[19:18]	RW	0x0	GPIO25 的上下拉使能： 10：使能上拉 01：使能下拉
17:16	pe_ctrl2[17:16]	RW	0x0	GPIO24 的上下拉使能： 10：使能上拉 01：使能下拉
15:14	pe_ctrl2[15:14]	RW	0x0	GPIO23 的上下拉使能： 10：使能上拉 01：使能下拉
13:12	pe_ctrl2[13:12]	RW	0x0	GPIO22 的上下拉使能： 10：使能上拉 01：使能下拉

11:10	pe_ctrl2[11:10]	RW	0x0	GPIO21 的上下拉使能： 10：使能上拉 01：使能下拉
9:8	pe_ctrl2[9:8]	RW	0x0	GPIO20 的上下拉使能： 10：使能上拉 01：使能下拉
7:6	pe_ctrl2[7:6]	RW	0x0	GPIO19 的上下拉使能： 10：使能上拉 01：使能下拉
5:4	pe_ctrl2[5:4]	RW	0x0	GPIO18 的上下拉使能： 10：使能上拉 01：使能下拉
3:2	pe_ctrl2[3:2]	RW	0x0	GPIO17 的上下拉使能： 10：使能上拉 01：使能下拉
1:0	pe_ctrl2[1:0]	RW	0x0	GPIO16 的上下拉使能： 10：使能上拉 01：使能下拉

5.3.3.3. PU_CTRL1

- 地址偏移：0x3c（pu_ctrl1）

位数	名称	方向	复位值	描述
31:19	NA	--	--	--
19:18				GPIO39
17:16	pu_ctrl1[17:16]	RW	0x0	GPIO38 的上下拉使能： 10：使能上拉 01：使能下拉
15:14	pu_ctrl1[15:14]	RW	0x0	GPIO37 的上下拉使能： 10：使能上拉 01：使能下拉
13:12	pu_ctrl1[13:12]	RW	0x0	GPIO36 的上下拉使能： 10：使能上拉 01：使能下拉
11:10	pu_ctrl1[11:10]	RW	0x0	GPIO35 的上下拉使能： 10：使能上拉 01：使能下拉
9:8	pu_ctrl1[9:8]	RW	0x0	GPIO34 的上下拉使能： 10：使能上拉 01：使能下拉
7:6	pu_ctrl1[7:6]	RW	0x0	GPIO33 的上下拉使能： 10：使能上拉

				01：使能下拉
5:4	pu_ctrl1[5:4]	RW	0x0	GPIO32 的上下拉使能： 10：使能上拉 01：使能下拉
3	pu_ctrl1[3]	RW	0x0	SSI0TX 下拉使能：(上拉不使能) 1：使能 0：不使能
2	pu_ctrl1[2]	RW	0x0	SSI0RX 下拉使能：(上拉不使能) 1：使能 0：不使能
1	pu_ctrl1[1]	RW	0x0	SSI0SSN 上拉使能：(下拉不使能) 1：使能 0：不使能
0	pu_ctrl1[0]	RW	0x0	SSI0CLK 下拉使能：(上拉不使能) 1：使能 0：不使能

5.3.4. 端口数据输出寄存器（GPIO_PORT_DRx）

- 地址偏移：
  - GPIO_PORT_DR0：0x0
  - GPIO_PORT_DR1：0x4

  GPIO Portn 的对应寄存器位为 GPIO_PORT_DRx[y]，其中 x=n/16，y=n%16，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:16	DR_WE	W	0x0	DR 写使能位 0：不允许写 1：允许写 DR_WE[n]控制 DR[n]，如 DR_WR[0]控制 DR[0]
15:0	DR	RW	0x0	相应端口设置为“输出”模式时，该寄存器中的值决定 IO 端口的输出。读寄存器，返回值为上一次写入该寄存器的值。 0：低电平 1：高电平

5.3.5. 端口数据方向寄存器（GPIO_PORT_DDRx）

- 地址偏移：
  - GPIO_PORT_DDR0：0x20
  - GPIO_PORT_DDR1：0x24

GPIO Portn 的对应寄存器位为 GPIO_PORT_DRx[y]，其中 x=n/16，y=n% 16，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:16	DDR_WE	W	0x0	DDR 写使能位 0：不允许写 1：允许写 DDR_WE[n]控制 DDR[n]， 如 DDR_WR[0]控制 DDR[0]
15:0	DDR	RW	0x0	IO 端口的方向。 0：输入方向 1：输出方向

5.3.6. 外部端口输入寄存器（GPIO_EXT_PORTx）

- 地址偏移：
  - GPIO_EXT_PORT0：0x40
  - GPIO_EXT_PORT0：0x44

GPIO Portn 的对应寄存器位为 GPIO_EXT_PORTx[y]，其中 x=n/32，y=n%32，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:0	DIN	R	0x0	当端口配置为“输入”，读该寄存器的相应 位可以得到对应 IO 端口的值。如果端口配置为“输出”，读该寄存器得到的是端口数据寄存器（GPIO_PORT_DRx）的值。

5.3.7. 中断控制寄存器（GPIO_INTR_CTRLx）

- 地址偏移：
  - GPIO_INTR_CTRL0：0x100
  - GPIO_INTR_CTRL1：0x104
  - GPIO_INTR_CTRL2：0x108
  - GPIO_INTR_CTRL3：0x10C
  - GPIO_INTR_CTRL4：0x110
  - GPIO_INTR_CTRL5：0x114
  - GPIO_INTR_CTRL6：0x118
  - GPIO_INTR_CTRL7：0x11C

GPIO Portn 的对应寄存器位为 GPIO_INTR_CTRLx[y*4+3: y*4]，其中 x=n/4，y=n%4，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:20	NA	--	--	保留
15:12	MODE_WE	W	0x0	中断模式写使能位 0：不允许写 1：允许写 MODE_WE[0]控制 MODE0 MODE_WE[1]控制 MODE1 MODE_WE[2]控制 MODE2 MODE_WE[3]控制 MODE3
15:12	MODE3	RW	0x0	MODE3[3:1]为 IO 端口中断模式模式设置 0：高电平中断 1：低电平中断 2：gpio_clk 上升沿中断 3：gpio_clk 下降沿中断 4：gpio_clk 沿中断 5：pclk 上升沿中断 6：pclk 下降沿中断 7：pclk 沿中断 MODE3[0]为 IO 端口中断模式使能 0：不使能中断模式 1：使能中断模式
11:8	MODE2	RW	0x0	MODE2[3:1]为 IO 端口中断模式模式设置 0：高电平中断 1：低电平中断

				2：gpio_clk 上升沿中断 3：gpio_clk 下降沿中断 4：gpio_clk 沿中断 5：pclk 上升沿中断 6：pclk 下降沿中断 7：pclk 沿中断 MODE2[0]为 IO 端口中断模式使能 0：不使能中断模式 1：使能中断模式
7:4	MDOE1	RW	0x0	MODE1[3:1]为 IO 端口中断模式模式设置 0：高电平中断 1：低电平中断 2：gpio_clk 上升沿中断 3：gpio_clk 下降沿中断 4：gpio_clk 沿中断 5：pclk 上升沿中断 6：pclk 下降沿中断 7：pclk 沿中断 MODE1[0]为 IO 端口中断模式使能 0：不使能中断模式 1：使能中断模式
3:0	MDOE0	RW	0x0	MODE0[3:1]为 IO 端口中断模式模式设置 0：高电平中断 1：低电平中断 2：gpio_clk 上升沿中断 3：gpio_clk 下降沿中断 4：gpio_clk 沿中断 5：pclk 上升沿中断 6：pclk 下降沿中断 7：pclk 沿中断 MODE0[0]为 IO 端口中断模式使能 0：不使能中断模式 1：使能中断模式

5.3.8. 去毛刺使能寄存器（GPIO_DEBOUNCEx）

- 地址偏移：
  - GPIO_DEBOUNCE0：0x180
  - GPIO_DEBOUNCE1：0x184

  GPIO Portn 的对应寄存器位为 GPIO_DEBOUNCEx[y]，其中 x=n/16，y =n%16，“/”为除法运算符，“%”为求余运算符。若对 GPIO 中断检测时间有严格要求，建议不使能去毛刺操作。

位数	名称	方向	复位值	描述
31:16	DEBEN_WE	Rw	0x0	DEBAN 写使能位 0：不允许写 1：允许写 DEBEN_WE[n]控制 DEBEN [n]， 如 DEBEN_WE[0]控制 DEBEN[0]
15:0	DEBEN	W	0x0	IO 端口去毛刺操作使能 0：不使能去毛刺操作 1：使能去毛刺操作

5.3.9. 中断原始状态寄存器（GPIO_INTR_RAW0）

- 地址偏移：
  - GPIO_INTR_RAW0：0x1a0

GPIO Portn 的对应寄存器位为 GPIO_INTR_RAWx[y]，其中 x=n/32，y =n%32，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:0	RAW	R	0x0	IO 端口中断原始状态 0：没有中断 1：有中断

5.3.10. 中断清除寄存器（GPIO_INTR_CLR0）

- 地址偏移：
  - GPIO_INTR_CLR0：0x1b0

GPIO Portn 的对应寄存器位为 GPIO_INTR_CLRx[y]，其中 x=n/32，y=n%32，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:0	CLR	W	0x0	IO 端口中断清除 0：不清除中断 1：清除中断 当中断类型为高电平中断或者为低电平中断时，此寄存器的操作无效。

5.3.11. 中断屏蔽寄存器（GPIO_INTR_MASK_C0x）

- 地址偏移：
  - GPIO_INTR_MASK_C00：0x200
  - GPIO_INTR_MASK_C01：0x204

GPIO Portn 的对应寄存器位为 GPIO_INTR_MASK_C0x[y]，其中 x=n/16，y=n%16，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:16	MASK_WE	W	0x0	MASK 写使能位。 0：不允许写 1：允许写 MASK_WE[n]控制 MASK[n]，如 MASK_WR[0]控制 MASK[0]
15:0	MASK	RW	0x0	IO 端口中断屏蔽 0：不屏蔽中断 1：屏蔽中断

5.3.12. 中断状态寄存器（GPIO_INTR_STATUS_C00）

- 地址偏移：
  - GPIO_INTR_CLR0：0x1b0

GPIO Portn 的对应寄存器位为 GPIO_INTR_STATUS_C0x[y]，其中 x=n/32，y=n%32，“/”为除法运算符，“%”为求余运算符。

位数	名称	方向	复位值	描述
31:0	STATUS	R	0x0	IO 端口中断状态 0：没有中断 1：有中断

5.4. 工作流程

本节分别给出数据输入、数据输出和中断输入三种工作模式的工作流程。

5.4.1. 数据输入

将 GPIO 某端口设置为数据输入模式流程：

1. 向端口数据方向寄存器（GPIO_PORT_DDRx，x=0~1）的相应比特位写‘0’，配置为输入。
2. 读外部端口输入寄存器（GPIO_EXT_PORT0），与端口相对应的比特位反映端口的值。

5.4.2. 数据输出

将 GPIO 某端口设置为数据输出模式流程：

1.向端口数据方向寄存器（GPIO_PORT_DDRx，x=0~1）的相应比特位写‘1’，配置为输出。
2.向端口数据寄存器（GPIO_PORT_DRx，x=0~1）的相应比特位写‘0’或‘1’，端口输出低电平或者高电平。

5.4.3. 中断输入

将 GPIO 某端口设置为中断输入模式流程：
1.向端口数据方向寄存器（GPIO_PORT_DDRx，x=0~1）的相应比特位写‘0’，配置为输入；
2.通过去毛刺使能寄存器（GPIO_DEBOUNCEx，x=0~1）选择是否使用去毛刺操作；
3.通过中断控制寄存器（GPIO_INTR_CTRLx，x=0~7）配置中断类型；
4.产生中断请求，进入中断服务程序，在退出中断服务程序之前，需要清除中断；
a)对于电平触发的中断，可以通过查询中断原始状态寄存器（GPIO_INTR_STATUS_C00）直至中断源消失，或者设置中断屏蔽寄存器（GPIO_INTR_MASK_C0x，x=0~1）屏蔽中断；
b)对于沿触发的中断，通过向中断清除寄存器（GPIO_INTR_CLR0）的相应比特写‘1’来清除中断；

第 6 章 精简直接存储器存取控制器（DMA）

6.1. 概述

直接存储器存取（direct memory access DMA）使用硬件传输数据的方法，用来提供存储器与存储器之间和外设与存储器之间的高速数据传输。无需 CPU 干预，节省了 CPU 资源。每个通道相互独立，专门管理独自通道的访问请求。

- DMA 的基地址：0x50001000

6.1.1. 主要特性

- 4 个独立的可配置通道；
- 支持外设与存储器之间的传输
- 支持存储器与存储器之间的传输
- 通道 2 和 通道 3 支持软件可配的位宽
- 可配置数据源和目标数据区的传输宽度(字节、半字、全字)
- 支持硬件优先级（优先级可配。同优先级下，通道号越低，优先级越高）
- 支持 32 Byte FIFO 深度

6.1.2. 功能描述

DMA 控制器和 CPU 核心共享系统数据总线，执行直接存储器数据传输。当 CPU 和 DMA 同时访问相同的目标（RAM 或外设）时，DMA 请求会暂停 CPU 访问系统总线若干 的周期，总线仲裁器会执行循环调度，以保证 CPU 在获得访问资源后的正常执行。

6.1.3. DMA 优先级

优先级可配，可通过 CFGx.CH_PROOR 配置通道优先级，7 为最高优先级，0 为最低优先级。优先级相同的通道，通道号越低优先级越高。

注：

当 DMA 控制器在同一时间接收到多个请求时，仲裁器将根据硬件优先级来决定响应哪一个外设请求。

如果 DMA 正在为一个低优先级的通道传输数据时，一个高优先级的通道请求产生了，那

么 DMA 将会先把低优先级通道的数据传输完毕，然后再来处理高优先级的通道要求。

6.1.4. 传输总长度和数据位宽

CTLx.寄存器中，BLOCK_TS 为传输的总长度，单位为源数据位宽(SRC_TR_WIDTH)。SRC_TR_WIDTH、DST_TR_WIDTH 分别为源数据位宽，目标数据位宽。SRC_MSIZE、DEST_MSIZE 为 Burst length。

例如：

BLOCK_TS=5.

SRC_TR_WIDTH=0x2.（4byte）

DST_TR_WIDTH=0x2.（4byte）

SRC_MSIZE=0x1.（待传输的数据项为 4）

DEST_MSIZE=0x1.（待传输的数据项为 4）

DMA 传输的总长度位 5(BLOCK_TS)*4(SRC_TR_WIDTH)=20byte。

DMA 收到一次传输请求后，将待传输的数据项 4*4byte=16byte 的数据读取到 DMAFIFO 中。再由 FIFO 搬往目标地址。

DMA 再次收到一次传输请求，将剩余的 20-16=4byte 的数据搬往目标地址。

注：SRC_TR_WIDTH*SRC_MSIZE 的数据量不能超过 DMA 通道 FIFO 深度。如果是内存与外设之间的传输，SRC_TR_WIDTH*SRC_MSIZE 需要搬运的数据量也不能超过外设 FIFO 的深度。

6.2. 信号及接口描述

DMA 接口描述如下表所示：

表 6-1 DMA 接口描述

名称	宽度	方向	描述	备注
AHB Master 总线接口				
AHB Slave 总线接口				
硬件握手信号				
dmas_req[15:0]	16	I	DMA 数据请求信号	内部信号
dmas_ack[15:0]	16	O	DMA 数据传输结束信号	内部信号
中断信号				
dmas_int	1	O	DMA 接口中断信号	内部信号

6.3. DMA 寄存器地址映射

DMA 寄存器地址映射如下表所示，这些寄存器被映射到系统的地址空间，可由 M0 进行配置。

- DMA 的基地址为：0x50001000

表 6-2 DMA 寄存器表

寄存器名	描述	方向 宽度	地址偏移	复位值
DMA_SAR[0]	通道 0 源起始地址	RW 32bits	0x00	0x0000
DMA_DAR[0]	通道 0 目的地址	RW 32bits	0x08	0x0000
DMA_CTL_L[0]	通道 0 控制寄存器 L	RW 23bits	0x18	0x0000
DMA_CTL_H[0]	通道 0 控制寄存器 H	RW 12bits	0x1C	0x0000
DMA_CFG_L[0]	通道 0 配置寄存器 L	RW 13bits	0x40	0x0000
DMA_CFG_H[0]	通道 0 配置寄存器 H	RW 16bits	0x44	0x0000
DMA_SAR[1]	通道 1 源起始地址	RW 32bits	0x58	0x0000
DMA_DAR[1]	通道 1 目的地址	RW 32bits	0x60	0x0000
DMA_CTL_L[1]	通道 1 控制寄存器 L	RW 23bits	0x70	0x0000
DMA_CTL_H[1]	通道 1 控制寄存器 H	RW 12bits	0x74	0x0000
DMA_CFG_L[1]	通道 1 配置寄存器 L	RW 13bits	0x98	0x0000
DMA_CFG_H[1]	通道 1 配置寄存器 H	RW 16bits	0x9C	0x0000
DMA_SAR[2]	通道 2 源起始地址	RW 32bits	0xB0	0x0000
DMA_DAR[2]	通道 2 目的地址	RW 32bits	0xB8	0x0000
DMA_CTL_L[2]	通道 2 控制寄存器 L	RW 23bits	0xC8	0x0000
DMA_CTL_H[2]	通道 2 控制寄存器 H	RW 12bits	0xCC	0x0000
DMA_CFG_L[2]	通道 2 配置寄存器 L	RW 13bits	0xF0	0x0000
DMA_CFG_H[2]	通道 2 配置寄存器 H	RW 16bits	0xF4	0x0000
DMA_SAR[3]	通道 3 源起始地址	RW 32bits	0x108	0x0000
DMA_DAR[3]	通道 3 目的地址	RW 32bits	0x110	0x0000
DMA_CTL_L[3]	通道 3 控制寄存器 L	RW 23bits	0x120	0x0000
DMA_CTL_H[3]	通道 3 控制寄存器 H	RW 12bits	0x124	0x0000
DMA_CFG_L[3]	通道 3 配置寄存器 L	RW 13bits	0x148	0x0000
DMA_CFG_H[3]	通道 3 配置寄存器 H	RW 16bits	0x14C	0x0000
DMA_RAW_TFR	IntTfr 中断原始寄存器	RW 2bits	0x2C0	0x0000
DMA_RAW_BLOCK	IntBlock 中断原始寄存器	RW 2bits	0x2C8	0x0000

DMA_RAW_SRC_TRAN	IntSrcTran 中断原始寄存器	RW 2bits	0x2D0	0x0000
DMA_RAW_DST_TRAN	IntDstTran 中断原始寄存器	RW 2bits	0x2D8	0x0000
DMA_RAW_ERR	IntErr 中断原始寄存器	RW 2bits	0x2E0	0x0000
DMA_STATUS_TFR	IntTfr 中断状态寄存器	RW 2bits	0x2E8	0x0000
DMA_STATUS_BLOCK	IntBlock 中断状态寄存器	RW 2bits	0x2F0	0x0000
DMA_STATUS_SRC_TRAN	IntSrcTran 中断状态寄存器	RW 2bits	0x2F8	0x0000
DMA_STATUS_DST_TRAN	IntDstTran 中断状态寄存器	RW 2bits	0x300	0x0000
DMA_STATUS_ERR	IntErr 中断状态寄存器	RW 32bits	0x308	0x0000
DMA_MASK_TFR	IntTfr 中断屏蔽寄存器	RW 11bits	0x310	0x0000
DMA_MASK_BLOCK	IntBlock 中断屏蔽寄存器	RW 10bits	0x318	0x0000
DMA_MASK_SRC_TRAN	IntSrcTran 中断屏蔽寄存器	RW 11bits	0x320	0x0000
DMA_MASK_DST_TRAN	IntDstTran 中断屏蔽寄存器	RW 10bits	0x328	0x0000
DMA_MASK_ERR	IntErr 中断屏蔽寄存器	R 11bits	0x330	0x0000
DMA_CLEAR_TFR	IntTfr 中断清除寄存器	RW 2bits	0x338	0x0000
DMA_CLEAR_BLOCK	IntBlock 中断清除寄存器	RW 2bits	0x340	0x0000
DMA_CLEAR_SRC_TRAN	IntSrcTran 中断清除寄存器	RW 2bits	0x348	0x0000
DMA_CLEAR_DST_TRAN	IntDstTran 中断清除寄存器	RW 2bits	0x350	0x0000
DMA_CLEAR_ERR	IntErr 中断清除寄存器	RW 2bits	0x358	0x0000
DMA_STATUS_INT	中断状态寄存器	RW 5bits	0x360	0x0000
DMA_REQ_SRC_REG	源软件交互请求寄存器	RW 10bits	0x368	0x0000
DMA_REQ_DST_REG	目标软件交互请求寄存器	RW 10bits	0x370	0x0000
DMA_SGL_RQ_SRC_REG	源单交互请求寄存器	RW 10bits	0x378	0x0000
DMA_SGL_RQ_DST_REG	目标单交互请求寄存器	RW 10bits	0x380	0x0000
DMA_LST_SRC_REG	源最新交互请求寄存器	RW 10bits	0x388	0x0000
DMA_LST_DST_REG	目标最新交互请求寄存器	RW 10bits	0x390	0x0000
DMA_CFG_REG	DMA 配置寄存器	RW 1bits	0x398	0x0000
DMA_CH_EN_REG	DMA 通道使能寄存器	RW 10bits	0x3A0	0x0000
DMA_ID_REG	DMA ID 寄存器	RW 32bits	0x3A8	0x0000
DMA_TEST_REG	DMA 测试寄存器	RW 1bits	0x3B0	0x0000

DMA_LP_TIME_OUT_REG	DMA 超时寄存器	RW 4bits	0x3B8	0x0000
DMA_COMP_PARAM_6_L	DMA 组件参数寄存器 6L	RW 10bits	0x3C8	0x0000
DMA_COMP_PARAM_6_H	DMA 组件参数寄存器 6H	RW 31bits	0x3CC	0x0000
DMA_COMP_PARAM_5_L	DMA 组件参数寄存器 5L	RW 31bits	0x3D0	0x0000
DMA_COMP_PARAM_5_H	DMA 组件参数寄存器 5H	RW 31bits	0x3D4	0x0000
DMA_COMP_PARAM_4_L	DMA 组件参数寄存器 4L	RW 31bits	0x3D8	0x0000
DMA_COMP_PARAM_4_H	DMA 组件参数寄存器 4H	RW 31bits	0x3DC	0x0000
DMA_COMP_PARAM_3_L	DMA 组件参数寄存器 3L	RW 31bits	0x3E0	0x0000
DMA_COMP_PARAM_3_H	DMA 组件参数寄存器 3H	RW 31bits	0x3E4	0x0000
DMA_COMP_PARAM_2_L	DMA 组件参数寄存器 2L	RW 31bits	0x3E8	0x0000
DMA_COMP_PARAM_2_H	DMA 组件参数寄存器 2H	RW 32bits	0x3EC	0x0000
DMA_COMP_PARAM_1_L	DMA 组件参数寄存器 1L	RW 32bits	0x3F0	0x0000
DMA_COMP_PARAM_1_H	DMA 组件参数寄存器 1H	RW 30bits	0x3F4	0x0000
DMA_VER_ID_L	DMA 组件 ID 寄存器 L	RW 32bits	0x3F8	0x0000
DMA_VER_ID_H	DMA 组件 ID 寄存器 H	RW 32bits	0x3FC	0x0000

6.4. DMA 寄存器

6.4.1. 通道目的地址寄存器（DMA_SAR）

- 地址偏移：
  - DMA_SAR[0]: 0x00
  - DMA_SAR[1]: 0x58
  - DMA_SAR[2]: 0xB0
  - DMA_SAR[3]: 0x108

位数	名称	方向	复位值	描述
31：0	SAR	RW	0x0	当前 DMA 传输的源地址。每次源传输后更新。CTLx 寄存器中的 SINC 字段决定地址在通过块传输的每次源传输中是增加、减少还是保持不变。

6.4.2. 通道清除寄存器（DMA_DAR）

- 地址偏移：
  - DMA_DAR[0]:0x08
  - DMA_DAR[1]:0x60
  - DMA_DAR[2]:0xB8
  - DMA_DAR[3]:0x110

位数	名称	方向	复位值	描述
32：0	DAR	RW	0x0	DMA 传输的当前目的地址。每次目的地转移后更新。CTLx 寄存器中的 DINC 字段决定在整个块传输过程中，地址在每个目标传输中是增加、减少还是保持不变。

6.4.3. 通道控制寄存器 L（DMA_CTL_L）

- 地址偏移：
  - DMA_CTL_L[0]:0x18
  - DMA_CTL_L[1]:0x70
  - DMA_CTL_L[2]:0xC8
  - DMA_CTL_L[3]:0x120

位数	名称	方向	复位值	描述
22:20	TT_FC	Rw	0x0	传输类型和流量控制；
16:14	SRC_MSIZE	Rw	0x0	源长度： 0x0 数据项的数量是 1 0x1 数据项的数量是 4 0x2 传输数据项的数量是 8 0x3 传输数据项的数量是 16 0x4 传输数据项的数量是 32 0x5 传输数据项的数量是 64 0x6 传输数据项的数量是 128 0x7 传输的数据项数为 256
13:11	DEST_MSIZE	Rw	0x0	长度： 0x0:数据项的数量是 1 0x1 数据项的数量是 4 0x2 数据项的数量是 8 0x3 数据项的数量是 16 0x4 数据项的数量是 32 0x5 传输数据项的数量是 64 0x6 传输数据项的数量是 128 0x7 传输的数据项数为 256
10:9	SINC	Rw	0x0	源地址增量： 0x0 (SINC_0):增加源地址 0x1 (SINC_1):减少源地址 0x2 (SINC_2):源地址没有变化 0x3 (SINC_3):源地址无变化
8:7	DINC	Rw	0x0	目的地址增量。 0x0 (DINC_0):增加目的地址 0x1 (DINC_1):减少目的地址 0x2 (DINC_2):目的地址没有变化 0x3 (DINC_3):目的地址没有变化
6：4	SRC_TR_WIDTH	Rw	0x0	源传输宽度： 0x0 (SRC_TR_WIDTH_0):源转移宽度 8 位

				0x1 (SRC_TR_WIDTH_1):源转移宽度 16 位 0x2 (SRC TR 宽度 2):源转移宽是 32 位 0x3 (SRC_TR_WIDTH_3):源转移宽度是 64 位 0x4 (SRC_TR_WIDTH_4):源转移宽度是 128 位 0x5 (SRC TR 宽度 5):源转移宽度是 256 位 0x6 (SRC_TR_WIDTH_6):源转移宽度是 256 位 0x7 (SRC_TR_WIDTH_7):源转移宽度是 256 位
3：1	DST_TR_WIDTH	Rw	0x0	目的传输宽度: 0x0 (DST TR WIDTH 0):目的传输宽度为 8 位 0x1 (DST_TR WIDTH_1):目的传输宽度为 16 位 0x2 (DST TR WIDTH 2):目的传输宽度为 32 位 0x3 (DST_TR_WIDTH_3):目的传输宽度为 64 位 0x4 (DST TR WIDTH 4):目的传输宽度为 128 位 0x5 (DST_TR_WIDTH_5):目标传输宽度为 256 位 0x6 (DST TR WIDTH 6):目的传输宽度为 256 位 0x7 (DST_TR_WIDTH_7):目标传输宽度为 256 位
0	INT_EN	Rw	0x0	中断使能位。 0x0 关闭中断 0x1 中断开启

6.4.4. 通道控制寄存器 H（DMA_CTL_H）

- 地址偏移：
  - DMA_CTL_H[0]:0x1C
  - DMA_CTL_H[1]:0x74
  - DMA_CTL_H[2]:0xCC
  - DMA_CTL_H[3]:0x124

位数	名称	方	复位	描述
11:0	BLOCK_TS	RW	0x0	块传输大小。 当 DW_ahb_dmac 是流控制器时，用户在通道启用之前写入该字段，以指示块大小。

6.4.5. 通道配置寄存器 L（DMA_CFG_L）

- 地址偏移：
  - DMA_CFG_L[0]:0x40
  - DMA_CFG_L[1]:0x98
  - DMA_CFG_L[2]:0xF0
  - DMA_CFG_L[3]:0x148

位数	名称	方向	复位值	描述
19	SRC_HS_POL	RW	0x0	源握手接口极性： 0x0:源握手接口极性为高 0x1:源握手接口极性为低
18	DST_HS_POL	RW	0x0	目的握手接口极性。 0x0：目的握手接口极性为高 0x1：目的握手接口极性为低
11	HS_SEL_SRC	RW	0x0	源软件或硬件握手选择。该寄存器选择哪个握手接口(硬件或软件)对该通道上的源请求是活动的。如果来源外设是内存，那么这个位被忽略。 0x0:硬件握手接口。 0x1:软件握手接口。
10	HS_SEL_DST	RW	0x0	目标软件或硬件握手选择。此寄存器选择哪个握手接口(硬件或软件)对该通道上的目标请求是活动的。如果目标外设是内存，那么这个位将被忽略。 0x0:硬件握手接口。 0x1:软件握手接口。

9	FIFO_EMPTY	RW	0x0	通道 FIFO 状态。指示通道 FIFO 中是否有数据剩余: 0x0：通道 FIFO 不为空 0x1：通道 FIFO 为空
8	CH_SUSP	RW	0x0	频道挂起。挂起所有来自源的 DMA 数据传输，直到清除此位。
7：5	CH_PRIOR	RW	0x0	通道优先级。优先级为 7 的优先级最高，0 的优先级最低。该字段必须在 0 到 DMAH_NUM_CHANNELS-1 的范围内。

6.4.6. 通道配置寄存器 H（DMA_CFG_H）

- 地址偏移：
  - DMA_CFG_H[0]:0x44
  - DMA_CFG_H[1]:0x9C
  - DMA_CFG_H[2]:0xF4
  - DMA_CFG_H[3]:0x14c

位数	名称	方向	复位值	描述
14：11	DEST_PER	RW	0x0	目的硬件接口。 分配一个硬件握手接口(0: DMAH_NUM_HS_INT-1)到通道 x 的目标，如果 CFGx。HS_SEL_DST 字段为 0;否则，该字段将被忽略。然后，通道可以通过指定的硬件握手接口与连接到该接口的目标外设通信。
10：7	SRC_PER	RW	0x0	源硬件接口 分配一个硬件握手接口(0: DMAH_NUM_HS_INT-1)到通道 x 的源，如果 CFGx。HS_SEL_SRC 字段为 0;否则，该字段将被忽略。然后，通道可以通过指定的硬件握手接口与连接到该接口的源外设通信。
4：2	PROTCTL	RW	0x0	保护用于驱动 AHB HPROT[3:1]总线的控制位。AMBA 规范建议 HPROT 的默认值表示非缓存、非缓冲的特权数据访问。重置值用于指示这种访问。

1	FIFO_MODE	RW	0x0	FIFO 模式选择。确定在突发事务请求得到服务之前，FIFO 中需要有多少空间或数据可用。 0x0 :用于指定传输宽度的单个 AHB 传输的数据； 0x1:目的传输的可用数据大于等于 FIFO 深度的一半，源传输的可用空间大于 FIFO 深度的一半。
0	FCMODE	RW	0x0	流量控制模式。 0x0 (FCMODE_0):源事务请求发生时提供服务。启用数据预取功能 0x1 (FCMODE 1):源事务请求直到目标事务请求发生时才得到服务。在这种模式下，从源传输的数据量受到限制，从而保证在目的端终止块之前将数据传输到目的端。禁用数据预取。

6.4.7. DMA_RAW_TFR

- 地址偏移：0x2C0

位数	名称	方向	复位值	描述
1：0	RAW	RW	0x0	intfr 中断的原始状态: 0x0 (INACTIVE) 0x1 (ACTIVE)

6.4.8. DMA_RAW_BLOCK

- 地址偏移：0x2C8

位数	名称	方向	复位值	描述
1：0	RawBlock	RW	0x0	IntBlock 中断的原始状态

6.4.9. DMA_RAW_SRC_TRAN

- 地址偏移：0x2D0

位数	名称	方向	复位值	描述
1:0	RawSrcTranRAW	RW	0x0	IntBlock 中断的原始状态

6.4.10. DMA_RAW_DST_TRAN

- 地址偏移：0x2D8

位数	名称	方向	复位值	描述
1：0	RawDstTranRAW	RW	0x0	IntDstTran 中断的原始状态

6.4.11. DMA_RAW_ERR

- 地址偏移：0x2E0

位数	名称	方向	复位值	描述
1：0	RawErrRAW	RW	0x0	IntErr 中断的原始状态

6.4.12. DMA_STATUS_TFR

- 地址偏移：0x2E8

位数	名称	方向	复位值	描述
1：0	StatusTfr	RW	0x0	IntErr 中断状态

6.4.13. DMA_STATUS_BLOCK

- 地址偏移：0x2F0

位数	名称	方向	复位值	描述
1：0	StatusBlock	RW	0x0	IntBlock 中断状态

6.4.14. DMA_STATUS_SRC_TRAN

- 地址偏移：0x2F8

位数	名称	方向	复位值	描述
1：0	StatusSrcTran	RW	0x0	IntSrcTran 中断状态

6.4.15. DMA_STATUS_DST_TRAN

- 地址偏移：0x300

位数	名称	方向	复位值	描述
1：0	StatusDstTran	RW	0x0	IntDstTran 中断状态

6.4.16. DMA_STATUS_ERR

- 地址偏移：0x308

位数	名称	方向	复位值	描述
1：0	StatusErr	RW	0x0	IntErr 中断状态

6.4.17. DMA_MASK_TFR

- 地址偏移：0x310

位数	名称	方向	复位值	描述
9:8	INT_MASK_WE	W	0x0	中断屏蔽写使能
1：0	INT_MASK	RW	0x0	IntTfr 中断的屏蔽

6.4.18. DMA_MASK_BLOCK

- 地址偏移：0x318

位数	名称	方向	复位值	描述
9:8	INT_MASK_WEBLOCK	W	0x0	中断屏蔽写使能
1：0	INT_MASKBLOCK	RW	0x0	IntBlock 中断的屏蔽

6.4.19. DMA_MASK_SRC_TRAN

- 地址偏移：0x320

位数	名称	方向	复位值	描述
9:8	INT_MASK_WETran	W	0x0	中断屏蔽写使能
1：0	INT_MASKTran	RW	0x0	IntSrcTran 中断的屏蔽

6.4.20. DMA_MASK_DST_TRAN

- 地址偏移：0x328

位数	名称	方向	复位值	描述
9:8	INT_MASK_WEDT	W	0x0	中断屏蔽写使能
1：0	INT_MASKDT	RW	0x0	IntDstTran 中断的屏蔽

6.4.21. DMA_MASK_ERR

- 地址偏移：0x330

位数	名称	方向	复位值	描述
9:8	INT_MASK_WEERR	W	0x0	中断屏蔽写使能
1：0	INT_MASKERR	RW	0x0	IntErr 中断的屏蔽

6.4.22. DMA_CLEAR_TFR

- 地址偏移：0x338

位数	名称	方向	复位值	描述
1：0	ClearTfr	RW	0x0	IntTfr 中断清除

6.4.23. DMA_CLEAR_BLOCK

- 地址偏移：0x340

位数	名称	方向	复位值	描述
1：0	ClearBlock	RW	0x0	IntBlock 中断清除

6.4.24. DMA_CLEAR_SRC_TRAN

- 地址偏移：0x348

位数	名称	方向	复位值	描述
1：0	ClearSrcTran	RW	0x0	IntSrcTran 中断清除

6.4.25. DMA_CLEAR_DST_TRAN

- 地址偏移：0x350

位数	名称	方向	复位值	描述
1：0	ClearDstTran	RW	0x0	IntDstTran 中断清除

6.4.26. DMA_CLEAR_ERR

- 地址偏移：0x358

位数	名称	方向	复位值	描述
1：0	ClearErr	RW	0x0	IntErr 中断清除

6.4.27. DMA_STATUS_INT

- 地址偏移：0x360

位数	名称	方向	复位值	描述
4	ERR	R	0x0	StatusErr 寄存器的内容
3	DSTT	R	0x0	statussttran 寄存器的内容
2	SRCT	R	0x0	StatusSrcTran 寄存器的内容
1	BLOCK	R	0x0	StatusBlock 寄存器的内容
0	TFR	R	0x0	或 StatusTfr 寄存器的内容

6.4.28. DMA_REQ_SRC_REG

- 地址偏移：0x368

位数	名称	方向	复位值	描述
9:8	SRC_REQ_WE	W	0x0	源软件交互请求写使能
1：0	SRC_REQ	RW	0x0	源软件交互请求

6.4.29. DMA_REQ_DST_REG

- 地址偏移：0x370

位数	名称	方向	复位值	描述
9:8	DST_REQ_WE	RW	0x0	目标软件交互请求写使能
1：0	DST_REQ	RW	0x0	目标软件交互请求

6.4.30. DMA_SGL_RQ_SRC_REG

- 地址偏移：0x378

位数	名称	方向	复位值	描述
9:8	SRC_SGLREQ_WE	RW	0x0	源单交互请求写使能
1：0	SRC_SGLREQ	RW	0x0	源单交互请求

6.4.31. DMA_SGL_RQ_DST_REG

- 地址偏移：0x380

位数	名称	方向	复位值	描述
9:8	DST_SGLREQ_WE	RW	0x0	目标单交互请求写使能
1：0	DST_SGLREQ	RW	0x0	目标单交互请求

6.4.32. DMA_LST_SRC_REG

- 地址偏移：0x388

位数	名称	方向	复位值	描述
9:8	LSTSRC_WEQ_WE	RW	0x0	源最后交互请求写使能
1：0	LSTSRC	RW	0x0	源最后交互请求

6.4.33. DMA_LST_DST_REG

- 地址偏移：0x390

位数	名称	方向	复位值	描述
9:8	LSTDST_WE	RW	0x0	目标最后交互请求写使能
1：0	LSTDST	RW	0x0	目标最后交互请求

6.4.34. DMA_CFG_REG

- 地址偏移：0x398

位数	名称	方向	复位值	描述
1	DMA_EN	RW	0x0	dma 使能

6.4.35. DMA_CH_EN_REG

- 地址偏移：0x3A0

位数	名称	方向	复位值	描述
9:8	CH_EN_WE	RW	0x0	通道使能
1：0	CH_EN	RW	0x0	通道使能

6.4.36. DMA_ID_REG

- 地址偏移：0x3A8

位数	名称	方向	复位值	描述
31：0	DMA_ID	RW	0x0	硬件编码 dma 外设 ID

6.4.37. DMA_TEST_REG

- 地址偏移：0x3B0

位数	名称	方向	复位值	描述
0	TEST_SLV_IF	RW	0x0	DMA 测试寄存器

6.4.38. DMA_LP_TIME_OUT_REG

- 地址偏移：0x3B8

位数	名称	方向	复位值	描述
3：0	DMA_LP_TIMEOUT	RW	0x0	低功耗计数器寄存器的超时值

6.4.39. DMA_COMP_PARAM_6_L

- 地址偏移：0x3C8

位数	名称	方向	复位值	描述
9:8	LSTDST_WE	RW	0x0	--
1：0	LSTDST	RW	0x0	--

6.4.40. DMA_COMP_PARAM_6_H

- 地址偏移：0x3CC

位数	名称	方向	复位值	描述
30：28	CH7_FIFO_DEPTH	R	0x0	DMAH_CH7_FIFO_DEPTH 参数
27:25	CH7_SMS	R	0x0	DMAH_CH7_SMS 参数
24：22	CH7_LMS	R	0x0	DMAH_CH7_LMS 参数
21:19	CH7_DMS	R	0x0	DMAH_CH7_DMS 参数
18:16	CH7_MAX_MULT_SIZE	R	0x0	DMAH_CH7_MULT_SIZE 参数
15:14	CH7_FC	R	0x0	DMAH_CH7_FC 参数
13	CH7_HC_LLP	R	0x0	DMAH_CH7_HC_LLP 参数

12	CH7_CTL_WB_EN	R	0x0	DMAH_CH7_CTL_WB_EN 参数
11	CH7_MULTI_BLK_EN	R	0x0	DMAH_CH7_MULTI_BLK_EN 参数
10	CH7_LOCK_EN	R	0x0	DMAH_CH7_LOCK_EN 参数
9	CH7_SRC_GAT_EN	R	0x0	DMAH_CH7_SRC_GAT_EN 参数
8	CH7_DST_SCA_EN	R	0x0	DMAH_CH7_DST_SCA_EN 参数
7	CH7_STAT_SRC	R	0x0	DMAH_CH7_STAT_SRC 参数
6	CH7_STAT_DST	R	0x0	DMAH_CH7_STAT_DST 参数
5:3	CH7_STW	R	0x0	DMAH_CH7_STW 参数
2:0	CH7_DTW	R	0x0	DMAH_CH7_DTW 参数。

6.4.41. DMA_COMP_PARAM_5_L

- 地址偏移：0x3D0

位数	名称	方向	复位值	描述
30：28	CH6_FIFO_DEPTH	R	0x0	DMAH_CH6_FIFO_DEPTH 参数
27:25	CH6_SMS	R	0x0	DMAH_CH6_SMS 参数
24：22	CH6_LMS	R	0x0	DMAH_CH6_LMS 参数
21:19	CH6_DMS	R	0x0	DMAH_CH6_DMS 参数
18:16	CH6_MAX_MULT_SIZE	R	0x0	DMAH_CH6_MULT_SIZE 参数
15:14	CH6_FC	R	0x0	DMAH_CH6_FC 参数
13	CH6_HC_LLP	R	0x0	DMAH_CH6_HC_LLP 参数
12	CH6_CTL_WB_EN	R	0x0	DMAH_CH6_CTL_WB_EN 参数
11	CH6_MULTI_BLK_EN	R	0x0	DMAH_CH6_MULTI_BLK_EN 参数
10	CH6_LOCK_EN	R	0x0	DMAH_CH6_LOCK_EN 参数
9	CH6_SRC_GAT_EN	R	0x0	DMAH_CH6_SRC_GAT_EN 参数
8	CH6_DST_SCA_EN	R	0x0	DMAH_CH6_DST_SCA_EN 参数
7	CH6_STAT_SRC	R	0x0	DMAH_CH6_STAT_SRC 参数
6	CH6_STAT_DST	R	0x0	DMAH_CH6_STAT_DST 参数
5:3	CH6_STW	R	0x0	DMAH_CH6_STW 参数
2:0	CH6_DTW	R	0x0	DMAH_CH6_DTW 参数。

6.4.42. DMA_COMP_PARAM_5_H

- 地址偏移：0x3D4

位数	名称	方向	复位值	描述
30：28	CH5_FIFO_DEPTH	R	0x0	DMAH_CH5_FIFO_DEPTH 参数
27:25	CH5_SMS	R	0x0	DMAH_CH5_SMS 参数
24：22	CH5_LMS	R	0x0	DMAH_CH5_LMS 参数
21:19	CH5_DMS	R	0x0	DMAH_CH5_DMS 参数
18:16	CH5_MAX_MULT_SIZE	R	0x0	DMAH_CH5_MULT_SIZE 参数
15:14	CH5_FC	R	0x0	DMAH_CH5_FC 参数
13	CH5_HC_LLP	R	0x0	DMAH_CH5_HC_LLP 参数
12	CH5_CTL_WB_EN	R	0x0	DMAH_CH5_CTL_WB_EN 参数
11	CH5_MULTI_BLK_EN	R	0x0	DMAH_CH5_MULTI_BLK_EN 参数
10	CH5_LOCK_EN	R	0x0	DMAH_CH5_LOCK_EN 参数
9	CH5_SRC_GAT_EN	R	0x0	DMAH_CH5_SRC_GAT_EN 参数
8	CH5_DST_SCA_EN	R	0x0	DMAH_CH5_DST_SCA_EN 参数
7	CH5_STAT_SRC	R	0x0	DMAH_CH5_STAT_SRC 参数
6	CH5_STAT_DST	R	0x0	DMAH_CH5_STAT_DST 参数
5:3	CH5_STW	R	0x0	DMAH_CH5_STW 参数
2:0	CH5_DTW	R	0x0	DMAH_CH5_DTW 参数。

6.4.43. DMA_COMP_PARAM_4_L

- 地址偏移：0x3D8

位数	名称	方向	复位值	描述
30：28	CH4_FIFO_DEPTH	R	0x0	DMAH_CH4_FIFO_DEPTH 参数
27:25	CH4_SMS	R	0x0	DMAH_CH4_SMS 参数
24：22	CH4_LMS	R	0x0	DMAH_CH4_LMS 参数
21:19	CH4_DMS	R	0x0	DMAH_CH4_DMS 参数
18:16	CH4_MAX_MULT_SIZE	R	0x0	DMAH_CH4_MULT_SIZE 参数
15:14	CH4_FC	R	0x0	DMAH_CH4_FC 参数

13	CH4_HC_LLP	R	0x0	DMAH_CH4_HC_LLP 参数
12	CH4_CTL_WB_EN	R	0x0	DMAH_CH4_CTL_WB_EN 参数
11	CH4_MULTI_BLK_EN	R	0x0	DMAH_CH4_MULTI_BLK_EN 参数
10	CH4_LOCK_EN	R	0x0	DMAH_CH4_LOCK_EN 参数
9	CH4_SRC_GAT_EN	R	0x0	DMAH_CH4_SRC_GAT_EN 参数
8	CH4_DST_SCA_EN	R	0x0	DMAH_CH4_DST_SCA_EN 参数
7	CH4_STAT_SRC	R	0x0	DMAH_CH4_STAT_SRC 参数
6	CH4_STAT_DST	R	0x0	DMAH_CH4_STAT_DST 参数
5:3	CH4_STW	R	0x0	DMAH_CH4_STW 参数
2:0	CH4_DTW	R	0x0	DMAH_CH4_DTW 参数。

6.4.44. DMA_COMP_PARAM_4_H

- 地址偏移：0x3DC

位数	名称	方向	复位值	描述
30：28	CH3_FIFO_DEPTH	R	0x0	DMAH_CH3_FIFO_DEPTH 参数
27:25	CH3_SMS	R	0x0	DMAH_CH3_SMS 参数
24：22	CH3_LMS	R	0x0	DMAH_CH3_LMS 参数
21:19	CH3_DMS	R	0x0	DMAH_CH3_DMS 参数
18:16	CH3_MAX_MULT_SIZE	R	0x0	DMAH_CH3_MULT_SIZE 参数
15:14	CH3_FC	R	0x0	DMAH_CH3_FC 参数
13	CH3_HC_LLP	R	0x0	DMAH_CH3_HC_LLP 参数
12	CH3_CTL_WB_EN	R	0x0	DMAH_CH3_CTL_WB_EN 参数
11	CH3_MULTI_BLK_EN	R	0x0	DMAH_CH3_MULTI_BLK_EN 参数
10	CH3_LOCK_EN	R	0x0	DMAH_CH3_LOCK_EN 参数
9	CH3_SRC_GAT_EN	R	0x0	DMAH_CH3_SRC_GAT_EN 参数
8	CH3_DST_SCA_EN	R	0x0	DMAH_CH3_DST_SCA_EN 参数
7	CH3_STAT_SRC	R	0x0	DMAH_CH3_STAT_SRC 参数
6	CH3_STAT_DST	R	0x0	DMAH_CH3_STAT_DST 参数
5:3	CH3_STW	R	0x0	DMAH_CH3_STW 参数
2:0	CH3_DTW	R	0x0	DMAH_CH3_DTW 参数。

6.4.45. DMA_COMP_PARAM_3_L

- 地址偏移：0x3E0

位数	名称	方向	复位值	描述
30：28	CH2_FIFO_DEPTH	R	0x0	DMAH_CH2_FIFO_DEPTH 参数
27:25	CH2_SMS	R	0x0	DMAH_CH2_SMS 参数
24：22	CH2_LMS	R	0x0	DMAH_CH2_LMS 参数
21:19	CH2_DMS	R	0x0	DMAH_CH2_DMS 参数
18:16	CH2_MAX_MULT_SIZE	R	0x0	DMAH_CH2_MULT_SIZE 参数
15:14	CH2_FC	R	0x0	DMAH_CH2_FC 参数
13	CH2_HC_LLP	R	0x0	DMAH_CH2_HC_LLP 参数
12	CH2_CTL_WB_EN	R	0x0	DMAH_CH2_CTL_WB_EN 参数
11	CH2_MULTI_BLK_EN	R	0x0	DMAH_CH2_MULTI_BLK_EN 参数
10	CH2_LOCK_EN	R	0x0	DMAH_CH2_LOCK_EN 参数
9	CH2_SRC_GAT_EN	R	0x0	DMAH_CH2_SRC_GAT_EN 参数
8	CH2_DST_SCA_EN	R	0x0	DMAH_CH2_DST_SCA_EN 参数
7	CH2_STAT_SRC	R	0x0	DMAH_CH2_STAT_SRC 参数
6	CH2_STAT_DST	R	0x0	DMAH_CH2_STAT_DST 参数
5:3	CH2_STW	R	0x0	DMAH_CH2_STW 参数
2:0	CH2_DTW	R	0x0	DMAH_CH2_DTW 参数。

6.4.46. DMA_COMP_PARAM_3_H

- 地址偏移：0x3E4

位数	名称	方向	复位值	描述
30：28	CH1_FIFO_DEPTH	R	0x0	DMAH_CH1_FIFO_DEPTH 参数
27:25	CH1_SMS	R	0x0	DMAH_CH1_SMS 参数
24：22	CH1_LMS	R	0x0	DMAH_CH1_LMS 参数
21:19	CH1_DMS	R	0x0	DMAH_CH1_DMS 参数
18:16	CH1_MAX_MULT_SIZE	R	0x0	DMAH_CH1_MULT_SIZE 参数
15:14	CH1_FC	R	0x0	DMAH_CH1_FC 参数

13	CH1_HC_LLP	R	0x0	DMAH_CH1_HC_LLP 参数
12	CH1_CTL_WB_EN	R	0x0	DMAH_CH1_CTL_WB_EN 参数
11	CH1_MULTI_BLK_EN	R	0x0	DMAH_CH1_MULTI_BLK_EN 参数
10	CH1_LOCK_EN	R	0x0	DMAH_CH1_LOCK_EN 参数
9	CH1_SRC_GAT_EN	R	0x0	DMAH_CH1_SRC_GAT_EN 参数
8	CH1_DST_SCA_EN	R	0x0	DMAH_CH1_DST_SCA_EN 参数
7	CH1_STAT_SRC	R	0x0	DMAH_CH1_STAT_SRC 参数
6	CH1_STAT_DST	R	0x0	DMAH_CH1_STAT_DST 参数
5:3	CH1_STW	R	0x0	DMAH_CH1_STW 参数
2:0	CH1_DTW	R	0x0	DMAH_CH1_DTW 参数。

6.4.47. DMA_COMP_PARAM_2_L

- 地址偏移：0x3E8

位数	名称	方向	复位值	描述
30：28	CH0_FIFO_DEPTH	R	0x0	DMAH_CH0_FIFO_DEPTH 参数
27:25	CH0_SMS	R	0x0	DMAH_CH0_SMS 参数
24：22	CH0_LMS	R	0x0	DMAH_CH0_LMS 参数
21:19	CH0_DMS	R	0x0	DMAH_CH0_DMS 参数
18:16	CH0_MAX_MULT_SIZE	R	0x0	DMAH_CH0_MULT_SIZE 参数
15:14	CH0_FC	R	0x0	DMAH_CH0_FC 参数
13	CH0_HC_LLP	R	0x0	DMAH_CH0_HC_LLP 参数
12	CH0_CTL_WB_EN	R	0x0	DMAH_CH0_CTL_WB_EN 参数
11	CH0_MULTI_BLK_EN	R	0x0	DMAH_CH0_MULTI_BLK_EN 参数
10	CH0_LOCK_EN	R	0x0	DMAH_CH0_LOCK_EN 参数
9	CH0_SRC_GAT_EN	R	0x0	DMAH_CH0_SRC_GAT_EN 参数
8	CH0_DST_SCA_EN	R	0x0	DMAH_CH0_DST_SCA_EN 参数
7	CH0_STAT_SRC	R	0x0	DMAH_CH0_STAT_SRC 参数
6	CH0_STAT_DST	R	0x0	DMAH_CH0_STAT_DST 参数
5:3	CH0_STW	R	0x0	DMAH_CH0_STW 参数
2:0	CH0_DTW	R	0x0	DMAH_CH0_DTW 参数。

注：参数值如下

0x0 (TRANS_WIDTH_PROGRAMMABLE):Programmable

0x1 (TRANS WIDTH 8):硬码通道 2 的目标传输宽度到 8 位

0x2 (TRANS_WIDTH_16):硬码通道 2 的目标传输宽度到 16 位

0x3 (TRANS WIDTH 32):硬码通道 2 的目标传输宽度到 32 位

0x4 (TRANS WIDTH 64):硬码通道 2 的目标传输宽度到 64 位

0x5 (TRANS WIDTH_128):硬码通道 2 的目标传输宽度到 128 位

0x6 (TRANS_WIDTH_256):硬码通道 2 的目的传输宽度为 256 位

0x7 (RESERVED):保留

6.4.48. DMA_COMP_PARAM_2_H

- 地址偏移：0x3EC

位数	名称	方向	复位值	描述
31:28	CH7_MULTI_BLK_TYPE	R	0x0	DMAH_CH7_MULTI_BLK_TYPE 参数
27:24	CH6_MULTI_BLK_TYPE	R	0x0	DMAH_CH6_MULTI_BLK_TYPE 参数
23:20	CH5_MULTI_BLK_TYPE	R	0x0	DMAH_CH5_MULTI_BLK_TYPE 参数
19:16	CH4_MULTI_BLK_TYPE	R	0x0	DMAH_CH4_MULTI_BLK_TYPE 参数
15:12	CH3_MULTI_BLK_TYPE	R	0x0	DMAH_CH3_MULTI_BLK_TYPE 参数
11:8	CH2_MULTI_BLK_TYPE	R	0x0	DMAH_CH2_MULTI_BLK_TYPE 参数
7:4	CH1_MULTI_BLK_TYPE	R	0x0	DMAH_CH1_MULTI_BLK_TYPE 参数
3:0	CH0_MULTI_BLK_TYPE	R	0x0	DMAH_CH0_MULTI_BLK_TYPE 参数

注：参数值如下，其中 x=[0-7]

0x0 (PROGRAMMABLE):允许所有类型的多支持

0x1 (CONT_RELOAD):只允许 SARx 连续的多块传输;DAR 和 CTL 从它们的初始值重新加载;

0x2 (RELOAD CONT):只允许多块传输，其中 SARx 和 CTL 从它们的初始值重新加载;

0x3 (RELOAD_RELOAD):只允许多块传输，其中 SARx，DARx 和 CTLx 从其初始值重新加载;

0x4 (CONT_LLP):只允许 SARx 连续的多块传输;DARx、CTLx 和 LLPx 从下一个链表项加载;

0x5 (RELOAD_LLP):只允许从初始值重新加载 SARO 的多块传输;DARx、CTLx 和 LLPx 是从下一个链表项加载的;

0x6 (CNT_LLP):只允许从下一个链表项加载 SARx，CTLx 和 LLPx 的多块传输;DAR 是连续的;

0x7 (LLP RELOAD):只允许从下一个链表项加载 SARx、CTLx 和 LLPx 的多块传输;DAR 从其初始值重新加载;

0x8 (LLP_LLP):只允许从下一个链表项加载 SARx、DARx、CTLx 和 LLPx 的多块传输。

6.4.49. DMA_COMP_PARAM_1_L

- 地址偏移：0x3F0

位数	名称	方向	复位值	描述
31:28	CH7_MAX_BLK_SIZE	R	0x0	DMAH_CH7_MAX_BLK_SIZE 参数
27:24	CH6_MAX_BLK_SIZE	R	0x0	DMAH_CH6_MAX_BLK_SIZE 参数
23:20	CH5_MAX_BLK_SIZE	R	0x0	DMAH_CH5_MAX_BLK_SIZE 参数
19:16	CH4_MAX_BLK_SIZE	R	0x0	DMAH_CH4_MAX_BLK_SIZE 参数
15:12	CH3_MAX_BLK_SIZE	R	0x0	DMAH_CH3_MAX_BLK_SIZE 参数
11:8	CH2_MAX_BLK_SIZE	R	0x0	DMAH_CH2_MAX_BLK_SIZE 参数
7:4	CH1_MAX_BLK_SIZE	R	0x0	DMAH_CH1_MAX_BLK_SIZE 参数
3:0	CH0_MAX_BLK_SIZE	R	0x0	DMAH_CH0_MAX_BLK_SIZE 参数

注：参数值如下

0x0 : (MAX_BLOCK_SIZE 3)
0x1 : (MAX_BLOCK_SIZE 7)
0x2 : (MAX_BLOCK_SIZE 15)
0x3 : (MAX_BLOCK_SIZE 31)
0x4 : (MAX_BLOCK_SIZE 63)
0x5 : (MAX_BLOCK_SIZE 127)
0x6 : (MAX_BLOCK_SIZE 255)
0x7 : (MAX_BLOCK_SIZE 511)
0x8 : (MAX_BLOCK_SIZE 1023)
0x9 : (MAX_BLOCK_SIZE 2047)
0xa : (MAX_BLOCK_SIZE 4095)

6.4.50. DMA_COMP_PARAM_1_H

- 地址偏移：0x3F4

位数	名称	方向	复位值	描述
29	STATIC_ENDIAN_SELECT	R	0x0	DMAH_STATIC_ENDIAN_SELECT 参数
28	ADD_ENCODED_PARAMS	R	0x0	0x0 (FALSE):未启用添加编码参数 0x1 (TRUE):启用添加编码参数 0x0 (STATIC_ENDIAN_FALSE):没有通过 coreConsultant 配置 Endianness 0x1 (STATIC_ENDIAN_TRUE): 通过

				coreConsultant 静态配置 Endianness
27:23	NUM_HS_INT	R	0x0	握手接口数 0x0-0x10
22:21	M1_HDATA_WIDTH	R	0x0	0x0 DATA BUS WIDTH 32 0x1 DATA_BUS_WIDTH_64 0x2 DATA_BUS_WIDTH_128 0x3 DATA_BUS_WIDTH_256
20:19	M2_HDATA_WIDTH	R	0x0	0x0 DATA BUS WIDTH 32 0x1 DATA_BUS_WIDTH_64 0x2 DATA_BUS_WIDTH_128 0x3 DATA_BUS_WIDTH_256
18:17	M3_HDATA_WIDTH	R	0x0	0x0 DATA BUS WIDTH 32 0x1 DATA_BUS_WIDTH_64 0x2 DATA_BUS_WIDTH_128 0x3 DATA_BUS_WIDTH_256
16:15	M4_HDATA_WIDTH	R	0x0	0x0 DATA BUS WIDTH 32 0x1 DATA_BUS_WIDTH_64 0x2 DATA_BUS_WIDTH_128 0x3 DATA_BUS_WIDTH_256
14:13	S_HDATA_WIDTH	R	0x0	0x0 DATA BUS WIDTH 32 0x1 DATA_BUS_WIDTH_64 0x2 DATA_BUS_WIDTH_128 0x3 DATA_BUS_WIDTH_256
12:11	NUM_MASTER_INT	R	0x0	0x0 NUM_MST_INTERFACE_1 0x1 NUM_MST_INTERFACE_2 0x2 NUM_MST_INTERFACE_3 0x3 NUM_MST_INTERFACE_4

10:8	NUM_CHANNELS	R	0x0	0x0 NUM_CHANNEL_1 0x1 NUM_CHANNEL_2 0x2 NUM CHANNEL 3 0x3 NUM_CHANNEL_4 0x4 NUM_CHANNEL_5 0x5 NUM CHANNEL_6 0x6 NUM_CHANNEL_7 0x7 NUM_CHANNEL_8
3	MAX_ABRST	R	0x0	0x0 (FAI SE):最大 AMBA 长度不受软件控制 0x1 (TRUE):通过写入通道配置寄存器，将最大 AMBA 长度限制为软件控制下的值
2:1	INTR_IO	R	0x0	0x0 (ALL INT):所有中断相关的信号做输出 0x1 (TYPE INT):只有 TYPE 中断相关的信号作为输出 0x2 (COMBINED_INT):只有与中断相关的组合信号作为输出 0x3 (RESERVED):预留
0	BIG_ENDIAN	RW	0x0	0x0 (FALSE):大端 0x1 (TRUE):小端

6.4.51. DMA_VER_ID_L

- 地址偏移：0x3F8

位数	名称	方向	复位值	描述
31：0	DMA_COMP_TYPE	RW	0x0	DMA 组件类型编号= h44571110

6.4.52. DMA_VER_ID_H

- 地址偏移：0x3FC

位数	名称	方向	复位值	描述
31：0	DMA_COMP_VERSION	RW	0x0	DMA 组件版本

6.5. 工作流程

6.5.1. 单 Block 传输

1. 读取通道启用寄存器以选择一个空闲(禁用)通道;

2. 通过写入中断清除寄存器:clearfr, ClearBlock, ClearSrcTran, clearsttran 和 ClearErr，清除先前 DMA 传输中通道上的任何挂起中断。读取中断原始状态和中断状态寄存器确认所有中断已被清除。

3. 配置下列通道寄存器:

a.在通道x的SARx寄存器中写入起始源地址（DMA_SAR[x]）；

b.在通道x的DARx寄存器中写入起始目的地址(DMA_DAR[x])；

C.按照第 1 步配置CTLx和CFGx，将LLPx寄存器设置为 0；

d.在通道x的CTLx寄存器中写入DMA传输的控制信息(DMA_CTL[x])。例如，在寄存器中，可以配置如下:

(1) 设置传输类型和通过配置实现流量控制设备的 TT_FC 的 CTLx 寄存器。

(2) 设置传输特性，例如:

- ✧ SRC_TR_WIDTH 字段中源的传输宽度。
- ✧ DST_TR_WIDTH 字段中目的地的传输宽度。
- ✧ 源所在的 SMS 字段中的源主层。
- ✧ 目标所在 DMS 字段中的目标主层。
- ✧ 在 SINC 字段中为源的递增/递减或固定地址。
- ✧ 在 SINC 字段中为源的递增/递减或固定地址。

e. 将通道配置信息写入通道x的CFGx寄存器（DMA_CFG[x]）。

f.启用scatter(参数CHx_DST_SCA_EN= True和CTLx.DST_SCATTER_EN)，为通道x编程DSRx寄存器(DMA_SAR[x])。

4.在写入 ChEnReg 之前，确保 DmaCfgReg 寄存器的 0 位是启用的。

5.源和目的请求单个和爆发 DMA 事件，以便传输块数据。DW_ahb_dmac 在完成时确认区块中的每个事件并执行区块传输。

6.一旦传输完成，硬件设置中断并禁用通道。此时，可以响应 Block Complete 或 Transfer Complete 中断，或者轮询传输完成原始中断状态寄存器(RawTfr[n]， n=通道号)，直到它被硬件设置，以便检测传输何时完成。注意，如果使用这种轮询，软件必须确保在通道启用之前，通过写入中断清除寄存器 ClearTfr[n]来清除传输完成中断。

6.5.2. DMA 低功耗

DMA 模块自带低功耗控制，在上电默认状态下是关闭的，可以通过配置 CPR 和 DMA 寄存器打开：

1.配置 DMA 低功耗控制寄存器 DMA_LP_CTL 为 0x11F0F。

2.配置 CPR_LP_CTL.DMA_BUS_LP_EN 为 1。这样 DMA 在空闲但有时钟的状态下功耗会小一些

第 7 章 通用异步收发器（UART）

7.1. 概述

UART 是通用异步收发控制器，用于芯片和外部设备进行数据接收和发送。用户可以对传输的字符长度、波特率和奇偶校验等进行设置。XC65xx 中集成 3 个 UART：UART0、UART1、UART2，在 CTL_APB 总线上。其中 UART0 有流控接口，UART1、UART2 上没有流控接口。3 个 UART 其他的配置都相同。可以被 DMA 访问。**UART0 和 UART1 可以映射到 gpio0-gpio35 上，UART2_RX 映射 gpio31,UART2_TX 映射 gpio32.**

7.1.1. 主要特性

UART 具备以下特点

- 支持自动流控 AFC 模式；
- 支持可编程 THRE 中断模式；
- 支持使用 FIFO，可设置 FIFO 中断阈值；
- 收发 FIFO 深度为 16，宽度为 8bits；
- 可编程通讯波特率，默认串口波特率为 115200，最大传输速率为 1Mbps；
- 可设置串行数据传输帧格式；
- 支持 DMA 硬件握手；
- 支持时钟关断保护功能；

7.1.2. 功能描述

UART 的原理框图如下图所示：

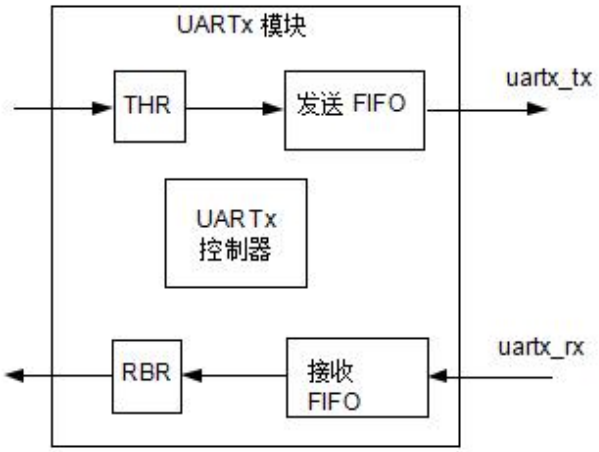

图 7-1 UARTx 结构框图

UARTx（x=0，1，2）模块包括 UARTx 控制器、接收 FIFO、发送 FIFO、RBR（接收缓存寄存器）和 THR（发送保持寄存器）。UART0、UART1 和 UART2 接收 FIFO 和发送 FIFO 深度均为 16，宽度为 8bits。接收缓存寄存器 RBR 和发送保持寄存器 THR 都是 8bits 寄存器。

用户只能访问 RBR 和 THR，不能访问收/发 FIFO。当需要获得通过 UART 串口接收到的数据时需要读 RBR，RBR 会自动从接收 FIFO 中取数；当要通过 UART 串口发送数据时需要将数据写入 THR，THR 会自动把数据送往发送 FIFO，并通过串口发送出去。

7.1.3. 波特率计算

UART 对波特率要求比较精确的模块。uartx_clk(x=0,1)使用了这种分频。使用两级分频得到分频时钟，第一级为 G8 整数分频，第二级为浮点分数分频。分频公式如下：

$$freq(clkout) = \frac{freq(clkin) * CLKx\_GR * CLKx\_MUL}{16 * CLKx\_DIV}$$

(1) clkin 表示时钟源头，clkout 表示目标时钟，

(2) 如果希望配置出占空比为 1:1 的 clkout，则 CLK_GR 允许的配置范围为 1、2、4 或 8，这种情况下占空比仅决定于 MUL 和 DIV；否则，CLK_GR 可任意配置；

(3) MUL 推荐设置为偶数，否则分数分频的结果可能不对；DIV 和 MUL 都不能置为 0。

(4) 推荐的配置顺序：

a. 配置 CLKx_GR

b. 配置 CLKx_MUL 和 CLKx_DIV 为期望值

波特率分频计算公式：

$$BaudRate = \frac{f_{uart0\_mclk}}{16 * divisor}$$

UART0 工作时钟(uart0_clk)分频示例：

浮点分频逻辑的源时钟均为 mclk_in（默认频率为 32MHz）。

当配置 UART0 的波特率为 115200 时，当 divisor 为 1 时，需要得到的 f_{uart0_mclk}=115200*16=1.8432Mhz，divisor 在 uart 寄存器中可以配置）。由此得到：

$$\frac{UART0_{MUL}}{UART0_{DIV}} = \frac{f_{uart0\_mclk} * 16}{f_{mclk\_in} * UART0\_CLK\_GR} = \frac{f_{uart0\_mclk} * 16}{f_{mclk\_in} * 8} = \frac{1.8432M * 16}{32M * 8} = \frac{72}{625}$$

最终 UART0 波特率为 115200 时的 CPR 寄存器的配置为：

- CPR_UART0_CLK_GRCTL.UART0_CLK_GR = 8
- CPR_UART0_CLK_CTL.UART0_CLK_MUL = 72
- CPR_UART0_CLK_CTL.UART0_CLK_DIV = 625

同理可以得到 UART0 波特率为 1MHz 时的 CPR 寄存器的配置为：

- CPR_UART0_CLK_GRCTL.UART0_CLK_GR = 8
- CPR_UART0_CLK_CTL.UART0_CLK_MUL = 1
- CPR_UART0_CLK_CTL.UART0_CLK_DIV = 1

7.1.4. 串口协议

uart 传输数据时的串行数据格式如图所示。

图 7-2 UARTx 串行数据帧格式

串行数据的一帧数据包括起始位、数据位、奇偶校验位和停止位。数据位可设置为 5bits~8bits，停止位可设置为 1bit、1.5bits、2bits，奇偶校验位可选择有或者没有，长度 1bit。串行数据的格式可以通过寄存器 UARTx_TCR（x=0，1，2）进行设置。

7.1.5. 工作模式

UART0 支持非流控模式和自动流控（AFC）模式这两种工作方式，UART1 和 UART2 只支持非流控模式。

7.1.5.1. 非流控模式

UARTx（x=0，1）默认工作在非流控模式。在非流控模式下 UARTx（x=0，1）用 uartx_tx 发送数据，用 uartx_rx 接收数据。整个发送和接收的过程需要软件进行控制。

7.1.5.2. 自动流控（AFC）模式

在流控模式下，UART 设备之间的数据传输过程可以由硬件自主完成，不需软件另外操作。UART0 的自动流控模式可通过寄存器 UART0_MCR 的 bit[5]AFCE 进行配置。需要注意的是使能自动流控模式之前必须先使能接收和发送 FIFO。

图 7-3 是的 UART0 接口和 UART 外设的连接框图，uart0_rx、uart0_tx、uart0_cts 和 uart0_rts 是的 UART0 脚信号，rx、tx、cts 和 rts 是芯片外部 UART 设备管脚信号。RTS 为寄存器 UART0_MCR 的 bit[1]；CTS 为寄存器 UART0_MSR 的 bit[4]。

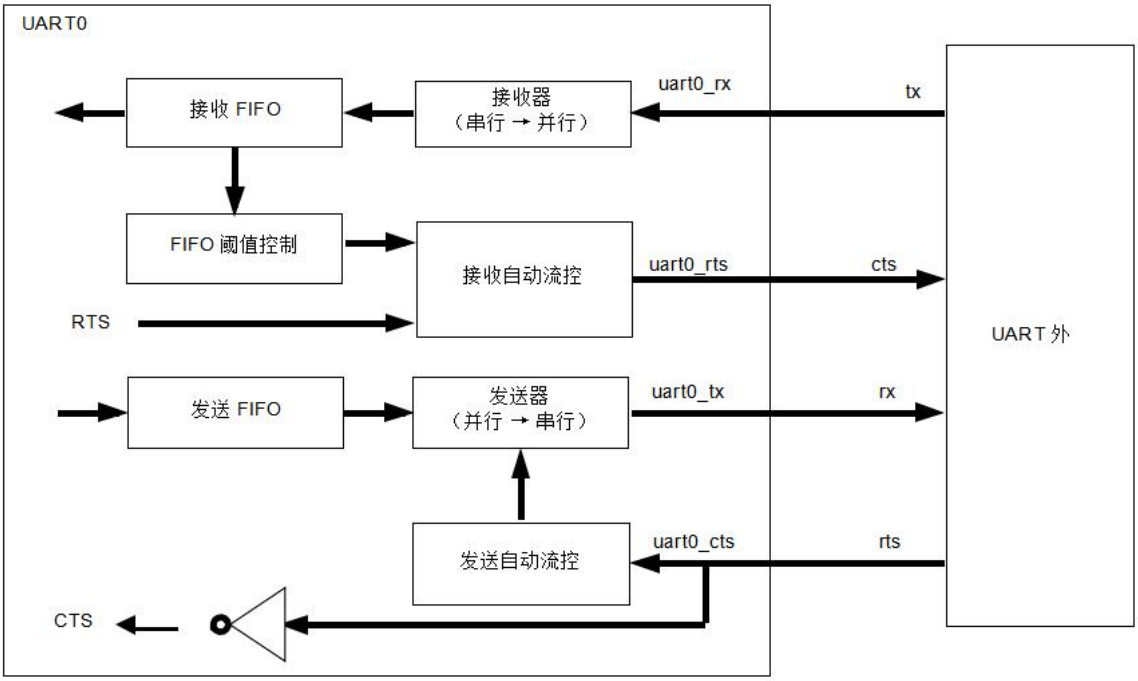

图 7-3 UART0 自动流控模式管脚连接图

UART0 自动流控模式下可以同时进行数据的接收和数据的发送，即接收自动流控和发送自动流控。

7.1.5.3. 接收自动流控（AUTO RTS）

当 UART0 置成自动流控模式，准备接收数据的时候，uart0_rts 会输出低电平，通知与此相连 UART 外设可以发送数据，当接收 FIFO 中收到的数据达到 FIFO 控制寄存器 UART0_FCR.RCVR_Trigger 设置的数值时，uart0_rts 会输出高电平，通知 UART 外设停止发送数据，等待内部 master 通过 RBR 寄存器从接收 FIFO 中读取数据，当接收 FIFO 处于完全空状态的时候，uart0_rts 才会再次输出低电平，通知 UART 外设继续发送数据，进而达到连续接收数据的功能。

接收 FIFO 的阈值（UART0_FCR.RCVR_Trigger）可以设置为以下数值：1 个字符，FIFO 四分之一满，FIFO 半满，FIFO 差 2 个字符满。

需要特别说明的是，在 uart0_rts 变为高电平（通知 UART 外设停止发送数据）的时候，因为下一帧数据已经进入到 UART 外设的发送模块，所以为了保证数据不丢失，UART0 还是会接收这一帧数据，所以将接收 FIFO 的阈值设置为距离 FIFO 全满还差 2 个字符的情况能够保证数据传输的安全性，并且发挥接收 FIFO 的最大效率。

接收自动流控模式的信号关系如图 7-4 所示，N 表示接收 FIFO 的阈值设置值。

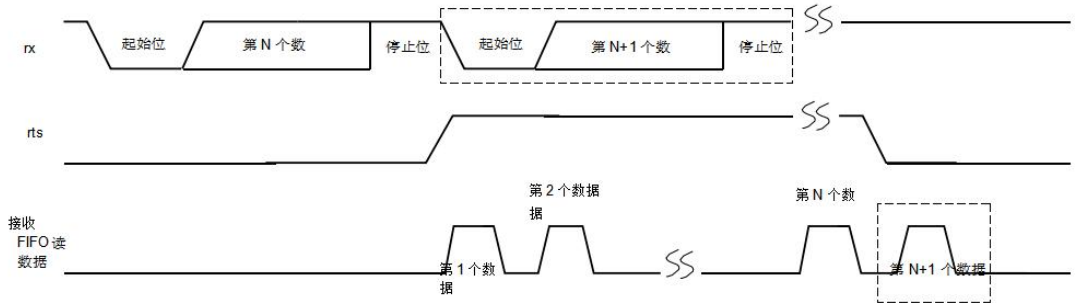

图 7-4　接收自动流控模式信号关系图

7.1.5.4. 发送自动流控（AUTO CTS）

当 UART0 配置成自动流控模式，准备发送数据的时候，会检查 uart0_cts 信号的输入状态，如果该信号为 0，表示相连接的 UART 外设可以接收数据，这时 UART0 才会发送数 据，如果该信号为 1，表示 UART 外设不能接收数据，则 UART0 会停止发送数据等待外设准备好。

需要特别注意的是，如果 uart0_cts 信号变为 1 的时间发生在当前正在发送的一帧数据的停止位之后，那么 UART0 会将之后的一帧数据传输完之后才会停止发送数据，所以接收数据的 UART 外设需要预留出足够的空间接收该帧数据。

发送自动流控模式的信号关系如图 7-5 所示。

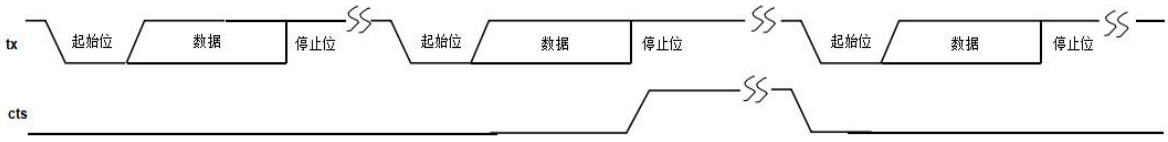

图 7-5 发送自动流控模式信号关系图

7.1.5.5. 中断模式

UARTx（x=0，1，2）有多种中断类型，每一种中断都有自己的中断 ID 和中断优先级。中断 ID 和中断优先级均不可设置。读中断寄存器 UARTx_IIR（x=0，1，2）的 IID 位可以得到各中断 ID 号。以下分别对各种中断的中断 ID，优先级，产生条件、清除方法进行说明。

7.1.5.5.1. 中断优先级

UARTx（x=0，1，2）的每个中断都有自己的优先级，一共有 5 个中断优先级，1 是最高优先级，5 是最低优先级。各个中断的优先级为确定的，不可更改。

当不同优先级的中断同时发生时，送到 CPU 的中断标志信号（uartx_intr（x=0，1，2））为有效。读取中断 ID 时，会首先得到高优先级的中断的 ID，之后清除该中断，中断标志信号仍旧为有效；再读取中断 ID，会得到低优先级的中断 ID，之后清除该中断，如果此时没有其他中断发生，中断标志信号变为无效（0），表示 UARTx（x=0，1，2）无任何中断发生。具体的处理流程如下图所示。

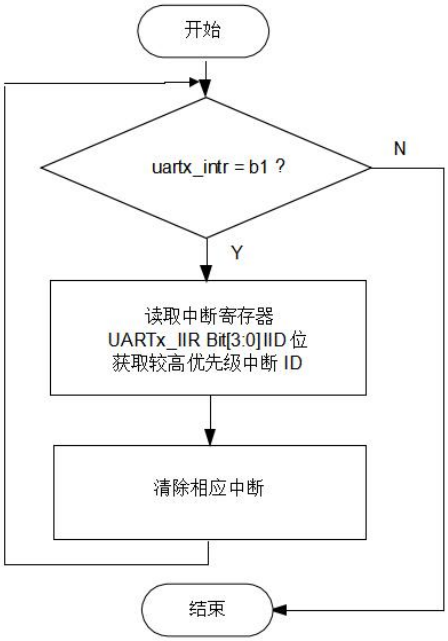

图 7-6 UARTx(x=0,1,2)中断处理流程图

当不同中断同时发生时，不会丢失中断，只是需要先处理高优先级的中断，后处理低优先级的。

7.1.5.5.2. 流控状态改变中断

中断 ID 号：0000，中断优先级：4

- 中断发生条件：

  1. 使能该中断，即设置中断使能寄存器 UART0_IER.EMSI=1。
  2. 不使能自动流控模式。
  3. uart0_cts 管脚信号电平发生变化。

  如果满足上面三个条件就会产生**流控状态改变中断**。

  读中断寄存器 UART0_IIR.IID 位可得到流控状态改变中断的 ID 号。

- 清除中断方法：

  读流控状态寄存器 UART0_MSR，可以清除流控状态改变中断。

7.1.5.5.3. 发送保持寄存器 THR 空中断

中断 ID 号：0010，中断优先级：3

在 FIFO 使能和 FIFO 不使能时发生发送保持寄存器 THR 空中断的条件是不同的，清除中断的方法也不相同，所以下面分为两种情况进行说明。

1. FIFO 使能，即设置 FIFO 控制寄存器 UARTx_FCR.FIFO_Enable（x=0，1，2）位为 1。

   中断发生条件分为两种情况。

   a. 非可编程 THRE 模式下，即中断使能寄存器 UARTx_IER.PTIME（x=0，1，2）位为 0。

      - 中断发生条件：
        - 使能该中断，即设置中断使能寄存器 UARTx_IER.ETH（x=0，1，2）位为 1；
        - 发送 FIFO 完全为空（0），发送保持寄存器 THR 为空。
        - 满足了上面两个条件就会发生发送保持寄存器 THR 空中断。当发生了发送保持寄存器 THR 空中断时，中断寄存器 UARTx_IIR.IID（x=0，1，2）为 0010，且传输状态寄存器 UARTx_TSR.THRE（x=0，1，2）为 1。
      - 清除中断方法：
        - 往发送保持寄存器 UARTx_THR（x=0，1，2）中写数，使它充发送 FIFO，使得发送 FIFO 处于非空状态，这样中断就会被清除。

   b. 可编程 THRE 模式下，即中断使能寄存器 UARTx_IER.PTIME（x=0，1，2）为 1。

      - 中断发生条件
        - 使能该中断，设置中断使能寄存器 UARTx_IER.ETHEI（x=0，1，2）为 0。
        - 发送 FIFO 中的数据数目小于等于 FIFO 控制寄存器 UARTx_FCR（x=0，1，

2）的 bit5，bit4 设置的 THRE 中断阈值。

- 如果满足上面两个条件会发生发送保持寄存器 THR 空中断。当发生了该中断时，中断寄存器 UARTx_IIR.IID（x=0，1，2）为 0010。

- 清除中断方法：
  - 在 FIFO 使能状态发生的中断，清除中断的方法相同，都是往发送保持寄存器 UARTx_THR（x=0，1，2）中写数，使它填充发送 FIFO，直到不满足中断发生的条件，这样中断就会自动被清除。

2. FIFO 不使能，即设置 FIFO 控制寄存器 UARTx_FCR.FIFO_Enable（x=0，1，2）为 0。
   - 中断发生条件：
     - FIFO 不使能时，只要发送保持寄存器 UARTx_THR（x=0，1，2）为空就会产生 THRE 中断。当发生了发送保持寄存器 THR 空中断时，中断寄存器 UARTx_IIR.IID（x=0，1，2）为 0010，且传输状态寄存器 UARTx_TSR.THRE（x=0，1，2）为 1。
   - 清除中断方法
     - FIFO 不使能时，往发送保持寄存器 UARTx_THR（x=0，1，2）中写 1 个字符就会清除该中断。但是当 THRE 变为 0 后，再往 UARTx_THR（x=0，1，2）中写数时，原有的数据就会丢失。

综上所述，只要符合以上任何一组中断产生条件就会产生发送保持寄存器 THR 空中断，同时 UARTx_IIR.IID（x=0，1，2）就会显示中断 ID 号 0010。

7.1.5.5.4. 接收数据有效中断

中断 ID 号：0100，中断优先级：2

在 FIFO 使能和 FIFO 不使能时发生接收数据有效中断的条件是不同的，清除中断的方法也不相同，所以下面分为两种情况进行说明。

1. FIFO 使能，设置 FIFO 控制寄存器 UARTx_FCR.FIFO_Enable（x=0，1，2）为 1。
   - 中断发生条件
     - 使能该中断，即中断使能寄存器 UARTx_IER.ERDAI（x=0，1，2）为 1。
     - 接收 FIFO 中接收到的数据个数达到该中断设定的阈值（写 FIFO 控制寄存器 UARTx_FCR.RCVR_Trigger（x=0，1，2）设置）时产生接收数据有效中断。
     - 如果满足上面两个条件会发生接收数据有效中断。
   - 清除中断方法

- FIFO 使能时，读接收缓存寄存器 UARTx_RBR（x=0，1，2），使接收 FIFO 中的数据个数小于设定的阈值就可以清除接收数据有效中断。

2. FIFO 不使能，即设置 FIFO 控制寄存器 UARTx_FCR.FIFO_Enable（x=0，1，2）为 0。
   - 中断发生条件
     - 使能该中断，即中断使能寄存器 UARTx_IER.ERDAI（x=0，1，2）为 1。
     - 接收缓存寄存器 UARTx_RBR（x=0，1，2）中有一个 8 比特字符时会产生接收数据有效中断。
     - 如果满足上面两个条件会发生接收数据有效中断。
   - 清除中断方法
     - FIFO 不使能时，读接收缓存寄存器 UARTx_RBR（x=0，1，2），取出其中的字符就会清除接收数据有效中断。

7.1.5.5.5. 字符超时中断

中断 ID 号：1100，中断优先级：2

- 中断发生条件：
  - 只有 FIFO 使能时才可能发生此中断。当接收 FIFO 里至少还有一个字符，但是接收 FIFO 在 4 个字符的传输时间里没有收到数据或者没有从接收 FIFO 中读取数据，此时会发生字符超时中断。读中断寄存器位 UARTx_IIR.IID（x=0，1，2）可得到字符超时中断的 ID 为 1100。
- 清除中断方法：
  - 读接收缓存寄存器 UARTx_RBR（x=0，1，2），这样 UARTx_RBR（x=0，1，2）就会自动访问接收 FIFO，读取其中的数据，进而清除字符超时中断。

7.1.5.5.6. 错误中断

中断 ID 号：0110，中断优先级：1

- 中断发生条件：
  - 如果已经使能了该中断，即中断使能寄存器 UARTx_IER.ETSI（x=0，1，2）为 1。在传输过程中发生溢出错误、奇偶校验错误、帧格式错误和停止错误时都会产生错误中断。读中断寄存器 UARTx_IIR.IID（x=0，1，2）可得到错误中断的 ID 为 0110。读取传输状态寄存器 UARTx_TSR（x=0，1，2）的 bit[4：1]可得到发生的是哪种错误。
- 清除中断方法：
  - 无论是哪种错误引起的错误中断，读传输状态寄存器 UARTx_TSR（x=0，1，2）就能清 除错误中断。

4 种错误发生的条件各不相同，下面分别详细说明每种错误产生的条件。

a. 溢出错误

- 相应的错误状态显示在传输状态寄存器 UARTx_TSR.OE（x=0，1，2）。在 FIFO 使能和 FIFO 不使能时发生溢出错误的条件是不同的，所以分两种条件进行说明。
  - 当 FIFO 不使能时，在下一个数据到来之前接收缓存寄存器 UARTx_RBR（x=0，1，2）中的数据必须被读走，否则 UARTx_RBR（x=0，1，2）中的数据将被新来的数据覆盖，就会产生溢出错误，从而引发错误中断。
  - 当FIFO 使能时，若接收 FIFO 已满，新接收到的数据将会丢失（FIFO 中的数据不会丢失），从而产生溢出错误，引发错误中断。

b. 奇偶校验错误

- 相应的错误状态显示在传输状态寄存器 UARTx_TSR.PE（x=0，1，2）。若传输控制寄存器 UARTx_TCR.PEN（x=0，1，2）设置为 1，当传输过程中如果发生奇偶校验错误，传输状态寄存器 UARTx_TSR.PE（x=0，1，2）就会变成 1。

c. 帧格式错误

- 相应的错误状态显示在传输状态寄存器 UARTx_TSR.FE（x=0，1，2）。当接收到的数据 中缺少 1 个有效的 stop 位时出现帧格式错误。

d. 停止错误

- 相应的错误状态显示在传输状态寄存器 UARTx_TSR.BI（x=0，1，2）。当 uartx_rx（x=0，1，2）长时间保持 0 超过起始时间、数据时间、校验和停止时间之和时该错误发生。停止错误 发生时 UARTx（x=0，1，2）只接收到一个“0”字符。

7.1.5.5.7. 忙检测中断

中断 ID 号：0111，中断优先级：5

- 中断发生条件：
  - 当 UARTx（x=0，1，2）正在进行数据传输时（此时状态寄存器 UARTx_USR=1（x=0，1，2）），对 UARTx_TCR（x=0，1，2）寄存器进行写操作会产生忙检测中断。读中断寄存器 UARTx_IIR.IID（x=0，1，2）可得到忙检测中断的 ID 为 0111。
- 清除中断方法：
  - 读状态寄存器 UARTx_USR（x=0，1，2）就会清除忙检测中断。

7.1.5.5.8. 时钟关断保护功能

UARTx（x=0，1，2）模块和 CPR 模块可以共同实现对其工作时钟的关断保护功能，通过 CPR_LP_CTL.UARTx_CLK_OFF_PROTECT_EN（x=0，1，2）寄存器进行配置，具体的配置方式见 CPR 模块的相关章节。

如果使能了该保护功能，那么在 UARTx（x=0，1，2）模块在进行数据传输的过程中，M4 可以在 CPR 中直接进行关断 uartx_clk（x=0，1，2）的操作，硬件会自动对 UARTx（x=0，1，2）的时钟进行保护，待数据传输完成之后才会关断其时钟，以保证数据不会丢失。

如果不使能该功能，那么不管在什么情况下，只要 M0 通过 CPR 模块对 UARTx（x=0，1，2）的时钟进行了关断操作，那么时钟会立即关断，而不管 UARTx（x=0，1，2）是否在进行工作。

7.1.5.6. 信号及接口描述

图中是与 UARTx 相关的信号及接口描述图。其中 UART 的 DMA 请求信号与 DMAS 相连，两个中断信号均送到 M0。

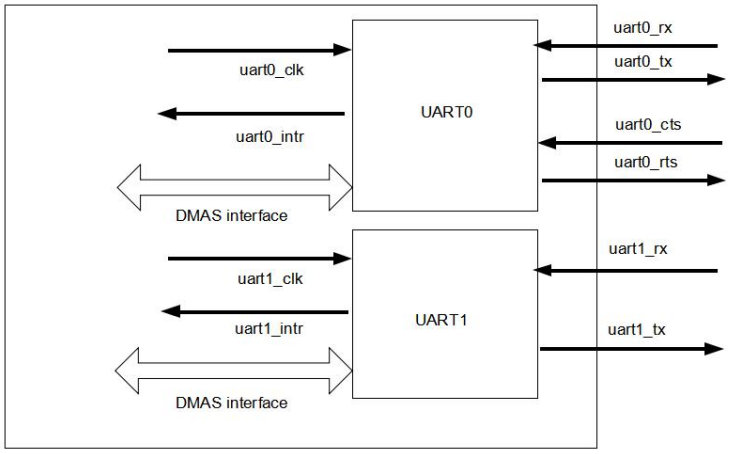

图 7-7 UART 信号及接口描述图

表 7-1 UARTx(x=0,1,2)信号及接口描述表

名称	宽度	方向	描述	备注
CTL_APB 接口信号				内部信号
DMAS 接口信号				
dma_uart0_tx			UART0 发送与 DMA 之间的硬件握手信号（存储器->DMAS->UART0）	内部信号
dma_uart0_rx			UART0 接收与 DMA 之间的硬件握手信号（UART0->DMA->存储器）	内部信号
dma_uart1_tx			UART1 发送与 DMA 之间的硬件握手信号（存储器->DMA->UART1）	内部信号
dma_uart1_rx			UART1 接收与 DMA 之间的硬件握手信号（UART1->DMA->存储器）	内部信号

UART0 信号				
uart0_clk	1	I	UART0 工作时钟，来自 CPR，默认时钟是 1.8432MHz（对应的串口波特率为 115200bps），可在 CPR 中配置 UART0 的工作时钟最高为 16MHz（对应串口波特率为 1Mbps），具体配置见 CPR 模块寄存器描述。	内部信号
uart0_rx	1	I	UART0 的串行输入信号	管脚信号
uart0_tx	1	O	UART0 的串行输出信号	管脚信号
uart0_rts	1	O	请求发送输出信号	管脚信号
uart0_cts	1	I	允许发送输入信号	管脚信号
UART1 信号				
uart1_clk	1	I	UART1 工作时钟，来自 CPR，默认时钟是 1.8432MHz（对应的串口波特率为 115200bps），可在 CPR 中配置 UART1 的工作时钟最高为 16MHz（对应串口波特率为 1Mbps），具体配置见 CPR 模块寄存器描述。	内部信号
uart1_rx	1	I	UART1 的串行输入信号	管脚信号
uart1_tx	1	O	UART1 的串行输出信号	管脚信号
UART2 信号				
Uart2_clk	1	I	UART2 工作时钟，来自 CPR，默认时钟是 1.8432MHz（对应的串口波特率为 115200bps），可在 CPR 中配置 UART2 的工作时钟最高为 16MHz（对应串口波特率为 1Mbps），具体配置见 CPR 模块寄存器描述。	内部信号
Uart2_rx	1	I	UART2 的串行输入信号	管脚信号
Uart2_tx	1	O	UART2 的串行输出信号	管脚信号
中断				
uart0_intr	1	O	UART0 中断输出，直接送往 M0	内部信号
uart1_intr	1	O	UART1 中断输出，直接送往 M0	内部信号
Uart2_intr	1	O	UART2 中断输出，直接送往 M0	内部信号

7.2. 寄存器地址映射

表 7-2 uart 寄存器表

寄存器名	描述	方向&宽度		地址偏移	复位值
UARTx_RBR	接收缓存寄存器	R	8bits	0x00	0x00
UARTx_THR	发送保持寄存器	W	8bits	0x00	0x00

UARTx_DLL	除数锁存低位	RW	8bits	0x00	0x00
UARTx_IER	中断使能寄存器	RW	8bits	0x04	0x00
UARTx_DLH	除数锁存高位	RW	8bits	0x04	0x00
UARTx_IIR	中断寄存器	R	8bits	0x08	0x01
UARTx_FCR	FIFO 控制寄存器	W	8bits	0x08	0x00
UARTx_TCR	传输控制寄存器	RW	8bits	0x0C	0x00
UARTx_MCR	流控控制寄存器	RW	6bits	0x10	0x00
UARTx_TSR	传输状态寄存器	R	8bits	0x14	0x60
UARTx_MSR	流控状态寄存器	R	5bits	0x18	0x10
UARTx_USR	状态寄存器	R	1bit	0x7C	0x0

7.3. UART 寄存器

- UART0 的基地址为：0x40010000
- UART1 的基地址为：0x40011000
- UART2 的基地址为：0x40011400

注：本节寄存器名称中 x 的取值为 0，1，2。

7.3.1. 接收缓存寄存器（UARTx_RBR）

- 地址偏移：0x00

注意：当 DLAB=0 时，对地址偏移 0x00 作“读”操作，访问的是该寄存器。

位数	名称	方向	复位值	描述
7：0	RBR	R	0x00	暂存来自串行输入端口的数据：在 UARTx_TSR 传输状态寄存器的 DR=1 时，标志数据已经准备好，可以对 UARTx_RBR 进行读操作。对其进行读操作时，UARTx_RBR 会自动读取接收 FIFO 中的数据。

7.3.2. 发送保持寄存器（UARTx_THR）

- 地址偏移：0x00

注意：当 DLAB=0 时，对地址偏移 0x00 作“写”操作，访问的是该寄存器。

位数	名称	方向	复位值	描述
7：0	THR	W	0x0	暂存即将通过串行输出端口输出的数据：在 UARTx_TSR 传输状态寄存器的 THRE=1 时，有两个含义： ● 当 UARTx_TSR 传输状态寄存器 THRE=1 表示“发送保持寄存器空中断”时，可以往 UARTx_THR 寄存器中写数据； ● 当 UARTx_TSR 传输状态寄存器 THRE=1 表示“FIFO 满”时，应该等待，不能往 UARTx_THR 寄存器中写数据，否则新写入数据会丢失。 当对 UARTx_THR 进行写操作时，UARTx_THR 会自动将数据写入发送 FIFO。

7.3.3. 除数所存低位（UARTx_DLL）

- 地址偏移：0x00

当 UARTx_TCR.DLAB=1 时，对地址偏移 0x00 作“读/写”操作，访问的是该寄存器。设置除数寄存器后，至少要等待（2×除数×16）个 uartx_clk（x=0，1，2）时钟周期后才能开始收发数据。在对 UARTx_DLL（x=0，1，2）操作前必须确保 DLAB=1，并且 UARTx（x=0，1，2）不忙（UARTx_USR（x=0，1，2）的 bit[0]为 0），否则操作无效。

位数	名称	方向	复位值	描述
7：0	DLL	RW	0x00	锁存除数：用于存 16 位除数的低 8 位，该除数用于控制波特率。 波特率＝（串行时钟（uartx_clk 频率）/（16×除数)

7.3.4. 除数所存高位（UARTx_DLH）

- 地址偏移：0x04

当 UARTx_TCR.DLAB=1 时，对地址偏移 0x04 作“读/写”操作，访问的是该寄存器。设置除数寄存器后，至少要等待（2×除数×16）个 uartx_clk（x=0，1，2）时钟周期后才能开始收发数据。在对 UARTx_DLH（x=0，1，2）操作前必须确保 DLAB=1，并且 UARTx（x=0，1，2）不忙（UARTx_USR（x=0，1，2）的 bit[0]为 0），否则操作无效。

位数	名称	方向	复位值	描述
7：0	DLH	RW	0x00	锁存除数：用于存 16 位除数的高 8 位，该除数用于控制波特率。 波特率＝（串行时钟（uartx_clk 频率）/（16×除数)

7.3.5. 中断使能寄存器（UARTx_IER）

- 地址偏移：0x04

当 DLAB=0 时，对地址偏移 0x04 作“读/写”操作，访问的是该寄存器。

位数	名称	方向	复位值	描述
7	PTIME	RW	0x0	设置可编程 THRE 中断 0：非可编程 THRE 中断模式 1：可编程 THRE 中断模式

6：4	N/A	—	—	保留
3	EMSI	RW	0x0	使能流控状态改变中断 中断优先级 4 0：不使能 1：使能
2	ETSI	RW	0x0	使能错误中断： 中断优先级 1 0：不使能 1：使能
1	ETHEI	RW	0x0	使能发送保持寄存器 THR 空中断 中断优先级 3 0：不使能 1：使能
0	ERDAI	RW	0x0	使能接收数据有效中断和字符超时中断 中断优先级 2 0：不使能 1：使能

7.3.6. 中断寄存器（UARTx_IIR）

- 地址偏移：0x08

位数	名称	方向	复位值	描述
7：6	FS	R	0x0	是否使能 FIFO 00：FIFO 不使能 11：FIFO 使能 缺省设置或复位时默认 FIFO 不使能。
5：4	N/A	—	0x0	保留。此字段一定为 00
3：0	IID	R	0x1	中断 ID 号：1 是最高优先级，5 是最低优先级。 0000：流控状态改变中断，由 UARTx_IER 中断使能寄存器 EMSI 位控制其使能，中断优先级：4 0001：没有发生中断

				0010：发送保持寄存器 THR 空中断 UARTx_IER 寄存器 ETHEI 位控制其使能中断优先级：3 0100：接收数据有效中断，由 UARTx_IER 寄存器 ERDAI 位控制其使能中断优先级：2 0110：错误中断，由 UARTx_IER 寄存器 ETSI 位控制其使能，中断优先级：1 0111：忙检测中断，中断优先级：5 1100：字符超时中断，由 UARTx_IER 寄存器 ERDAI 位控制其使能，中断优先级：2

7.3.7. FIFO 控制寄存器（UARTx_FCR）

- 地址偏移：0x08

位数	名称	方向	复位值	描述
7：6	RCVR_Trigger	W	0x0	设置接收 FIFO 中断的阈值。 00：FIFO 中有 1 个字符报中断 01：FIFO 1/4 满报中断 10：FIFO 1/2 满报中断 11：FIFO 差 2 个字符满报中断
5：4	TX_Empty_Trigger	W	0x0	设置发送 FIFO 的 THRE 中断阈值。此参数仅在 FIFO 使能和可编程 THRE 中断模式下用来设置 THRE 阈值中断。 00：FIFO 空报中断 01：FIFO 中还有 2 个数据报中断 10：FIFO3/4 空报中断 11：FIFO1/2 空报中断
3	NA	—	—	保留
2	XMIT_FIFO_Reset	W	0x0	清空发送 FIFO：该位自动清零。 0：不做任何操作 1：清空发送 FIFO

1	RCVR_FIFO_Reset	W	0x0	清空接收 FIFO：该位自动清零。 0：不做任何操作 1：清空接收 FIFO
0	FIFO_Enable	W	Cx0	启用发送/接收 FIFO：任何时候此位改变设置时，接收/发送 FIFO 复位。 0：不使能 FIFO 1：使能 FIFO

7.3.8. 传输控制寄存器（UARTx_TCR）

- 地址偏移：0x0C

控制接收和发送数据帧格式。当状态寄存器 UARTx_USR.BUSY=1（x=0，1，2）时，对此寄存器进行写操作会产生忙检测中断。

位数	名称	方向	复位值	描述
7	DLAB	RW	0x0	UARTx_DLL，UARTx_DLH 操作使能位。 只有当 UARTx 不忙（UARTx_USR.BUSY=0） 时，对该位写操作有效。 0：UARTx_DLL，UARTx_DLH 操作不使能。 1：操作使能。可以去操作 UARTx_DLL，UARTx_DLH，操作完成后必须设置 DLAB=0
6	Break	RW	0x0	停止控制位：用于流控模式往该位写“1”，发送 1 个停止位信号使 uartx_tx 线为低电平，该低电平一直保持到往该位写“0”。 往该位写“0”，uartx_tx 才恢复为高电平。
5	NA	—	—	保留
4	EPS	RW	0x0	奇偶校验选择位。 只有当 UARTx 不忙（UARTx_USR.BUSY=0）时，对该位写操作有效。 0：奇校验 1：偶校验
3	PEN	RW	0x0	奇偶校验使能位。 只有当 UARTx 不忙（UARTx_USR.BUSY=0）时，对该位写

				操作有效。 0：奇偶校验不使能 1：奇偶校验使能
2	STOP	RW	0x0	设置 STOP 位的个数。 只有当 UARTx 不忙（UARTx_USR.BUSY=0）时，对该位写操作有效。 0：1bit 停止位 1：当数据位设置为 5 位时，将产生 1.5bits 的停止位；当数据位设置为其他位时，将产生 2bits 停止位。
1：0	CLS	RW	0x0	设置传输一帧数据的总的数据位的个数。只有当 UARTx 不忙（UARTx_USR.BUSY=0） 时，对该位写操作有效。 00：5bits 01：6bits 10：7bits 11：8bits

7.3.9. 流控控制寄存器（UARTx_MCR）

- 地址偏移：0x10

位数	名称	方向	复位值	描述
5	AFCE	RW	0x0	是否使能 AFC 模式。 当 FIFO 使能时此参数用于 AFC 控制（必须保证先使能 FIFO，再使能 AFC）。 0：不使能 AFC 模式。 1：使能 AFC 模式。
4：2	NA	—	—	保留
1	RTS	RW	0x0	在非流控模式下，用户可以通过配置 RTS，来控制 uartx_rts 的输出，即为一个 GPIO 端口。 0：uartx_rts 输出 1。 1：uartx_rts 输出 0 在流控模式下，该位寄存器和接收 FIFO 中断共同控制 uartx_rts 的输出，以实现接收数据的流控控制。如果要实现

				双方向 AFC 模式，该位必须设置为 1。 0：流控模式下不接收数据，只有发送数据。 1：使能双方向流控模式，可同时接收发送。
0	NA	—	—	保留

7.3.10. 传输状态寄存器（UARTx_TSR）

- 地址偏移：0x14

指示当前发送/接收数据的状态，可随时被程序读取。错误中断（0110）发生时，该寄存器指示相应的中断状态。

位数	名称	方向	复位值	描述
7	RX FIFO Error	R	0x0	接收 FIFO 错误。 FIFO 使能时该位有效。如果接收过程中 FIFO 出现了奇偶校验错误，或者帧格式错误，或者停止错误就会产生接收 FIFO 错误中断。读 UARTx_TSR 可以清除接收 FIFO 错误中断。 0：正常 1：发生接收 FIFO 错误
6	NA	—	—	保留
5	THRE	R	0x1	发送保持寄存器 UARTx_THR 空中断。 FIFO 不使能时，当 UARTx_THR 发送保持寄存器空，发生发送保持寄存器空中断。 FIFO 使能时，在非可编程 THRE 模式下，当 UARTx_THR 空时（此时 FIFO 一定空），发生发送保持寄存器空中断。 FIFO 使能时，在可编程 THRE 模式下，此位不再表示发送保持寄存器 THR 空中断是否发生，表示的是 FIFO 是否为满，当 FIFO 为满时，THRE=1。

				0：正常 1：发生发送保持寄存器空中断（FIFO 不使能，FIFO 使能并且在非可编程 THRE 模式下） 1：发送 FIFO 满（FIFO 使能并且在可编程 THRE 模式下）
4	BI	R	0x0	停止错误。 0：正常 1：发生停止错误
3	FE	R	0x0	帧格式错误。 0：正常 1：发生帧格式错误
2	PE	R	0x0	奇偶校验错误。 0：正常 1：发生奇偶校验错误
1	OE	R	0x0	接收 FIFO 溢出错误。 0：正常 1：发生溢出错误
0	DR	R	0x0	接收数据准备好：可以读取数据。 0：接收数据没有准备好 1：表示 UARTx_RBR（接收缓存寄存器）中或接收 FIFO 中至少还有一个字符可以被读取。当在 FIFO 不使能时读 UARTx_RBR 或者 FIFO 使能时接收 FIFO 空，该位被清 0。

7.3.11. 流控状态寄存器（UARTx_MSR）

- 地址偏移：0x18

位数	名称	方向	复位值	描述
4	CTS	R	—	管脚信号 uartx_cts 的值的反。 1：uartx_cts 为低电平 0：uartx_cts 为高电平 uartx_cts 是流控状态输入管脚信号，用于 AFC

				模式。当启用 AFC 时，且 FIFO 使能，uartx_cts 高电平（无效）将会使发送无效。 该位的复位值与外部管脚信号有关，UART0 的该位寄存器由于 uart0_cts 管脚是复用的，在处于非 UART 模式下，该位寄存器复位值为 0，在 UART 模式下，复位值是 uart0_cts 的反。
3：1	NA	—	—	保留
0	DCTS	R	—	标志从前一次读取 UARTx_MSR，到当前时刻为止，uartx_cts 管脚信号电平发生变化情况。用于流控模式。该位寄存器在进行读操作 之后自动清 0。 0：没有变化 1：发生变化

7.3.12. 状态寄存器（UARTx_USR）

- 地址偏移：0x7C

  该寄存器指示 UARTx（x=0，1，2）的状态。

位数	名称	方向	复位值	描述
0	BUSY	R	0x0	UART 状态：用于指示 UARTx 的工作状态 0：UARTx 处于空闲状态或不工作状态 1：UARTx 正在传输数据

7.4. 工作流程

下面给出了 3 种推荐采用的工作流程。分别为非流控模式下的数据收发，自动流控模式下的数据收发和 DMA 模式下的数据收发。每种工作流程中配置 UARTx（x=0,1,2）的部分，每一步的先后顺序不能调整，如果调整的话可能会导致 UARTx（x=0,1,2）工作不正常。

以下的工作流程都是使 UARTx（x=0,1,2）工作在串口波特率为 115200 的默认情况下，如果想配置不同的串口波特率，需要在这些操作之前加上配置 CPR 的部分，使得 UARTx（x=0,1,2）的工作时钟满足要求。具体配置方法可参见 CPR 模块。

7.4.1. 非流控模式下的数据收发

(1) 首先要对 UARTx（x=0,1,2）进行初始化，设置 UARTx_FCR（x=0,1,2）控制寄存器完成使能 FIFO，复位接收 FIFO，复位发送 FIFO，发送 FIFO 阈值，接收 FIFO 阈值；

(2) 写 UARTx_TCR（x=0,1,2）设置数据传输格式以及 DLAB=1；

(3) 写 UARTx_DLH（x=0,1,2）配置除数锁存高位,写 UARTx_DLL（x=0,1,2）配置除数锁存低位，设置波特率；

(4) 写 UARTx_TCR（x=0,1,2）的 DLAB=0，数据传输格式设置同步骤 2；

(5) 之后等待至少（2×除数×16）个 uartx_clk（x=0,1,2）时钟周期，否则会产生错误；

(6) 写 UARTx_IER（x=0,1,2）中断使能寄存器，设置 ERDAI=1，ETHEI=1，使能 FIFO 接收和发送中断，配置 PTIME=1 使能可编程 THRE；

(7) 当发送 FIFO 中的数据小于 UARTx_FCR（x=0,1,2）设定的发送 FIFO 阈值时，发生发送保持寄存器空中断，此时可以向发送 FIFO 写入数据进行发送；

(8) 当接收 FIFO 中的数据大于等于 UARTx_FCR（x=0,1,2）设定的接收 FIFO 阈值时，发生接收数据有效中断，可以将接收 FIFO 中的数据读出；

7.4.2. 自动流控（AFC）模式下的数据收发

(1) 首先要对 UART0 进行初始化，设置 UART0_FCR 控制寄存器完成使能 FIFO，复位接收 FIFO，复位发送 FIFO，发送 FIFO 阈值，接收 FIFO 阈值；

(2) 写 UART0_TCR 设置数据传输格式以及 DLAB=1；

(3) 写 UART0_DLH 除数锁存高位，UART0_DLL 除数锁存低位，设置波特率；

(4) 写 UART0_TCR 的 DLAB=0，数据传输格式设置同步骤 2；

(5) 之后等待至少（2×除数×16）个 uart0_clk 时钟周期，否则会产生错误；

(6) 写流控控制寄存器 UART0_MCR，配置 AFCE=1，使得 UART0 进入 AFC 模式；配置 RTS=1，使能双向 AFC 模式；

(7) 写 UART0_IER 中断使能寄存器，设置 ERDAI=1，ETHEI=1，使能 FIFO 接收和发送中断，配置 PTIME=1 使能可编程 THRE；

(8) 当发送 FIFO 中的数据小于 UART0_FCR 设定的发送 FIFO 阈值时，发生发送保持寄存器空中断，此时可以向发送 FIFO 写入数据进行发送；

(9) 当接收 FIFO 中的数据大于等于 UART0_FCR 设定的接收 FIFO 阈值时，发生接收数据有效中断，可以将接收 FIFO 中的数据读出。

7.4.3. DMA 模式下的的数据收发

(1) 首先要对 UARTx（x=0,1）进行初始化，设置 UARTx_FCR（x=0,1）控制寄存器完成使能 FIFO，复位接收 FIFO，复位发送 FIFO，发送 FIFO 阈值，接收 FIFO 阈值；

(2) 写 UARTx_TCR（x=0,1）设置数据传输格式以及 DLAB=1；

(3) 写 UARTx_DLH（x=0,1）除数锁存高位，UARTx_DLL（x=0,1,2）除数锁存低位，设置波特率；

(4) 写 UARTx_TCR（x=0,1）的 DLAB=0，数据传输格式设置同步骤 2；

(5) 之后等待至少（2×除数×16）个 uartx_clk（x=0,1）时钟周期，否则会产生错误；

(6) 再配置 DMA，DMA 和 UARTx（x=0,1）的硬件握手如表 7-3 所示。

表 7-3 DMA 和 UARTx（x=0，1）的硬件握手信号

DMAC 的通道号	寄存器名
0	UART0 发送（存储器->DMA->UART0）
1	UART1 发送（存储器->DMA->UART1）
8	UART0 接收（UART0->DMA->存储器）
9	UART1 接收（UART1->DMA->存储器）

根据握手信号配置 DMA 相应的通道寄存器；

1. 使能 DMA 的通道，配置 DMA 的中断使能控制寄存器，使能 blk 结束中断；

2. 等待 DMA 的相应通道的 blk 结束中断，就可以完成数据的传输。

3. 在使用 DMA 进行数据传输的时候，自动流控模式（AFC）和可编程 THRE 功能可自由选择，视系统需要而定。

第 8 章 SysTick

8.1. 概述

SysTick—系统定时器是属于 CM0 内核中的一个外设，内嵌在 NVIC 中。Cortex-M 处理器之所以集成 SysTick，是为了将其用作 OS 的时钟节拍，这样 Cortex-M 处理器之间就可以保持统一，方便 OS 的管理和移植。

8.2. SysTick 中断时间

SysTick 定时器的计数器是向下递减计数的，计数一次的时间 Tdec=1/SYSCLK， 当重装载数值寄存器的值 VALUELOAD 减到 0 的时候，产生中断，可知中断一次的时间，中断时间 **Tint**=VALUELOAD*Tdec=VALUELOAD/SYSCLK，其中 SYSCLK =32MHZ。如果 VALUELOAD 设置为 32，那中断一次的时间 **Tint**=32/32M=1us。 不过 1us 的中断没啥意义，整个程序的重心都花在进出中断上了，根本没有时间处理其他的任务。SysTick_Config（）的形我们配置为 xc_clock_hfclk_in_get( )/ 100=32M/100=320000，从刚刚分析我们知道这个形参的值最终是写到重装载寄存器 LOAD 中的，从而可知我们现在把 SysTick 定时器中断一次的时间 TINT=320000/32M=10ms。

8.3. SysTick 寄存器

SysTick 有 4 个寄存器，如下表所示。

表 8-1 SysTick 寄存器表

序号	寄存器名	地址	描述
1	CTRL	0xE000E010	SysTick 控制和状态寄存器
2	LOAD	0xE000E014	SysTick 重装载数值寄存器
3	VAL	0xE000E018	SysTick 当前数值寄存器
4	CALIB	0xE000E01C	SysTick 校准数值寄存器

8.3.1. SysTick 控制和状态寄存器

- 地址偏移：0x10

位数	名称	方向	复位值	描述
31:17	Reserved	—	—	Reserved

16	COUNTFLAG	RO	0	当 SysTick 计时器计数达到零时设置为 1。 通过读取该寄存器来清零。
15:3	Reserved	—	—	Reserved.
2	CLKSOURCE	R/W	0	CLKSOURCE=1 表示内核时钟用于 SysTick 定时器。否则将使用参考时钟频率（取决 MCU 设计）。
1	TICKINT	R/W	0	SysTick 中断使能器。 TICKINT=1，则 SysTick 定时器倒计时到 0 时会产生 SysTick 异常。 TICKINT=0，则 SysTick 定时器倒计时到 0 时无动作。
0	ENABLE	R/W	0	当 ENABLE=1 时，SysTick 计时器启用。 否则计数将被禁用。

8.3.2. SysTick 重装载数值寄存器

地址偏移：0x14

位数	名称	方向	复位值	描述
31:24	Reserved	—	—	Reserved
23:0	RELOAD	R/W	Undefined	指定 SysTick 计时器的重装载值。

8.3.3. SysTick 当前数值寄存器

- 地址偏移：0x18

位数	名称	方向	复位值	描述
31:24	Reserved	—	—	Reserved
23:0	CURRENT	R/W	Undefined	读取时返回 SysTick 计时器的当前值。 向该寄存器写入任何值以清除寄存器并将 COUNTFLAG 设置为 0。 （这不会导致 SysTick 异常生成。）

8.3.4. SysTick 校准数值寄存器

- 地址偏移：0x1C

位数	名称	方向	复位值	描述
31	NOREF	R	—	如果读为 1，则表示 SysTick 始终使用内核时钟进行计数，因为没有可用的外部参考时钟。如果为 0，则外部参考时钟可用且可以使用。该值取决于 MCU 设计。
30	SKEW	R	—	如果设置为 1，则 TENMS 位字段不准确。 这值取决于 MCU 设计。
29:24	Reserved	—	—	Reserved.
23:0	TENMS	R	—	该值取决于 MCU 设计。

第 9 章 定时/计数器（TIMER）

9.1. 概述

XC65xx 芯片提供有 4 个独立的 32 位的 Timer，2 个独立的 32 位的 AOTimer，**AOTimer 的时钟源只能是 32k**，每个定时器都是完全独立的，可以一起同步操作。它们都可以工作于单次运行模式和循环运行两种模式，并且 Timer 的时钟频率和位宽都可以配置，Timer 的原理框图如下。

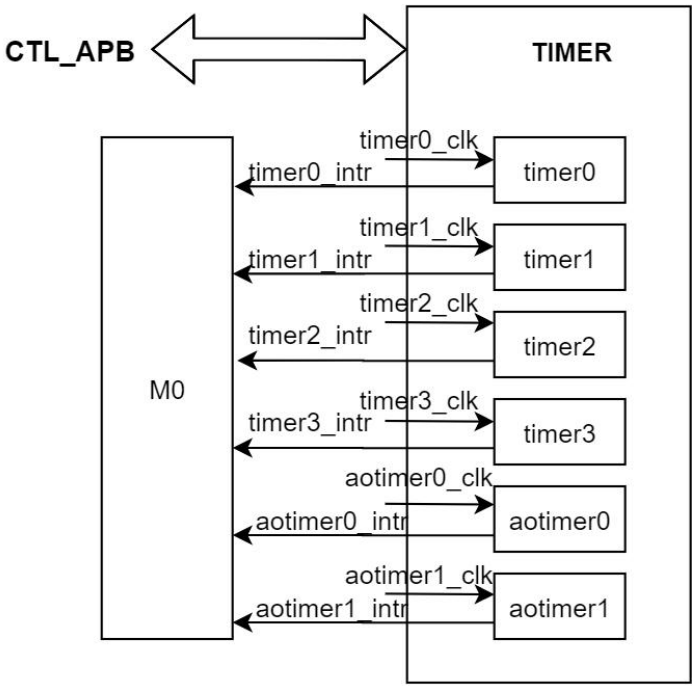

图 9-1 Timer 原理框图

9.2. 主要特性

- 4 个 32 位定时器，每个定时器定时时间最大为 2^32 个时钟周期。
- 支持定时器中断。
- 每个定时器都有独立的时钟
- 每个定时器都有独立的中断
- 支持同时清除 4 个定时器中断

9.3. 功能描述

定时器从计数初值开始向下计数，当计数值为 0 时会产生一个中断。在单次运行模式下，用户写入计数值载入寄存器 TIMERx_TLC（x=0，1，2，3）的值是定时器第一次计数的计数初值，之后定时器的计数初值为 0xFFFFFFFF；在循环运行模式下，用户写入计数值载入寄存器 TIMERx_TLC（x=0，1，2，3）的值就是定时器的计数初值。定时器的计数频率，即定时器的工作时钟可配，详见 CPR 模块寄存器 CPR_TIMERx_CLK_CTL（x=0，1，2，3）。为使定时器计数精确，请确保 TIMERx_clk（x=0，1，2，3）的频率小于 APB_clk 的 3 倍。

9.3.1. Timer 时钟源

Timer 使用的时钟源来自 CPR，由 CPR_TIMERx_CLK_CTL 控制寄存器选择分频来源及分频比。AOTimer 使用的时钟源来自 32K。

9.3.2. Timer 时钟频率

Timer 的分频来源可以选择 32MHz 或者 32KHz，拿 TIMER0 举例来说，TIMER0 的时钟源由 CPR_TIMER0_CLK_CTL 控制寄存器的 TIMER0_CLKSEL 位来进行选择，其分频比受分频来源影响：

- 当分频来源为 32MHz 时，其分频比由 TIMER0_CLK1_DIV 位决定。分频公式为

  2*(TIMER0_CLK1_DIV+1)

- 当分频来源为 32KHz 时，其分频比由 TIMER0_CLK0_DIV 位决定。分频公式为

  2*(TIMER0_CLK0_DIV+1)

9.4. Timer 工作模式

定时器有两种工作模式：单次运行模式、循环运行模式，默认工作在单次运行模式。在定时器 TIMERx（x=0，1，2，3）不使能的情况下，可通过对 TIMERx_TCR（x=0，1，2，3）控制寄存器 TMS 位的写操作来选择工作模式。TMS=0，工作于单次运行模式；TMS=1，工作于循环运行模式。

- **单次运行模式：**定时器第一次计数的初值为 TIMERx_TLC 内设定的值，产生中断以后的计数初值是 0xFFFFFFFF。因此如果用户希望定时器中断只产生一次，可以采用这种工作模式，这样就有足够的时间在下一个中断到来之前关闭定时器。
- **循环运行模式：**定时器计数初值始终为 TIMERx_TLC 内设定的值。因此如果用户希望定时器中断是周期性产生的，则可采用循环运行模式。

9.5. 信号及接口描述

表 9-1 TIMER 信号及接口描述表

名称	宽度	方向	描述	备注
CTL_APB 总线接口				内部信号
定时器信号				
timer0_clk	1	I	定时器 0 的输入时钟，来自 CPR，由 CPR_TIMER0_CLK_CTL 控制寄存器选择分频来源及分频比。	内部信号
timer1_clk	1	I	定时器 1 的输入时钟，来自 CPR，由 CPR_TIMER1_CLK_CTL 控制寄存器选择分频来源及分频比。	内部信号
timer2_clk	1	I	定时器 2 的输入时钟，来自 CPR，由 CPR_TIMER2_CLK_CTL 控制寄存器选择分频来源及分频比。	内部信号
timer3_clk	1	I	定时器 3 的输入时钟，来自 CPR，由 CPR_TIMER3_CLK_CTL 控制寄存器选择分频来源及分频比。	内部信号
AOtimer0_clk	1	I	AOTIMER0 的输入时钟，时钟源来自 32k	内部信号

AOtimer1_clk	1	I	AOTIMER1 的输入时钟，时钟源来自 32k	内部信号
中断				
timer0_intr	1	O	timer0 中断输出，送出给 M0。	内部信号
timer1_intr	1	O	timer1 中断输出，送出给 M0。	内部信号
timer2_intr	1	O	timer2 中断输出，送出给 M0。	内部信号
timer3_intr	1	O	timer3 中断输出，送出给 M0。	内部信号
Aotimer0_intr	1	O	Aotimer0 中断输出，送出给 M0。	内部信号
Aotimer1_intr	1	O	Aotimer1 中断输出，送出给 M0。	内部信号

9.6. TIMER 寄存器地址映射

- TIMER 的基地址为：0x40003000

表 9-2 timer 寄存器表

寄存器名	描述	方向&宽度	地址偏移	复位值
TIMER0 的寄存器				
TIMER0_TLC	计数值载入寄存器	RW 32bits	0x00	0x0000
TIMER0_TCV	当前计数值寄存器	R 32bits	0x04	0x0000
TIMER0_TCR	控制寄存器	RW 3bits	0x08	0x0
TIMER0_TIC	中断清除寄存器	R 1bit	0x0C	0x0
TIMER0_TIS	中断状态寄存器	R 1bit	0x10	0x0
TIMER1 的寄存器				
TIMER1_TLC	计数值载入寄存器	RW 32bits	0x14	0x0000
TIMER1_TCV	当前计数值寄存器	R 32bits	0x18	0x0000
TIMER1_TCR	控制寄存器	RW 3bits	0x1C	0x0
TIMER1_TIC	中断清除寄存器	R 1bit	0x20	0x0
TIMER1_TIS	中断状态寄存器	R 1bit	0x24	0x0
TIMER2 的寄存器				
TIMER2_TLC	计数值载入寄存器	RW 32bits	0x28	0x0000
TIMER2_TCV	当前计数值寄存器	R 32bits	0x2C	0x0000
TIMER2_TCR	控制寄存器	RW 3bits	0x30	0x0

TIMER2_TIC	中断清除寄存器	R 1bit	0x34	0x0
TIMER2_TIS	中断状态寄存器	R 1bit	0x38	0x0
TIMER3 的寄存器				
TIMER3_TLC	计数值载入寄存器	RW 32bits	0x3C	0x0000
TIMER3_TCV	当前计数值寄存器	R 32bits	0x40	0x0000
TIMER3_TCR	控制寄存器	RW 3bits	0x44	0x00
TIMER3_TIC	中断清除寄存器	R 1bit	0x48	0x00
TIMER3_TIS	中断状态寄存器	R 1bit	0x4C	0x00

9.7. TIMER 寄存器

9.7.1. 计数值寄存器（TIMER_TLC）

- 地址偏移：
  - TIMER0_TLC：0x00
  - TIMER1_TLC：0x14
  - TIMER2_TLC：0x28
  - TIMER3_TLC：0x3C

注意：只能在定时器不使能时，对该寄存器做“写”操作。在定时器已经使能后，对该寄存器进行“写”操作，会得到不可预知的结果。

位数	名称	方向	复位值	描述
31:0	TLC	RW	0x00	载入计数器计数初值：计数器计数的初值，该值应大于等于 0x4。

9.7.2. 当前计数值寄存器（TIMER_TCV）

- 地址偏移：
  - TIMER0_TCV：0x04
  - TIMER1_TCV：0x18
  - TIMER2_TCV：0x2C
  - TIMER3_TCV：0x40

位数	名称	方向	复位值	描述
31:0	TCV	R	0x00	当前计数器的值：显示当前计数器计数值。

9.7.3. 控制寄存器（TIMER_TCR）

- 地址偏移：
  - TIMER0_TCR：0x08
  - TIMER1_TCR：0x1C
  - TIMER2_TCR：0x30
  - TIMER3_TCR：0x44

注意：只能在定时器没有使能时，对该寄存器的 bit1 做写操作。在定时器已经使能后，对 bit1 进行写操作，会得到不可预知的结果。

位数	名称	方向	复位值	描述
2	TIM	RW	0x00	定时器中断屏蔽： 0：不屏蔽定时器中断。 1：屏蔽定时器中断。
1	TMS	RW	0x00	定时器模式选择： 0：单次运行模式。 1：循环运行模式。
0	TES	RW	0x00	定时器使能选择： 0：不使能定时器。 1：使能定时器。

9.7.4. 中断清除寄存器（TIMER_TIC）

- 地址偏移：
  - TIMER0_TIC：0x0C
  - TIMER1_TIC：0x20
  - TIMER2_TIC：0x34
  - TIMER3_TIC：0x48

位数	名称	方向	复位值	描述
0	TIC	R	0x00	中断清除：读该寄存器将清除定时器中断，因此该寄存器的值也被清 0。当计数器记到设定值时，中断又会被重新置起，此时该寄存器的值为 1。 1：定时器发生中断 0：定时器没有发生中断

9.7.5. 中断状态寄存器（TIMER_TIS）

- 地址偏移：
  - TIMER0_TIS：0x10
  - TIMER1_TIS：0x24
  - TIMER2_TIS：0x38
  - TIMER3_TIS：0x4C

位数	名称	方向	复位值	描述
0	TIS	R	0x00	中断状态标志：标志定时器的中断状态，若中断被屏蔽，则该值标志中断屏蔽后的中断状态。对该寄存器进行读操作不会影响定时器的中断。 1：定时器发生中断 0：定时器没有发生中断

9.8. 工作流程

用户在编写程序时建议采用以下流程：

上电后定时器处于不使能状态（即寄存器 TIMERx_TCR 位 TES=0），请顺序执行 2、3、4。如果已经对定时器做过某些操作，请先读 TIMERx_TCR 寄存器。若定时器在使能状态（即寄存器 TIMERx_TCR 位 TES=1），先写 TIMERx_TCR 寄存器，设置定时器在不使能状态，即写寄存器 TIMERx_TCR 位 TES=0，然后顺序执行 2、3、4。

(1)、写 TIMERx_TCR 寄存器 TMS 位，设置定时器的工作模式。

a)若需要设置定时器工作在用户定义工作模式。

i.写 TIMERx_TCR 寄存器 TMS=1。

ii.写 TIMERx_TLC 寄存器一个大于等于 0x4 的数，设置定时器计数初值。

iii.执行 3。

b)若需要设置定时器工作在单次运行模式。

i.写 TIMERx_TCR 寄存器 TMS=0。

ii.写 TIMERx_TLC 寄存器一个大于等于 0x4 的数，设置定时器计数初值。

iii.执行 3。

(2)、写 TIMERx_TCR 寄存器 TIM=0，设置定时器中断非屏蔽。

(3)、写 TIMERx_TCR 寄存器 TES 位，使能定时器。

(4)、定时器计数到 0，产生中断。

(5)、进入中断服务子程序，读 TIMER_TIC 寄存器，清除定时器中断。

注：本节寄存器 TIMERx 为 TIMER0，TIMER1，TIMER2，TIMER3。

第 10 章 看门狗定时器（WDT）

10.1. 概述

WDT 监视 M0，一旦出现 M0 死机，WDT 会产生系统复位信号送往 CPR，只要系统复位发生，CPR 会根据寄存器配置复位全芯片或者仅复位 M0，从而实现对系统的保护。

WDT 模块可以被 M0 访问。

- WDT 的基地址为：0x40004000

10.1.1. 主要特性

- 计数器宽度 32bits
- 有 2 种工作模式，用户可以根据需要选择不同的工作模式。
- 计数器计数到 0 时发生超时。
- 用户可以编程设置超时时间。
- 用户可以在 WDT 工作期间修改超时时间。
- 用户可以在任意时刻重启计数器。
- 用户可以通过写 CPR 模块的 CPR_RSTCTL_CTLAPB_SW 寄存器（模块软复位控制寄存器）使 WDT 复位。
- WDT 产生的系统复位信号送往 CPR，CPR 会根据 CPR_RSTCTL_WDTRST_MASK 寄存器（WDT 复位屏蔽寄存器）的配置来进行全芯片复位或者仅 M0 复位。
- 休眠模式下复位芯片。

10.1.2. 工作模式

WDT 有两种工作模式，用户可以设置控制寄存器 WDT_CR 的 RMOD 位来选择需要的工作模式。计数器向下计数，计数到 0，WDT 发生超时。发生超时后，根据用户设置的工作模式，WDT 会产生不同的响应。

当 WDT 使能后只能通过软件写 CPR 模块来复位。

工作模式 0：当计数器计到 0 时，直接产生复位信号送至 CPR 模块。

工作模式 1：这是 WDT 默认的工作模式。当计数器计到 0 时，发生超时中断，若当发生下一次超时时，软件仍未清除此中断，那么 WDT 就会产生复位信号送至 CPR 模块。

工作模式 0 和工作模式 1 产生的复位信号，送往 CPR，CPR 模块接收到 wdt_sys_rst_n 信号后根据 CPR_RSTCTL_WDTRST_MASK 寄存器（WDT 复位屏蔽寄存器）的配置来进行全芯片复位或者仅 M0 复位。

10.1.3. WDT 使能初期

图 10-1 中 wdt_pclk 是 WDT 工作时钟（更详细内容参见信号与接口描述）；当前计数 值寄存器 WDT_CCVR 反映的是计数器当前值，用户可以读此寄存器得到计数器当前值（更详细内容参见当前计数值寄存器 WDT_CCVR）。

为了说明使能初期 WDT 计数器的工作情况，以 WDT 在工作模式 1 时为例进行介绍。

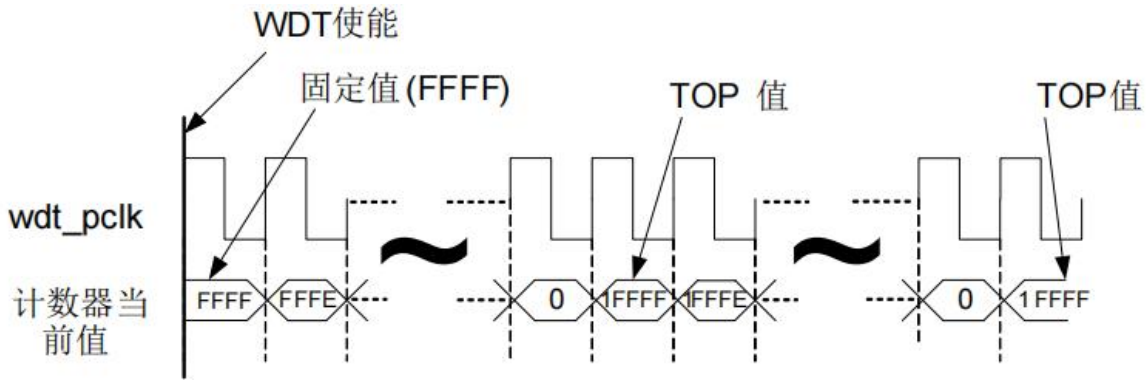

图 10-1 WDT 使能初期时序图

图中的参数设置：

- 超时范围寄存器 WDT_TORR 位 TOP=0001（计数器装载值为 0x1FFFF）
- 控制寄存器 WDT_CR 位 RMOD=1

WDT 上电后使能时，计数器第一次装载的值固定为 0xFFFF，第一次超时后计数器会 按照 TOP 值进行计数。

10.1.4. WDT 计数器装载

WDT 在 3 种情况下计数器会装载计数值：

- 芯片上电并且复位后，WDT 使能时，计数器第一次装载的值固定为 0xFFFF
- 以后再发生超时时，WDT 计数器会装载 TOP 位的值
- 计数器重启（写 0x76 到计数器重启寄存器 WDT_CRR）后，WDT 计数器装载入 WDT_TORR 寄存器 TOP 位所对应的计数值

当用户在 WDT 工作过程中修改 TOP 值时，要想立即生效，就重启计数器，WDT 计数 器会立刻装载修改后的 TOP 值；否则，就等到发生超时时，WDT 计数器会装载修改后的 TOP 值。

10.1.5. WDT 计数器重启

往计数器重启寄存器 WDT_CRR 中写入 0x76 就能重启 WDT 计数器，写入其它值无效。重启计数器对 WDT 有两个重要影响：

- 重启计数器可以清除 WDT 中断
- 重启计数器会使 WDT 读超时范围寄存器 WDT_TORR 的 TOP 位，然后把 TOP 位的值装入计数器

因此，当用户写 WDT_CRR=0x76 时，如果此时有中断，WDT 会先清除中断，然后把 TOP 位的值装入计数器。

10.2. 中断模式

用户选择 WDT 工作模式 1 时，当计数器计数到 0 发生超时时，会产生 WDT 中断。WDT 中断不影响计数器的计数。

发生 WDT 中断后，下列 3 种方式可以清除中断：

1.读中断清除寄存器 WDT_ICR 可以清除中断。这种清除中断的方法不会影响计数器的计数。

2.写 0x76 到计数器重启寄存器 WDT_CRR 可以清除中断。这种方法会影响计数器 的计数值。因为 WDT 清除中断后会读超时范围寄存器 WDT_TORR 的 TOP 位，并装载 TOP 的值。

3.写 CPR 模块的 CPR_RSTCTL_CTLAPB_SW 可以复位 WDT，此复位信号可以清 除中断。

10.2.1. M0 复位

WDT 的中断送给 M0，如果没有得到 M0 的响应，那么 WDT 就会发出复位信号送给 CPR 模块。如果需要对 WDT 进行复位，就需要软件对 CPR 模块进行操作以实现对 WDT 的复位。

芯片上电后，当用户选择工作模式 1，并使能 WDT 后，先发生超时中断，若 M0 在下一次超时来临时还没有清除此中断，则产生 M0 复位信号 wdt_sys_rst_n 并送给 CPR 模块，CPR 模块接收到 wdt_sys_rst_n 信号后根据 CPR_RSTCTL_WDTRST_MASK 寄存器（WDT 复位屏蔽寄存器）的配置来进行全芯片复位或者仅 M0 复位。

wdt_pclk 是 WDT 工作时钟（更详细内容参见信号与接口描述）；当前计数值寄存器 WDT_CCVR 反映的是计数器当前值，用户可以读此寄存器得到计数器当前值（更详细内容参见当前计数值寄存器 WDT_CCVR）；wdt_intr 是 WDT 的中断输出信号（更详细内容参见信号与接口描述）；读中断清除寄存器表示清除当前中断（更详细内容参见中断清除寄存器 WDT_ICR）；wdt_sys_rst_n 是 WDT 输出给 CPR 模块的复位信号（更详细内容参见信号与接口描述）。

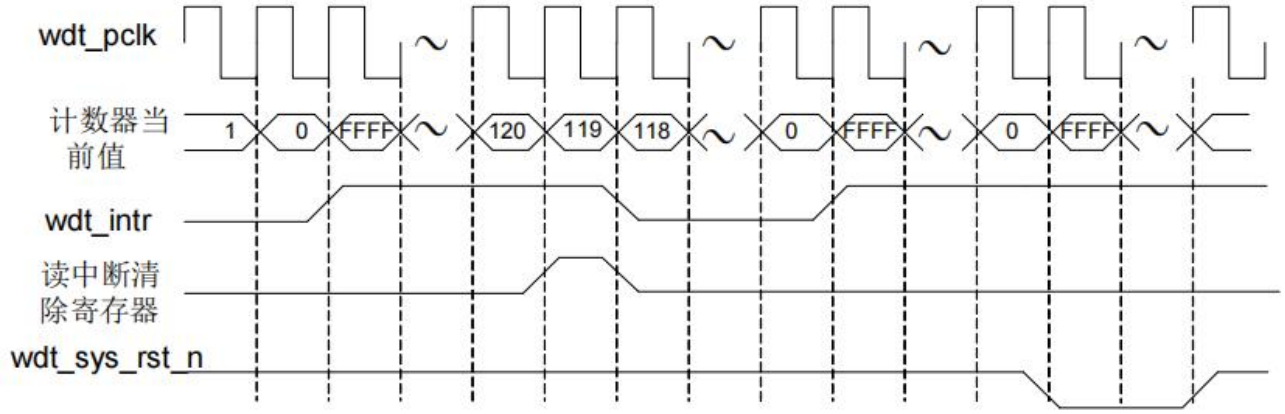

图 10-2　WDT 在工作模式 1 下的 M0 复位图

图 10-2 说明了 WDT 在工作模式 1 下 M0 的复位情况。

上图中参数设置：

超时范围寄存器 WDT_TORR 位 TOP=0000（计数器装载值为 0xFFFF）

超时范围寄存器 WDT_CR 位 RMOD=1

图 10-2 中发生第一次超时时 wdt_intr 变高，软件读中断清除寄存器后清除 WDT 中断。第二次超时时 wdt_intr 中断将一直保持，直到三次超时来临，此时，wdt_sys_rst_n 变低，M0 复位信号有效，wdt_intr 中断仍然有效，WDT 计数器继续计数。wdt_sys_rst_n 保持 N 个 wdt_pclk 时间后会自动变高，其中 N 由 WDT 控制寄存器（WDT_CR）的 RPL 决定，WDT 仍然在计数，当计数到 0，发生超时中断，计数器重载 TOP 值。

10.3. 信号及接口描述

WDT 信号及接口描述图 10-3 所示：

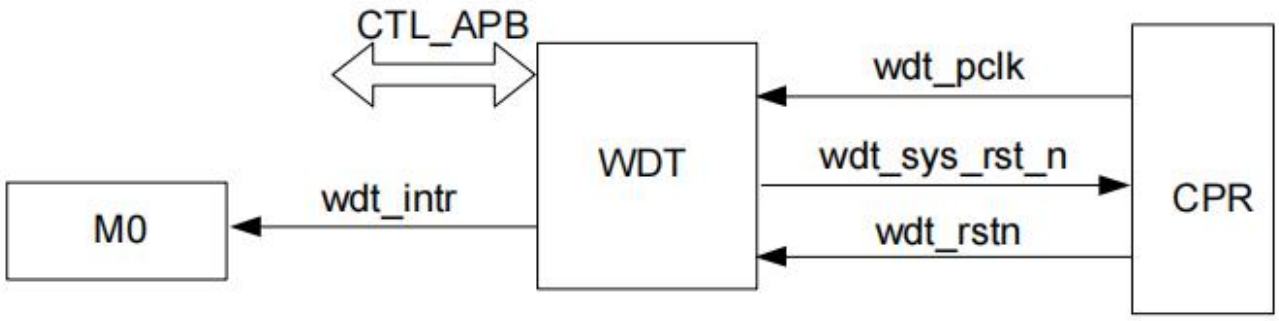

图 10-3 WDT 信号及接口描述图

WDT 信号及接口描述表如表 10-1 所示：

表 10-1 WDT 信号及接口描述表

名称	宽度	方向	描述	备注
CTL_APB 总线接口				内部信号
时钟接口				
Wdt_clk	1	I	WDT 总线时钟，来自 CPR 模块，时钟频率可以通过软件对 CPR 模块进行配置	内部信号
复位信号				
wdt_sys_rst_n	1	O	M0 复位信号，输出给 CPR 模块	内部信号
wdt_rstn	1	I	来自 CPR 的复位 WDT 的信号	内部信号
中断				
Wdt_intr	1	O	WDT 中断输出，输出给 M0	内部信号

10.4. 看门狗寄存器地址映射

- WDT 的基地址为：0x40004000

表 10-2 WDT 寄存器地址映射表

寄存器名	描述	方向&宽度	地址偏移	复位值
WDT_CR	控制寄存器	RW 5bits	0x00	0x00
WDT_TORR	超时范围寄存器	RW 4bits	0x04	0x0
WDT_CCVR	当前计数值寄存器	R 32bits	0x08	0xFFFF
WDT_CRR	计数器重启寄存器	W 8bits	0x0C	0x0
WDT_STAT	中断状态寄存器	R 1bit	0x10	0x0
WDT_ICR	中断清除寄存器	R 1bit	0x14	0x0

10.5. 看门狗寄存器

10.5.1. 控制寄存器（WDT_CR）

- 地址偏移：0x00

位数	名称	方向	复位值	描述
31：5	NA	--	--	保留
4：2	RPL	RW	0x00	复位脉冲宽度。 M0 复位信号，维持多少个 pclk 时钟周期 000：2pclk 时钟周期 001：4pclk 时钟周期 010：8pclk 时钟周期 011：16pclk 时钟周期 100：32pclk 时钟周期 101：64pclk 时钟周期 110：128pclk 时钟周期 111：256pclk 时钟周期
1	RMOD	RW	0x00	工作模式。 WDT 发生超时的时候有两种不同的响应，由此产生了 WDT 的两种工作模式。用户可以选择自己需要的工作模式。 0：工作模式 0，直接产生复位信号送给 CPR 1：工作模式 1，先产生一个中断，如果在下一个超时来临之前软件没有清除中断，那么会产生复位。
0	WDT_EN	RW	0x00	WDT 使能。 软件设置 WDT 使能后，任何对该位写 0 的操作都是无效操作。该位只能被 M0 复位清除。 0：WDT 不使能 1：WDT 使能

10.5.2. 超时范围寄存器（WDT_TORR）

- 地址偏移：0x04

位数	名称	方向	复位值	描述
31：4	NA	--	--	保留。
3：0	TOP	RW	0x00	设置超时周期：此字段用于设置 WDT 装载的计数器值。如果用户想在计数器计数过程中修改此字段，重启计数器或发生超时，修改后的值都会被装载入计数器。更详细的描述见 WDT 计数器装载，用户只有以下 16 种选择，每种选择对应的计数器值如下： 0000：0xFFFF 0001：0x1FFFF 0010：0x3FFFF 0011：0x7FFFF 0100：0xFFFFF 0101：0x1FFFFF 0110：0x3FFFFF 0111：0x7FFFFF 1000：0xFFFFFF 1001：0x1FFFFFF 1010：0x3FFFFFF 1011：0x7FFFFFF 1100：0xFFFFFFF 1101：0x1FFFFFFF 1110：0x3FFFFFFF 1111：0x7FFFFFFF

10.5.3. 当前计数值寄存器（WDT_CCVR）

- 地址偏移：0x08

位数	名称	方向	复位值	描述
31：0	CCVR	R	0xFFFF	当前计数值：记录当前计数器的计数值。

10.5.4. 计数器重启寄存器（WDT_CRR）

- 地址偏移：0x0C

位数	名称	方向	复位值	描述
31：8	NA	--	--	保留。
7：0	CPR	W	0x0	重启计数器。 用户可向该字段写入 0x76，计数器将会重启，写其它任何值对该字段没有影响。用户可以在任何时候对该寄存器进行写操作。重启计数器会清除当前中断。更详细的描述见 WDT 计数器重启。

10.5.5. 中断状态寄存器（WDT_STAT）

- 地址偏移：0x10

位数	名称	方向	复位值	描述
31：1	NA	--	--	保留。
0	STAT	R	0x00	当前中断状态：显示 WDT 当前中断状态。 0：没有发生中断 1：发生中断

10.5.6. 中断清除寄存器（WDT_ICR）

地址偏移：0x14

位数	名称	方向	复位值	描述
31：1	NA	--	--	保留。
0	ICR	R	0x00	清除中断：读此寄存器会清除 WDT 当前中断。

10.6. 工作流程

下面给出 WDT 工作模式 1 的操作流程，其他 WDT 流程类似：

(1) 芯片上电，芯片复位；

(2) 写 CPR 模块寄存器 CPR_RSTCTL_CTLAPB_SW 的 WDT_RSTN 位为 1，复位 WDT；

(3) 写 CPR 模块寄存器 CPR_RSTCTL_CTLAPB_SW 的 WDT_RSTN 位为 0，恢复 WDT 的正常工作；

(4) 写 CPR 模块寄存器 RSTCTL_WDTRST_MASK，配置 M0 复位还是全芯片复位；

(5) 写 CPR 模块寄存器 LP_CTL，配置 wdt_pclk_en=1，使用 32k 时钟作为 wdt 定时器的时钟；

(6) 写 CPR 模块寄存器 LP_CTL，配置 wdt_tclk_en=1，使能 wdt 定时器的时钟；

(7) 写控制寄存器 WDT_CR 的 RMOD=1，设置 WDT 工作模式 1；

(8) 写超时范围计数器 WDT_TORR 的 TOP 位，设置超时时间，第一次计数值为 0xffff；

(9) 写控制寄存器 WDT_CR 的 WDT_EN=1，使能 WDT；

(10) 计数器开始计数，当计数器计数到 0 时，发生超时中断，进入中断服务子程序；

(11) 在 WDT 中断复位程序中，通过读中断清除寄存器 WDT_ICR 来清除中断；若 M0 在下一次超时来临时还没有清除此中断，则产生 M0 复位信号。如需要重新设置超时时间，则执行 11。否则跳过 11、12 两步从 9 开始执行；

(12) 如果需要修改超时时间，写超时范围寄存器 WDT_TORR 的 TOP 位，若需要超时时间立即生效则执行 12，否则新的超时时间在超时中断发生后生效，直接从 9 开始执行；

(13) 写计数器重启寄存器 WDT_CRR=0x76，然后从 9 开始执行。

第 11 章 模数转换（ADC）

11.1. 概述

通用 ADC 接口模块，数据采集精度为 12bits。RF 侧通用 ADC 包括 4 个输入通道，接口模块可以控制 4 通道的实时切换。以下所说“通道”，均指的是 RF 侧 ADC 的输入通道。通用 ADC 接口模块接收 RF 侧通用 ADC 某一通道的数据后，会把数据和通道信息一起，按一定格式缓存在 FIFO 中。FIFO 数据可以被 DMA 或者 CPU 读取。

注意事项：

- adc 一共 16 个通道，其中 ch0~ch7，ch12,ch13 通道为外部输入通道；
  ch8 用来测试内部 1/2*AVDD；
  ch9 用来测试 1/3*LDO_IN(锂电池可用)；
  ch10 用来测试温度传感器
  ch11 通道接 PGA 输出。PGA 增益 0~42.4dB,1.6dB step，支持差分和单端输入。
  ch12 接比较器负端；
- micp 接通道 4 即 GPIO0，micn 接通道 3 即 GPIO18。运放输入正端接 ch13,负端接 ch14，输出端接 ch15，支持正负端可切换。
- 16 个周期出一个数，adc 时钟最高支持 8MHz，默认配置成 1MHz，一般建议配置成 1/2/4MHz。adc 满量程电压范围可以配置成 0~2.4V 或者 0~AVDD。默认配置成 0~2.4V（电源电压 3.3V）。
- 常温下理论精度 1.3%，实测精度<2%。GADC_VREF_SEL（0x24<1>）默认是 0，0~2.4V，改成 1，0~AVDD。Adc 满量程电压和电源电压有关。在电源电压 3.3V 下为 2.4V。电源电压每降低 0.1V，Adc 满量程电压降低 0.0064V。(但是在 3.6V 下例外，为 2.5749V)。
- 通道设置后多久可以进行采样操作。默认设置 adc 时钟频率 1M，采样一个数据为 16 个时钟周期(16us)，adc fifo 容量 32 个数据。Time_wait = 16*32 = 512us,所以设置 0.512ms 后可以进行 adc 的数据采集，驱动会对 32 个数据取平均得出一个数据从而减少误差。采用中断方式采集可以节省部分时间。

11.1.1. 主要特性

- 12 个外部电压采样通道、4 个内部采样通道。
- 支持 4 通道实时切换，可以通过配置寄存器，实时切换通道。
- 硬件自动切换通道功能、软件自动切换通道功能。

- 可选 PGA，提供增益调节选择。
- 支持电池电压，核心温度检测。
- 数据采样时钟沿选择功能。
- 自动滤除通道切换期间无效数据功能。
- 深度 16，位宽 32bit 的数据 FIFO

11.1.2. 功能描述

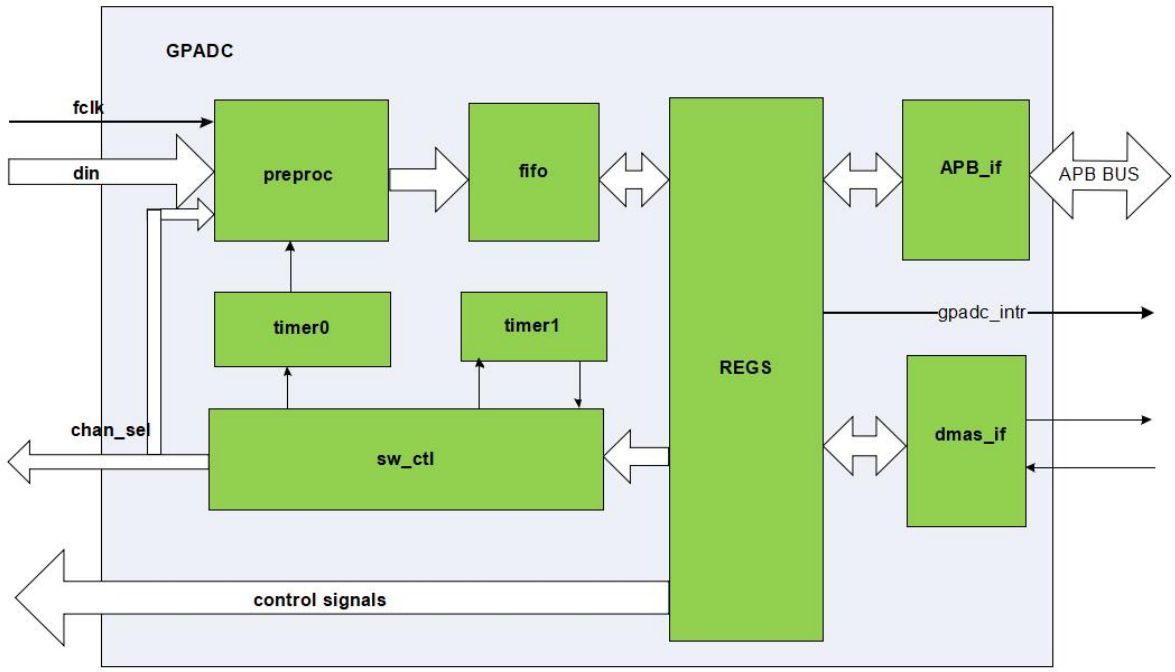

图 11-1 ADC 模块结构框图

ADC 模块结构框图如上图所示，fclk 为随路时钟，din[9:0]为输入数据，chan_sel[3:0]为 RF ADC 的输入通道选择信号，control signals 为 ADC 接口模块控制 RF ADC 的控制信号，包括以下信号：gpadc_ctl_mux、rst_gpadc、diffsel_gpadc、pd_vcmgpadc、pd_gpadc，以上这些信号的详细说明，参考信号及接口描述。

11.1.3. FIFO 数据格式

通用ADC 接收模块接收 12bit 宽的RF 侧通用 ADC 数据后，添加通道信息，格式如下图 11-2 所示，存储于FIFO 中（FIFO 的位宽x 深度为 32x16)，其中，先到的 RF 侧通用 ADC 数据（din[11:0]）和通道信息（chan_sel[3:0]）存在高位。

31		16	0
Chan_sel[3:0]	Din[11:0]	Chan_sel[3:0]	Din[11:0]

图 11-2 FIFO 缓存数据格式

11.1.4. 硬件自动切换通道功能

当 GPADC_MAIN_CTL.AUTO_SW=0 时，通道自动切换功能关闭，输出给 ADC 的通道选择信号由寄存器决定；

当 GPADC_MAIN_CTL.AUTO_SW=1 时，通道自动切换功能打开，输出给 ADC 的通道选择信号每隔一定的时间自动切换，切换时间间隔由计时器 timer1 决定。通过 GPADC_CHAN_CTL.CHAN_AUTO[3:0]和 GPADC_CHAN_CTL.VCM_CHAN_AUTO[1:0] 选择参与自动切换的通道，其中，CHAN_AUTO[0]~ CHAN_AUTO[3]分别对应单端输入通道 1、2、3、4；VCM_CHAN_AUTO[0]和 VCM_CHAN_AUTO[1]分别对应差分输入的通道 1、2 差分和通道 3、4 差分。

例如：当 CHAN_AUTO[3:0] = 4’b1111, VCM_CHAN_AUTO=2’b11 时，chan_sw[3:0] 循环输出 4’b0001 -> 4’b0010 -> 4’b0100 -> 4’b1000 -> 4’b0011 -> 4’b1100 ；当 CHAN_AUTO[3:0] = 4’b1111, VCM_CHAN_AUTO=2’b00 时，chan_sw[3:0] 循环输出 4’b0001 -> 4’b0010 -> 4’b0100 -> 4’b1000 ； 当 CHAN_ATUO=4’b0000 ，VCM_CHAN_AUTO=2’b11 时，chan_sw[3:0] 循环输出 4’b0011 -> 4’b1100。

11.1.5. 数据采样沿选择功能

通过 ADC 接口模块支持上升沿和下降沿采样数据，可以根据实际需要，配置寄存器 ADC_MAIN_CTL，设置为上升沿或者下降沿采集数据。

11.1.6. 自动滤除通道切换期间无效数据无效数据功能

由于 RF 侧通用 ADC 在通道切换期间，可能产生无效数据，所以通用 ADC 接口模块支持无效数据的自动滤除功能。配置寄存器 ADC_TIMER0，设置通道切换等待时间，每次发生通道切换时，接口模块在等待时间内，不接收数据。设置等待时间为 0，则该功能无效。

11.2. 信号及接口描述

ADC 信号及接口描述如下表所示。

表 11-1 ADC 接口信号描述

名称	宽度	方向	描述	备注
rstn	1	I	复位信号，低有效	内部信号
APB 从设备接口				内部信号
射频接口信号				
fclk	1	I	RF GPADC 随路时钟	内部信号
din	10	I	RF GPADC 输出数据	内部信号
gpadc_ctl_mux	1	0	RF GPADC 控制信号选择: 0：SPI 1：DIGITAL GPADC	内部信号
rst_gpadc	1	0	RF_GPADC 复位信号	内部信号
diffsel_gpadc	1	0	RF GPADC 输入模式控制： 0：差分输入 1：单端输入；	内部信号
pd_vcmgpadc	1	0	RF GPADC 差分输入 buffer 断电指示： 0：不断电 1：断电	内部信号
pd_gpadc	1	0	ADC 断电指示： 0：不断电 1：断电	内部信号
chan_sel	4	0	RF GPADC 输入通道选择，共 4 个通道，每 bit 对应一个通道： 0：关闭 1：打开	内部信号
其他信号				
gpadc_intr	1	0	中断信号	内部信号
dmas_rx_req	1	0	DMA 读数据请求信号	内部信号
dmas_rx_ack	1	I	DMA 读数据完成信号	内部信号

11.3. ADC 寄存器地址映射

- ADC 的基地址：0x40018000

表 11-2 ADC 寄存器表

寄存器名	描述	方向&宽度	地址偏移	复位值
ADC_MAIN_CTL	主要控制寄存器	RW 16bits	0x00	0x8
ADC_CHAN_CTL	ADC 通道控制寄存器	RW 12bits	0x04	0x0
ADC_FIFO_CTL	FIFO 控制寄存器	RW 4bits	0x08	0x8
ADC_TIMER0	通道切换时间计数器	RW 8bits	0x0C	0x0
ADC_TIMER1	通道自动切换时间间隔计数器	RW 16bits	0x10	0x0
ADC_INT	中断状态寄存器	RC1 2bits	0x14	0x0
ADC_INT_RAW	中断原始状态寄存器	R 2bits	0x18	0x0
ADC_INT_EN	中断使能寄存器	RW 2bits	0x1C	0x0
ADC_FIFO	FIFO 读数据接口	R 32bits	0x20	0x0
ADC_RF_CTL	ADC 控制寄存器	RW 5bits	0x24	0x801

11.4. ADC 寄存器描述

11.4.1. 主要控制器（ADC_MAIN_CTL）

- 地址偏移：0x00

位数	名称	方向	复位值	描述
15:12	EXT_SAMPLE_NUM	RW	0X02	选择每次外部 trigger 信号满足采样的数据数目 0：每次外部触发采样 1 个数据 n:每次外部触发满足，采样 n+1 个数据 注：需要和 fifo 阈值寄存器配合产生中断
11	EXT_DATA_MODE	RW	0X00	0：每个 32bit 包括{din1,din0}两组信号。 1：每个 32bit 包括 din 一组信号，高 16bit 无效
10:8	EXT_TRIGGER_SEL	RW	0X00	选择外部 trigger 信号 0~5：pwm0~pwm5 6：pwm_brk[1] 7：pwm_brk[2]
5：4	EXT_EDGE_SEL	RW	0x00	选择数据采样使用的触发源 0：内部自动采样 1：外部 trigger 信号上升沿触发采样 2：外部 trigger 信号上升沿触发采样 3：外部 trigger 信号上升和下降沿触发采样 注：外部信号可来自于 pwm 或外部管脚，在 pwm 模块中配置源头。
3	EDGE_SEL	RW	0x00	数据采样沿选择： 0：下降沿 1：上升沿
2	AUTO_SW	RW	0x01	ADC 输入通道自动切换功能： 0：关闭 1：打开
1	DMAS_ON	RW	0x00	DMAS_ON 功能开关： 0：关闭 1：打开
0	ADC_EN	RW	0x01	ADC 模块开关： 0：关闭 1：打开

11.4.2. ADC 通道控制寄存器（ADC_CHAN_CTL）

- 地址偏移：0x04

位数	名称	方向	复位值	描述
31：12	NA	--	--	保留
11：8	CHAN_AUTO	RW	0x00	单端输入时，选择哪个通道参与自动通道切换功能，每 1bit 对应一个通道 0：不选择 1：选择
7：6	NA	--	--	保留
5：4	VCM_CHAN_AUTO	RW	0x00	差分输入的自动切换功能使能， bit0 对应通道 1 和 2 的差分， bit1 对应通道 3 和 4 的差分
3：0	SELECT_CHAN	RW	0x0	通道选择 0：选择 0 通道 ... 13：选择 13 通道

11.4.3. FIFO 控制寄存器（ADC_FIFO_CTL）

- 地址偏移：0x08

位数	名称	方向	复位值	描述
31：5	NA	--	--	保留
4	FIFO_FLUSH	RW	0x00	清空 FIFO： 0：不清空 1：清空
3：0	READ_REQ_THRESH	RW	0x8	读请求中断阈值

11.4.4. 通道切换时间计数器（ADC_TIMER0）

- 地址偏移：0x0C

位数	名称	方向	复位值	描述
31：8	NA	--	--	保留
7：0	SW_WAIT_CNT	RW	0x00	通道切换等待时间=总线时钟周期 默认值为 0us

11.4.5. 通道自动切换时间间隔计数器（ADC_TIMER1）

- 地址偏移：0x10

位数	名称	方向	复位值	描述
31：16	NA	--	--	保留
15：0	AUTO_SW_CNT	RW	0x00	通道自动切换时间间隔=总线时钟周期

11.4.6. 中断状态寄存器（ADC_INT）

- 地址偏移：0x14

位数	名称	方向	复位值	描述
31：2	NA	--	--	保留
1	FIFO_ERROR_INT	RW	0x00	FIFO 错误中断。 向本比特位写‘1’清除本中断状态位，同时清除相应的中断原始状态位。 0：没有读空或溢出错误中断 1：产生了读空或溢出错误中断
0	READ_REQ_INT	RW	0x00	FIFO 读请求中断。 向本比特位写‘1’清除本中断状态位，同时清除相应的中断原始状态位。 0：没有产生 FIFO 读请求中断 1：产生了 FIFO 读请求中断

11.4.7. 中断原始状态寄存器（ADC_INT_RAW）

- 地址偏移：0x18

位数	名称	方向	复位值	描述
31：2	NA	--	--	保留
1	FIFO_ERROR_RAW	R	0x00	FIFO 错误原始中断
0	READ_REQ_RAW	R	0x00	FIFO 读请求原始中断。

11.4.8. 中断使能寄存器（ADC_INT_EN）

- 地址偏移：0x1C

位数	名称	方向	复位值	描述
31：2	NA	--	--	保留
1	FIFO_ERROR_EN	RW	0x00	FIFO 溢出中断使能 0：不使能 1：使能
0	READ_REQ_EN	RW	0x00	FIFO 读请求中断使能 0：不使能

				1：使能

11.4.9. FIFO 读数据接口（ADC_FIFO）

- 地址偏移：0x20

位数	名称	方向	复位值	描述
31：0	FIFO_DOUT	R	0x0	FIFO 数据输出

11.4.10. ADC 控制寄存器（ADC_RF_CTL）

- 地址偏移：0x24

位数	名称	方向	复位值	描述
15：8	ADC_clkdiv	RW	0x08	ADC_CLK 分频比。 分频比为(div+1)*2。
4	adc_ctl_mux	RW	0x00	射频 ADC 控制信号源选择： 0：SPI 控制 1：数字 ADC 控制
3	rst_gpadc	RW	0x00	射频 ADC 复位控制信号
2	diffsel_gpadc	RW	0x00	射频 ADC 差分模式选择信号
1	gadc_vref_sel	RW	0x0	射频 ADC 测量电压范围选择信号 0：0-2.4V 1：0-AVDD
0	pd_gpadc	RW	0x1	射频 ADC 断电信号 0：不断电 1：断电

注：如果主频 32M，那么 adc 的频率是 16M，当 div=0 时，分频比为 2，adc 的采样频率则是 16M/2=8M；

11.5. 工作流程

ADC 的 4 个输入通道对应射频的 4 个管脚 ADC IN0~3，这 4 个管脚和数字管脚 GPIO13~16 复用，所以在使用 ADC 前，需要根据选择的输入通道，配置相应的 GPIO 管脚为输入模式。

ADC 的工作模式分为软件选择通道和硬件自动切换通道两种种工作模式，其中，硬件自动切换通道模式支持对应单端输入的单通道间的自动切换，比如 1->2->3->4；支持对应差分输入的双通道间的自动切换，比如 1+2->3+4；还支持单通道和双通道间的混合自动切换，比如 1+2->3->4。

11.5.1. 管脚复用的配置

由于 ADCIN0~4 分别与 GPIO13~16 复用管脚，ADC 选择使用某条通道时，需要将对应的 GPIO 管脚配置为输入方向，并且还要配置 RF 侧相关寄存器。

(1)、如果选择通道 0（对应管脚 ADCIN0）做为输入通道

a)配置 GPIO 寄存器 GPIO_PORT_DDR0[29]=1'b1

b)配置 GPIO 寄存器 GPIO_PORT_DDR0[13]=1'b0

c)配置射频侧寄存器 AUX_CTRL.sel_gpadc=4'b0001

(2)、如果选择通道 1（对应管脚 ADCIN1）做为输入通道

a)配置 GPIO 寄存器 GPIO_PORT_DDR0[30]=1'b1

b)配置 GPIO 寄存器 GPIO_PORT_DDR0[14]=1'b0

c)配置射频侧寄存器 AUX_CTRL.sel_gpadc=4'b0010

(3)、如果选择通道 2（对应管脚 ADCIN2）做为输入通道

a)配置 GPIO 寄存器 GPIO_PORT_DDR0[31]=1'b1

b)配置 GPIO 寄存器 GPIO_PORT_DDR0[15]=1'b0

c)配置射频侧寄存器 AUX_CTRL.sel_gpadc=4'b0100

(4)、如果选择通道 3（对应管脚 ADCIN3）做为输入通道

a)配置 GPIO 寄存器 GPIO_PORT_DDR1[16]=1'b1

b)配置 GPIO 寄存器 GPIO_PORT_DDR1[0]=1'b0

c)配置射频侧寄存器 AUX_CTRL.sel_gpadc=4'b1000

(5)、如果选择通道 0 和 1 差分输入（对应管脚 ADCIN0 和 ADCIN1）

a)配置 GPIO 寄存器 GPIO_PORT_DDR0[30:29]=2'b11

b)配置 GPIO 寄存器 GPIO_PORT_DDR0[14:13]=2'b00

c)配置射频侧寄存器 AUX_CTRL.sel_gpadc=4'b0011

(6)、如果选择通道 2 和 3 差分输入（对应管脚 ADCIN2 和 ADCIN3）

a)配置 GPIO 寄存器 GPIO_PORT_DDR0[31]=1'b1

c)配置 GPIO 寄存器 GPIO_PORT_DDR1[16]=1'b1

d)配置 GPIO 寄存器 GPIO_PORT_DDR1[0]=1'b0

e)配置射频侧寄存器 AUX_CTRL.sel_gpadc=4'b1100

11.5.2. 软件选择通道的工作流程

(1) 配置寄存器 CPR_RSTCTL_CTLAPB_SW（0x40000100）：先使 ADC_RSTN=0，再使 ADC_RSTN=1，软复位 ADC 模块；

(2) 配置寄存器 CPR_CTLAPBCLKEN_GRCTL（0x40000070）：ADC_PCLK_EN=1，使能 ADC 模块的时钟；

(3) 配置寄存器 ADC_TIMER0，因为 RFADC 在切换通道期间，输出数据不可用，通过配置寄存器 ADC_TIMER0，定义通道切换开始至输出有效数据的时间，在此期间，DIGITALADC 不接收数据；

(4) 配置寄存器 ADC_FIFO_CTL：先使 FIFO_FLUSH=1，再使 FIFO_FLUSH=0，对 FIFO 进行一次清空操作；然后根据需求，配置读请求中断阈值（READ_REQ_THRESH[3:0]），默认值为 8；

(5) 配置寄存器 ADC_CHAN_CTL 的 bit0~bit3，根据需要，打开相应通道开关；

(6) 配置寄存器 ADC_RF_CTL：gpadc_mux_ctl=1，pd_gpadc=0，根据 RFGPADC 输入需求，配置其他 bit 位；

(7) 配置寄存器 ADC_MAIN_CTL：ADC_EN=1，DMAS_ON=1，AUTO_SW=0，根据实际需要，配置 EDGE_SEL，选择采样数据的时钟边沿。

11.5.3. 硬件自动切换通道的工作流程

(1) 配置寄存器 CPR_RSTCTL_CTLAPB_SW（0x40000100）：先使 ADC_RSTN=0，再使 ADC_RSTN=1，软复位 ADC 模块；

(2) 配置寄存器 CPR_CTLAPBCLKEN_GRCTL（0x40000070）：ADC_PCLK_EN=1，使能 ADC 模块的时钟；

(3) 配置寄存器 ADC_TIMER0，因为 RFADC 在切换通道期间，输出数据不可用，通过配置寄存器 ADC_TIMER0，定义通道切换开始至输出有效数据的时间，在此期间，DIGITALADC 不接收数据；

(4) 配置寄存器 ADC_FIFO_CTL：先使 FIFO_FLUSH=1，再使 FIFO_FLUSH=0，对 FIFO 进行一次清空操作；然后根据需求，配置读请求中断阈值（READ_REQ_THRESH[3:0]），默认值为 8；

(5) 配置寄存器 ADC_CHAN_CTL：bit4~bit5(VCM_CHAN_AUTO[1:0])选择参与自动循环切换的差分通道，bit8~bit11（CHAN_AUTO[3:0]）选择参与自动循环切换的单端输入通道；

(6) 配置寄存器 ADC_TIMER1，设置自动切换时，每个通道的打开时间；

(7) 配置寄存器 ADC_RF_CTL：gpadc_mux_ctl=1，pd_gpadc=0，根据 RFADC 输入需求，配置其他 bit 位；

(8) 配置寄存器 ADC_MAIN_CTL：ADC_EN=1，DMAS_ON=1，AUTO_SW=1，根据实际需要，配置 EDGE_SEL，选择采样数据的时钟边沿。

第 12 章 实时计数器（RTC）

12.1. 概述

芯片内部集成了一个实时时钟（RTC 模块），可以实现天、时、分、秒的显示。产生定时中断等功能，RTC 也集成了一个可配置的 16 位计数器、对 5 个 GPIO 的输入检测功能和 32K 时钟校准计数功能。RTC 模块可被 M0 访问。

- RTC 的基地址为：0x40002000

12.1.1. 主要特性

- 各中断可分别屏蔽
- 可直接读取日、时、分、秒。
- 可产生三个定时中断。
- 可任意指定一周中的某一天或几天发生中断。
- 用户可以控制计数器是否使能。
- 提供 1 个可配置的 16 位计数器。
- 提供 5 个独立的 GPIO 输入检测功能。
- 提供 32K 时钟校准计数功能。

12.1.2. 功能描述

RTC 的基本功能为：提供一个长时期稳定运行的时钟，软件通过访问 RTC 模块的寄存器，可以确定当前的系统时间。

它的工作时钟为 32768Hz，计时可以精确到日、时、分、秒。软件可以通过设置相应的寄存器，控制 RTC 在指定的时间产生相应类型的中断。

RTC 内部集成了 1 个可配置的 16 位计数器（AO_TIMER），工作时钟为 32768Hz，提供 2 秒内的定时功能，可独立产生中断。

RTC 内部集成了对 5 个 GPIO 管脚的输入检测功能，可检测每个 GPIO 的电平、上升沿、下降沿等并触发中断，可配置对输入信号的去毛刺（Debounce）功能。RTC 内部集成了对提供 32K 时钟校准计数功能，使用高速时钟对 32K 时钟进行采样计数，确定 32K 时钟频率的偏差。

RTC 的组成部分主要包括：

- RTC 计数器
- AO_TIMER 计数器
- 32K 校准计数器

- GPIO 检测
- 总线接口
- 控制寄存器组
- 中断产生模块

M0 通过对寄存器的读写，控制计数器的工作，并设置中断产生的条件，读出时间信息。下图为 RTC 的结构框图：

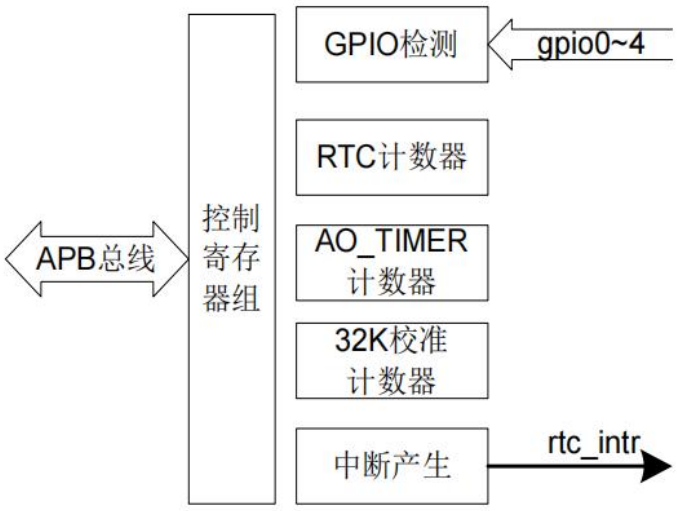

图 12-1 RTC 结构框图

12.1.3. 工作模式

12.1.3.1. 实时计时

实时计时功能提供一个长时期稳定运行的时间，软件通过访问寄存器可以确定当前的系统时间。

RTC 实时计时功能共可以产生七种中断，包括：

- 定时中断 1：中断产生的条件由定时匹配寄存器 1（RTC_CMR_ONE）设定。
- 定时中断 2：中断产生的条件由定时匹配寄存器 2（RTC_CMR_TWO）设定。
- 定时中断 3：中断产生的条件由定时匹配寄存器 3（RTC_CMR_THREE）设定。
- 秒中断：RTC_CCVR 中秒计数值变化时，中断条件满足。
- 分中断：RTC_CCVR 中分计数值变化时，中断条件满足。
- 小时中断：RTC_CCVR 中小时计数值变化时，中断条件满足。
- 天中断：RTC_CCVR 中天计数值变化时，中断条件满足。

这些中断送出给中断控制器 RTC_ICR，可被中断屏蔽位屏蔽，并具有各自的中断使能位和中断清除寄存器。用户可读取中断状态寄存器从而判断发生了哪个中断，并通过写相应的中断清除寄存器清除中断。

- 中断的使能
  - 用户通过写中断控制寄存器 RTC_ICR 的 0 到 6 位，可以控制相应的中断是否使能。用户也可以同时使能多种中断。
- 中断的屏蔽
  - 用户可设置中断控制寄存器 RTC_ICR 的 MASK 位从而设置是否屏蔽中断。
- 中断的清除
  - 用户通过写中断清除寄存器 RTC_EOI 的相应位，可清除相应的中断。
- 中断的产生
  - 中断使能后，相应的中断条件满足时，RTC 就会发出中断。

12.1.3.2. 16 位计数器

RTC 内部提供 16 位可配置的计数器（AO_TIMER），计数器在使能之后循环加 1，计数到配置的初始值时，产生中断。

配置 AO_TIMER_CTL.AO_TIMER_EN=0 不使能计数器可清除中断，计数器不使能。

配置 AO_TIMER_CTL.AO_TIMER_CLR=1 可清除中断，不改变计数器使能状态。

12.1.3.3. GPIO 检测

RTC 内部可以针对 5 个 GPIO（GPIO0~GPIO4）的输入信号进行检测，并产生中断。可以配置中断触发的模式。

触发模式有如下几种，配置 AOGPIO_INTR_MODEx（x=0~4）寄存器：

- 高电平中断
- 低电平中断
- rtc_clk 上升沿中断
- rtc_clk 下降沿中断
- rtc_clk 沿中断

GPIO 检测可以配置使能去毛刺(Debounce)操作，通过配置寄存器 AOGPIO_DEBOUNCE_Enx（x=0~4）配置。

使能去毛刺功能可以过滤掉 GPIOx(x=0~4)上小于 1 个 rtc_clk 时钟周期的输入脉冲。使能去毛刺功能会增加三个 rtc_clk 时钟周期的 GPIO 检测时间。

12.1.3.4. 32K 时钟校准计数

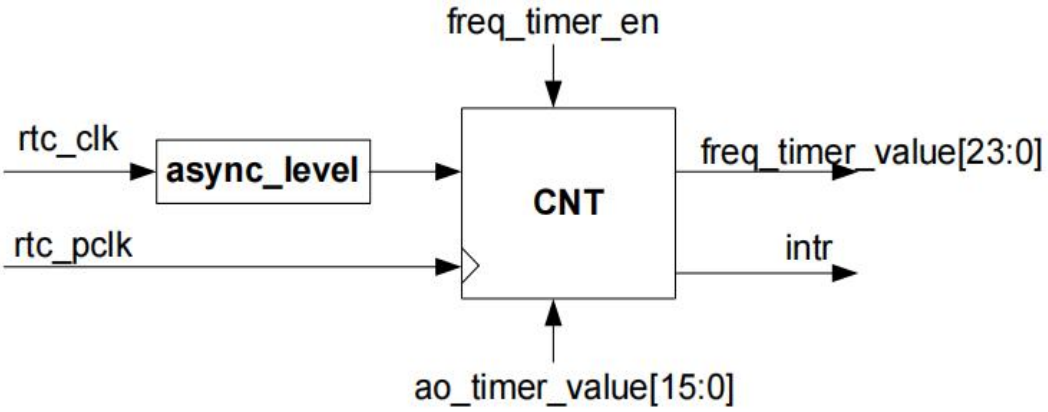

图 12-2 32K 时钟校准结构

校准计数器工作在 rtc_pclk 域，rtc_pclk 频率为 8/16/32MHz 可配。

校准计数器计数 ao_timer_value 个 32k 时钟周期，得到对应的 rtc_pclk 周期的计数值（freq_timer_value）。进而可以得到 32K 时钟的频率偏移。

举例来说，若 rtc_pclk 时钟频率为 16MHz，理想的 32K 时钟周期为 1/32768 s，理想的 32 个 32K 周期对应 15625 个 16M 时钟周期。配置 ao_timer_value = 32。则可以根据最终 freq_timer_value 得到平均每个 32K 时钟偏差：

clk_skew = (15625 - freq_timer_value)* T16M /32

12.1.4. 信号及接口的描述

RTC 模块的信号及接口描述图如下图所示。

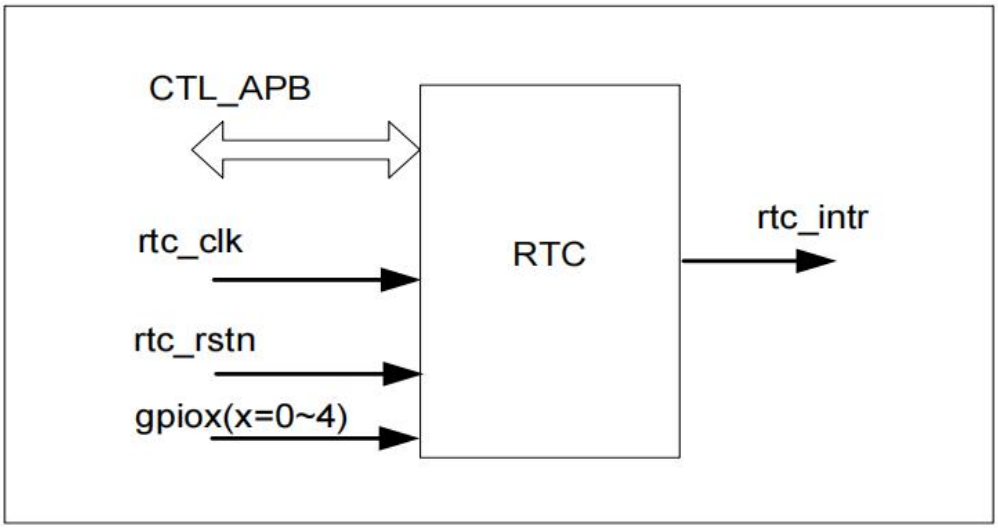

图 12-3 RTC 信号及接口描述图

RTC 模块的信号与接口表如下表所示

表 12-1 RTC 信号及接口描述表

名称	宽度	方向	描述	备注
CTL_APB 接口				内部信号
RTC 接口				
rtc_clk	1	I	RTC 工作时钟 32K	内部信号
中断				
rtc_intr	1	O	RTC 所有中断的输出	内部信号
复位信号				
rtc_rstn	1	I	控制 RTC 复位的信号	内部信号
GPIO 接口				
gpio_d[0]	1	I	外部 GPIO0	管脚信号
gpio_d[1]	1	I	外部 GPIO1	管脚信号
gpio_d[2]	1	I	外部 GPIO2	管脚信号
gpio_d[3]	1	I	外部 GPIO3	管脚信号
gpio_d[4]	1	I	外部 GPIO4	管脚信号

12.2. RTC 寄存器地址映射

- RTC 的基地址为：0x40002000

表 11-2 为 RTC 的寄存器描述和地址映射表。

表 12-2 RTC 寄存器描述和地址映射表

寄存器名	描述	方向&宽度	地址偏移	复位值
RTC_CCVR	当前计数值寄存器	R 32bits	0x00	0x0
RTC_CLR	计数器初始值设置寄存器	RW 32bits	0x04	0x0
RTC_CMR_ONE	定时匹配寄存器 1	RW 24bits	0x08	0x0
RTC_CMR_TWO	定时匹配寄存器 2	RW 24bits	0x0C	0x0
RTC_CMR_THREE	定时匹配寄存器 3	RW 24bits	0x10	0x0
RTC_ICR	中断控制寄存器	RW 9bits	0x14	0x0
RTC_ISR	中断状态寄存器	R 7bits	0x18	0x0
RTC_EOI	中断清除寄存器	W 7bits	0x18	0x0
RTC_WVR	当前周计数器值寄存器	R 3bits	0x1C	0x0
RTC_WLR	周计数器初值设置寄存器	RW 3bits	0x20	0x0

RTC_RAW_LIMIT	计数频率设置寄存器	RW 16bits	0x24	0x8000
RTC_SECOND_LIMIT	秒计数上限控制寄存器	RW 6bits	0x28	0x3c
RTC_MINUTE_LIMIT	分计数上限控制寄存器	RW 6bits	0x2c	0x3c
RTC_HOUR_LIMIT	时计数上限控制寄存器	RW 5bits	0x30	0x18
RTC_ISR_RAW	中断原始状态寄存器	R 7bits	0x34	0x0
RTC_RVR	当前秒内计数值寄存器	R 16bits	0x38	0x0
AO_TIMER_CTL	16 位计数器控制寄存器	RW 18bits	0x40	0x0
AO_GPIO_MODE	GPIO 模式配置寄存器	RW 20bits	0x44	0x0
AO_GPIO_CTL	GPIO 控制寄存器	RW 13bits	0x48	0x0
AO_ALL_INTR	AO 中断寄存器	RW 6bits	0x4C	0x0
FREQ_TIMER_VAL	FREQ_TIMER 值寄存器	R 24bits	0x50	0x0
RTC_CALI_CONFIG	校准配置寄存器	RW 5bit	0x54	0x0
RTC_CALI_RESULT	校准结果寄存器	RW 17bit	0x58	0x0
PWRKEY_CTRL0	power key 控制寄存器 0	RW 32bit	0x5C	0x0
PWRKEY_CTRL1	power key 控制寄存器 1	RW 32bit	0x60	0x0
PWRKEY_CTRL2	power key 控制寄存器 2	RW 32bit	0x64	0x0
PWRKEY_SOFT_CTRL0	power key 输出使能 0	RW 32bit	0x68	0x0
PWRKEY_SOFT_CTRL1	power key 输出使能 1	RW 32bit	0x6C	0x0
RCCAL_EN	校准使能寄存器	RW 1bit	0x70	0x0
RCCAL_LFCFG	低频时钟分频	RW 21bit	0x74	0x0
RCCAL_HFCFG	高频时钟分频	RW 28bit	0x78	0x0
RCCAL_LIM1	低频时钟计数器限制 1	RW 15bit	0x7C	0x0
RCCAL_LIM2	低频时钟计数器限制 2	RW 18bit	0x80	0x0
RCCAL_ADJ_CFG	校准调整配置	RW 32bit	0x84	0x0
RCCAL_ADJ_STEP	校准调整 step	RW 14bit	0x88	0x0
RCCAL_HFCNT_VAL	高频计数值	RW 32bit	0x8C	0x0
RCCAL_ADJ_VAL	调整值	R 14bit	0x90	0x0
RCCAL_STATUS	校准状态	R 4bit	0x94	0x0
RCCAL_INTEN	校准中断	RW 2bit	0x98	0x0
RCCAL_INTSTA	校准中断状态	RC1 2bit	0x9C	0x0
RCCAL_INTRAW	校准中断原始	R 2bit	0xA0	0x0
PWRKEY_CTRL3	power key 控制寄存器 3	RW 32bit	0xA4	0x0

12.3. RTC 寄存器描述

- RTC 的基地址为：0x40002000

12.3.1. 当前计数值寄存器（RTC_CCVR）

- 地址偏移：0x00

位数	名称	方向	复位值	描述
31：0	CCVR	R	0x0	当前计数值： 用于记录计数器当前计数值。 Bit 31：17：天计数值（0－32767） Bit 16：12：小时计数值（0－23） Bit 11：6：分计数值（0－59） Bit 5：0：秒计数值（0－59）

12.3.2. 计数器初始值设置寄存器（RTC_CLR）

- 地址偏移：0x04

位数	名称	方向	复位值	描述
31：0	CLR	RW	0x0	初次载入的计数值：向该寄存器写入新值以后，在 32K 计数器时钟的上升沿到来之后被载入计数器，计数器以载入值作为计数初值开始向上计数。 Bit 31：17：天计数值（0－32767） Bit 16：12：小时计数值（0－23） Bit 11：6：分计数值（0－59） Bit 5：0：秒计数值（0－59）

12.3.3. 定时匹配寄存器 1（RTC_CMR_ONE）

- 地址偏移：0x08

位数	名称	方向	复位值	描述
23：17	Week Match	RW	0x0	用于设置在周几产生中断： 1：允许产生中断 0：不允许产生中断 Bit 17 对应于周日 Bit 18 对应于周一 Bit 19 对应于周二 Bit 20 对应于周三 Bit 21 对应于周四 Bit 22 对应于周五 Bit 23 对应于周六
16：12	Hour Match	RW	0x0	用于设置中断产生时的小时计数
11：6	Minute Match	RW	0x0	用于设置中断产生时的分钟计数
5：0	Second Match	RW	0x0	用于设置中断产生时的秒计数

12.3.4. 定时匹配寄存器 2（RTC_CMR_TWO）

- 地址偏移：0x0C

位数	名称	方向	复位值	描述
23：17	Week Match	RW	0x0	用于设置在周几产生中断： 1：允许产生中断 0：不允许产生中断 Bit 17 对应于周日 Bit 18 对应于周一 Bit 19 对应于周二 Bit 20 对应于周三 Bit 21 对应于周四 Bit 22 对应于周五 Bit 23 对应于周六
16：12	Hour Match	RW	0x0	用于设置中断产生时的小时计数
11：6	Minute Match	RW	0x0	用于设置中断产生时的分钟计数
5：0	Second Match	RW	0x0	用于设置中断产生时的秒计数

12.3.5. 定时匹配寄存器 3（RTC_CMR_THREE）

- 地址偏移：0x10

位数	名称	方向	复位值	描述
23：17	Week Match	RW	0x0	用于设置在周几产生中断： 1：允许产生中断 0：不允许产生中断 Bit 17 对应于周日 Bit 18 对应于周一 Bit 19 对应于周二 Bit 20 对应于周三 Bit 21 对应于周四 Bit 22 对应于周五 Bit 23 对应于周六
16：12	Hour Match	RW	0x0	用于设置中断产生时的小时计数
11：6	Minute Match	RW	0x0	用于设置中断产生时的分钟计数
5：0	Second Match	RW	0x0	用于设置中断产生时的秒计数

12.3.6. 中断控制寄存器（RTC_ICR）

- 地址偏移：0x14

位数	名称	方向	复位值	描述
8	CntE	RW	0x0	计数器使能： 控制计数器是否使能。 0：计数器不使能 1：计数器使能
7	MASK	RW	0x0	中断屏蔽： 设置是否屏蔽中断输出 0：不屏蔽中断输出 1：屏蔽所有中断输出
6	T3E	RW	0x0	定时中断 3 使能： 设置是否使能定时中断 3 0：不使能定时中断 3 1：使能定时中断 3
5	T1E	RW	0x0	定时中断 1 使能：

				设置是否使能定时中断 1 0：不使能定时中断 1 1：使能定时中断 1
4	T2E	RW	0x0	定时中断 2 使能： 设置是否使能定时中断 2 0：不使能定时中断 2 1：使能定时中断 2
3	SeE	RW	0x0	秒中断使能： 设置是否使能秒中断 0：不使能秒中断 1：使能秒中断
2	MiE	RW	0x0	分中断使能： 设置是否使能分中断 0：不使能分中断 1：使能分中断
1	HoE	RW	0x0	小时中断使能： 设置是否使能小时中断 0：不使能小时中断 1：使能小时中断
0	DaE	RW	0x0	天中断使能： 设置是否使能天中断 0：不使能天中断 1：使能天中断

12.3.7. 中断状态寄存器（RTC_ISR）

- 地址偏移：0x18

位数	名称	方向	复位值	描述
6	T3F	R	0x0	中断状态： 该位是定时中断 3 的中断状态 0：没有发生中断 1：发生中断
5	T1F	R	0x0	中断状态： 该位是定时中断 1 的中断状态 0：没有发生中断 1：发生中断
4	T2F	R	0x0	中断状态： 该位是定时中断 2 的中断状态 0：没有发生中断 1：发生中断
3	SeF	R	0x0	中断状态：该位是秒中断的中断状态 0：没有发生中断 1：发生中断
2	MiF	R	0x0	中断状态：该位是分中断的中断状态 0：没有发生中断 1：发生中断
1	HoF	R	0x0	中断状态：该位是小时中断的中断状态 0：没有发生中断 1：发生中断
0	DaF	R	0x0	中断状态：该位是天中断的中断状态 0：没有发生中断 1：发生中断

12.3.8. 中断清除寄存器（RTC_EOI）

- 地址偏移：0x18（该寄存器和 RTC_ISR 共用同一个地址）

位数	名称	方向	复位值	描述
6	EOI_T3	W	0x0	清定时中断 3： 向寄存器该位写 1 可以清除定时中断 3
5	EOI_T1	W	0x0	清定时中断 1： 向寄存器该位写 1 可以清除定时中断 1
4	EOI_T2	W	0x0	清定时中断 2： 向寄存器该位写 1 可以清除定时中断 2
3	EOI_se	W	0x0	清秒中断： 向寄存器该位写 1 可以清除秒中断
2	EOI_mi	W	0x0	清分中断： 向寄存器该位写 1 可以清除分中断
1	EOI_ho	W	0x0	清小时中断： 向寄存器该位写 1 可以清除小时中断
0	EOI_da	W	0x0	清天中断： 向寄存器该位写 1 可以清除天中断

12.3.9. 当前周计数器值寄存器（RTC_WVR）

- 地址偏移：0x1C

位数	名称	方向	复位值	描述
2：0	WVR	R	0x0	记录当前日期为星期几： 3’b000：周日 3’b001：周一 3’b010：周二 3’b011：周三 3’b100：周四 3’b101：周五 3’b110：周六

12.3.10. 周计数器初值设置寄存器（RTC_WLR）

- 地址偏移：0x20

位数	名称	方向	复位值	描述
2：0	WLR	RW	0x0	设置写入 CLR 寄存器的天计数值对应星期几： 3’b000：周日 3’b001：周一 3’b010：周二 3’b011：周三 3’b100：周四 3’b101：周五 3’b110：周六 （该寄存器和 CLR 有相关性，写 CLR 时注意 WLR 寄存器的值是否匹配）

12.3.11. 计数频率设置寄存器（RTC_ RAW_LIMIT）

- 地址偏移：0x24

位数	名称	方向	复位值	描述
15：0	RAW_LIMIT	RW	0x8000	RAW_LIMIT 的值为计数 1 秒所需要的 rtc_mclk 时钟周期数。例如，当 rtc_mclk 时钟频率为 32768Hz 时，计数 1 秒需要 32768 个 rtc_mclk 时钟周期。当计数时钟不等于 32768Hz 时，可以设置 RAW_LIMIT 的值为实际采用的计数时钟的频率数。此外在调试时，通过将 RAW_LIMIT 设置为较小的值，可以减少等待时间。

12.3.12. 秒计数上限控制寄存器（RTC_SECOND_LIMIT）

地址偏移：0x28

位数	名称	方向	复位值	描述
5：0	SECOND_LIMIT	RW	0x3C	设置秒计数的上限，默认值为 60。主要用于调试，设置为较小的值，可以减少等待时间。通常情况下采用默认值。

12.3.13. 分计数上限控制寄存器（RTC_MINUTE_LIMIT）

地址偏移：0x2C

位数	名称	方向	复位值	描述
5：0	MINUTE_LIMIT	RW	0x3C	设置分计数的上限，默认值为 60。主要 用于调试，设置为较小的值，可以减少等待时间。通常情况下采用默认值。

12.3.14. 时计数上限控制寄存器（RTC_HOUR_LIMIT）

地址偏移：0x30

位数	名称	方向	复位值	描述
4：0	HOUR_LIMIT	RW	0x18	设置时计数的上限，默认值为 24。主要用于调试，设置为较小的值，可以减少等待时间。通常情况下采用默认值。

12.3.15. 中断原始状态寄存器（RTC_ISR_RAW）

- 地址偏移：0x34

位数	名称	方向	复位值	描述
6	T3F_raw	R	0x0	中断原始状态： 该位是定时中断 3 的原始中断状态 0：没有发生中断 1：发生中断
5	T1F_raw	R	0x0	中断原始状态： 该位是定时中断 1 的原始中断状态 0：没有发生中断 1：发生中断
4	T2F_raw	R	0x0	中断原始状态： 该位是定时中断 2 的原始中断状态 0：没有发生中断 1：发生中断
3	SeF_raw	R	0x0	中断原始状态： 该位是秒中断的原始中断状态 0：没有发生中断 1：发生中断
2	MiF_raw	R	0x0	中断原始状态： 该位是分中断的原始中断状态 0：没有发生中断 1：发生中断
1	HoF_raw	R	0x0	中断原始状态： 该位是小时的中断状态 0：没有发生中断 1：发生中断
0	DaF_raw	R	0x0	中断原始状态： 该位是天中断的原始中断状态 0：没有发生中断 1：发生中断

12.3.16. 当前秒内计数值寄存器（RTC_RVR）

- 地址偏移：0x38

位数	名称	方向	复位值	描述
15：0	RVR	R	0x0	当前秒内计数值： 记录 1 秒计数器的计数值。

12.3.17. 16 位计数器控制寄存器（AO_TIMER_CTL）

- 地址偏移：0x40

位数	名称	方向	复位值	描述
18	FREQ_TIMER_EN	RW	0x0	32K 校准计数器使能
17	AO_TIMER_EN	RW	0x0	计数器使能
16	AO_TIMER_CLR	RW	0x0	中断清除
15：0	AO_TIMER_VALUE	RW	0x0	计数器目标值，从 0 计数到该值时产生中断

12.3.18. GPIO 模式配置寄存器（AO_GPIO_MODE）

- 地址偏移：0x44

位数	名称	方向	复位值	描述
19	AOGPIO_DEBOUNCE_EN4	RW	0x0	GPIO4 debounce 功能使能。 1：有效
18:16	AOGPIO_INTR_MODE4	RW	0x0	GPIO4 触发模式 0：高电平 1：低电平 2：上升沿 3：下降沿 4：上升下降沿 其他：无效
15	AOGPIO_DEBOUNCE_EN3	RW	0x0	GPIO3 debounce 功能使能。 1：有效
14:12	AOGPIO_INTR_MODE3	RW	0x0	GPIO3 触发模式。 0：高电平 1：低电平 2：上升沿 3：下降沿 4：上升下降沿 其他：无效
11	AOGPIO_DEBOUNCE_EN2	RW	0x0	GPIO2 debounce 功能使能。

				1：有效
10：8	AOGPIO_INTR_MODE2	RW	0x0	GPIO2 触发模式。 0：高电平 1：低电平 2：上升沿 3：下降沿 4：上升下降沿 其他：无效
7	AOGPIO_DEBOUNCE_EN1	RW	0x0	GPIO1 debounce 功能使能。 1：有效
6：4	AOGPIO_INTR_MODE1	RW	0x0	GPIO1 触发模式。 0：高电平 1：低电平 2：上升沿 3：下降沿 4：上升下降沿 其他：无效
3	AOGPIO_DEBOUNCE_EN0	RW	0x0	GPIO0 debounce 功能使能。 1：有效
2：0	AOGPIO_INTR_MODE0	RW	0x0	GPIO0 触发模式。 0：高电平 1：低电平 2：上升沿 3：下降沿 4：上升下降沿 其他：无效

12.3.19. GPIO 控制寄存器（AO_GPIO_CTL）

- 地址偏移：0x48

位数	名称	方向	复位值	描述
12：8	AOGPIO_INTR_EN	RW	0x0	GPIO 中断使能
4：0	AOGPIO_INTR_CLR	RW	0x0	GPIO 中断清除

12.3.20. AO 中断寄存器（AO_ALL_INTR）

- 地址偏移：0x4C

位数	名称	方向	复位值	描述
5	AO_TIMER_INTR	R	0x0	ao_timer 中断标志
4：0	AOGPIO_INTR	R	0x0	gpio0~4 中断标志

12.3.21. FREQ_TIMER 值寄存器（FREQ_TIMER_VAL）

- 地址偏移：0x50

位数	名称	方向	复位值	描述
23：0	FREQ_TIMER_VALUE	R	0x0	32K 校准计数周期

12.3.22. （RTC_CALI_CONFIG）

- 地址偏移：0x54

位数	名称	方向	复位值	描述
4	cali_rc32k_enable	RW	0x0	Cali 32k 使能， 1：使能
3:0	cali_rc32k_alpha	RW	0x0	Cali 滤波因子，内部 IIR 滤波因子，配置值越大，窗口越宽

12.3.23. （RTC_CALI_RESULT）

- 地址偏移：0x58

位数	名称	方向	复位值	描述
16	cali_rc32k_factor_vld	R	0x0	校准结果输出有效指示
15:0	cali_rc32k_factor	R	0x0	校准结果输出

12.3.24. （PWRKEY_CTRL0）

- 地址偏移：0x5C

位数	名称	方向	复位值	描述
31:0	powkey_edge	RW	0x0	32 个 power key 电平选择，对应 gpio0~gpio31 1：低电平有效 0：高电平有效

12.3.25. （PWRKEY_CTRL1）

- 地址偏移：0x60

位数	名称	方向	复位值	描述
31:0	powkey_en	RW	0x0	32 个 power key 使能，对应 gpio0~gpio31

12.3.26. （PWRKEY_CTRL2）

- 地址偏移：0x64

位数	名称	方向	复位值	描述
31	powkey_inten	RW	0x0	power key 中断使能，1 使能
1	debounce_out	R	0x0	power key 中断原始状态
0	powkey_intr	R	0x0	power key 中断状态

12.3.27. （PWRKEY_SOFT_CTL0）

- 地址偏移：0x68

位数	名称	方向	复位值	描述
31:0	aogpio_soft_out_en	RW	0x0	32 个 power key 输出使能，对应 gpio0~gpio31

12.3.28. （PWRKEY_SOFT_CTL1）

- 地址偏移：0x6C

位数	名称	方向	复位值	描述
31:0	aogpio_soft_out_value	RW	0x0	32 个 power key 输出的值配置，对应 gpio0~gpio31

12.3.29. （RCCAL_EN）

- 地址偏移：0x70

位数	名称	方向	复位值	描述
0	ENABLE	RW	0x0	启用 RC 校准: 1: enable 0: disable

12.3.30. （RCCAL_LFCFG）

- 地址偏移：0x74

位数	名称	方向	复位值	描述
20	LFDIV_WEN	WC1	0x0	LFDIV 写入使能，1 个周期脉冲
19:0	LFDIV	RW	0x63	LFCLK（低频时钟）预分频器，用于产生分频后的信号脉冲： lfdiv_pulse 频率为低频时钟的 1/（lfdiv+1）

12.3.31. （RCCAL_HFCFG）

- 地址偏移：0x78

位数	名称	方向	复位值	描述
27:0	HFCNT_LOAD	RW	0x1869F	内部计数器“hcnt”初始负载值

12.3.32. （RCCAL_LIM1）

- 地址偏移：0x7C

位数	名称	方向	复位值	描述
14:0	LIM1	RW	0xA	第 1 步低频时钟调整的计数器限制

12.3.33. （RCCAL_LIM2）

- 地址偏移：0x80

位数	名称	方向	复位值	描述
17:0	LIM2	RW	0x32	第 2 步低频时钟调整的计数器限制

12.3.34. （RCCAL_ADJ_CFG）

- 地址偏移：0x84

位数	名称	方向	复位值	描述
31	FREQ_ADJ_INV	RW	0x0	0：频率低时增加步长 1：频率低时减少步长

29:16	FREQ_ADJ_OFFSET	RW	0x1FFF	LFCLK 调整初始偏移
15	FREQ_ADJ_HW_CLR	WC1	0x0	清除内部 freq_adj_hw，然后 rc_adj=freq_aj_offset，1 个周期脉冲
14	FREQ_ADJ_SOFTEN	RW	0x0	LFCLK 调整软件控制启用 1:LFCLK_ADJ_INTERNAL 使用此 reg 值 0:LFCLK_ADJ_INTERNAL 来自硬件
13:0	FREQ_ADJ_SOFT	RW	0x1FFF	LFCLK 调整软件控制值

12.3.35. （RCCAL_ADJ_STEP）

- 地址偏移：0x88

位数	名称	方向	复位值	描述
13:8	ADJ_STEP_LIM2	RW	0x4	LFCLK 调整极限 2 步 如果 lfdiv_ pulse 信号出现在 LIM2 和 128*LIM2 之间
5:0	ADJ_STEP_LIM1	RW	0x14	LFCLK 调整极限 1 步 如果 lfdiv_ pulse 信号出现在 LIM1 和 LIM2 之间

12.3.36. （RCCAL_HFCNT_VAL）

- 地址偏移：0x8C

位数	名称	方向	复位值	描述
31	HFCNT_DIR_VAL	R	0x0	
27:0	HFCNT_VAL	R	0x0	HFCNT conter 值，用于计算 LFCLK 频率

12.3.37. （RCCAL_ADJ_VAL）

- 地址偏移：0x90

位数	名称	方向	复位值	描述
13:0	RC_ADJ	R	0x1FFF	LFCLK 调整值，输出至 RC oscilor 输入 LFCLK_ADJ=LFCLK_ADJ_INTERNAL+LFCLK_OFFSET LFCLK_ADJ_INTERNAL 将在每次校准事件中进行调整

12.3.38. （RCCAL_STATUS）

- 地址偏移：0x94

位数	名称	方向	复位值	描述
3	LFCLK_MISS	R	0x0	LFCLK 频率未命中
2	LFCLK_ERROR	R	0x0	LFCLK 频率误差
1	LFCLK_WARN	R	0x0	LFCLK 频率警告
0	LFCLK_GOOD	R	0x0	LFCLK 频率良好

12.3.39. （RCCAL_INTEN）

- 地址偏移：0x98

位数	名称	方向	复位值	描述
1	RC_ERROR_EN	RW	0x0	校准错误中断启用
0	RC_EVENT_EN	RW	0x0	校准事件结束中断启用

12.3.40. （RCCAL_INTSTA）

- 地址偏移：0x9C

位数	名称	方向	复位值	描述
1	RC_ERROR	RC1	0x0	校准错误中断
0	RC_EVENT	RC1	0x0	校准事件结束中断状态，写入 1 以清除状态和原始状态

12.3.41. （RCCAL_INTRAW）

- 地址偏移：0xA0

位数	名称	方向	复位值	描述
1	RC_ERROR_RAW	R	0x0	校准错误中断原始状态
0	RC_EVENT_RAW	R	0x0	校准事件结束中断原始状态

12.3.42. （PWRKEY_CTRL3）

- 地址偏移：0xA4

位数	名称	方向	复位值	描述
31:17	powkey_edge_h	RW	0x0	8 个 power key 电平选择，对应 gpio32~gpio39 1：低电平有效 0：高电平有效
23:16	powkey_en_h	Rw	0x0	8 个 power key 使能，对应 qpio32~gpio39
15:8	aogpio_soft_out_en_h	Rw	0x0	8 个 power key 输出使能，对应 gpio32~gpio39
7:0	aogpio_soft_out_value_h	Rw	0x0	8 个 power key 输出的值配置，对应 gpio32~gpio39

12.4. 工作流程

本节给出如下编程参考流程：

12.4.1. 实时时钟配置

(1) 芯片上电后，设置周计数器初值设置寄存器（RTC_WLR）；
(2) 设置计数器初始值设置寄存器（RTC_CLR）；
(3) 配置相应的中断产生条件；
(4) 配置中断控制寄存器（RTC_ICR）的 CntE 位为 1，计数器开始计数；
(5) 当中断条件满足时，中断产生；
(6) 进入中断服务子程序，写中断清除寄存器中断清除寄存器（RTC_EOI），清除相应的中断。

12.4.2. AO_TIMER 配置

(1) 配置计数目标值（AO_TIMER_CTL.AO_TIMER_VALUE）；
(2) 使能计数器（AO_TIMER_CTL.AO_TIMER_EN=1）；
(3) 等待计数器计到目标值，产生中断；
(4) 进入中断服务程序，读取 AO_ALL_INTR.AO_TIMER_INTR 中断状态；
(5) 不使能计数器（AO_TIMER_CTL.AO_TIMER_EN=0），此时中断清除，计数器清 0 并停止计数。

12.4.3. GPIO 检测配置

(1) 配置 GPIO 触发中断模式、去毛刺功能（AO_GPIO_MODE）；
(2) 使能相应的 GPIO 检测（AO_GPIO_CTL.AOGPIO_INTR_Enx）；
(3) GPIO 信号触发中断；
(4) 进入中断服务程序，读取 AO_ALL_INTR.AOGPIO_INTR 中断状态；
(5) 不使能对应的 GPIO 检测，AO_GPIO_CTL.AO_GPIO_INTR_En[x]=0，x=0~4；
(6) 先配置 AO_GPIO_CTL.AOGPIO_INTR_CLR[x]=1，x=0~4，之后再配置；
AO_GPIO_CTL.AOGPIO_INTR_CLR[x]=0，x=0~4，清除对应的 GPIO 检测中断。

12.4.4. 32K 时钟校准计数配置

(1) 配置计数目标值（AO_TIMER_CTL.AO_TIMER_VALUE）；
(2) 使能计数器（AO_TIMER_CTL.AO_TIMER_EN=0）；
(3) 使能 32K 校准计数器（AO_TIMER_CTL.FREQ_TIMER_EN=1）；
(4) 等待计数器计到目标产生中断，此时校准计数器停止计数，并保持计数值；
(5) 进入中断服务程序，读取 AO_ALL_INTR 中断状态；
(6) 读取 FREQ_TIMER_VAL.FREQ_TIMER_VALUE 计数器的值；
(7) 不使能 32K 校准计数器（AO_TIMER_CTL.FREQ_TIMER_EN=0），此时中断清除，计数器清 0；
(8) AO_TIMER_VALUE 为 32K 时钟的整数倍周期值；
(9) FREQ_TIMER_VALUE 为对应 AO_TIMER_VALUE 个 32K 周期下的 pclk（默认为 16MHz）周期值；通过 FREQ_TIMER_VALUE 可以计算出 32K 时钟的偏差，进而用来校准 32K 时钟。

注：

AO_TIMER 计数器和 32K 校准计数器在同一时刻只能有一个工作，即 AO_TIMER_EN 与 FREQ_TIMER_EN 不能同时为 1。

第 13 章 脉冲宽度调制模块（PWM）

13.1. 概述

PWM 模块是芯片中的一个用于产生频率可调，占空比可调信号的模块，一共 12 路 pwm；4 路带互补（最高 16 位，无级可调，可以支持电机），2 路不带互补；pwm2,3 与 4,5 管脚位置互换一下（因为电机应用 pwm 可能会和 adc 冲突）；带互补功能的 PWM，死区时间可调范围增大 8 倍。高 3bit 控制倍数。PWM_TIMER 模块提供 6 路不带互补的 PWM，最高 12 位，基地址 0x40017D00。PWM_TIMER 的 6 路 pwmt[5:0]放在 gpio29-30,35-38 上。

- PWM0 的基地址为：0x40017000
- PWM1 的基地址为：0x40017040
- PWM2 的基地址为：0x40017400
- PWM3 的基地址为：0x40017440
- PWM4 的基地址为：0x40017800
- PWM5 的基地址为：0x40017840

13.1.1. 主要特性

- 8 位可调周期计数器。
- 占空比计数器计数周期固定为 100。
- 具有软件复位功能。
- 三种工作模式：只调占空比、只调频率、两者同时调节。
- PWM 刹车、捕获功能，低功耗可运行功能。

13.1.2. 功能描述

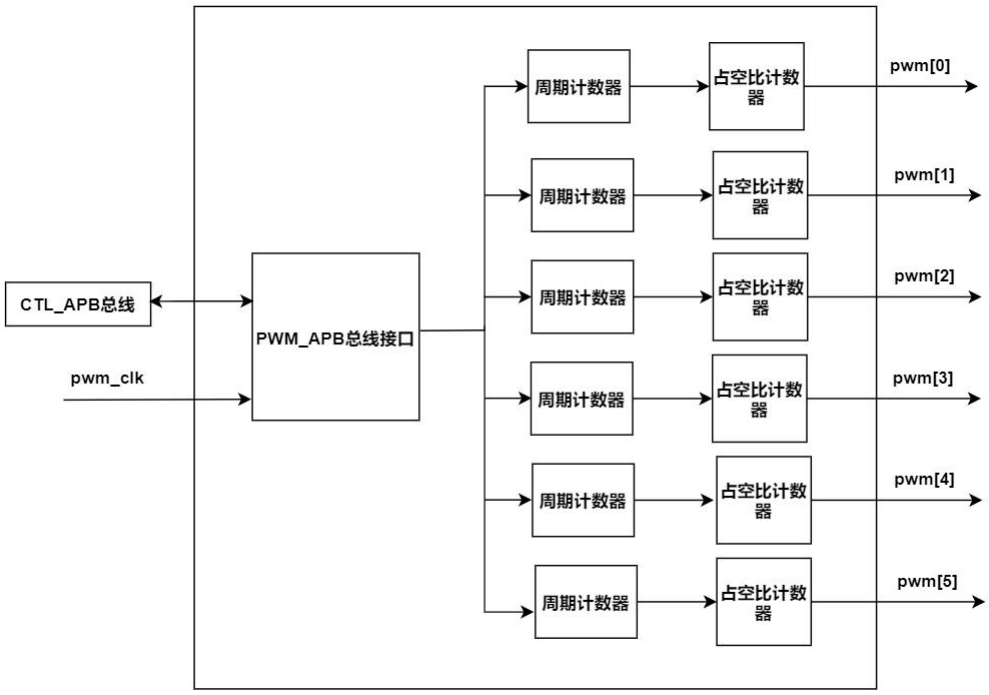

图 13-1 PWM 结构框图

脉冲宽度调制模块（PWM）能够产生 6 路完全独立的 PWM 信号。通过设置周期设置寄存器可以调整 PWM 脉冲信号的频率；设置占空比设置寄存器可以调 PWM 脉冲信号的占空比。

周期计数器从 0 开始计数，当计数值增加到周期设置寄存器的值时，占空比寄存器加 1，同时周期计数器回 0 重新开始计数。占空比计数器从 0 开始增加，当增加到占空比设置寄存器的值时，输出 pwm[x]（x=[0-5]）变为低电平且占空比计数器继续增加，当占空比计数器增加到 99 时回 0，同时将输出 pwm[x]（x=[0-5]）置为高电平，这样就输出了频率和占空比可调的脉冲信号，可用于控制 LCD 背景光亮度和产生各种需要的信号音等。

13.1.3. 工作模式

通过调整周期设置寄存器和占空比设置寄存器，PWMx（x=[0-5]）可以工作在三种模式下：

第一：可调频率，占空比不变；

第二：可调占空比，频率不变；

第三：同时调整占空比和频率。

- 调频模式

PWMx（x=[0-5]）工作在调频模式下，只通过调整周期设置寄存器来产生一个连续音调，即通过改变频率可以产生各种需要的信号音。由于占空比不变，占空比设置寄存器的值保持为上一次写入的值，输出音量保持不变。

- 调幅模式

PWMx（x=[0-5]）在此模式下，输出周期相同，但占空比不同的 PWM 信号，这时只需调节占空比设置寄存器。如果在 PWMx（x=[0-5]）信号输出后加一个低通滤波器，就可得到不同的电压，从而达到改变 LCD 背景光亮度的目的。

- 调频调幅模式

PWMx（x=[0-5]）可以同时改变频率和占空比，即同时改变周期设置寄存器和占空比设置寄存器的值。这时，就可同时改变输出声音的音量和音调。

13.1.4. 信号及接口描述

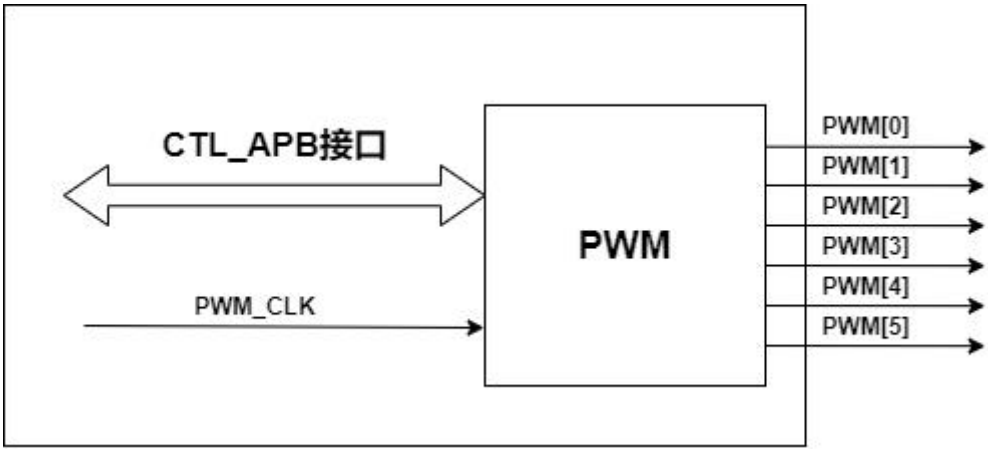

图 13-2 PWM 信号及接口框图

如图所示，PWM 接口信号由 PWM 时钟信号和 CTL_APB 信号以及 PWM 输出信号组成。下表是信号及接口描述表。

表 13-1 PWM 信号及接口描述表

名称	宽度	方向	描述	备注
CTL_APB 接口				内部信号
PWM 时钟信号				
pwm_clk	1	I	PWMx（x=[0-5]）工作时钟，默认频率 1.6MHz	内部信号
PWM 输出				
pwm[0]	1	O	PWM0 脉宽调制输出信号	管脚信号
pwm[1]	1	O	PWM1 脉宽调制输出信号	管脚信号
pwm[2]	1	O	PWM2 脉宽调制输出信号	管脚信号
pwm[3]	1	O	PWM3 脉宽调制输出信号	管脚信号
pwm[4]	1	O	PWM4 脉宽调制输出信号	管脚信号
pwm[5]	1	O	PWM5 脉宽调制输出信号	管脚信号

13.2. PWM 寄存器地址映射

PWM 寄存器地址映射

PWM 的基地址为：0x40017000

表 13-2 PWM 寄存器概述和地址映射表

寄存器	描述	方向和宽度	地址偏移	复位值
PWM_EN	PWM 使能寄存器	RW 7bit	0x00	0x0
PWM_UP	PWM 参数更新寄存器	RW 1bit	0x04	0x0
PWM_RST	PWM 复位寄存器	RW 1bit	0x08	0x0
PWM_PERIOD	PWM 周期设置寄存器	RW 8bit	0x0C	0xff
PWM_OCPY	PWM 占空比设置寄存器	RW 32bit	0x10	0x64
PWMCOMPEN	互补 PWM 输出寄存器	RW 1bit	0x14	0x0
PWMCOMPTIME	互补 PWM 死区控制寄存器	RW 3bit	0x18	0x0
PWM_BREAK_CTL	PWM 刹车寄存器	RW 20bit	0x20	0x0

PWM COMMON 寄存器地址映射

PWM 的基地址为：0x40017C00

表 13-3 PWM COMMON 寄存器概述和地址映射表

寄存器	描述	方向和宽度	地址偏移	复位值
CAP_TIM_VAL[0]	PWM 通道 0 捕获值寄存器	RW 16bit	0x00	0x0
CAP_TIM_VAL[1]	PWM 通道 1 捕获值寄存器	RW 16bit	0x04	0x0
CAP_TIM_VAL[2]	PWM 通道 2 捕获值寄存器	RW 16bit	0x08	0x0
PWM_CAP_TIM_INT	PWM 通道捕获值中断寄存器	RW 3bit	0x0C	0x0
PWM_CAP_TIM_INT_EN	PWM 通道捕获值中断使能寄存器	RW 3bit	0x10	0x0
PWM_CAP_TIM_INT_RAW	PWM 通道捕获值中断原始状态寄存器	RW 3bit	0x14	0x0

CAP_TIM_CTL[0]	PWM 通道 0 捕获控制寄存器	RW 24bit	0x18	0x0
CAP_TIM_CTL[1]	PWM 通道 1 捕获控制寄存器	RW 24bit	0x1C	0x0
CAP_TIM_CTL[2]	PWM 通道 2 捕获控制寄存器	RW 24bit	0x20	0x0

13.3. PWM 寄存器描述

注：本节寄存器名称中 PWMx 表示为 PWM0，PWM1，PWM2，PWM3，PWM4，PWM5。

13.3.1. PWM 使能寄存器（PWM_EN）

- 地址偏移：
  - PWM0_EN：0x00
  - PWM1_EN：0x40
  - PWM2_EN：0x400
  - PWM3_EN：0x440
  - PWM4_EN：0x800
  - PWM4_EN：0x840

注：bit6 仅在 pwm0 有，其他 pwm 该 bit 无效。

位数	名称	方向	复位值	描述
6	en_all	RW	0x0	同步使能，同时使能多组 PWM
5	sleep_en	RW	0x0	使能低功耗状态
4	sleep_incr	RW	0x0	PWMx 占空比自增或自减
3	EN_SEL	RW	0x0	PWMx 使能源选择
2：1	PWM_MODE	RW	0x0	PWM 占空比档位精度。 0：占空比精度为 1/100 1：占空比精度为 1/1000 2：占空比精度为 1/4000
0	PWM_EN	RW	0x0	PWM 使能控制位。此寄存器控制 PWM 内部计数器及输出的工作状态。 0：PWM 不使能 1：PWM 使能

13.3.2. PWM 参数更新寄存器（PWM_UP）

- 地址偏移：
  - PWM0_UP：0x04
  - PWM1_UP：0x44

- PWM2_UP：0x404
- PWM3_UP：0x444
- PWM4_UP：0x804
- PWM5_UP：0x844

位数	名称	方向	复位值	描述
31：1	NA	—	—	保留
0	UPDATE	RW	0x0	PWM 参数更新位。 向此位写 1 时，周期设置寄存器和占空比设置寄存器中的新值，只有在 PWM 不使能或者一个完整的 PWM 波形输出后才真正生效，然后此位自动清 0。 向此位写 0 无效。

注：对寄存器中的保留位写入，不起作用；读出始终为 0。

13.3.3. PWM 复位寄存器（PWM_RST）

- 地址偏移：
  - PWM0_RST：0x08
  - PWM1_RST：0x48
  - PWM2_RST：0x408
  - PWM3_RST：0x448
  - PWM4_RST：0x808
  - PWM5_RST：0x848

位数	名称	方向	复位值	描述
31：1	NA	—	—	保留
0	RESET	RW	0x0	PWM 模块复位。 此寄存器控制 PWM 复位操作 0：PWM 模块正常工作 1：PWM 模块进行复位

13.3.4. PWM 周期设置寄存器（PWM_PERIOD）

- 地址偏移：
  - PWM0_PERIOD：0x0C

- PWM1_PERIOD：0x4C
- PWM2_PERIOD：0x40C
- PWM3_PERIOD：0x44C
- PWM4_PERIOD：0x80C
- PWM5_PERIOD：0x84C

当计数器的值和 PERIOD 相匹配，计数器重新开始另一个周期。用下面的等式：

PWMO(Hz)=PWMCLK(Hz)/((PERIOD+1)*占空比计数器计数周期(100))

可以算出 PWM 信号的输出频率。

PWMO：PWM 输出的频率。

100：数字 100 代表占空比计数器计数周期，有 100、1000、4000 三个选项。

PERIOD：PWM 周期设置寄存器中的 7~0 位。

PWMCLK：PWM 的主时钟 pwm_clk，默认频率是 1.6MHZ，由于 pwm_clk 时钟频率，可以通过 CPR 模块中 PWM 分频控制寄存器调整，因此 PWM 输出频率范围是随着 pwm_clk 的变化而变化。

位数	名称	方向	复位值	描述
31：8	NA	—	—	保留
7：0	PERIOD	RW	0xFF	PERIOD 用于控制周期计数器的周期，在周期计数器增加到 PERIOD 时，周期计数器归 0，重新开始计数。

13.3.5. PWM 占空比设置寄存器（PWM_OCPY）

- 地址偏移：
  - PWM0_OCPY：0x10
  - PWM1_OCPY：0x50
  - PWM2_OCPY：0x410
  - PWM3_OCPY：0x450
  - PWM4_OCPY：0x810
  - PWM5_OCPY：0x850

PWM 占空比设置寄存器用来调节 PWM 的占空比，由于 PWM 占空比计数器的计数周期是 100，故与之比较用来产生占空比的 PWM 的占空比设置寄存器的值不能超过 99，否则会输出占空比为 1 的 PWM 信号。

位数	名称	方向	复位值	描述
31：12	OCPY_RATIO_CONFIG	RW	0x0	pwm_mode = 2 和 3 时候的占空比计数器的周期值

11：0	OCPY_RATIO	RW	0x64	OCPY_RATIO 用于控制占空比计数器的占空比，占空比设置寄存器用于和占空比计数器进行比较，相等时，PWM 输出为低电平；计数器由 99 归 0 时， PWM 输出为高电平。调节 OCPY_RATIO 的值便可调节占空比。 PWMx_EN 寄存器的 PWM_MODE 位决定占空比精度。 寄存器和占空比计数器相等时，PWM 输出为低电平 PWM_MODE=0:计数器由 99 归 0 时，PWM 输出为高电平 PWM_MODE=1:计数器由 999 归 0，PWM 输出为高电平 PWM_MODE=2:计数器由 3999 归 0 时，PWM 输出为高电平

13.3.6. 互补 PWM 输出寄存器（PWM_COMPEN）

- 地址偏移：
  - PWMCOMPEN0：0x14
  - PWMCOMPEN1：0x54
  - PWMCOMPEN2：0x414
  - PWMCOMPEN3：0x454
  - PWMCOMPEN4：0x814
  - PWMCOMPEN5：0x854

位数	名称	方向	复位值	描述
0	PWMCOMPEN	RW	0x0	使能 PWMx 的互补信号输出。 1：使能

13.3.7. 互补 PWM 死区控制寄存器（PWM_COMPTIME）

- 地址偏移：
  - PWMCOMPTIME0：0x18
  - PWMCOMPTIME1：0x58
  - PWMCOMPTIME2：0x418
  - PWMCOMPTIME3：0x458
  - PWMCOMPTIME4：0x818
  - PWMCOMPTIME5：0x858

位数	名称	方向	复位值	描述
5：0	PWMCOMPTIME	RW	0x0	控制反向互补信号的 pwm_inv 死区延迟 0：死区为 1 个 pwm_clk 始终周期 ... 7：死区为 8 个 pwm_clk 始终周期 ... 63:死区为 64 个 pwm_clk 始终周期

13.3.8. PWM 寄存器（PWM_BREAK_CTL）

- 地址偏移：
  - PWM_BREAK_CTL0：0x20
  - PWM_BREAK_CTL1：0x60
  - PWM_BREAK_CTL2：0x420
  - PWM_BREAK_CTL3：0x460
  - PWM_BREAK_CTL4：0x820
  - PWM_BREAK_CTL5：0x860

注：该寄存器仅在 pwm0 有。

位数	名称	方向	复位值	描述
19:17	PWM_BRK_ENABLE	RW	0x0	bit0：使能 pwm0 和 pwm1 的刹车功能，1 使能 bit1：使能 pwm2 和 pwm3 的刹车功能，1 使能 bit2：使能 pwm4 和 pwm5 的刹车功能，1 使能

16	PWM_BRK_SYNC	RW	0x0	内部硬件刹车信号的值，用于软件判断硬件刹车状态 1：硬件刹车信号有效 0：硬件刹车信号无效
14:12	PWM_BRK_MASK	RW	0x0	PWM_BRK 信号屏蔽，每位分别屏蔽一路 PWM_BRK[x],x=0~2 1：屏蔽 0：不屏蔽
11:9	PWM_BRK_INV	RW	0x0	PWM_BRK 信号反向，每位分别反向一路 PWM_BRK[x],x=0~2 1：反向 0：不反向
8	BRK_MODE	RW	0x0	硬件恢复模式和软件恢复模式选择： PWM_BRK=1 停止计数，且 PWM 信号立刻变为高阻，计数器暂停。 0:硬件恢复：PWM_BRK=0 开始计数，自动计数后，必须要等到计数器清 0，PWM 信号才可以从高阻恢复到输出。 1:软件恢复：PWM_BRK=0 之后，软件读取改 PWM_BRK 状态位确认之后，软件使能恢复计数，开始计数后，必须要等到计数器清 0，PWM 信号才可以从高阻恢复到输出。
7	BRK_DBC_EN	RW	0x0	PWM_BRK 信号的滤毛刺使能
6:1	BRK_DBC_STEP	RW	0x0	PWM_BRK 信号的滤毛刺步长。步长为 1 个 pwm_clk 时钟周期
0	BRK_CLEAR	RW	0x0	软件恢复模式退出刹车模式： 1：退出刹车模式 0：无效

13.3.9. CAP_TIM_VAL

- 地址偏移：
  - CAP_TIM_VAL[0]：0x40017C00+0x00
  - CAP_TIM_VAL[1]：0x40017C00+0x04
  - CAP_TIM_VAL[2]：0x40017C00+0x08

位数	名称	方向	复位值	描述
2：0	CAPTURE_VALUE	RW	0x0	捕获值

13.3.10. PWM_CAP_TIM_INT

- 地址偏移：0x40017C00+0x0C

位数	名称	方向	复位值	描述
2	CAPTURE_UPD2	RW	0x0	捕获中断通道 2
1	CAPTURE_UPD1	RW	0x0	捕获中断通道 1
0	CAPTURE_UPD0	RW	0x0	捕获中断通道 0

13.3.11. PWM_CAP_TIM_INT_EN

- 地址偏移：0x40017C00+0x10

位数	名称	方向	复位值	描述
2	capture_upd2_en	RW	0x0	捕获中断使能 2
1	capture_upd1_en	RW	0x0	捕获中断使能 1
0	capture_upd0_en	RW	0x0	捕获中断使能 0

13.3.12. PWM_CAP_TIM_INT_RAW

- 地址偏移：0x40017C00+0x14

位数	名称	方向	复位值	描述
2	capture_upd2_raw	RW	0x0	捕获中断原始状态 2
1	capture_upd1_raw	RW	0x0	捕获中断原始状态 1
0	capture_upd0_raw	RW	0x0	捕获中断原始状态 0

13.3.13. CAP_TIM_CTL

- 地址偏移：
  - CAP_TIM_VAL[0]：0x40017C00+0x18
  - CAP_TIM_VAL[1]：0x40017C00+0x1C
  - CAP_TIM_VAL[2]：0x40017C00+0x20

位数	名称	方向	复位值	描述
23:19	CLK_DIV_VAL	RW	0x0	捕获时钟分频系数
18:16	CAPTURE_SIG_SEL	RW	0x0	捕获信号选择
14	DBC_EN	RW	0x0	去抖使能
13:8	DBC_STEP	RW	0x0	去抖参数
5:4	CAPTURE_MODE	RW	0x0	捕获模式
3:2	CAPTURE_PSC	RW	0x0	捕获速率
1	capture_enable	RW	0x0	捕获使能
0	COMMON_CNT_ENABLE	RW	0x0	捕获计数使能

13.4. 工作流程

13.4.1. PWM 调频调幅模式

PWM 有三种工作模式：调频模式、调幅模式、调频调幅模式。

下面以调频调幅模式为例说明其软件操作流程：

1.配置 CPR 模块中 CPR_PWM_CLK_CTL 寄存器，设置 pwm_clk 的工作频率。配置 CPR 模块中 CPR_CTLAPBCLKEN_GRCTL 寄存器，使能 PWM 接口工作时钟。

2.然后配置 PWMx_EN.PWM_EN（x=[0-5]）位为 0，来暂停相应的 PWM 周期计数器和占空比计数器。

3.向周期设置寄存器和占空比设置寄存器写入新值。

4.向 PWMx_UP.UPDATE（x=[0-5]）位写 1，更新周期寄存器和占空比寄存器。

5.配置 PWMx_EN.PWM_EN（x=[0-5]）位为 1，PWM 以新设置的周期和占空比进行工作。如果需要改变周期和频率，重复 3，4 两步即可。对于调频模式，调幅模式，软件操作流程基本与

调频调幅模式相同，唯一区别是：调频模式只需设置周期设置寄存器，占空比设置寄存器保持不变；调幅模式只需设置占空比设置寄存器，周期设置寄存器保持不变。

13.4.2. PWM 中心对齐模式

1、配置 pwm_mode=2，选择为中心对称模式，原有的 pwm_mode=2（12bit 模式）不再使用，会用 pwm_mode=3 代替。
2、配置 OCPYx_RATIO_CONFIG 为计数中心最大值。配置 OCPYx_RATIO 为输出高电平阈值。
3、内部计数器归零后，才会更新下一次的配置参数。

13.4.3. PWM 通道使能 ADC 的数据采集

PWM 的输出的沿，可以触发 GADC 使能采样（作用和软件使能 GADC 采样相同）。

- 一次触发采样一次

1 使能 PWM 的通道 0，配置为 16~30K 之间;
2 配置 GADC 采样时钟 16M，采样率 1Msample;
3 配置 EXT_EDGE_SEL=1，EXT_SAMPLE_NUM=0;
4 配置 EXT_TRIGGER_SEL=0;
5 配置 GPADCfifo 阈值为 1;
6 则每次 PWM 上升沿，自动触发一次 GPADC 采样，并输出一次中断。

- 一次触发采样 6 次（EXT_SAMPLE_NUM=5）

1 使能 PWM 的通道 0，配置为 16~30K 之间;
2 配置 GADC 采样时钟 16M，采样率 1Msample;
3 配置 EXT_EDGE_SEL=1，EXT_SAMPLE_NUM=5;
4 配置 EXT_TRIGGER_SEL=0;
5 配置 GPADCfifo 阈值为 6;
6 则每次 PWM 上升沿，自动触发 6 次 GPADC 采样;

13.4.4. PWM 刹车信号

PWM_BRKx 信号有效电平可以配置。
3 个 PWM_BRKx 信号“或”产生最终内部刹车信号，每个信号使用 pwm_brk_mask 寄存器独立的屏蔽。
每个 PWM 可以独立的使能刹车功能。使用 pwm_brk_enale 寄存器。
PWM_BRK 信号滤毛刺，档位通过 pwm_dbc_step 寄存器配置，最大滤掉 64us 毛刺，该功能可以 bypass。

工作模式：

PWM_BRK=1 停止计数，且 PWM 信号立刻变为高阻，计数器暂停。

硬件恢复：PWM_BRK=0 开始计数，自动计数后，必须要等到计数器清 0，PWM 信号才可以从高阻恢复到输出。

软件恢复：PWM_BRK=0 之后，软件读取改 PWM_BRK 状态位确认之后，软件使能恢复计数，开始计数后，必须要等到计数器清 0，PWM 信号才可以从高阻恢复到输出。

第 14 章 I2C 接口控制器（I2C）

14.1. 概述

I2C 接口控制器主要实现了 I2C 接口。I2C 挂在 CTL_APB 总线上，提供多主机功能，控制所有 I2C 总线特定的时序、协议、仲裁。可以被 M0 访问。

- I2C 的基地址：0x40006000

14.1.1. 主要特性

- 2 线 I2C 串行接口
- 传输速度
  - 标准模式（100Kb/s）
  - 快速模式（400Kb/s）
  - 支持时钟同步
  - 不支持高速模式（HS MODE）
- 作为 I2CMaster/Slave 工作
- 支持多 Master 方式（总线仲裁）
- 仅支持 7bit 寻址
- 发送 FIFO 位宽为 8 bits，深度 16
- 接收 FIFO 位宽为 8 bits，深度 16

14.1.2. 功能描述

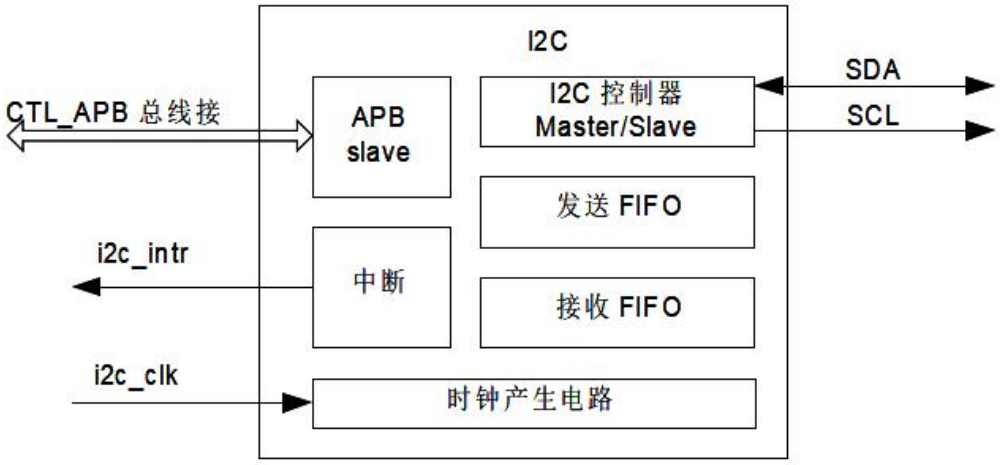

图 14-1 I2C 模块内部结构图

I2C 是 2 线串行接口，可以工作在标准模式（100Kb/s）、快速模式（400Kb/s）和高速

模式（3.4Mb/s）下，高速模式和快速模式向下兼容。例如高速模式器件可以和快速模式、标准模式器件在一个混合速度的 I2C 总线系统中通信，快速模式器件可以和标准模式器件在 0~100kbit/s 的 I2C 总线系统通讯，但是由于标准模式器件不向上兼容所以不能在快速模式 I2C 总线系统中工作，因为它们跟不上这么快的传输速率因而会产生不可预知的结果。

I2C 接口协议包括 Master 和 Slave，Master 负责时钟的产生和传输数据的控制，Slave 只进行数据的接收或者发送。数据传输的确认由数据接收方发出（既可以是 Master，也可以是 Slave）。协议支持多 Master/Slave，这时需要总线的仲裁。

每一个 Slave 有唯一的地址，当 Master 准备与 Slave 通信时，Master 发送一个开始条件，接着发送 Slave 的地址和读写控制比特（R/W）。Slave 接收到地址和读写控制比特之后，发送确认信号（ACK 脉冲）通知 Master 接到数据请求。

如果 Master 发送数据到 Slave，Slave 接收到一个字节数据后，发送确认信号（ACK 脉冲）通知 Master 接到数据，由 Master 发送结束标志以表明数据传输结束。

如果 Master 接收数据，Slave 发送一个字节数据，Master 发送确认（ACK 脉冲），数据传输持续到 Master 发出结束标志为止，此时，Master 在接收到最后一个字节数据之后不再发送 ACK 确认，而是发出结束标志，或者在发出 Restart 信号之后，寻址另一个 Slave。

下面介绍 I2C 协议包含的主要内容：

- 开始、结束条件协议
- Slave 寻址协议
- 数据传输协议

14.1.2.1.1. 开始、结束条件协议

当总线处于空闲状态时，sda 和 scl 均为高电平，这由外部总线上拉电路实现。当 Master 准备进行数据传输时：

- 发出开始条件，定义为：scl 保持高电平的条件下，sda 实现高电平到低电平的跳转。当 Master 决定结束数据传输时，
- 发出结束条件，定义为：scl 保持高电平的条件下，sda 实现低电平到高电平的跳转。

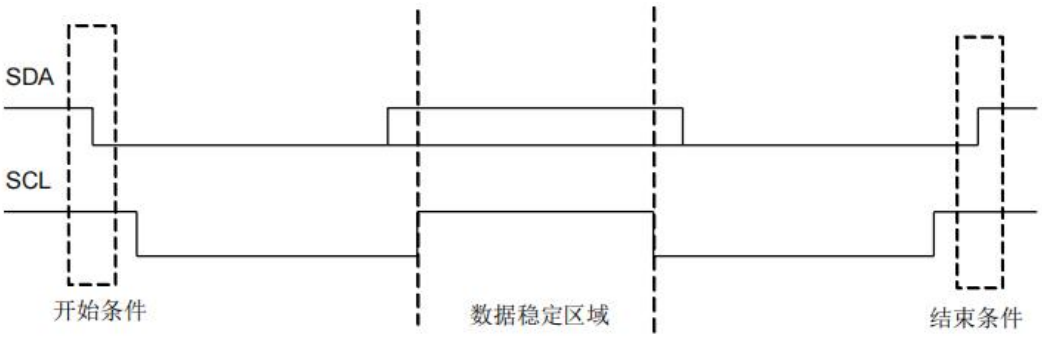

图 14-2 传输开始、结束条件图

14.1.2.1.2. Slave 寻址协议

I2C 协议使用 7 比特格式。在 7 比特格式中第一个字节的前 7 比特[7：1]为 Slave 地址，LSB（比特 0）为读写控制信号（R/W），若此比特为 0，Master 写数据，若为 1，Master 读数据。传输时，首先发送 MSB，最后发送 LSB。图 14-3 为 7 比特地址格式图。

图 14-3 协议 7bit 地址格式图

14.1.2.1.3. 数据传输协议

数据的发送以字节为单位，对于字节数没有限制。在 Master 发送地址和读写控制位或者数据之后，Slave 必须发送确认信号。当没有确认信号发出时，Master 发出结束标志中止此次数据传输。Master 发起数据传输，Slave 对每一个字节均需要发送确认脉冲，如图 14-4 所示。

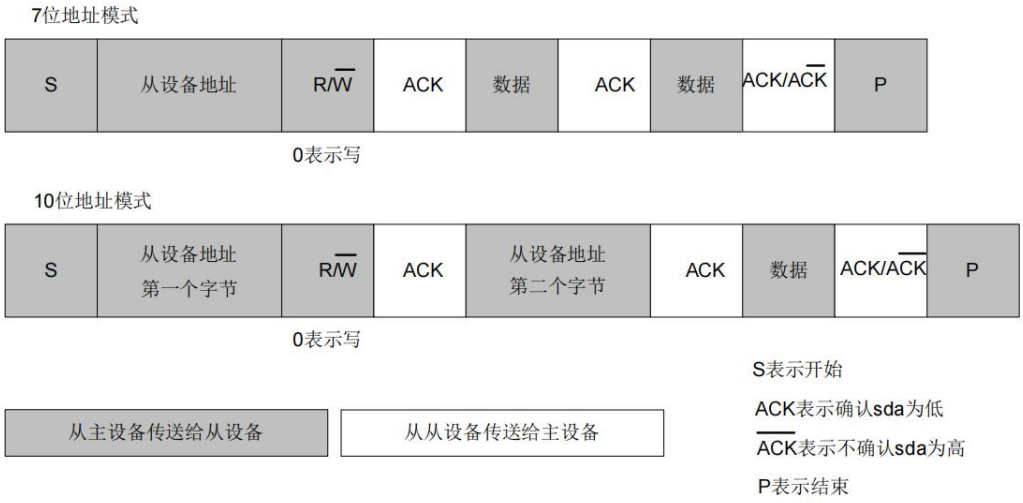

图 14-4 I2C Master 发送数据图

如果 Master 接收数据，在其接收到一个字节数据之后，会发出确认脉冲。对于最后一个字节，Master 不发送确认信号，Master 以这种方式通知 Slave 此字节为最后一个字节。

Slave 在没有接受到确认脉冲时，释放 sda 使得 Master 可以发送结束标志。如图 14-5 所示。

除了通过发送结束标志，Master 还可以通过发送 Restart（重新开始）标志来放弃总线，Restart 标志与开始标志类似，区别在于 Restart 标志发生在确认信号之后。此时，Master 可以与原来的 Slave 进行通信，也可以与其它 Slave 开始进行通信。

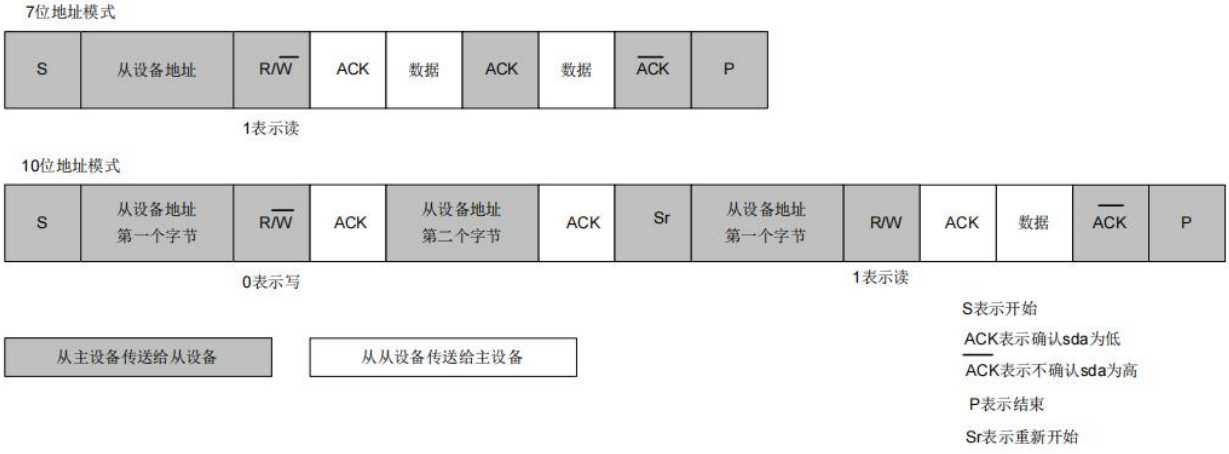

图 14-5 I2C Master 接收数据图

14.1.2.1.4. Start Byte 传输协议

I2C 模块作为 Slave 时，总是以最高速率采样 I2C 总线，而不需要 Start Byte 传输。I2C 模块作为 Master，对于需要 Start Byte 的 Slave，在每一次数据传输之前，发出 Start Byte。

Start Byte 为 7 个 0 和 1 个 1，如图 14-6 所示。在这种情况下，允许工作于低速查询总线方式的处理器采样数据线，直到识别到 0，此时处理器可以调整到 Master 所确定的数据

传输速率。slave 不对 Start Byte 进行响应和确认（因为它是一个保留的地址），slave 在 Restart 条件产生后复位。

Start Byte 协议的过程如下：

1.Master 发出开始条件

2.Master 发出一个 Start Byte

3.Master 发出 ACK 确认时钟脉冲

4.任何 Slave 不将 ACK 置 0

5.Master 发出一个 Restart 条件

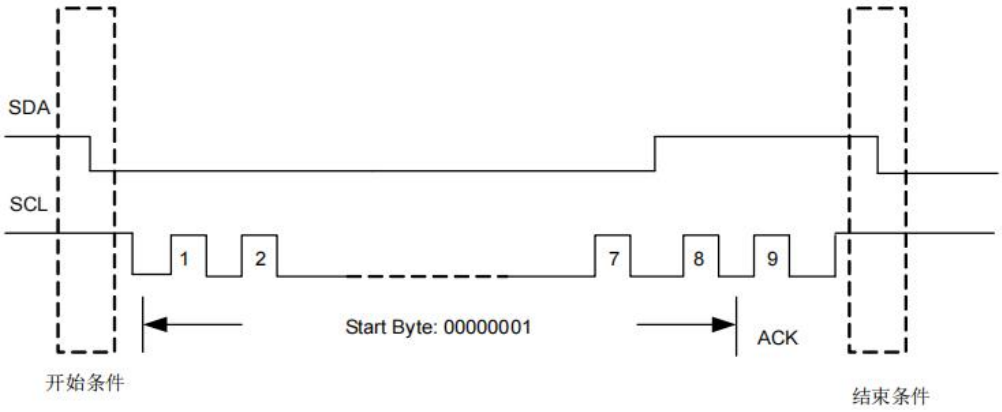

图 14-6 Start Byte 传输协议图

14.1.2.1.5. 发送 FIFO 管理和 STOP 信号的产生

当 I2C_DATA_CMD.STOP 设置为 1 时，当前字节发送完成时，不管发送 FIFO 是否为空，都产生 STOP 信号，如果发送 FIFO 非空，I2C 接口模块立即产生 START 信号去获取总线。

当 I2C_DATA_CMD.STOP 设置为 0 时，当前字节发送完成时，不管发送 FIFO 是否为空，都不产生 STOP 信号：

- 发送 FIFO 非空时，I2C 接口模块会根据 I2C_DATA_CMD.CMD 设置继续发送/接收字节。
- 发送 FIFO 空时，I2C 模块保持 SCL 低电平直到发送 FIFO 中有可用数据为止。
- 只有用户向 I2C_DATA_CMD 的 bit9（停止位）写 1 才会产生 STOP 信号。

I2C_DATA_CMD 的组成如图 14-7 所示。

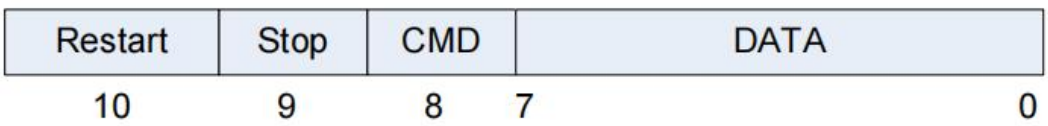

图 14-7 I2C_DATA_CMD 的构成

DATA：可读写，从 slave 接收到的数据或者准备发送到 slave 的数据。

CMD：只写，读写控制位，0 表示写操作，1 表示读操作。

Stop：只写，该位决定字节发送或接收完成是否产生 STOP 信号。

Restart：只写，该位决定字节发送或接收之前是否产生 RESTART 信号（在 restart 不使能时，用 STOP 后接着 START 信号代替）。

如下图所示，阐明了 master 作为发送器件时，发送 FIFO 变为空时总线上的变化，同时也说明了 STOP 信号产生的条件。

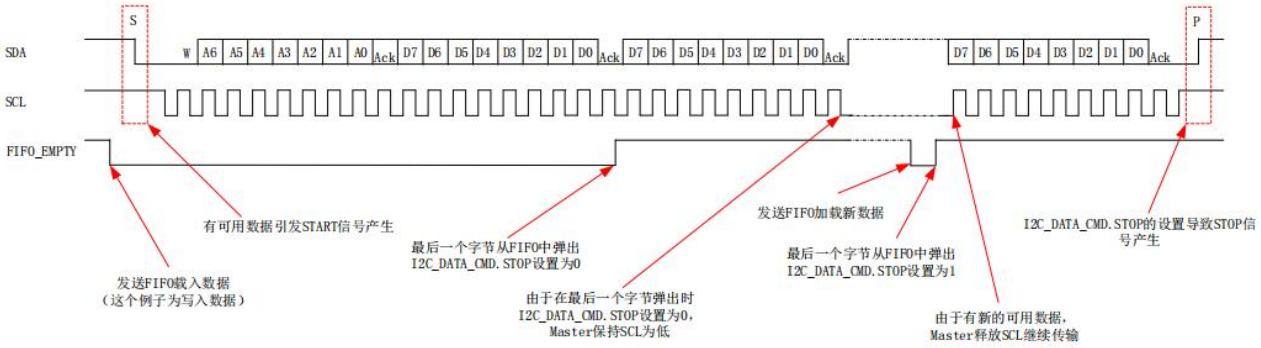

图 14-8 Master 发送，发送 FIFO 空时 STOP 信号的产生图

如下图所示，阐明了 master 作为接收器件时，发送 FIFO 变为空时总线上的变化，同时也说明了 STOP 信号。

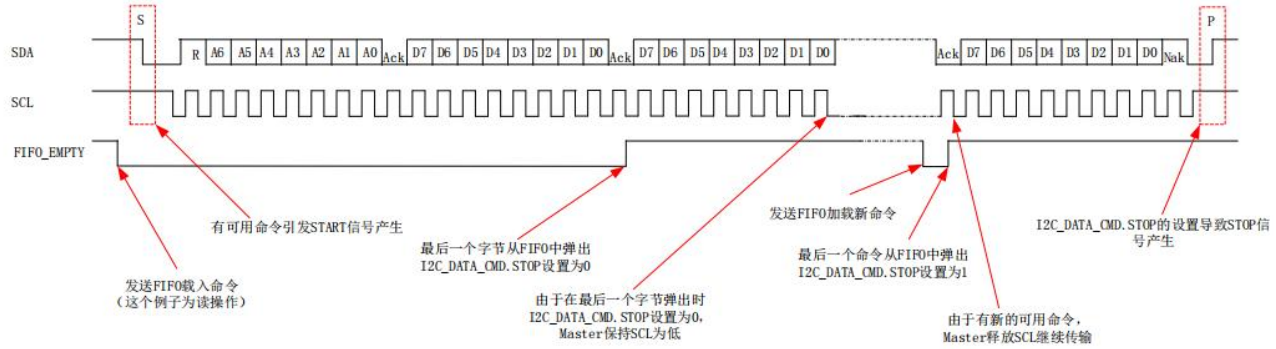

图 14-9 Master 接收，发送 FIFO 空时 STOP 信号的产生图

图 14-8 和图 14-9 阐明了用户可以通过配置控制 RESTART 信号的产生。如果 I2C_DATA_CMD.RESTART 设置为 1 并且 I2C_CON.I2C_RESTART_EN 设置为 1，在写入数据到 salve 或者从 slave 读出数据之前会产生 RESTART 信号。如果 I2C_DATA_CMD.RESTART 设置为 1 并且 I2C_CON.I2C_RESTART_EN 设置为 0，则会产生一个 STOP 信号紧跟着一个 START 信号来代替 RESTART 信号。图 14-10 阐明了 Master 做为发送器件时的这种情况。图 14-11 阐明了 Master 做为接收器件时的这种情况。

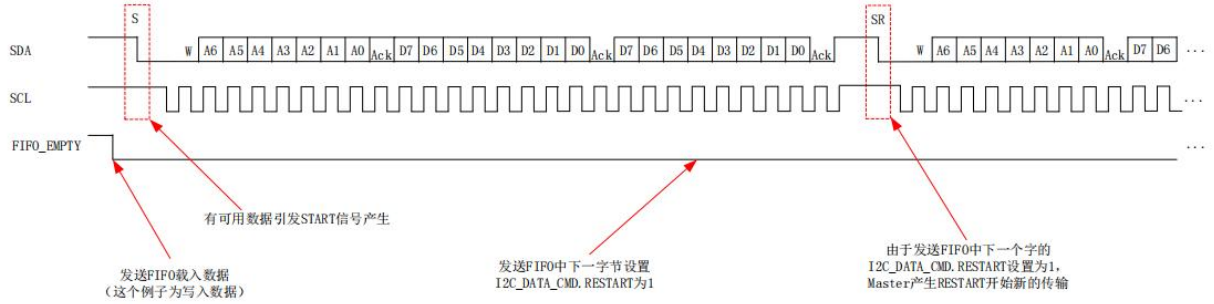

图 14-10 Master 发送，I2C_DATA_CMD.RESTART 设置为 1 时时序图

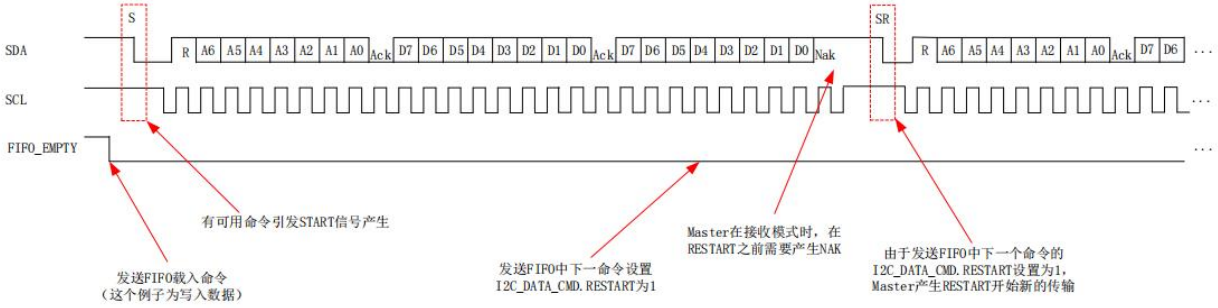

图 14-11 Master 接收，I2C_DATA_CMD.RESTART 设置为 1 时时序图

图 14-12 阐明了 Master 做为发送器件时，I2C_DATA_CMD.STOP 设置为 1 并且发送 FIFO 非空的情况。

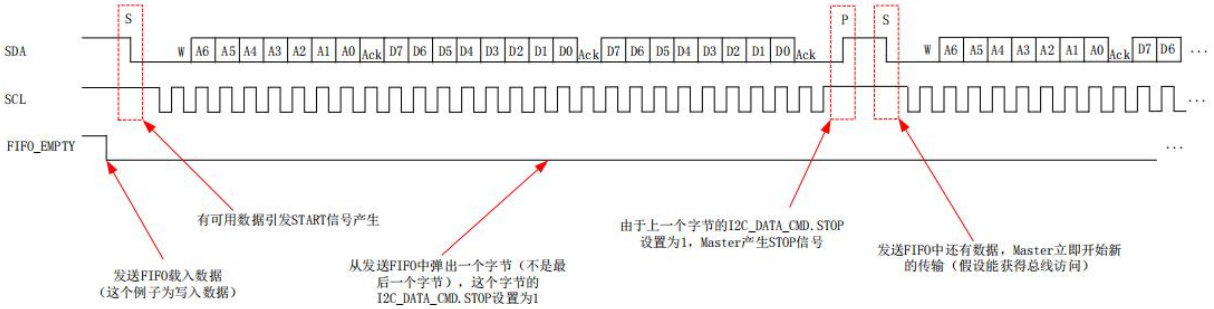

图 14-12 Master 发送，I2C_DATA_CMD.STOP 设置为 1 并且发生 FIFO 非空时时序图

图 14-13 阐明了 Master 做为发送器件时，I2C_DATA_CMD.RESTART 设置为 1 并且发送 FIFO 空之后第一个字节加载的情况。

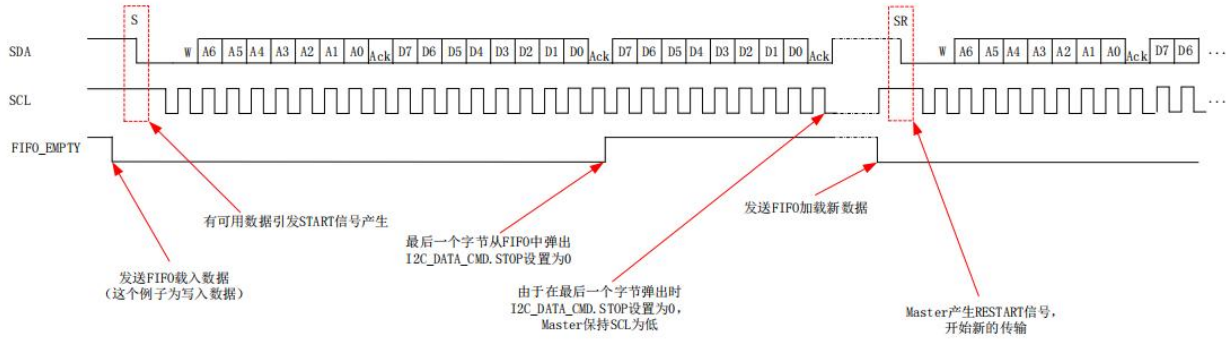

图 14-13 Master 发送，I2C_DATA_CMD.RESTART 设置为 1 并且发送 FIFO 空之后第一个字节

图 14-14 阐明了 Master 做为接收器件时，I2C_DATA_CMD.STOP 设置为 1 并且发送 FIFO

非空的情况。

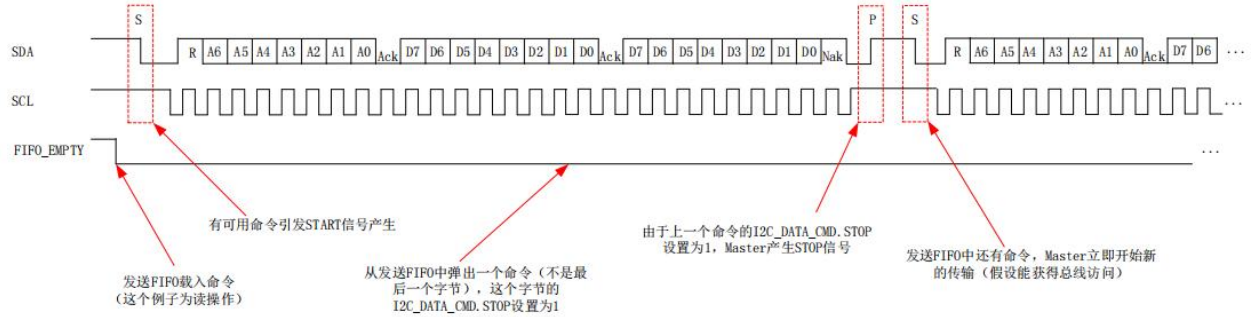

图 14-14 Master 接收，I2C_DATA_CMD.STOP 设置为 1 并且发送 FIFO 非空时序图

图 14-15 阐明了 Master 做为接收器件时，I2C_DATA_CMD.RESTART 设置为 1 并且发送 FIFO 空之后第一个命令加载的情况。

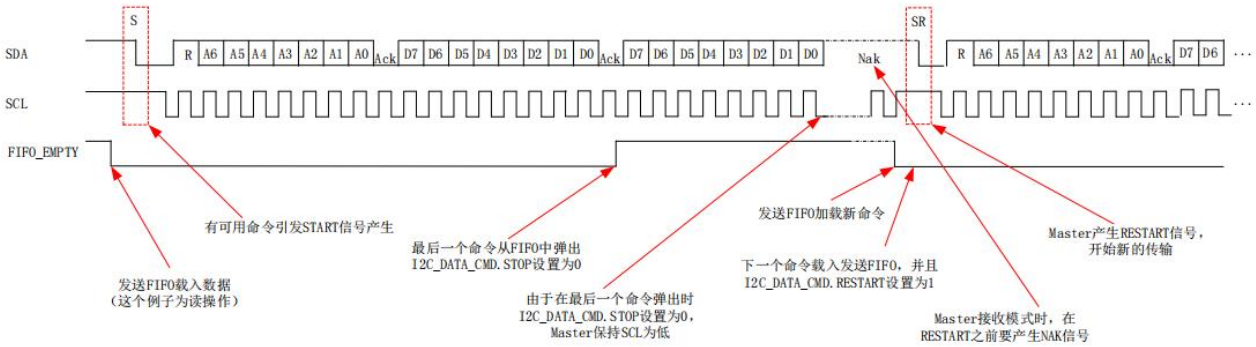

图 14-15 Master 接收，I2C_DATA_CMD.RESTART 设置为 1 并且发送 FIFO 空之后第一个命令

14.1.2.1.6. 多 Master 总线仲裁和时钟同步

I2C 接口模块支持多 Master，当多个 Master 准备在同一时刻进行数据传输时，需要总线仲裁和 scl 时钟同步。

总线仲裁：总线仲裁发生在 sda 线上，此时 scl 为 1。传输’1’的 Master 在其余 Master 传输’0’时失去总线，关闭其数据输出，但是 Master 可以继续产生 scl 信号直到此字节传输结束。如果所有的 Master 寻址相同的 Slave，仲裁允许进入数据级。

时钟同步：所有的 Master 生成自己的时钟，数据在 scl 为高时有效，时钟的同步通过 scl 连接线与（Wire-And）实现。

1. 当 Master 将 scl 置 0，Master 计数 scl 为低电平的时间，当达到时钟周期时，将 scl 置 1。
2. 如果此时有其它的 Master 将 scl 保持在 0，则 Master 进入 HIGH WAIT 状态，直到 scl 变成 1。
3. 所有的 Master 计算 scl 高电平的时间，高电平最短的 Master 首先将 scl 置 0，随之，所有的 Master 计算 scl 低电平的时间，低电平最长的 Master 强制其余的 Master

进入 HIGH WAIT 状态，至此，实现了时钟的同步。一种可选的方案是 Slave 保持 scl 为低电平，以实现降低传输速率。

14.1.2.1.7. SDA 保持时间

sda 相对于 scl 下降沿的数据保持时间可以通过寄存器 I2C_SDA_HOLD 来调整，以 i2c_clk 周期为单位，最小的调整周期为 1 个 i2c_clk。

当 I2C_SDA_HOLD 为 0 时，sda 数据保持时间为 1 个 i2c_clk。

I2C_SDA_HOLD 为其它值时，sda 数据保持时间为(I2C_SDA_HOLD-1)*i2c_clk。

I2C_SDA_HOLD 只有在 I2C_ENABLE 为 0 时才能配置，配置的最小值一定要大于数据保持时间的下限，最大值不能超过(SCL 的低电平时间-2)*i2c_clk。图 14-16 为 I2C_SDA_HOLD 设置为 3 的时序图。

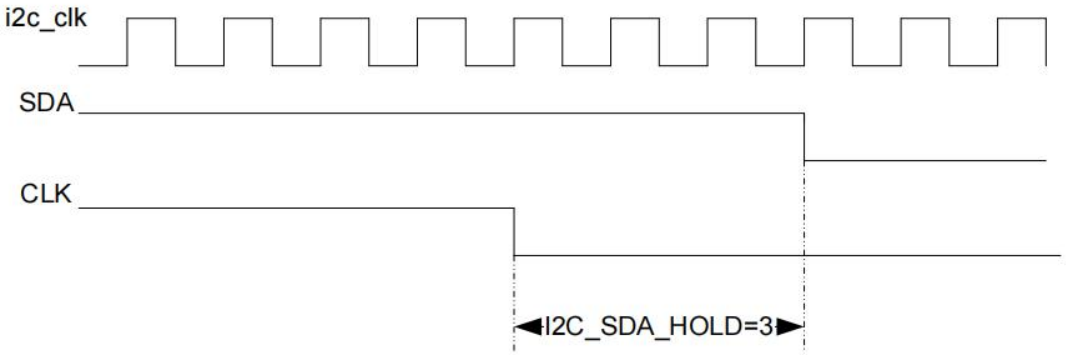

图 14-16 I2C_SDA_HOLD 设置为 3 的时序图

14.1.2.1.8. 中断模式

表 14-1 为本模块支持的所有中断列表。

中断状态寄存器 I2C_INTR_STAT 显示各个中断的状态。

中断原始状态寄存器 I2C_RAW_INTR_STAT 显示各个中断的原始状态。

中断使能寄存器 I2C_INTR_EN 包含各个中断的使能控制。

对于一个中断而言，只有当中断原始状态位为 1，并且相应的中断使能位为 1 时，才会触发中断管脚 i2c_intr 的有效状态，引起处理器的中断。本模块的中断输出（i2c_intr）送到 M0。

表 14-1 I2C 控制器中断模式列表

中断状态位	中断原始状态位	中断使能位	描述
GEN_CALL	R_GEN_CALL	EN_GEN_CALL	通用寻址中断
START_DET	R_START_DET	EN_START_DET	开始条件中断
STOP_DET	R_STOP_DET	EN_STOP_DET	结束条件中断
ACTIVITY	R_ACTIVITY	EN_ACTIVITY	活动状态中断

RX_DONE	R_RX_DONE	EN_RX_DONE	作为 slave 发送数据时，如果 master 没有响应的话，在发送最后一个字节后产生此中断
TX_ABRT	R_TX_ABRT	EN_TX_ABRT	作为 master，当接口模块无法完成处理器的命令时，产生中断。
RD_REQ	R_RD_REQ	EN_RD_REQ	作为 slave，当另外一个 master 尝试从 slave 读数据时产生此中断
TX_EMPTY	R_TX_EMPTY	EN_TX_EMPTY	TX FIFO 数据小于或等于 TX FIFO 阈值，产生中断
TX_OVER	R_TX_OVER	EN_TX_OVER	TX FIFO 写溢出错误中断
RX_FULL	R_RX_FULL	EN_RX_FULL	RX FIFO 数据大于或等于 RX FIFO 阈值，产生中断
RX_OVER	R_RX_OVER	EN_RX_OVER	RX FIFO 写溢出错误中断
RX_UNDER	R_RX_UNDER	EN_RX_UNDER	RX_FIFO 读空错误中断

14.1.3. 信号及接口描述

表 14-2 I2C 控制器接口描述表

名称	宽度	方向	描述	备注
CTL_APB 接口				内部信号
中断				
i2c_intr	1	O	I2C 中断输出，送到 M0	内部信号
I2C 接口				
i2c_clk	1	I	I2C 模块工作时钟，来自 CPR，默认频率 8M	内部信号
i2c_scl	1	IO	I2C 总线时钟	管脚信号
i2c_sda	1	IO	I2C 总线数据	管脚信号

14.2. I2C 寄存器地址映射

I2C 的基地址：0x40006000

寄存器地址映射和描述如下表所示：

表 14-3 I2C 寄存器概述和地址映射表

寄存器	描述	方向和宽度	地址偏移	复位值
I2C_CON	I2C 控制寄存器	RW 7bits	0x00	0x7D
I2C_TAR	I2C 目标地址寄存器	RW 13bits	0x04	0x1019
I2C_SAR	I2C 从地址寄存器	RW 10bits	0x08	0x055
I2C_DATA_CMD	I2C Rx/Tx 数据缓存和命令寄存器	RW 9 bits	0x10	0x0
I2C_SS_SCL_HCNT	标准速率 I2C 时钟高电平计数寄存器	RW 16bits	0x14	0x190
I2C_SS_SCL_LCNT	标准速率 I2C 时钟低电平计数寄存器	RW 16bits	0x18	0x1d6
I2C_FS_SCL_HCNT	快速速率 I2C 时钟高电平计数寄存器	RW 16bits	0x1C	0x3C
I2C_FS_SCL_LCNT	快速速率 I2C 时钟低电平计数寄存器	RW 16bits	0x20	0x82
I2C_INTR_STAT	I2C 中断状态寄存器	R 12bits	0x2C	0x0
I2C_INTR_EN	I2C 中断使能寄存器	RW 12bits	0x30	0x8FF
I2C_RAW_INTR_STAT	I2C 中断原始状态寄存器	R 12bits	0x34	0x0
I2C_RX_TL	接收 FIFO 阈值寄存器	RW 8bits	0x38	0x8
I2C_TX_TL	发送 FIFO 阈值寄存器	RW 8bits	0x3C	0x8
I2C_CLR_INTR	清除全局中断寄存器	R 1bit	0x40	0x0
I2C_CLR_RX_UNDER	清除 RX_UNDER 中断寄存器	R 1bit	0x44	0x0
I2C_CLR_RX_OVER	清除 RX_OVER 中断寄存器	R 1bit	0x48	0x0
I2C_CLR_RD_REQ	清除 RD_REQ 中断寄存器	R 1bit	0x50	0x0
I2C_CLR_TX_ABRT	清除 TX_ABRT 中断寄存器	R1bit	0x54	0x0
I2C_CLR_RX_DONE	清除 RX_DONE 中断寄存器	R1bit	0x58	0x0
I2C_CLR_ACTIVITY	清除 ACTIVITY 中断寄存器	R1bit	0x5c	0x0
I2C_CLR_STOP_DET	清除 STOP_DET 中断寄存器	R1bit	0x60	0x0
I2C_CLR_START_DET	清除 START 中断寄存器	R1bit	0x64	0x0
I2C_CLR_GEN_CALL	清除 GEN_CALL 中断寄存器	R1bit	0x68	0x0
I2C_ENABLE	I2C 使能寄存器	RW 1bit	0x6C	0x0
I2C_STATUS	I2C 状态寄存器	R 5bits	0x70	0x6
I2C_TXFLR	发送 FIFO 数据计数寄存器	R 5bits	0x74	0x0

I2C_RXFLR	接收 FIFO 数据计数寄存器	R 5bits	0x78	0x0
I2C_SDA_HOLD	SDA 保持时间寄存器	RW 16bits	0x7C	0x1
I2C_TX_ABRT_SOURCE	I2C 发送异常中止源寄存器	RW 16bits	0x80	0x0

14.3. I2C 寄存器描述

14.3.1. I2C 控制寄存器（I2C_CON）

- 地址偏移：0x00

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
6	I2C_SLAVE_DISABLE	RW	0x1	Slave 模式使能控制位 0：Slave 模式使能 1：Slave 模式不使能
5	I2C_RESTART_EN	RW	0x1	Master 模式下发出 Restart 的使能位 1：使能 0：不使能 如果禁止 Restart 指示，则无法实现： ● 一次传输发送多个字节 ● 一次传输中改变方向 ● 发出 Start Byte ● 7 字节地址和 10 字节地址下执行组合格式传输 ● 10 字节地址下执行读操作
4	I2C_10BITADDR_MASTER_R	R	0x1	Master 模式下地址模式的只读标志位 1：10-bit 地址模式 0：7-bit 地址模式
3	I2C_10BITADDR_SLAVE	RW	0x1	Slave 模式时可响应的地址模式 0：7-bit 地址模式 1：10-bit 地址模式
2：1	SPEED	RW	0x3	确定模块工作模式： 0：非法，导致工作在高速模式 1：标准模式（0 到 100kbit/s） 2：快速模式（≤400kbit/s） 3：保留
0	MASTER_MODE	RW	0x1	Master 使能控制 1：Master 使能 0：Master 不使能

注：I2C_SLAVE_DISABLE（Bit6）和 MASTER_MODE（Bit0）的某些取值组合会导致配置错误，下面列出了这两个比特的取值组合以及对应的状态：

- I2C_SLAVE_DISABLE=0，MASTER_MODE=0：Slave 模式
- I2C_SLAVE_DISABLE=0，MASTER_MODE=1：配置错误
- I2C_SLAVE_DISABLE=1，MASTER_MODE=0：配置错误

- I2C_SLAVE_DISABLE=1，MASTER_MODE=1：Master 模式

14.3.2. I2C 目标地址寄存器（I2C_TAR）

- 地址偏移：0x04

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
12	I2C_10BITADDR_MASTER_R	RW	0x1	确定 Master 模式下传输的地址模式 1：10-bit 地址模式 0：7-bit 地址模式
11	SPECIAL	RW	0x0	确定软件是否执行通用寻址或者发送 Start Byte 命令 1：执行 GC_OR_START 所描述的特殊 I2C 命令 0：忽略 GC_OR_START，正常使 I2C_TAR
10	GC_OR_START	RW	0x0	如果 SPECIAL＝1 时： 1：Start Byte 0：通用寻址地址 通用寻址时，只有写命令能够被行，如果发出读命令，则 TX_ABRT 置位。保持通用寻址模式直到 SPECIAL 复位。 如果 SPECIAL＝0 时：无定义。
9：0	I2C_TAR	RW	0x019	Master 数据传输的目标地址

14.3.3. I2C 从地址寄存器（I2C_SAR）

- 地址偏移：0x08

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
15:10	NA	–	–	保留
9：0	I2C_SAR	RW	0x055	Slave 数据传输的目标地址

14.3.4. I2C Rx/Tx 数据缓存和命令寄存器（I2C_DATA_CMD）

- 地址偏移：0x10

位数	名称	方向	复位值	描述
15:11	NA	–	–	保留
10	RESTART	W	0x0	该位控制在发送或接收一个字节之前是否产生 RESTART 信号。 RESTART=1： ● 当 I2C_CON.I2C_RESTART_EN 设置为 1 时，不管传输方向是否改变，都产生 RESTART 信号。 ● 当 I2C_CON.I2C_RESTART_EN 设置为 0 时，会产生 STOP 和 START 信号代替 RESTART 信号。 RESTART=0： ● 当 I2C_CON.I2C_RESTART_EN 设置为 1 时，只有当传输方向发生改变才产生 RESTART 信号。 ● 当 I2C_CON.I2C_RESTART_EN 设置为 0 时，会产生 STOP 和 START 信号代替 RESTART 信号。
9	STOP	W	0x0	该位控制在一个字节发送、接收之后是否产生 STOP 信号。 STOP=1： ● 不管发送 FIFO 是否为空，当前字节完成后产生 STOP 信号。发送 FIFO 非空时，I2C 接口模块会立即产生 START 信号去获取总线。

				STOP=0： ● 不管发送 FIFO 是否为空，当前字节完成后不产生 STOP 信号。发送 FIFO 非空时总线根据 I2C_DATA_CMD.CMD 位设置 续发送/接收字节；发送 FIFO 为空时，I2C 接口模块保持 SCL 低电平直到发送 FIFO 中有新的 可用命令
8	CMD	RW	0x0	读写控制。 CMD=1：读。 CMD=0：写。 ● 对于读模式，低 8 位被忽略。 ● 如果通用寻址之后发生读操作，产生 TX_ABRT 中断，除非 I2C_TAR 寄存器 SPECIAL 位被清除。 ● 如果收到 RD_REQ 后向此位写 1，产生 TX_ABRT 中断。
7：0	DAT	RW	0x0	包含需要发送或者接收到的 I2C。 总线数据

14.3.5. 标准速率 I2C 时钟高电平计数寄存器（I2C_SS_SCL_HCNT）

● 地址偏移：0x14

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
15：0	I2C_SS_SCL_HCNT	RW	0x190	为确保正确的 I/O 时序，此寄存器需要在任何数据传输之前设置，设置标准模式下 scl 高电平计数，最小的合法值为 6。

参数和 scl 的关系：

速率（Kbps）	i2c_clk（Mhz）	scl 高电平（最小值 us）	I2C_SS_SCL_HCNT	scl 高电平（实际值 us）
100	2	4	0x08	4.00
100	6.6	4	0x1B	4.09
100	10	4	0x28	4.00
100	75	4	0x12C	4.00
100	100	4	0x190	4.00
100	125	4	0x1F4	4.00

14.3.6. 标准速率 I2C 时钟低电平计数寄存器（I2C_SS_SCL_LCNT）

- 地址偏移：0x18

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
15：0	I2C_SS_SCL_LCNT	RW	0x1d6	为确保正确的 I/O 时序，此寄存器需要在任何数据传输之前设置，设置标准模式下 scl 低电平计数，最小的合法值为 8。

参数和 scl 的关系：

速率（Kbps）	i2c_clk（Mhz）	scl 低电平（最小值 us）	I2C_SS_SCL_LCNT	scl 低电平（实际值 us）
100	2	4.7	0x0A	5.00
100	6.6	4.7	0x20	4.85
100	10	4.7	0x2F	4.70
100	75	4.7	0x161	4.71
100	100	4.7	0x1D6	4.70
100	125	4.7	0x24C	4.70
100	1000	4.7	0x125C	4.70

14.3.7. 快速速率 I2C 时钟高电平计数寄存器（I2C_FS_SCL_HCNT）

- 地址偏移：0x1C

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
15：0	I2C_FS_SCL_HCNT	RW	0x3C	为确保正确的 I/O 时序，此寄存器需要在任何数据传输之前设置。 设置快速模式下 scl 高电平计数，最小的合法值为 6。

参数和 scl 的关系：

速率（Kbps）	i2c_clk（Mhz）	scl 高电平（最小值 us）	I2C_SS_SCL_HCNT	scl 高电平（实际值 us）
400	10	0.6	0x06	0.60
400	25	0.6	0x0F	0.60
400	50	0.6	0x1E	0.60
400	75	0.6	0x2D	0.60
400	100	0.6	0x3C	0.60
400	125	0.6	0x4B	0.60
400	1000	0.6	0x258	0.60

14.3.8. 快速速率 I2C 时钟低电平计数寄存器（I2C_FS_SCL_LCNT）

- 地址偏移：0x20

该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入，其余时刻写入数据无效。

位数	名称	方向	复位值	描述
15：0	I2C_FS_SCL_LCNT	RW	0x82	为确保正确的 I/O 时序，此寄存器需要在任何数据传输之前设置。 设置快速模式下 scl 低电平计数，最小的合法值为 8。

参数和 scl 的关系：

速率（Kbps）	i2c_clk（Mhz）	scl 低电平（最小值 us）	I2C_FS_SCL_LCNT	scl 低电平（实际值 us）
400	10	1.3	0x0D	1.30
400	25	1.3	0x21	1.32
400	50	1.3	0x41	1.30
400	75	1.3	0x62	1.31
400	100	1.3	0x82	1.30
400	125	1.3	0xA3	1.30
400	1000	1.3	0x514	1.30

14.3.9. I2C 中断状态寄存器（I2C_INTR_STAT）

- 地址偏移：0x2C

位数	名称	方向	复位值	描述
11	GEN_CALL	R	0x0	通用寻址请求标志 1： 收到通用寻址请求，接口模块将接收到的数据放入 RX 缓存中 0：无效
10	START_DET	R	0x0	开始条件标志： 1：总线出现开始条件 0：无效：
9	STOP_DET	R	0x0	结束条件标志： 1：总线出现结束条件 0：无效
8	ACTIVITY	R	0x0	活动状态标志 1： 接口模块活动状态，保持置位直到软件清除 0：无效
7	RX_DONE	R	0x0	作为 slave 发送数据时，如果 master 没有响应的话，在发送最后一个字节后产生此中断，表示发送结束
6	TX_ABRT	R	0x0	传输异常标志： 作为 master，当接口模块无法完成处

				理器的命令时，此位置 1 的条件包括： ● 地址发出之后无 slave 确认信号 ● 被寻址的 slave 没有发出数据 ● 仲裁丢失 ● I2C_RESTART_EN 置 0，但是处理器发出在没有 restart 条件就无法完成的命令 ● Start Byte 被确认 ● 通用寻址地址无确认 ● 读请求中断发生时，TX FIFO 有残留数据，或者当 I2C_RESTART_EN 不使能时，在数据传输时丢失总线的控制，又被当作 I2C slaved 使用 ● 通用寻址命令之后发出读命令 ● 响应 RD_REQ 之前，处理器 发出读命令 此位一旦置 1，发送和接收 FIFO 自动清空。 0：无效
5	RD_REQ	R	0x0	作为 slave，当另外一个 master 尝试从 slave 读数据时产生此中断。Slave 保持 I2C 总线为等待状态（SCL=0）直到中断服务被响应。 这表示 slave 被远端的 master 寻址，并且要求其发送数据。处理器必须响应这个中断，并将待发送的数据写入 I2C_DATA_CMD 寄存器。
4	TX_EMPTY	R	0x0	TX FIFO 空标志 1:TX FIFO 数据小于或者等于 TXFIFO 阈值。此比特随 FIFO 状态自动更新。 0：无效
3	TX_OVER	R	0x0	TX FIFO 写溢出标志

				1：TX FIFO 写溢出错误 0：无效
2	RX_FULL	R	0x0	RX FIFO 满标志 1：RX FIFO 数据大于或者等于 RX FIFO 阈值。此比特随 FIFO 状态自动更新。 0：无效
1	RX_OVER	R	0x0	RX FIFO 写溢出标志 1：RX FIFO 写溢出错误 0：无效
0	RX_UNDER	R	0x0	RX FIFO 读空标志 1：RX_FIFO 读空错误 0：无效

14.3.10. I2C 中断使能寄存器（I2C_INTR_EN）

- 地址偏移：0x30

位数	名称	方向	复位值	描述
11	EN_GEN_CALL	RW	0x1	通用寻址中断使能 1：中断使能 0：不使能
10	EN_START_DET	RW	0x0	开始标志中断使能 1：中断使能 0：不使能
9	EN_STOP_DET	RW	0x0	结束标志中断使能 1：中断使能 0：不使能
8	EN_ACTIVITY	RW	0x0	活动状态中断使能 1：中断使能 0：不使能
7	EN_RX_DONE	RW	0x0	Slave 发送数据完成中断 1：中断使能 0：不使能
6	EN_TX_ABRT	RW	0x1	传输异常中断使能

				1：中断使能 0：不使能
5	EN_RD_REQ	RW	0x1	Slave 收到发送数据请求中断 1：中断使能 0：不使能
4	EN_TX_EMPTY	RW	0x1	TX FIFO 空中断使能 1：中断使能 0：不使能
3	EN_TX_OVER	RW	0x1	TX FIFO 写满中断使能 1：中断使能 0：不使能
2	EN_RX_FULL	RW	0x1	RX FIFO 满中断使能 1：中断使能 0：不使能
1	EN_RX_OVER	RW	0x1	RX FIFO 写满中断使能 1：中断使能 0：不使能
0	EN_RX_UNDER	RW	0x1	RX FIFO 读空中断使能 1：中断使能 0：不使能

14.3.11. I2C 中断原始状态寄存器（I2C_RAW_INTR_STAT）

- 地址偏移：0x34

位数	名称	方向	复位值	描述
11	R_GEN_CALL	R	0x0	对应中断状态寄存器 GEN_CALL 位 1：有效 0：无效
10	R_START_DET	R	0x0	对应中断状态寄存器 START_DET 位 1：有效 0：无效
9	R_STOP_DET	R	0x0	对应中断状态寄存器 STOP_DET 位 1：有效

				0：无效
8	R_ACTIVITY	R	0x0	对应中断状态寄存器 ACTIVITY 位 1：有效 0：无效
7	R_RX_DONE	R	0x0	对应中断状态寄存器 RX_DONE 位 1：有效 0：无效
6	R_TX_ABRT	R	0x0	对应中断状态寄存器 TX_ABRT 位 1：有效 0：无效
5	R_RD_REQ	R	0x0	对应中断状态寄存器 RD_REQ 位 1：有效 0：无效
4	R_TX_EMPTY	R	0x0	对应中断状态寄存器 TX_EMPTY 位 1：有效 0：无效
3	R_TX_OVER	R	0x0	对应中断状态寄存器 TX_OVER 位 1：有效 0：无效
2	R_RX_FULL	R	0x0	对应中断状态寄存器 RX_FULL 位 1：有效 0：无效
1	R_RX_OVER	R	0x0	对应中断状态寄存器 RX_OVER 位 1：有效 0：无效
0	R_RX_UNDER	R	0x0	对应中断状态寄存器 RX_UNDER 位 1：有效 0：无效

14.3.12. 接收 FIFO 阈值寄存器（I2C_RX_TL）

- 地址偏移：0x38

位数	名称	方向	复位值	描述
7：0	RX_TL	RW	0x8	RXFIFO 阈值。 当 RX FIFO 数据大于 RX FIFO 阈值时，引发 RX_FULL 中断。 当设置成 0 时，表示 RX FIFO 数据个数为 1 时发出中断，依次类推。当设置的值大于 15 时，本寄存器的值均 15 0x0：1 个 0x1：2 个 …… 0xF：16 个

14.3.13. 发送 FIFO 阈值寄存器（I2C_TX_TL）

- 地址偏移：0x3C

位数	名称	方向	复位值	描述
7：0	TX_TL	RW	0x8	TX FIFO 阈值。 当 TX FIFO 中数据个数小于或者等于 TX FIFO 阈值，引发 TX_EMPTY 中断。 当设置的值大于 15 时，本寄存器的值均为 15。 注意与 I2C_RX_TL 不同，当设置为 1 时，表示 TX FIFO 数据项为 1 时就会发出中断。 0x0：0 个 0x1：1 个 …… 0xF：15 个

14.3.14. 清除全局中断寄存器（I2C_CLR_INTR）

- 地址偏移：0x40

位数	名称	方向	复位值	描述
0	CLR_INTR	R	0x0	读此寄存器清除组合中断，各个独立中断以及 I2C_TX_ABRT_SOURCE 寄存器。

14.3.15. 清除 RX_UNDER 中断寄存器（I2C_CLR_RX_UNDER）

- 地址偏移：0x44

位数	名称	方向	复位值	描述
0	CLR_RX_UNDER	R	0x0	读此寄存器清除中断 RX_UNDER

14.3.16. 清除 RX_OVER 中断寄存器（I2C_CLR_RX_OVER）

- 地址偏移：0x48

位数	名称	方向	复位值	描述
0	CLR_RX_OVER	R	0x0	读此寄存器清除中断 RX_OVER

14.3.17. 清除 TX_OVER 中断寄存器（I2C_CLR_TX_OVER）

- 地址偏移：0x4C

位数	名称	方向	复位值	描述
0	CLR_TX_OVER	R	0x0	读此寄存器清除中断 TX_OVER

14.3.18. 清除 RD_REQ 中断寄存器（I2C_CLR_RD_REQ）

- 地址偏移：0x50

位数	名称	方向	复位值	描述
0	CLR_RD_REQ	R	0x0	读此寄存器清除中断 RD_REQ

14.3.19. 清除 TX_ABRT 中断寄存器（I2C_CLR_TX_ABRT）

- 地址偏移：0x54

位数	名称	方向	复位值	描述
0	CLR_TX_ABRT	R	0x0	读此寄存器清除中断 TX_ABRT

14.3.20. 清除 RX_DONE 中断寄存器（I2C_CLR_RX_DONE）

- 地址偏移：0x58

位数	名称	方向	复位值	描述
0	CLR_RX_DONE	R	0x0	读此寄存器清除中断 RX_DONE。

14.3.21. 清除 ACTIVITY 中断寄存器（I2C_CLR_ACTIVITY）

- 地址偏移：0x5C

位数	名称	方向	复位值	描述
0	CLR_ACTIVITY	R	0x0	读此寄存器清除中断 ACTIVITY，I2C 模块处于激活状态时无法清除该中断，只有当 I2C 处于非激活状态时才能清除。

14.3.22. 清除 STOP_DET 中断寄存器（I2C_CLR_STOP_DET）

- 地址偏移：0x60

位数	名称	方向	复位值	描述
0	CLR_STOP_DET	R	0x0	读此寄存器清除中断 STOP_DET。

14.3.23. 清除 START 中断寄存器（I2C_CLR_START_DET）

- 地址偏移：0x64

位数	名称	方向	复位值	描述
0	CLR_START_DET	R	0x0	读此寄存器清除中断 START_DET。

14.3.24. 清除 GEN_CALL 中断寄存器（I2C_CLR_GEN_CALL）

- 地址偏移：0x68

位数	名称	方向	复位值	描述
0	CLR_GEN_DET	R	0x0	读此寄存器清除中断 GEN_CALL。

14.3.25. I2C 使能寄存器（I2C_ENABLE）

- 地址偏移：0x6C

位数	名称	方向	复位值	描述
0	CLR_GEN_DET	RW	0x0	接口模块使能控制。 1：使能 0：禁止 接口模块处于激活状态时，软件不能进行禁止设置。可通过 I2C 状态寄存器（I2C_STATUS）来查询模块是否处于激活状态。 使能/禁止存在两个 i2c_clk 延迟。

14.3.26. I2C 状态寄存器（I2C_STATUS）

- 地址偏移：0x70

位数	名称	方向	复位值	描述
4	RFF	R	0x0	RX FIFO 满。 1：RX FIFO 满 0：RX FIFO 不满
3	RFNE	R	0x0	RX FIFO 不空。 1：RX FIFO 不空 0：RX FIFO 空
2	TFE	R	0x1	TX FIFO 空。 1：TX FIFO 空 0：TX FIFO 不空

1	TFNF	R	0x1	TX FIFO 不满。 1：TX FIFO 不满 0：TX FIFO 满
0	ACTIVITY	R	0x0	激活状态。 1：激活 0：不激活

14.3.27. 发送 FIFO 数据计数寄存器（I2C_TXFLR）

- 地址偏移：0x74

I2C 被禁止，传输异常中止或者 Slave Bulk 异常中止时，该寄存器清 0。

位数	名称	方向	复位值	描述
4：0	TXFLR	R	0x0	TX FIFO 有效数据项个数

14.3.28. 接收 FIFO 数据计数寄存器（I2C_RXFLR）

- 地址偏移：0x78

I2C 被禁止或者传输异常中止时，该寄存器清 0。

位数	名称	方向	复位值	描述
4：0	RXFLR	R	0x0	RX FIFO 有效数据项个数

14.3.29. SDA 保持时间（I2C_SDA_HOLD）

- 地址偏移：0x7C

设置 SDA 保持时间。该寄存器只有在模块不使能时（相对应 I2C_ENABLE 寄存器置 0）才能被写入。

位数	名称	方向	复位值	描述
15：0	I2C_SDA_HOLD	RW	0x1	设置以 i2c_clk 为单位的 SDA 保持时间

14.3.30. I2C 发送异常中止源寄存器（I2C_TX_ABRT_SOURCE）

- 地址偏移：0x80

指示 TX_ABRT 源，读该寄存器或者处理器发出清除所有中断时清 0。

位数	名称	方向	复位值	描述
15：13	NA	—	—	保留
12	ABRT_LOST	RW	0x0	1：Master:丢失总线控制权，或者

				本寄存器 bit14 置 1，表明发送数据的 Slave 丢失总线控制权。
11	ABRT_MASTER_DIS	RW	0x0	1:处理器企图使用禁止的 Master。
10	ABRT_10B_RD_NORSTRT	RW	0x0	1：Restart 被禁止，Master 发送一个 10 比特地址读命令。
9	ABRT_SBYTE_NORSTRT	RW	0x0	1：Restart 被禁止，处理器企图发出 Start Byte。
8	NA	—	—	保留
7	ABRT_SBYTE_ACKDET	RW	0x0	1：Master 发送 Start Byte 而 Start Byte 得到确认。
6	NA	—	—	保留
5	ABRT_GCALL_READ	RW	0x0	1：Master 发出通用寻址，而处理器紧接着发出读请求。
4	ABRT_GCALL_NOACK	RW	0x0	1：Master 发出通用寻址而无 Slave 响应。
3	ABRT_TXDATA_NOACK	RW	0x0	1：Master 收到地址确认，但是发送数据之后无确认。
2	ABRT_10ADDR2_NOACK	RW	0x0	1：Master 处于 10 比特地址模式，而第二地址字节无确认。
1	ABRT_10ADDR1_NOACK	RW	0x0	1：Master 处于 10 比特地址模式，第一地址字节无确认。
0	ABRT_7B_ADDR_NOACK	RW	0x0	1：Master 处于 7 比特地址模式，地址字节无确认。

14.4. 工作流程

14.4.1. I2C 作为 Master 发送数据

(1) 读取 I2C 状态寄存器，判断 I2C 是否处于激活状态，如果 I2C_STATUS 的 bit0 为 0 则进行下一步，如果为 1 则等待；

(2) 写 0 到 I2C_ENABLE 寄存器，不使能 I2C 接口控制器；

(3) 配置 I2C_CON 寄存器，设置 I2C 控制接口能否发出 restart 指示，选择 I2C 接口控制器工作方式（标准模式或者快速模式），master 使能；

(4) 根据选择的工作方式，配置相对应的 I2C 时钟高、低电平计数寄存器和 I2C_SDA_HOLD 寄存器，使 scl 信号和 sda 信号满足时序要求；

(5) 配置 I2C_TX_TL 寄存器，设置发送 FIFO 阈值；

(6) 读 I2C_CON 寄存器，通过 I2C_10BITADDR_MASTER_R 比特可知当前是工作在 10-bit 地址模式还是 7-bit 地址模式；

(7) 写 I2C_TAR 寄存器，设置接收数据的 slave 地址，设置是否增加 StartByte 以及是否启动通用寻址，通过 I2C_10BITADDR_MASTER 比特设置 I2C 发起者 7-bit 地址模式；

(8) 配置中断使能控制寄存器 I2C_INTR_EN，使能发送 FIFO 空中断，该空中断与发送 FIFO 的阈值有关。

(9) 写 1 到 I2C_ENABLE 寄存器，使能 I2C 接口控制器；

(10) 每次接收到发送 FIFO 空中断，向 I2C_DATA_CMD 写要发送的数据和配置状态（RESTART、STOP、CMD），直到所要发送的数据发送完成。

14.4.2. I2C 作为 Master 接收数据

(1) 读取 I2C 状态寄存器，判断 I2C 是否处于激活状态，如果 I2C_STATUS 的 bit0 为 0 则进行下一步，如果为 1 则等待；

(2) 写 0 到 I2C_ENABLE 寄存器，不使能 I2C 接口控制器；

(3) 配置 I2C_CON 寄存器，设置 I2C 控制接口能否发出 restart 指示，选择 I2C 接口控制器工作方式（标准模式或者快速模式），master 使能；

(4) 根据选择的工作方式，配置相对应的 I2C 时钟高、低电平计数寄存器和 I2C_SDA_HOLD 寄存器，使 scl 信号和 sda 信号满足时序要求；

(5) 配置 I2C_RX_TL 寄存器，设置接收 FIFO 阈值；

(6) 配置 I2C_TX_TL 寄存器，设置发送 FIFO 阈值；

(7) 读 I2C_CON 寄存器，通过 I2C_10BITADDR_MASTER_R 比特可知当前是工作在 10-bit 地址模式还是 7-bit 地址模式；

(8) 写 I2C_TAR 寄存器，设置发送数据的 slave 地址，设置是否增加 StartByte 以及是否启动通用寻址，通过 I2C_10BITADDR_MASTER 比特设置 I2C 发起 10-bit 地址模式或者 7-bit 地址模式；

(9) 配置中断使能寄存器 I2C_INTR_EN 寄存器，使能 I2C 接口控制器的接收 FIFO 满中断；

(10) 写 1 到 I2C_ENABLE 寄存器，使能 I2C 接口控制器；

(11) 由于发送 FIFO 的深度为 16，而接收数据的数据量是通过向 I2C_DATA_CMD.CMD 写 1 来控制的，故当要接收的数据量超过 16 个时，要先向 I2C_DATA_CMD.CMD 写 1，并且配置 I2C_DATA_CMD 的状态位（RESTART、STOP），等到发送 FIFO 有空间时再向 I2C_DATA_CMD.CMD 写 1 及配置 I2C_DATA_CMD 的状态位（RESTART、STOP）。例如要求

I2C 接收 18 个数据时，并且每个字节接收之前不产生 RESART，接收之后也不产生 STOP，先连续向 I2C_DATA_CMD（发送 FIFO）写 16 次 0x100，根据 I2C_TX_TL 的数值和发送 FIFO 的空中断，来判断当发送 FIFO 中有两个空位后再写 1 次 0x100 和 1 次 0x300（最后一次设置 I2C_DATA_CMD.STOP 为 1 以便产生 STOP 信号）；

(12) 根据发送 FIFO 的空满状态和写 1 到 I2C_DATA_CMD.CMD 的次数，等待接收 FIFO 满中断，读出数据，清除接收满中断，等待下一个接收 FIFO 满中断或者传输结束中断，直到所有的数据接收完。

14.4.3. I2C 作为 Slave 发送数据

(1) 配置 I2C_CON 寄存器的 I2C_SLAVE_DISABLE 为 0，MASTER_MODE 为 0，选择 Slave 模式；
(2) 设置 I2C_CON.I2C_10BITADDR_SLAVE，确定可以响应的地址类型；
(3) 设置 I2C 的 I2C_SAR，设置从地址；
(4) 设置 I2C 时钟高电平低电平计数器，确定时钟波形；
(5) 设置 I2C 的接收发送 FIFO 的阈值；
(6) 设置 I2C 的使能控制寄存器为 1，使能 I2C 模块；
(7) 使能 I2C 的所有中断，以便进行数据传输；
(8) I2C 检测 RDREQ 中断，当检测到中断后写 I2C_DATA_CMD 寄存器，将数据发送出去。

14.4.4. I2C 作为 Slave 接收数据

(1) 配置 I2C_CON 寄存器的 I2C_SLAVE_DISABLE 为 0，MASTER_MODE 为 0，选择 Slave 模式；
(2) 设置 I2C_CON.I2C_10BITADDR_SLAVE，确定可以响应的地址类型；
(3) 设置 I2C 的 I2C_SAR，设置从地址；
(4) 设置 I2C 时钟高电平低电平计数器，确定时钟波形；
(5) 设置 I2C 的接收发送 FIFO 的阈值；
(6) 设置 I2C 的使能控制寄存器为 1，使能 I2C 模块；
(7) 使能 I2C 的所有中断，以便进行数据传输；
(8) I2C 检测 RXFULL 中断（bit2oftheI2C_INTR_STAT），当检测到中断后读 I2C_DATA_CMD 寄存器，从而接收数据。

14.4.5. I2C 高、低电平计数寄存器配置

在使用 I2C 模块进行数据传输之前，必须正确配置*CNT 寄存器，确保正确的 I/O 时序。有效的计数寄存器包括：I2C_SS_SCL_HCNT，I2C_SS_SCL_LCNT，I2C_FS_SCL_HCNT，I2C_FS_SCL_LCNT。

*_LCNT 配置 scl 保持低电平的 i2c_clk 时钟数，最小值为 8，是由 I2C 控制器在 scl 下降沿驱动 sda 的时间决定的；

*_HCNT 配置 scl 保持高电平的 i2c_clk 时钟数，最小值为 6，是由 I2C 控制器在 scl 为高时采样 sda 的时间决定的；

I2C 控制器会在*_LCNT+1 的基础上产生 scl 低电平；在*_HCNT+8 的基础上产生 scl 高电平。I2C 控制器能产生的最小 scl 低电平时间为 9 个 i2c_clk 时钟；最小的 scl 高电平时间为 14 个 i2c_clk 时钟；

根据 I2C 规范中时序的要求和 I2C 接口控制器的特性，i2c_clk 和高低电平计数值之间的依赖关系如表所示

工作模式	i2c_clk(MHz)	SCL 低电平计数	SCL 低电平编程值	SCL 低电平时间	SCL 高电平计数	SCL 高电平编程值	SCL 低电平时间
SS	2.7	13	12	4.7us	14	6	5.2us
FS	12	16	15	1.33	14	6	1.16us

下面的方法能计算出合适的 I2C 时钟高、低电平值（以快速模式为例），为了适应该公式，i2c_clk 必须不能小于表 3-28 中的值。该方法计算出的值可能高于 I2C 规范中规定的速率，这时需要减小高电平或低电平的值，以适应规范要求。

I2C_FS_SCL_yCNT =(ROUNDUP (MIN_SCL_ytime *OSCFREQ，0))

注：y = H，L； ROUNDUP 是 EXCEL 中的四舍五入函数。

其中：MIN_SCL_Htime = 最小高电平时间。

MIN_SCL_Htime 在标准速率下是 4000ns；在快速模式下是 600ns。

MIN_SCL_Ltime = 最小低电平时间。

MIN_SCL_Ltime 在标准速率下是 4700ns；在快速模式下是 1300ns。

OSCFREQ = i2c_clk 时钟频率（Hz）。

举例说明：

OSCFREQ = 100 MHz

I2C 模式为快速模式，即 400 kbit/s

MIN_SCL_HIGHtime = 600 ns.

MIN_SCL_LOWtime = 1300 ns

I2C_FS_SCL_HCNT = （ROUNDUP（600 ns * 100 MHz，0））

I2C_FS_SCL_HCNT = 60

I2C_FS_SCL_LCNT = （ROUNDUP（1300 ns * 100 MHz，0））

I2C_FS_SCL_LCNT = 130

另一种计算方法是根据 I2C 工作时钟频率和 I2C 数据传输的最小时钟高低电平时间来配置标准、快速 I2C 时钟高、低电平计数寄存器的值。

例如 I2C 的工作时钟为 13Mhz，I2C 工作于标准速率模式时 scl 信号保持为高电平的最短时间要求是 4000ns，则 I2C_SS_SCL_HCNT 的寄存器的值 value0 要满足：（value0/13Mhz）≥4000ns，得到 value0≥42，即 I2C_SS_SCL_HCNT 的寄存器的值大于等于 42 时满足要求。

I2C_SS_SCL_LCNT 的寄存器的值 value1 要满足:（value1/13Mhz）≥4700ns，得到 value1≥62，即 I2C_SS_SCL_LCNT 的寄存器的值大于等于 62 时满足要求。

第 15 章 串行外设接口（SPI）

串行外设接口（SPI）是一种高速的、全双工的、同步的通信总线。SPI 以主从方式工作，通常是有一个主设备和一个或多个从设备，一般 SPI 需要 4 根线，但是也可以使用三根线(单向传输)。

SPI 模块可以被 M0 和 DMAS 访问。

- SPI0 的基地址为：0x40013000
- SPI1 的基地址为：0x40014000
- SPI2 的基地址为：0x40014800

15.1. SPI 概述

15.1.1. 主要特性

- 具有可分别使能的中断。
- 用户可调整数据传输速率。
- 用户可设置数据帧格式（4—16 位）。
- 接收 FIFO 和发送 FIFO 深度为 16，宽度为 16bits。
- 支持 Motorola SPI 协议和 TI SSP 协议。
- SPI 同步串行主设备。
- SPI 与 DMA 的接收硬件握手和发送硬件握手。

15.1.2. 功能描述

SPI 是全双工的同步串行接口，SPI 作为主设备接口，产生串口时钟（ssi_clk）和串口使能/帧同步信号（ssi_ssn0）。如果 SPI 不使能，不进行串口传输，并且串口时钟（ssi_clk）保持在无效状态，不同的串口协议决定无效状态为高或者为低。

数据传输由主设备发起，发起的条件是：SPI 使能、发送 FIFO 中至少有一个数据并且从设备选择寄存器 SPI_SE 的 SS0 控制位为 1。

SPI 作为主设备的协议选择是通过 CPR 模块的 SSI 控制寄存器（CPR_SSI_CTRL）和 SPI 模块的 SPI 控制寄存器 0（SPI_CTRL0）共同选择来完成的。

Motorola SPI 协议设置需要进行以下操作步骤：

1、设置 CPR 模块 SSI 控制寄存器（CPR_SSI_CTRL）的 bit[0]为‘1’。

2、SPI 模块的 SPI 控制寄存器 0（SSI_CTRL0）的 bit[5：4]为‘00’。

Motorola SPI 协议是一个 4 线、全双工的串口协议。根据串口时钟（ssi_clk）的相位（SCPH）和极性（SCPOL）的不同，有 4 种组合。在主设备 SPI 处于不使能或者空闲的状态

下，主设备输出的从设备选择信号 ssi_ssn0 保持高电平。

在使用Motorola SPI 协议进行传输之前，主设备和从设备必须配置相同的 SCPH 和SCPOL 参数。

- 时钟相位（SCPH）参数决定数据传输是在从设备使能信号（ssi_ssn0）的下降沿开始还是在串口时钟（ssi_clk）的第一个变化沿开始。通过 SPI 控制寄存器 0（SPI_CTRL0）的 SCPH 比特来配置。
  - SCPH=0，在从设备使能信号（ssi_ssn0）的下降沿开始发送数据，并在串口时钟（ssi_clk）的第一个沿采样输入数据。
  - SCPH=1，在从设备使能信号（ssi_ssn0）有效后，第一个串口时钟（ssi_clk）的变化沿开始发送数据，并在第二个串口时钟（ssi_clk）的变化沿采样输入数据。
- 时钟极性（SCPOL）参数决定串口时钟（ssi_clk）在无效的状态下是高电平还是低电平。通过 SPI 控制寄存器 0（SPI_CTRL0）的 SCPOL 比特来配置。
  - SCPOL=0，串口时钟（ssi_clk）在无效的状态下保持低电平。
  - SCPOL=1，串口时钟（ssi_clk）在无效的状态下保持高电平。

图 15-1 是 SCPH=0，SCPOL=0 的 SPI0 协议时序图。

图 15-2 是 SCPH=1，SCPOL=0 的 SPI0 协议时序图。

图 15-3 是 SCPH=0，SCPOL=1 的 SPI0 协议时序图。

图 15-4 是 SCPH=1，SCPOL=1 的 SPI0 协议时序图。

图中 ssi_ssn0、ssi_clk 对于主设备 SPI 为输出；ssi_tx 为输出，ssi_rx 为输入。

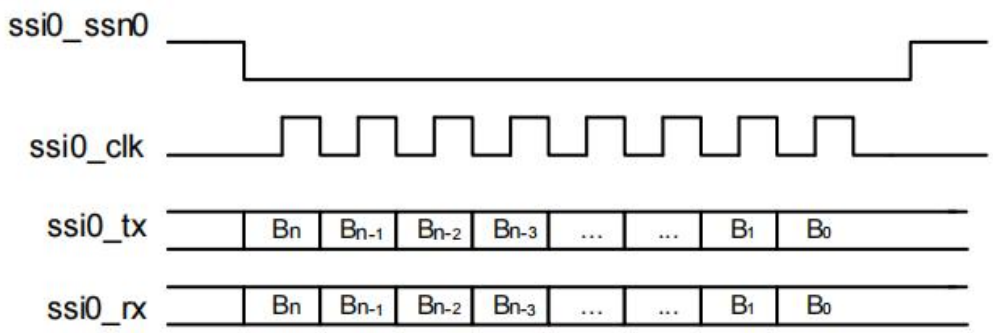

图 15-1 SPI0 协议时序图（SCPH=0，SCPOL=0）

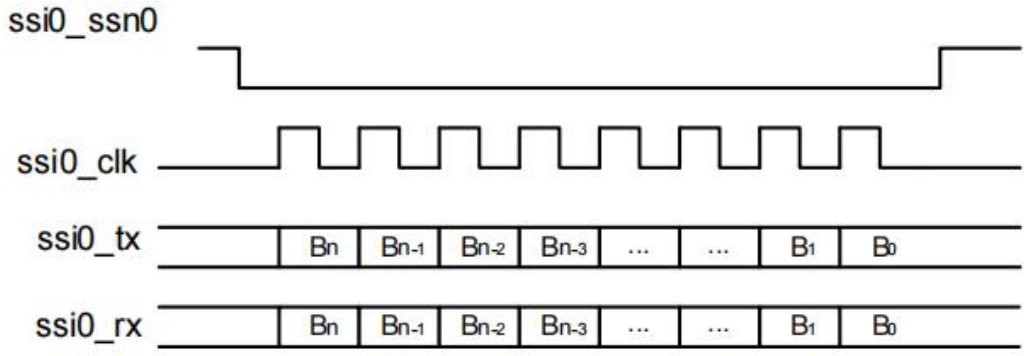

图 15-2 SPI0 协议时序图（SCPH=1，SCPOL=0）

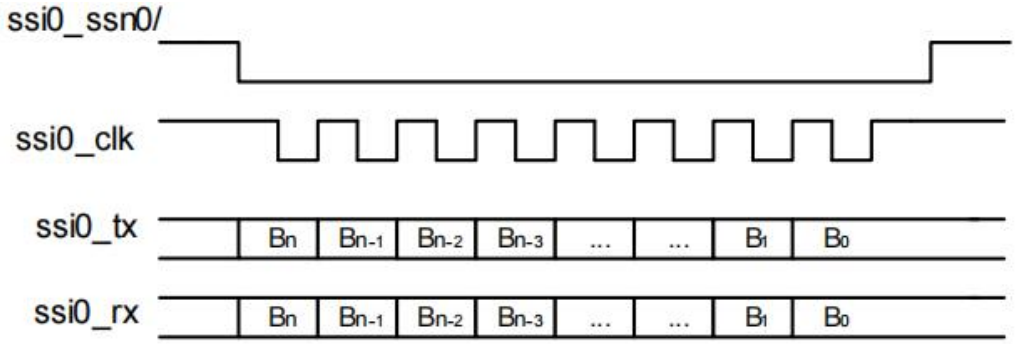

图 15-3 SPI0 协议时序图（SCPH=0，SCPOL=1）

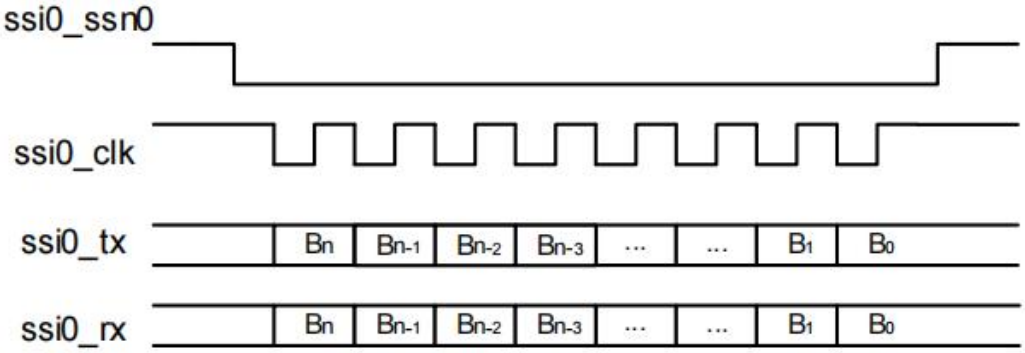

图 15-4 SPI0 协议时序图（SCPH=1，SCPOL=1）

15.1.2.1. 工作模式

SPI 作为主设备与外部从设备进行通信，传输是由主设备 SPI 发起。

SPI 在数据传输过程中可以有如下三种工作模式，通过 SPI 控制寄存器 0（SPI_CTRL0）的 TMOD 位设置。

- 发送和接收
- 只发送，仅发
- 只接收，仅收

SPI 可能的状态：不使能，空闲以及 BUSY。

SPI 还提供了一种测试模式，在 SPI 内部实现环回。

15.1.2.2. 发送和接收

当 TMOD=0x00，SPI 的发送和接收都有效。

SPI 按照设置好的帧格式（串口协议）将发送 FIFO 的数据通过 ssi_tx 送给从设备，同时接收来自 ssi_rx 的数据并存入到接收 FIFO 中。

SPI 传输开始的条件是：SPI 使能、发送 FIFO 中至少有一个数据并且从设备选择寄存器 SPI_SE 的 SS0 控制位为 1。

SPI 传输结束的条件是：发送 FIFO 为空。对于连续的数据传输，软件必须保证在所有数据发送完之前，发送 FIFO 不曾完全空（0 个）过；否则 SPI 进行的传输将是不连续的传输。

15.1.2.3. 只发送，仅发

当 TMOD=0x01，SPI 的发送有效，而接收无效。

SPI 按照设置好的帧格式（串口协议）将发送 FIFO 的数据通过 ssi_tx 送给从设备，同时接收来自 ssi_rx 的数据，但是不会将接收到的数据存入到接收 FIFO 中。在进入该工作模式前，应该先不使能所有接收中断。

SPI 传输开始的条件是：SPI 使能、发送 FIFO 中至少有一个数据并且从设备选择寄存器 SPI_SE 的 SS0 控制位为 1。

SPI 传输结束的条件是：发送 FIFO 为空。对于连续的数据传输，软件必须保证在所有数据发送完之前，发送 FIFO 不曾完全空（0 个）过；否则 SPI 进行的传输将是不连续的传输。

15.1.2.4. 只接收，仅收

当 TMOD=0x02，SPI 的接收有效，而发送无效。

SPI 按照设置好的帧格式(串口协议)反复发送最后一个发送的数据,同时接收来自 ssi_rx 的数据，并存入到接收 FIFO 中。在进入该工作模式前，应该先不使能所有发送中断。

SPI 传输开始的条件是：SPI 使能、发送 FIFO 中至少有一个数据并且从设备选择寄存器 SPI_SE 的 SS0 控制位为 1。

SPI 传输结束的条件是：由 SPI 控制寄存器 1（SPI_CTRL1）的 NDF（帧个数）来控制的。当 SPI 接收到 NDF（帧个数）+1 个数据后，结束本次传输。

15.1.2.5. SPI 状态

SPI 可能处于的状态有：不使能，空闲以及 BUSY。

- 不使能

  SPI 在 SPI 使能寄存器（SPI_EN）的 SEN=0 时，处于不使能状态。

- 空闲

主设备（SPI）的 SEN=1，但是不满足其他传输开始的条件，比如发送 FIFO 为空或者从设备选择都为 0，此时的 SPI 处于空闲状态。

- BUSY

当 SPI 在传输的进行过程中，SPI 处于 BUSY 状态，并且 SPI 状态寄存器（SPI_STS）的 BUSY 标志用于指示是否处于该状态。

15.1.2.6. 测试模式

SPI 可以工作于测试模式，是在 SPI 内部做环回，将发送寄存器的输出连接到接收寄存器的输入。这样就可以在没有外部从设备的时候，对 SPI 做一些功能性的测试，自己发送自己接收。

15.1.3. 中断模式

SPI 对外有一个中断请求信号 ssi_intr，SPI 的中断请求可以单独使能。

15.1.3.1. 发送 FIFO 空中断（TXE_INTR）

当发送 FIFO 中的数据个数小于等于软件设置的阈值时，发送 FIFO 空中断（TXE_INTR）的原始中断状态（TXEIR）置‘1’。发送 FIFO 空中断（TXE_INTR）的中断状态（TXEIS）是否置‘1’，还要看中断使能位（TXEIE）是否有效。

阈值是通过 SPI 发送 FIFO 阈值寄存器（SPI_TXFTL）的 TFT 来设置的。

如果向发送 FIFO 写数据，使 FIFO 中的数据个数超过阈值时，硬件自动清除该中断。

15.1.3.2. 发送 FIFO 上溢出中断（TXO_INTR）

当发送 FIFO 中的数据已经完全满（8 个）时，CTL_APB 总线接口还试图向发送 FIFO 中写数据，发送 FIFO 上溢出中断（TXO_INTR）的原始中断状态（TXOIR）置‘1’，总线试图写入发送 FIFO 的值丢失。发送 FIFO 上溢出中断（TXO_INTR）的中断状态（TXOIS）是否置‘1’，还要看中断使能位（TXOIE）是否有效。

通过读 SPI 发送 FIFO 上溢出中断清除寄存器（SPI_TXOIC）或者 SPI 中断清除寄存器（SPI_IC）可以清除该中断。

15.1.3.3. 接收 FIFO 满中断（RXF_INTR）

当接收 FIFO 中的数据个数大于等于软件设置的阈值时，接收 FIFO 满中断（RXF_INTR）的原始中断状态（RXFIR）置‘1’。接收 FIFO 满中断（RXF _INTR）的中断状态（RXFIS）是否置‘1’，还要看中断使能位（RXFIE）是否有效。

阈值是通过 SPI 接收 FIFO 阈值寄存器（SPI_RXFTL）的 RFT 来设置的。

如果 CTL_APB 总线读取接收 FIFO 数据，使 FIFO 中的数据个数小于阈值时，硬件自动清除该

中断。

15.1.3.4. 接收 FIFO 上溢出中断（RXO_INTR）

当接收 FIFO 中的数据已经完全满（8 个）时，仍然从外部接收数据写入接收 FIFO，接收 FIFO 上溢出中断（RXO_INTR）的原始中断状态（RXOIR）置‘1’，从外部接收的数据丢失。接收 FIFO 上溢出中断（RXO_INTR）的中断状态（RXOIS）是否置‘1’，还要看中断使能位（RXOIE）是否有效。

通过读 SPI 接收 FIFO 上溢出中断清除寄存器（SPI_RXOIC）或者 SPI 中断清除寄存器（SPI_IC）可以清除该中断。

15.1.3.5. 接收 FIFO 下溢出中断（RXU_INTR）

当接收 FIFO 中的数据已经完全空（0 个）时，CTL_APB 总线接口还试图从接收 FIFO 中读数据，接收 FIFO 下溢出中断（RXU_INTR）的原始中断状态（RXUIR）置‘1’，并且总线从接收 FIFO 中读回的数为 0。接收 FIFO 下溢出中断（RXU_INTR）的中断状态（RXUIS）是否置‘1’，还要看中断使能位（RXUIE）是否有效。

通过读 SPI 接收 FIFO 下溢出中断清除寄存器（SPI_RXUIC）或者 SPI 中断清除寄存器（SPI_IC）可以清除该中断。

15.1.4. 信号及接口描述

图 15-5 为 SPI 的接口信号图，包括芯片内部信号和管脚信号。

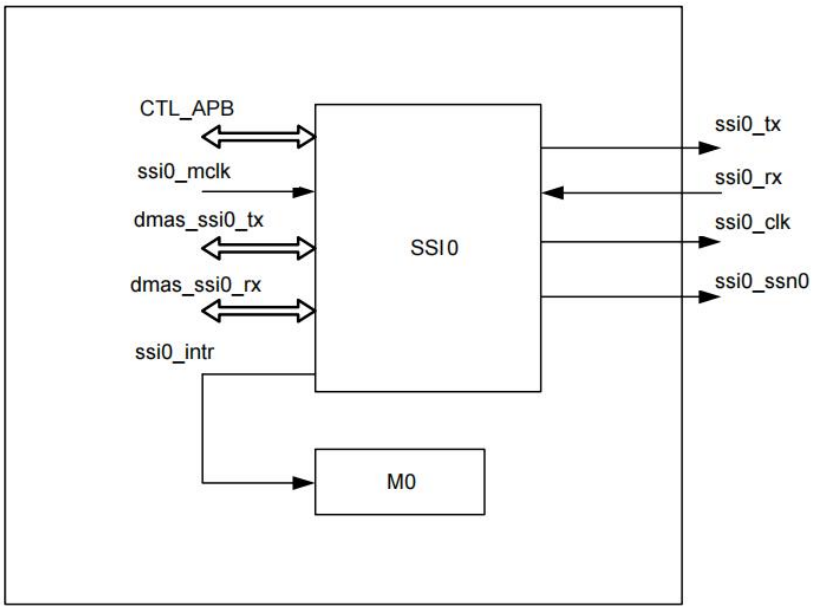

图 15-5 SPI 接口信号图

表 15-1 是 SPI 的接口信号描述。

表 15-1 SPI 的接口信号描述

名称	宽度	方向	描述	备注
CTL_APB 接口				
dmas_ssi0_tx			SPI0 发送与 DMA 之间的硬件握手信号：存储器->DMA->SPI0	内部信号
dmas_ssi0_rx			SPI0 接收与 DMA 之间的硬件握手信号：SPI0->DMA->存储器	内部信号
dmas_ssi1_tx			SPI1 发送与 DMA 之间的硬件握手信号：存储器->DMA->SPI1	内部信号
dmas_ssi1_rx			SPI1 接收与 DMA 之间的硬件握手信号：SPI1->DMA->存储器	内部信号
dmas_ssi2_tx			SPI2 发送与 DMA 之间的硬件握手信号：存储器->DMA->SPI2	内部信号
dmas_ssi2_rx			SPI2 接收与 DMA 之间的硬件握手信号：SPI2->DMA->存储器	内部信号
SPI0 信号				
ssi0_mclk	1	I	SPI0 工作时钟，来自 CPR 模块，默认频率为 8MHz，详细配置参见 CPR 模块的 CPR_SSI0_MCLK_CTL 寄存器	内部信号
ssi0_tx	1	O	SPI0 的数据输出信号	管脚信号
ssi0_rx	1	I	SPI0 的数据输入信号	管脚信号
ssi0_clk	1	O	SPI0 的输出时钟信号	管脚信号
ssi0_ssn	1	O	SPI0 的从设备选择信号低电平有效	管脚信号
SPI0 中断				
ssi0_intr	1	O	SPI0 中断输出，送给 M0	内部信号
SPI1 信号				
ssi1_mclk	1	I	SPI1 工作时钟，来自 CPR 模块，默认频率为 8MHz，详细配置参见 CPR 模块的 CPR_SSI1_MCLK_CTL 寄存器	内部信号
ssi1_tx	1	O	SPI1 的数据输出信号	管脚信号
ssi1_rx	1	I	SPI1 的数据输入信号	管脚信号
ssi1_clk	1	O	SPI1 的输出时钟信号	管脚信号

ssi1_ssn	1	O	SPI1 的从设备选择信号低电平有效	管脚信号
SPI1 中断				
ssi1_intr	1	O	SPI1 中断输出，送给 M0	内部信号
SPI2 信号				
ssi2_mclk	1	I	SPI2 工作时钟，来自 CPR 模块，默认频率为 8MHz，详细配置参见 CPR 模块的 CPR_SSI2_MCLK_CTL 寄存器	内部信号
ssi2_tx	1	O	SPI2 的数据输出信号	管脚信号
ssi2_rx	1	I	SPI2 的数据输入信号	管脚信号
ssi2_clk	1	O	SPI2 的输出时钟信号	管脚信号
ssi2_ssn	1	O	SPI2 的从设备选择信号低电平有效	管脚信号
SPI2 中断				
ssi2_intr	1	O	SPI2 中断输出，送给 M0	内部信号

15.2. SPI 寄存器地址映射

- SPI0 的基地址为：0x40013000
- SPI1 的基地址为：0x40014000
- SPI2 的基地址为：0x40014800

表 15-2 是 SPI 的寄存器概述和地址映射。

表 15-2 SPI 寄存器映射表

寄存器名	描述	方向和宽度	地址偏移	复位值
SPI_CTRL0	SPI 控制寄存器 0	RW 12bits	0x00	0x07
SPI_CTRL1	SPI 控制寄存器 1	RW 16bits	0x04	0x00
SPI_EN	SPI 使能寄存器	RW 1bit	0x08	0x00
SPI_SE	SPI 从设备选择寄存器	RW 2bits	0x10	0x00
SPI_BAUD	SPI 波特率选择寄存器	RW 16bits	0x14	0x00
SPI_TXFTL	SPI 发送 FIFO 阈值寄存器	RW 3bits	0x18	0x00
SPI_RXFTL	SPI 接收 FIFO 阈值寄存器	RW 3bits	0x1C	0x00
SPI_TXFL	SPI 发送 FIFO 有效值寄存器	R 4bits	0x20	0x00
SPI_RXFL	SPI 接收 FIFO 有效值寄存器	R 4bits	0x24	0x00
SPI_STS	SPI 状态寄存器	R 5bits	0x28	0x06
SPI_IE	SPI 中断使能寄存器	RW 5bits	0x2C	0x3F
SPI_IS	SPI 中断状态寄存器	R 5bits	0x30	0x00
SPI_RIS	SPI 原始中断状态寄存器	R 5bits	0x34	0x00
SPI_TXOIC	SPI 发送 FIFO 上溢出中断清除寄存器	R 1bit	0x38	0x00
SPI_RXOIC	SPI 接收 FIFO 上溢出中断清除寄存器	R 1bit	0x3C	0x00
SPI_RXUIC	SPI 接收 FIFO 下溢中断清除寄存器	R 1bit	0x40	0x00
SPI_IC	SPI 中断清除寄存器	R 1bit	0x48	0x00
SPI_DMAS	SPI 的 DMA 控制寄存器	RW 2bits	0x4C	0x00
SPI_DMATDL	SPI 的 DMA 发送阈值寄存器	RW 3bits	0x50	0x00
SPI_DMARDL	SPI 的 DM 接收阈值寄存器	RW 3bits	0x54	0x00
SPI_DATA	SPI 数据寄存器	RW 16bits	0x60	0x00

15.3. SPI 寄存器描述

15.3.1. SPI 控制寄存器 0（SPI_CTRL0）

- 地址偏移：0x00

该寄存器控制串行数据的传输。当 SPI 使能时，不能修改该寄存器的值。

位数	名称	方向	复位值	描述
24	SSTE	RW	0x0	SPI 超总线帧格式使能
22：21	SPI_FRF	RW	0x0	SPI 模式。 00：STD SPI 01：DUAL SPI 02：QUAD API 03：QCTAL SPI
15:12	SSI_CFS	RW	0x0	控制长度
11	SRL	RW	0x0	移位寄存器环回：控制进入测试模式。该模式下，连接发送寄存器的输出到接收寄存器的输入。 0：正常模式。 1：环回模式。
10	SLV_OE	RW	0x0	从机输出使能 0x1:从机输出不使能 0x0:从机输出使能
9：8	TMOD	RW	0x0	传输模式。 用于控制 SPI 的传输模式。该控制不影响 SPI 传输的双工性，只指示发送/接收的数据是否有效。 在“仅发”模式下，通过 ssi0_rx 接收到的数据无效，不存放到接收 FIFO 中。 在“仅收”模式下，通过 ssi0_tx 发送的数据无效。SPI 将在传输过程中，反复发送第一个写入发送 FIFO 的数据。 00：收发 01：仅发 10：仅收 11：保留
7	SCPOL	RW	0x0	串行时钟极性控制：控制串口时钟 ssi0_clk 停止时的电平。 0：ssi0_clk 停止时，ssi0_clk 保持低电平 1：ssi0_clk 停止时，ssi0_clk 保持高电平

6	SCPH	RW	0x0	串行时钟的相位控制：控制串口时序，决定何时开始传输数据。 0：在 ssi0_ssn0/ssi0_ssn1 的下降沿，开始数据传输。 1：在 ssi0_ssn0/ssi0_ssn1 有效后的第一个 ssi0_clk 的变化沿，开始数据传输。
5：4	FRF	RW	0x0	帧格式控制。 00：Motorola SPI 协议。 01：TI SSP 协议。 其它：保留。
3：0	DFS	RW	0x7	数据帧长度。 0x00~0x02：保留。 0x03~0x0F：对应 4~16 比特串行数据传输。

15.3.2. SPI 控制寄存器 1（SPI_CTRL1）

- 地址偏移：0x04

在“仅收”模式下，该寄存器可以控制串口传输何时结束。当 SPI 使能时，不能修改该寄存器的值。

位数	名称	方向	复位值	描述
15：0	NDF	RW	0x0	数据帧的个数。 当 SPI 工作在“仅收”模式下（TMOD=10），控制 SPI0 持续接收的数据帧个数。 该 NDF 值为 N，SPI 将持续接收 N＋1 个数据后，结束串口传输。SPI 最多连续接收 64K 个数据帧。

15.3.3. SPI 使能寄存器（SPI_EN）

- 地址偏移：0x08

通过该寄存器使能 SPI 和禁止 SPI。

位数	名称	方向	复位值	描述
0	SEN	RW	0x0	SPI 使能控制。 SPI 禁用时，所有收发立即停止，并清空接收 FIFO 和发送 FIFO。 SPI 使能时，不能修改某些控制寄存器的值。在 SPI 禁用时，可以节省系统的功耗。 0：不使能

				1：使能

15.3.4. SPI 从设备选择寄存器（SPI_SE）

- 地址偏移：0x10

通过该寄存器使能来自 SPI 的从设备选择。当 SPI 使能或者 SPI 处于 BUSY 状态，不能写该寄存器。

位数	名称	方向	复位值	描述
1	SS1	RW	0x0	从设备选择 1。 如果设置为‘1’，从设备使能信号（ssi_ssn1）会在传输开始时有效。 对该 bit 写‘0’或‘1’，只有在传输开始后，才会在端口信号 ssi_ssn1 上有所体现。 0：不选择从设备 1 1：选择从设备 1
0	SS0	RW	0x0	从设备选择 0。 如果设置为‘1’，从设备使能信号（ssi_ssn0）会在传输开始时有效。对该 bit 写‘0’或‘1’，只有在传输开始后，才会在端口信号 ssi_ssn0 上有所体现。 0：不选择从设备 0。 1：选择从设备 0。

15.3.5. SPI 波特率选择寄存器（SPI_BAUD）

- 地址偏移：0x14

通过该寄存器设置串口时钟（ssi_mclk）的频率。当 SPI 使能时，不能写该寄存器。

位数	名称	方向	复位值	描述
15:0	SCKDV	RW	0x0	SPI 时钟分频。 最低位固定为 0，所以分频比为偶数。如果该位为 0，那么相应的 ssi_clk 没有时钟输出。 ssi_clk 有如下公式： ssi_clk = ssi_mclk/SCKDV SCKDV 是 2 到 65534 之间的偶数。

15.3.6. SPI 发送 FIFO 阈值寄存器（SPI_TXFTL）

- 地址偏移：0x18

该寄存器设置发送 FIFO 的阈值。当 SPI 使能时，不能写该寄存器。

位数	名称	方向	复位值	描述
2：0	TFT	RW	0x0	发送 FIFO 阈值。 用于设置触发发送空中断（TXE_INTR）的发送 FIFO 阈值。 0x00~0x07：分别对应发送 FIFO 中数据少于等于 0~7 时，触发发送空中断（TXE_INTR）。

15.3.7. SPI 接收 FIFO 阈值寄存器（SPI_RXFTL）

- 地址偏移：0x1C

该寄存器设置接收 FIFO 的阈值。当 SPI 使能时，不能写该寄存器。

位数	名称	方向	复位值	描述
2：0	RFT	RW	0x0	接收 FIFO 阈值。 用于设置触发接收满中断（RXF_INTR）的接收 FIFO 阈值。 0x00~0x07：分别对应发送 FIFO 中数据大于等于 1~8 时，触发接收满中断（RXF_INTR）。

15.3.8. SPI 发送 FIFO 有效值寄存器（SPI_TXFL）

- 地址偏移：0x20

该寄存器包含发送 FIFO 有效数据个数。

位数	名称	方向	复位值	描述
3：0	TXTFL	R	0x0	发送 FIFO 有效值：发送 FIFO 有效数据个数。

15.3.9. SPI 接收 FIFO 有效值寄存器（SPI_RXFL）

- 地址偏移：0x24

该寄存器包含接收 FIFO 有效数据个数。

位数	名称	方向	复位值	描述
3：0	RXTFL	R	0x0	接收 FIFO 有效值：接收 FIFO 有效数据个数。

15.3.10. SPI 状态寄存器（SPI_STS）

- 地址偏移：0x28

该寄存器用于指示 SPI 当前传输状态、FIFO 状态和可能发生的发送/接收错误。

位数	名称	方向	复位值	描述
4	RFF	R	0x0	接收 FIFO 满。 当接收 FIFO 完全满时（8 个），该比特设置。 当接收 FIFO 有一个或者多个空位时，该比特清零。 0：接收 FIFO 未满。 1：接收 FIFO 满。
3	RFNE	R	0x0	接收 FIFO 非空。 当接收 FIFO 包含有一个或者多个数据时，该比特设置。 当接收 FIFO 为空时，该比特清零。 0：接收 FIFO 空。 1：接收 FIFO 非空。
2	TFE	R	0x1	发送 FIFO 空。 当发送 FIFO 完全空时（0 个），该比特设置。 当发送 FIFO 包含一个或多个数据时，该比特清零。 0：发送 FIFO 非空。 1：发送 FIFO 空。
1	TFNF	R	0x1	发送 FIFO 非满。 当发送 FIFO 有一个或者多个空位时，该比特设置。 当发送 FIFO 完全满时（8 个），该比特清零。 0：发送 FIFO 满。 1：发送 FIFO 未满。
	BUSY	R	0x0	SPI BUSY 标志。 如果该比特设置，标志 SPI 正在进行串行传输。 如果 SPI 处于空闲或者不使能的状态，该比特清零。 0：SPI 空闲或者不使能。 1：SPI 正在进行串行传输。

15.3.11. SPI 中断使能寄存器（SPI_IE）

- 地址偏移：0x2C

该寄存器使能或者不使能所有的 SPI 产生的中断。

位数	名称	方向	复位值	描述
4	RXFIE	RW	0x1	接收 FIFO 满中断使能。 0：RXF_INTR 中断不使能。 1：RXF_INTR 中断使能。
3	RXOIE	RW	0x1	接收 FIFO 上溢出中断使能。 0：RXO_INTR 中断不使能。 1：RXO_INTR 中断使能。
2	RXUIE	RW	0x1	接收 FIFO 下溢出中断使能。 0：RXU_INTR 中断不使能。 1：RXU_INTR 中断使能。
1	TXOIE	RW	0x1	发送 FIFO 上溢出中断使能。 0：TXO_INTR 中断不使能。 1：TXO_INTR 中断使能。
0	TXEIE	RW	0x1	发送 FIFO 空中断使能。 0：TXE_INTR 中断不使能。 1：TXE_INTR 中断使能。

15.3.12. SPI 中断状态寄存器（SPI_IS）

- 地址偏移：0x30

该寄存器为 SPI 的中断经过使能后的状态。

位数	名称	方向	复位值	描述
4	RXFIS	R	0x0	接收 FIFO 满中断状态。 0：RXF_INTR 中断经使能后无效。 1：RXF_INTR 中断经使能后有效。
3	RXOIS	R	0x0	接收 FIFO 上溢出中断状态。 0：RXO_INTR 中断经使能后无效。 1：RXO_INTR 中断经使能后有效。
2	RXUIS	R	0x0	接收 FIFO 下溢出中断状态。 0：RXU_INTR 中断经使能后无效。 1：RXU_INTR 中断经使能后有效。

1	TXOIS	R	0x0	发送 FIFO 上溢出中断状态。 0：TXO_INTR 中断经使能后无效。 1：TXO_INTR 中断经使能后有效。
0	TXEIS	R	0x0	发送 FIFO 空中断状态。 0：TXE_INTR 中断经使能后无效。 1：TXE_INTR 中断经使能后有效。

15.3.13. SPI 原始中断状态寄存器（SPI_RIS）

- 地址偏移：0x34

该寄存器为 SPI 的中断未经使能的状态。

位数	名称	方向	复位值	描述
4	RXFIR	R	0x0	接收 FIFO 满原始中断状态。 0：RXF_INTR 未经使能的中断无效。 1：RXF_INTR 未经使能的中断有效。
3	RXOIR	R	0x0	接收 FIFO 上溢出原始中断状态。 0：RXO_INTR 未经使能的中断无效。 1：RXO_INTR 未经使能的中断有效。
2	RXUIR	R	0x0	接收 FIFO 下溢出原始中断状态。 0：RXU_INTR 未经使能的中断无效。 1：RXU_INTR 未经使能的中断有效。
1	TXOIR	R	0x0	发送 FIFO 上溢出原始中断状态。 0：TXO_INTR 未经使能的中断无效。 1：TXO_INTR 未经使能的中断有效。
0	TXEIR	R	0x0	发送 FIFO 空原始中断状态。 0：TXE_INTR 未经使能的中断无效。 1：TXE_INTR 未经使能的中断有效。

15.3.14. SPI 发送 FIFO 上溢出中断清除寄存（SPI_TXOIC）

- 地址偏移：0x38

位数	名称	方向	复位值	描述
0	TXOIC	R	0x0	清除发送 FIFO 上溢出中断。 寄存器值为相应中断状态。读该寄存器清除 TXO_INTR 中断，写无效。

15.3.15. SPI 接收 FIFO 上溢出中断清除寄存（SPI_RXOIC）

- 地址偏移：0x3C

位数	名称	方向	复位值	描述
0	RXOIC	R	0x0	清除接收 FIFO 上溢出中断。 寄存器值为相应中断状态。 读该寄存器清除 RXO_INTR 中断，写无效。

15.3.16. SPI 接收 FIFO 下溢中断清除寄存器（SPI_RXUIC）

- 地址偏移：0x40

位数	名称	方向	复位值	描述
0	RXUIC	R	0x0	清除接收 FIFO 下溢出中断： 寄存器值为相应中断状态。 读该寄存器清除 RXU_INTR 中断，写无效。

15.3.17. SPI 中断清除寄存器（SPI_IC）

- 地址偏移：0x48

位数	名称	方向	复位值	描述
0	IC	R	0x0	清除中断： 如果 TXO_INTR、RXO_INTR、RXU_INTR 中断状态有效，那么寄存器值为‘1’，否则为‘0’。读该寄存器清除 TXO_INTR、RXO_INTR、RXU_INTR 中断，写无效。

15.3.18. SPI 的 DMA 控制寄存器（SPI_DMAS）

- 地址偏移：0x4C

该寄存器控制与 DMAS 的接口是否使能。

位数	名称	方向	复位值	描述
1	TDMAE	RW	0x0	发送 DMA 接口使能。 使能/不使能发送的 DMA 接口 0：发送 DMA 接口不使能 1：发送 DMA 接口使能。
0	RDMAE	RW	0x0	接收 DMA 接口使能。 使能/不使能接收的 DMA 接口

				0：接收 DMA 接口不使能 1：接收 DMA 接口使能。

15.3.19. SPI 的 DMA 发送阈值寄存器（SPI_DMATDL）

- 地址偏移：0x50

位数	名称	方向	复位值	描述
2：0	DMATDL	RW	0x0	DMA 发送数据阈值。 控制发送数据过程中，产生 DMA 传输请求的阈值；如果发送 FIFO 中的数据个数等于小于本阈值，并且 TDMAE=1，发出 DMA 传输请求。

15.3.20. SPI 的 DMA 接收阈值寄存器（SPI_DMARDL）

- 地址偏移：0x54

位数	名称	方向	复位值	描述
2：0	DMARDL	RW	0x0	DMA 接收数据阈值。 控制接收数据过程中，产生 DMA 传输请求的阈值；如果接收 FIFO 中的数据个数等于大于本阈值，并且 RDMAE=1，发出 DMA 传输请求。

15.3.21. SPI 数据寄存器（SPI_DATA）

- 地址偏移：0x60

SPI 数据寄存器（SPI_DATA）为发送/接收 FIFO 的缓存。读该寄存器，得到接收 FIFO 中的数据。写该寄存器，数据会送到发送 FIFO。只有在 SPI 使能后，才可以写该寄存器；当 SPI 不使能时，接收/发送 FIFO 处于复位状态。

位数	名称	方向	复位值	描述
15：0	DR	RW	0x0	数据寄存器。 由于数据帧小于等于 16bits，写数据时保证右对齐；读数据时，返回的数据也是自动右对齐的。 读：接收 FIFO 写：发送 FIFO

15.4. 工作流程

15.4.1. 从设备（SPI）非 DMA 传输模式

(1) 配置 SPI_CTRL0 的 FRF 位，选择 SPI 支持的帧格式是 SSP 还是 SPI。

(2) 如果 SPI 在使能状态，通过写‘0’到 SPI 使能寄存器（SPI_EN）的 SEN 位，不使能 SPI。

(3) 设置 SPI 的传输控制寄存器，对这些寄存器的操作顺序没有限制。

- 设置 SPI 控制寄存器 0（SPI_CTRL0），设置与主设备相同的串口时钟极性（SCPOL）和串口时钟相位（SCPH）；
- 设置 SPI 发送 FIFO 阈值寄存器（SPI_TXFTL）和 SPI 接收 FIFO 阈值寄存器（SPI_RXFTL）；
- 通过 SPI 中断使能寄存器（SPI_IE）设置所需中断使能。

(4) 写‘1’到 SPI 使能寄存器（SPI_EN）的 SEN 位，使能 SPI。

(5) 如果工作在收发模式或者只发模式，向发送 FIFO 写数据，通过写 SPI 数据寄存器（SPI_DATA）实现。

(6) SPI 已经准备好传输数据，传输是通过外接主设备发起的。

(7) 当 SPI 开始传输数据，此时 SPI 状态寄存器（SPI_STS）的 BUSY 标志为‘1’。如果发生发送 FIFO 空中断（TXE_INTR），处理器响应中断，并向发送 FIFO 中写数据（写 SPI_DATA）。如果发生接收 FIFO 满中断（RXF_INTR），处理器响应中断，并从接收 FIFO 中读数据（读 SPI_DATA）。

(8) 外接主设备控制传输的结束。

(9) 如果不是工作在“仅发”模式，从接收 FIFO 中将所有的数据读出。

(10) 通过写‘0’到 SPI 使能寄存器（SPI_EN）的 SEN 位，不使能 SPI。

15.4.2. 从设备（SPI）DMA 传输模式（仅发送）

(1) 配置 SPI_CTRL0 的 FRF 位，选择 SPI 支持的帧格式是 SSP 还是 SPI。

(2) 如果 SPI 在使能状态，通过写‘0’到 SPI 使能寄存器（SPI_EN）的 SEN 位，不使能 SPI。

(3) 设置 SPI 的传输控制寄存器，对这些寄存器的操作顺序没有限制。

- 设置 SPI 控制寄存器 0（SPI_CTRL0），设置与主设备相同的串口时钟极性（SCPOL）和串口时钟相位（SCPH）。
- 设置 SPI 发送 FIFO 阈值寄存器（SPI_TXFTL）。

(4) 配置 DMA 与 SPI 通过硬件握手的方式输出数据。

- DMA 中指定两个通道 3 和 11 分别作为 SPI 的发送和接收通道。

- 对于 SPI 的发送通道，选择 DMAS 相应通道的目的端数据宽度（DST_WIDTH）等于 16bits。
- 写‘1’到 SPI 使能寄存器（SPI_EN）的 SEN 位，使能 SPI。
- 使能 DMA，并使能 DMA 的 SPI 发送通道。

(5) SPI 已经准备好传输数据，传输是通过外接主设备发起的。

(6) 当 SPI 开始传输数据，此时 SPI 状态寄存器（SPI_STS）的 BUSY 标志为‘1’。如果 SPI 发生 DMA 发送请求，DMA 向发送 FIFO 写数据（写 SPI_DATA）。

(7) 外接主设备控制传输的结束。

(8) 通过写‘0’到 SPI 使能寄存器（SPI_EN）的 SEN 位，不使能 SPI。

第 16 章 正交编解码（QDEC）

16.1. 概述

集成了一个正交解码器，从而支持正交编码输出的外设。正交解码器（QDEC）从 HID 设备/传感器中获取输入信号，从中译码得到步进（step）与方向信息。正交解码器（QDEC）的管脚和其他功能模块管脚复用。

- QDEC 的基地址为：0x40016000

16.1.1. 主要特性

- 支持 QEI 接口的设备和 Sensor；
- 最多支持 3 轴的输入设备，3 个 16 位符号型计数器提供步进与方向信息（X，Y，Z）；
- 支持窄脉冲毛刺过滤功能；
- 采样时钟频率和采样间隔可配置；
- 支持相对位移和绝对位移的判定和计算；
- 通过内置缓存支持轨迹记录；
- 支持多种中断请求，包括上下溢中断、非法输入中断和 Event 中断等；
- 可选 FIFO 缓存模式与 BYPASS 直通模式；

16.1.2. 功能描述

正交解码器（QDEC）支持正交编码输出的外设的解码，从而获取设备的步进与方向信息。QDEC 能够支持正交编码的三轴输入设备（如三轴陀螺仪，空鼠等）的译码。

正交编码是一种成对的数字信号，通称为 A 和 B，两信号相位差 90 度，输出呈现格雷码规律。通过格雷码，可以检测方向和相对位置。

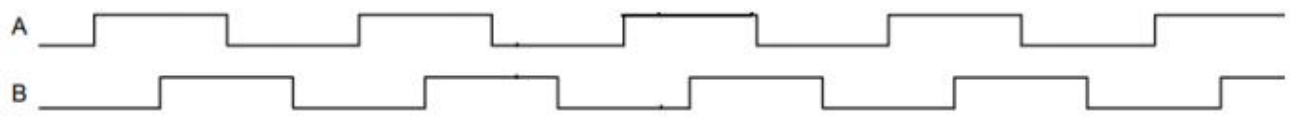

图 16-1 正交编码输出格式

16.1.3. 结构框图

模块的结构框图如图 3-72 所示。

- QDEC_FILTER：前置输入信号过滤，可以滤除窄脉宽毛刺，此功能可以通过软件配置关闭/使能。窄脉宽毛刺指的是宽度小于 2 个采样时钟周期的输入信号；
- QDEC_DECODER(QDEC_CHn)：QDEC 译码器，分别对各个通道的 QEI 输入进行译码，得到 16 位的有符号计数值；
- QDEC_SAMPLE：采样模块，通过软件配置的采样间隔，周期性的对 QDEC 译码器的输出值进行采样，并送至输出；
- QDEC_FIFO(nFIFO)：采样数据缓存 FIFO，用来缓存 QDEC 译码的输出信息，通过记录输出，获取移动的轨迹信息；
- QDEC_ASYNC：异步处理模块，用来处理从 clk 域到 pclk 域，以及从 pclk 域到 clk 域的信息交换；
- QDEC_REGIF：QDEC 的寄存器模块，包含 QDEC 的控制、状态、输出等信息；
- APBBUSIF：QDEC 模块的 APB 接口。

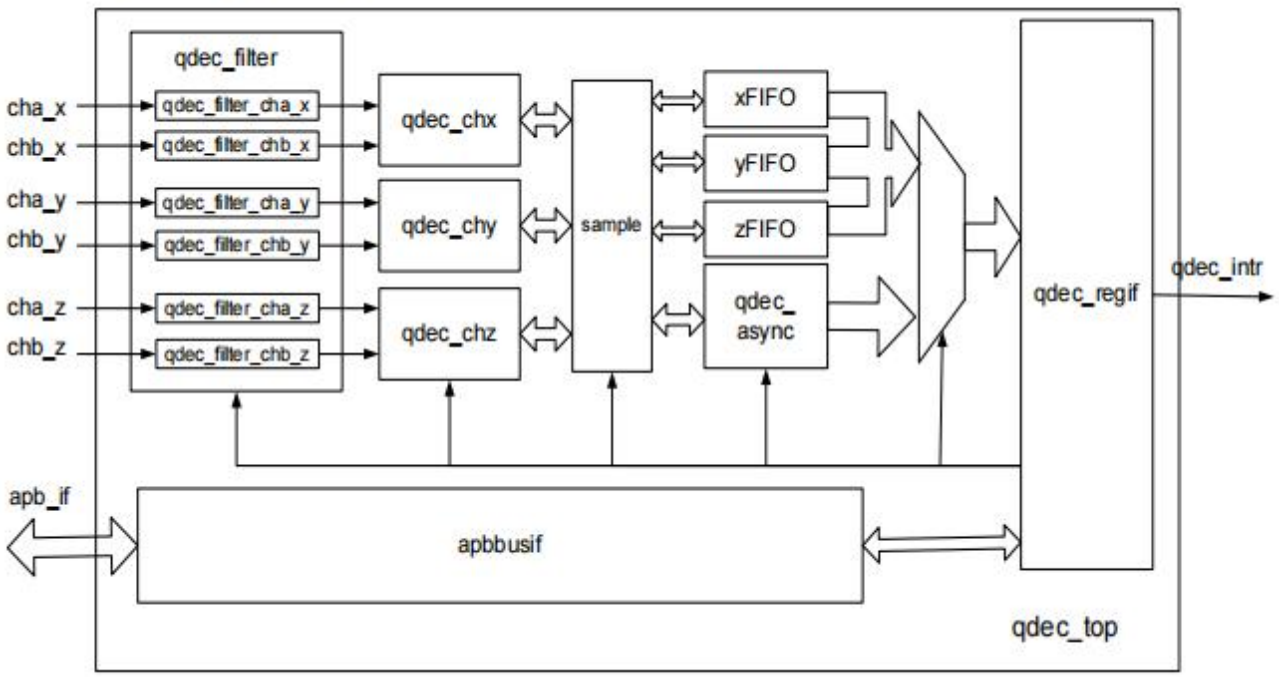

图 16-2 正交解码器（QDEC）模块框图

16.1.4. 中断说明

QDEC 支持 7 种类型的中断，包括 QDEC Event 中断、QDEC 输入非法组合中断、QDEC 上下溢中断、QDEC FIFO 中断等，详见表 中断说明表所示

中断说明表

中断状态位	中断原始状态位	中断使能位	描述
QDEC_EVT_INT	QDEC_EVT_INT_RAW	QDEC_EVT_INT_EN	QDEC Event 中断
QDEC_INV_INT	QDEC_INV_INT_RAW	QDEC_INV_INT_EN	QDEC 输入信号非法中断
QDEC_OVF_INT	QDEC_OVF_INT_RAW	QDEC_OVF_INT_EN	QDEC 计数器上溢出中断
QDEC_UDF_INT	QDEC_UDF_INT_RAW	QDEC_UDF_INT_EN	QDEC 计数器下溢出中断
QDEC_FIFO_AF	QDEC_FIFO_AF_RAW	QDEC_FIFO_AF_EN	QDEC FIFO almost full 中断
QDEC_FIFO_EP	QDEC_FIFO_EP_RAW	QDEC_FIFO_EP_EN	QDEC FIFO 读空中断
I2S_FIFO_OV	I2S_FIFO_OV_RAW	I2S_FIFO_OV_EN	QDEC FIFO 溢出中断

16.1.5. 信号及接口描述

表 3-43 是信号及接口描述表。

QDEC 信号及接口描述表

名称	宽度	方向	描述	备注
APB 从设备接口				内部信号
QDEC 时钟与控制信号				内部信号
clk	1	I	采样时钟输入	内部信号
rstn	1	I	QDEC 复位信号	内部信号
qdec_intr	1	O	QDEC 中断输出	内部信号
Scan_mode	1	I	scan_mode 选择	内部信号
QDEC 时钟与控制信号				管脚信号
qdax	1	I	正交解码器 X 轴的 channel A	管脚信号
qdbx	1	I	正交解码器 X 轴的 channel B	管脚信号
qday	1	I	正交解码器 Y 轴的 channel A	管脚信号
qdby	1	I	正交解码器 Y 轴的 channel B	管脚信号
qdaz	1	I	正交解码器 Z 轴的 channel A	管脚信号

qdbz	1	I	正交解码器 Z 轴的 channel B	管脚信号

16.2. 寄存器地址映射

这些寄存器被映射到系统的地址空间，可由 M0 进行配置。

- QDEC 的基地址为：0x40001600

寄存器名	描述	方向和宽度	地址偏移	复位值
QDEC_CTL	QDEC 控制寄存器	RW 1bits	0x00	0x0000
QDEC_SAMP_CTL	QDEC 采样控制寄存器	RW 26bits	0x04	0x0000
QDEC_SAMPLE	QDEC 采样状态寄存器	RW 16bits	0x08	0x0000
QDEC_ACC		R 32bits	0x0C	0x0000
QDEC_ACC_R		W 32bits	0x10	0x0000
QDEC_DB		RW 16bits	0x14	0x0000
QDEC_DB_R		RW 18bits	0x18	0x0000
QDEC_INT	QDEC 中断寄存器	RC 7bits	0x1C	0x0000
QDEC_INT_MSK		R 7bits	0x20	0x0000
QDEC_STAT		RW 7bits	0x24	0x0000

16.3. 寄存器描述

- 寄存器地址：0x40001600

16.3.1. 控制寄存器（QDEC_CTL）

- 地址偏移：0x00

位数	名称	方向	复位值	描述
31:7	--	--	--	保留
6	DB_FILTER_EN	RW	0x0	
5	SINGLE_SAMPLE_RST_EN	RW	0x0	
4	AUTO_CLR_EN	RW	0x0	
3	SOFT_CLR	RW	0x0	
2	SOFT_RST	RW	0x0	

1	QDEC_START	RW	0x0	
0	QDEC_EN	RW	0x0	QDEC 使能位： 1：使能正交解码器； 0：不使能正交解码器

16.3.2. （QDEC_SAMP_CTL）

- 地址偏移：0x04

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.3. （QDEC_SAMPLE）

- 地址偏移：0x08

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.4. （QDEC_ACC）

- 地址偏移：0x0C

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.5. （QDEC_ACC_R）

- 地址偏移：0x10

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.6. （QDEC_DB）

- 地址偏移：0x14

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.7. （QDEC_DB_R）

- 地址偏移：0x18

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.8. （QDEC_INT）

- 地址偏移：0x20

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.9. （QDEC_INT_MSK）

- 地址偏移：0x24

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器

16.3.10. （QDEC_STAT）

- 地址偏移：0x2C

位数	名称	方向	复位值	描述
31:20	--	--	--	保留
19:16	DB_SAMP_DIV	RW	0x0	
11:8	QDEC_PTS	RW	0x0	
4：0	QDEC_DIVIDE	RW	0x0	QDEC Acc 分压器